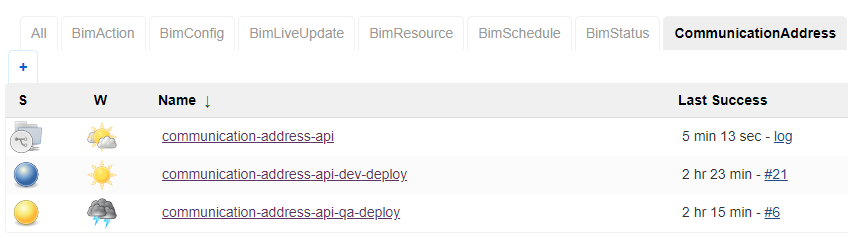
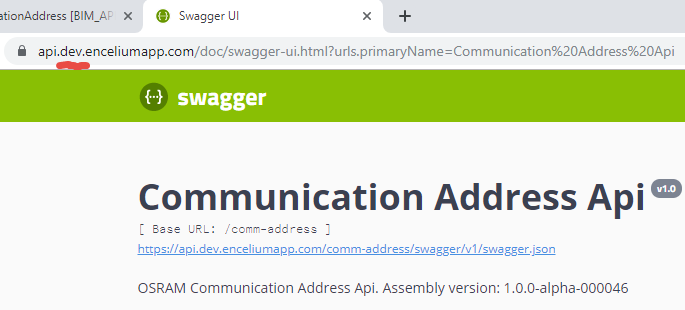
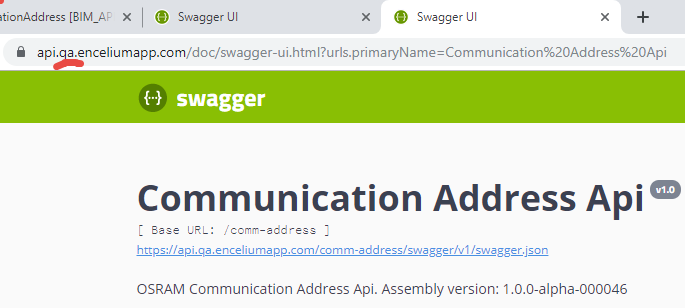
Run scripts against multiple environments with different credentials in CI/CD pipeline

# 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.



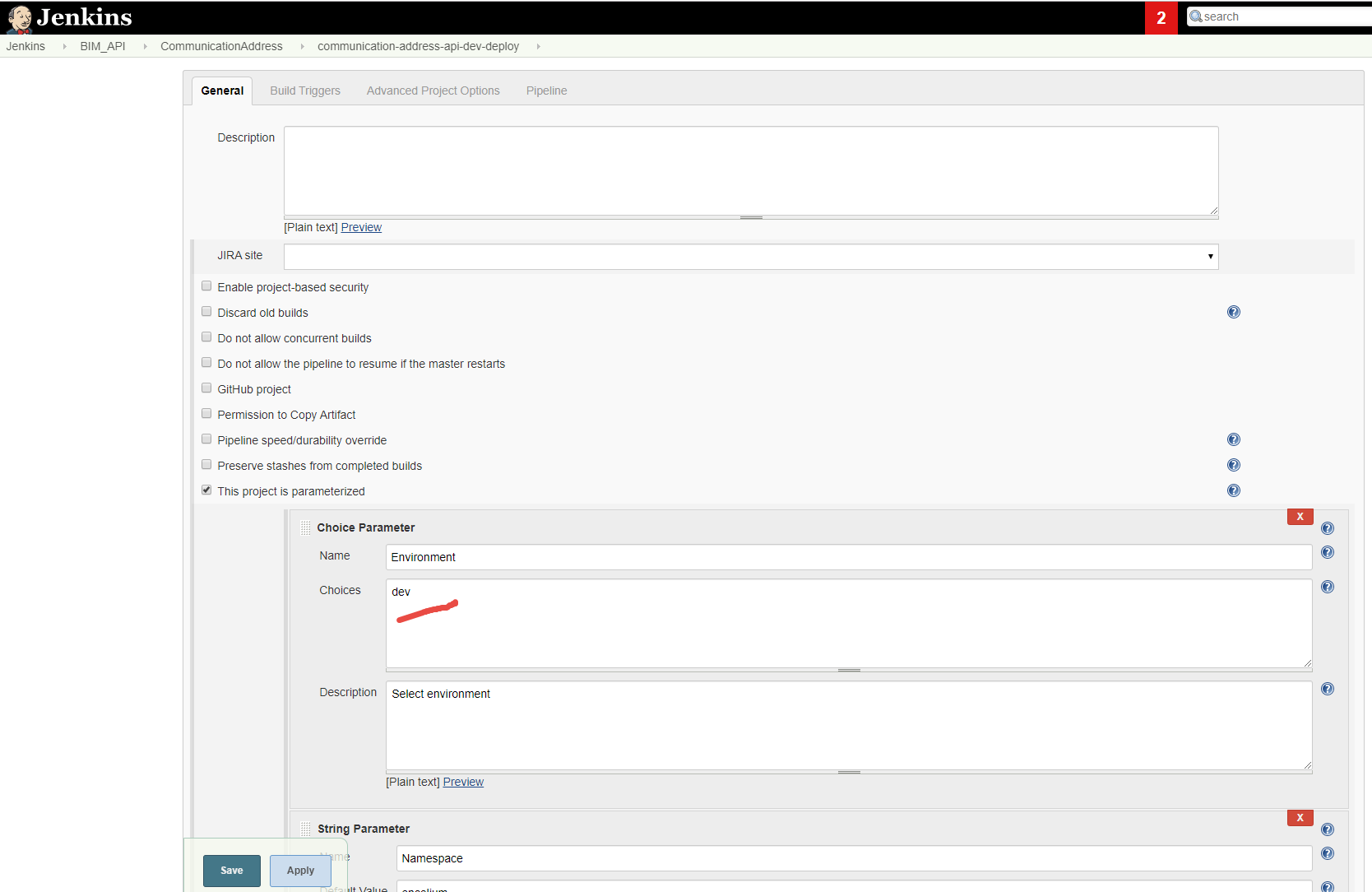
We currently only have 2 environments, dev and qa. In the future, we will also have staging and production environments. Each of the environment has its own deployment project as displayed above. “bim\_communication\_address\_api” is one of the API/microservice we have, inside the “bim\_communication\_address\_api” folder, you can see “Jenkinsfile” and “Deploy.Jenkinsfile”. The “Jenkinsfile” oversees 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” oversees 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 containerized application has been deployed by Kubernetes/Helm, the corresponding service’s URL will be populated as displayed below, and our QA sanity scripts will be running against the dev URL, and both sanity and regression scripts will be running against the qa URL.

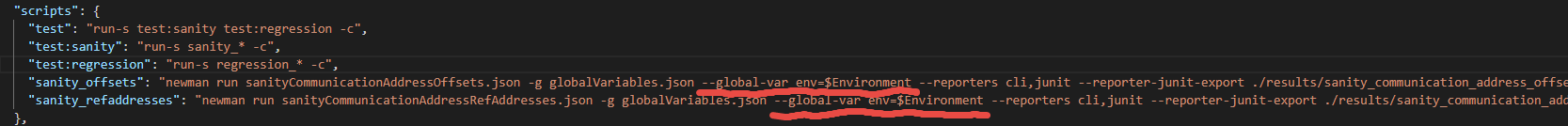
As a result, in order to run postman scripts againest different URLs inside CICD pipeline, we need to generate the URL, URL credentials (currently is basic authentication of username and password) these 3 parameters from Jenkins GUI or shared jenkins groovy libraries, and pass them into our postman scripts which are 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 with the same name “Environment”, but differnt value “qa”.



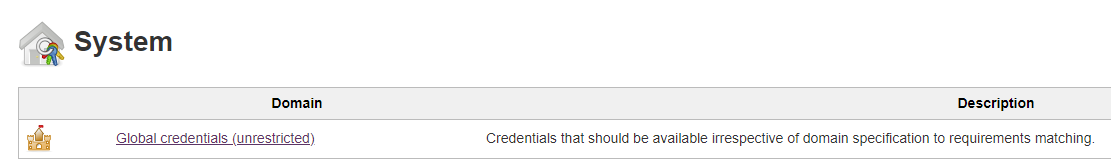
This Environment variable from Jenkins can be easily passed into newman command as --global-var env=$Environment to overwrite our postman global variables “env” which was defined inside “globalVariables.json”. The “globalVariables.json” is located under the “bim\_communication\_address\_api\qa\postman” folder. Our API URLs are [https://api.dev.enceliumapp.com](https://api.dev.enceliumapp.com/) for dev environment and [https://api.qa.enceliumapp.com](https://api.qa.enceliumapp.com/) for qa environment, and we use “env” variable to construct part of the URL during test. The “env” variable should have value “dev” when testing dev URL and should have value “qa” when testing qa URL. Below is the “package.json” file if we want to pass in --global-var env=$Environment into newman command.



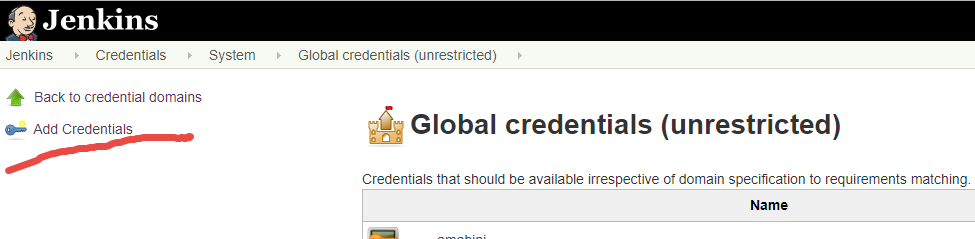
However, to execute npm run locally (no in Jenkins, but in my PC) will have a problem after above change. I tried to set system variable on my local PC to have a variable named as “Environment”, but newman command cannot recognized it. Let us put this problem aside, we still need to overwrite the API username and password since under different environment, these credentials are different.



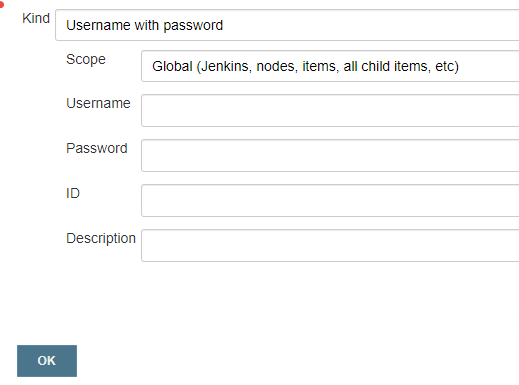
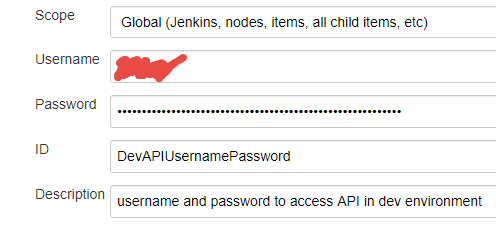
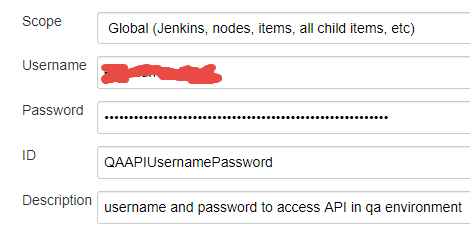
From Jenkins GUI, click the left-hand side “Credentials” selection from the menu, you will notice that there are many different types of credentials have been created by our DevOps person. I created 2 new sets of credentials here, one set for dev environment username and password (named as DevAPIUsernamePassword),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



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.



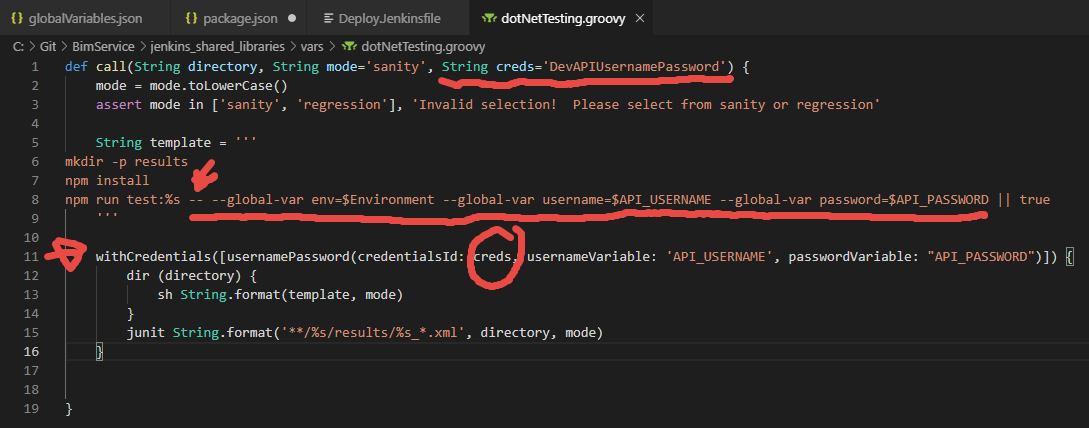
You can choose different kind of credentials, for my case, I picked “Username with password”. Fill in the Username and Password fields. The ID filed will be the name you pass in to your script. The result for dev and qa credentials are in the middle and right below. In case you made any mistake, you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

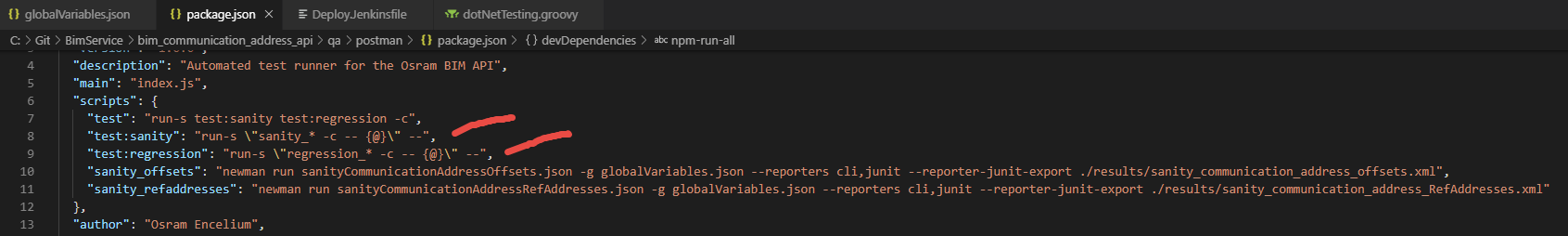
# Step 2: pass in Jenkins parameters to package.json

Let me just show you the result.

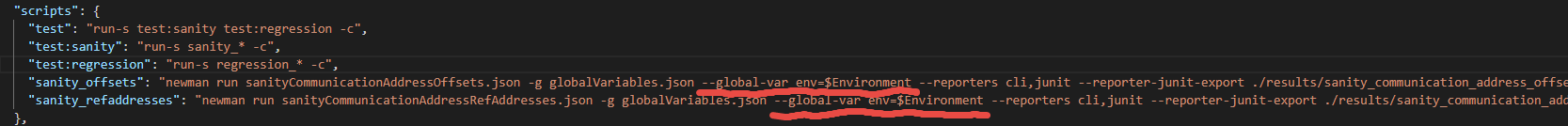
First, from “jenkins\_shared\_libraries\vars” directory, find the “dotNetTesting.groovy” file. Line 11 wrapped original code block from line 12 to line 15. The use of line 11 is based on <https://jenkins.io/doc/pipeline/steps/credentials-binding/>, search for “usernamePassword”. Basically, it takes the credentialId which I just defined in Jenkins GUI, 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 4th image below). The withCredentials function gets the credentials from Jenkins by id, then passes the username and password from the Jenkins credentials to variable API\_USERNAME and API\_PASSWORD.



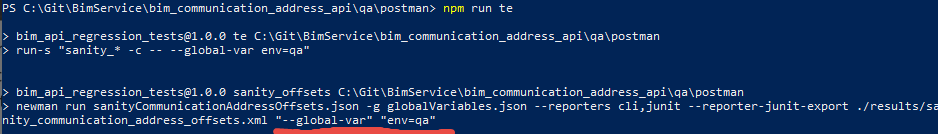
At this point, API\_USERNAME and API\_PASSWORD are the same as Environment parameter defined in the Jenkins deployment projects, so they can be used in the same way in line 8 of “dotNetTesting.groovy” file above. Note that you need to write -- first before pass in --gloable-var). Before the change, line 8 was npm run test:%s || true originally.



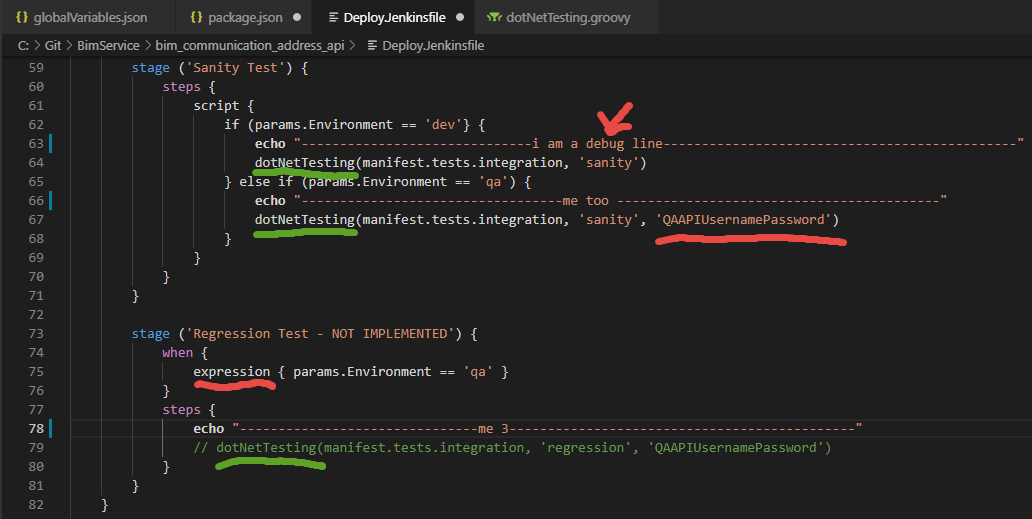
The benefit of line 8 is to solve the problem that I mentioned in step 1, if we pass in --global-var env=$Environment --global-var username=$API\_USERNAME --global-var password=$API\_PASSWORD inside the “package.json” line 10 and 11 under “newman run”, then I can no longer run these scripts on my local PC. 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. Note “run-s test:sanity test:regression -c” is actually “npm run-s test:sanity test:regression -c”, just the “npm” can be ignored. [npm run-s](https://github.com/mysticatea/npm-run-all/blob/master/docs/run-s.md) will run given npm-scripts sequentially. This command is the shorthand of npm-run-all -s. -c is an option of npm run-s, which means “--continue-on-error”. Line 8 of “package.json” aggregated all the tests 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. You can debug locally by trying to change the “test:sanity” line in “package.json” to the format that you think would work, and run it via npm run in cmd. Note [{@}](https://github.com/mysticatea/npm-run-all/blob/master/docs/run-s.md) is an argument placeholder, when debug, please replace it with -global-var env=qa. For example, I find "te": "run-s \"sanity\_\* -c -- --global-var env=qa\"" also works. Observe the error message to improve, good luck, hehe.



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. The “dotNetTesting” function is used under “Sanity Test” and “Regression Test” stages in the “Deploy.Jenkinsfile”.



Couple things to note:

1, you cannot 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 use “expression”, then it will always be true, and you can never falls into the “else if” branch.

2, the use of echo line is 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” is displayed on the top (see below), 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 fall into.

