Project One Micro Service Integration with Jenkins

# Required Software

Vagrant: <https://www.vagrantup.com/>

Java 8: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

Eclipse or STS: <https://www.eclipse.org/> OR <https://spring.io/tools/sts>

Maven: <https://maven.apache.org/>

Postman: <https://www.getpostman.com/apps>

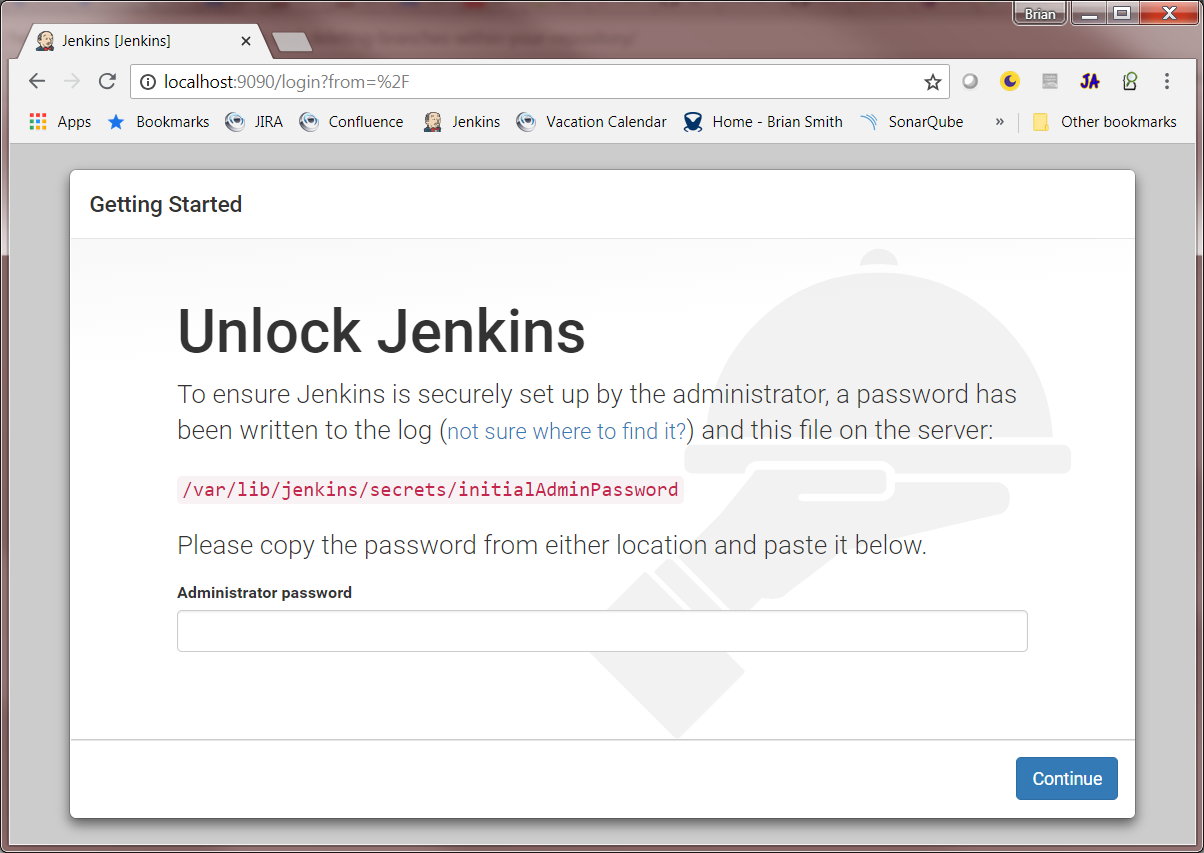
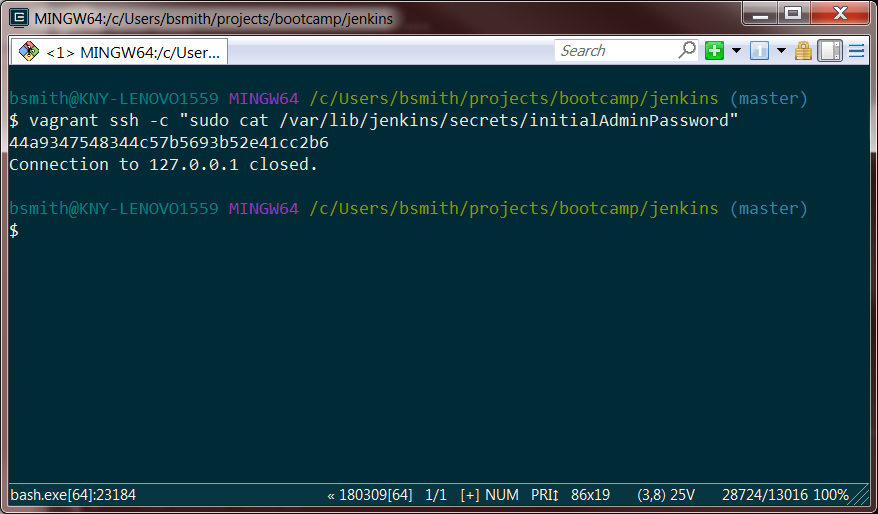
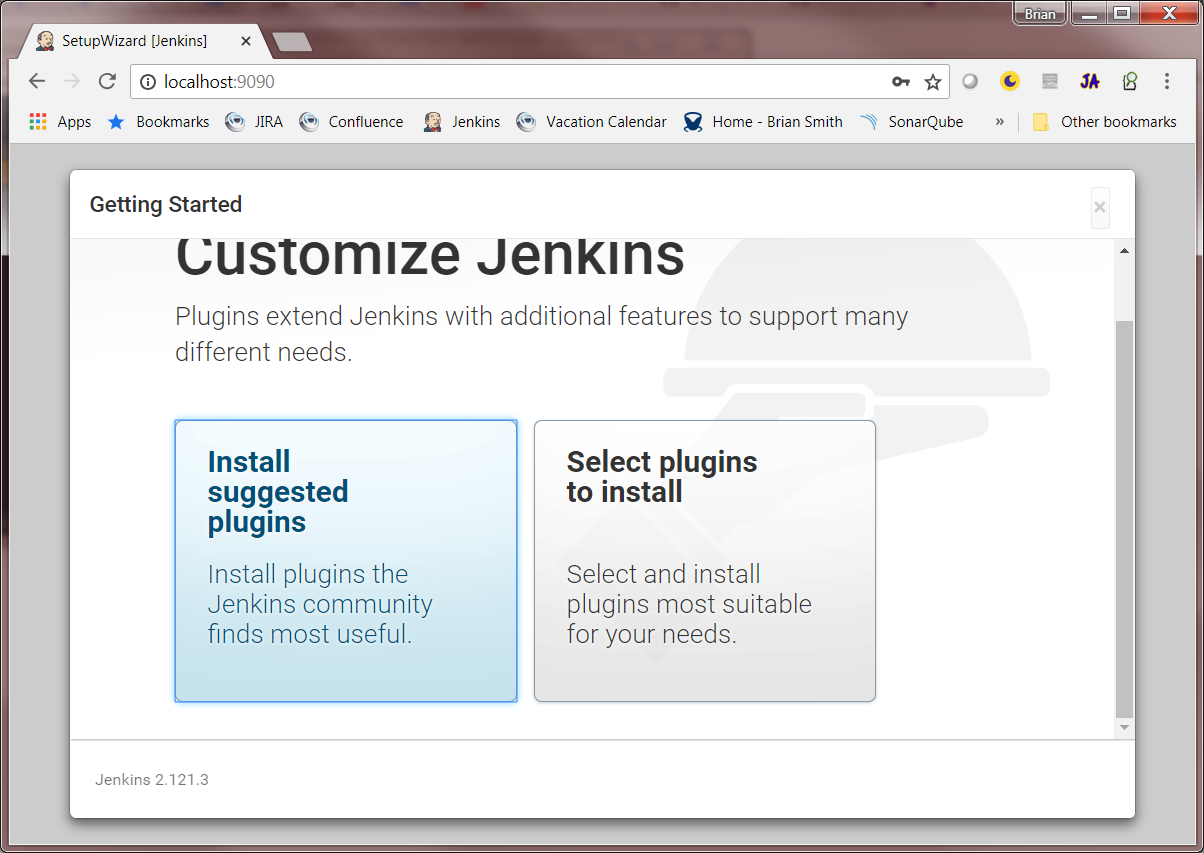
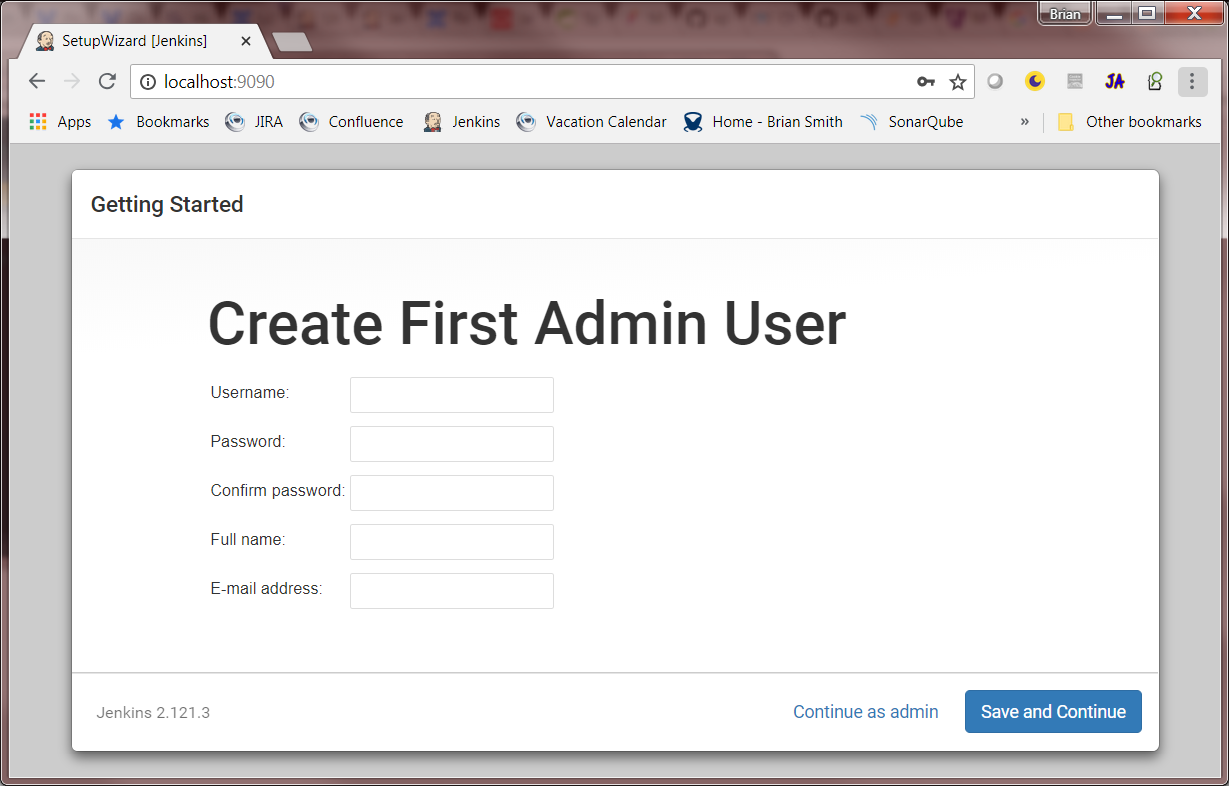
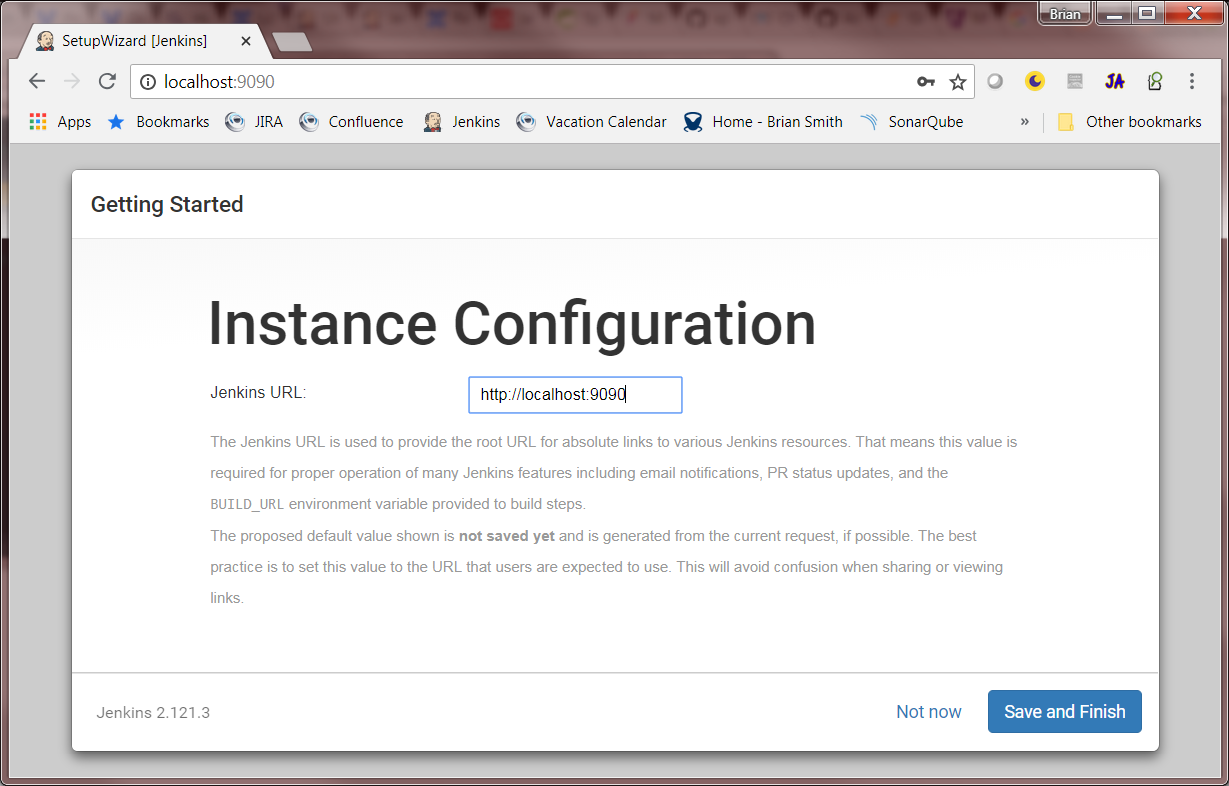
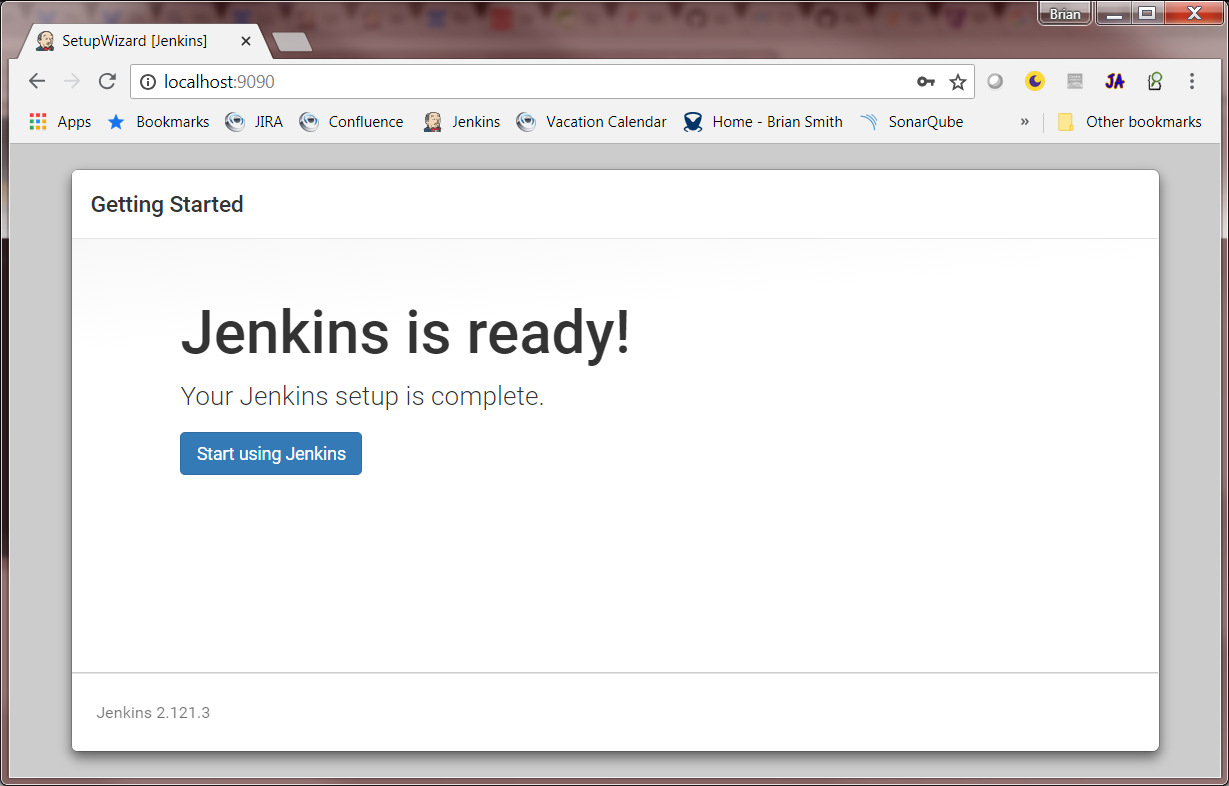
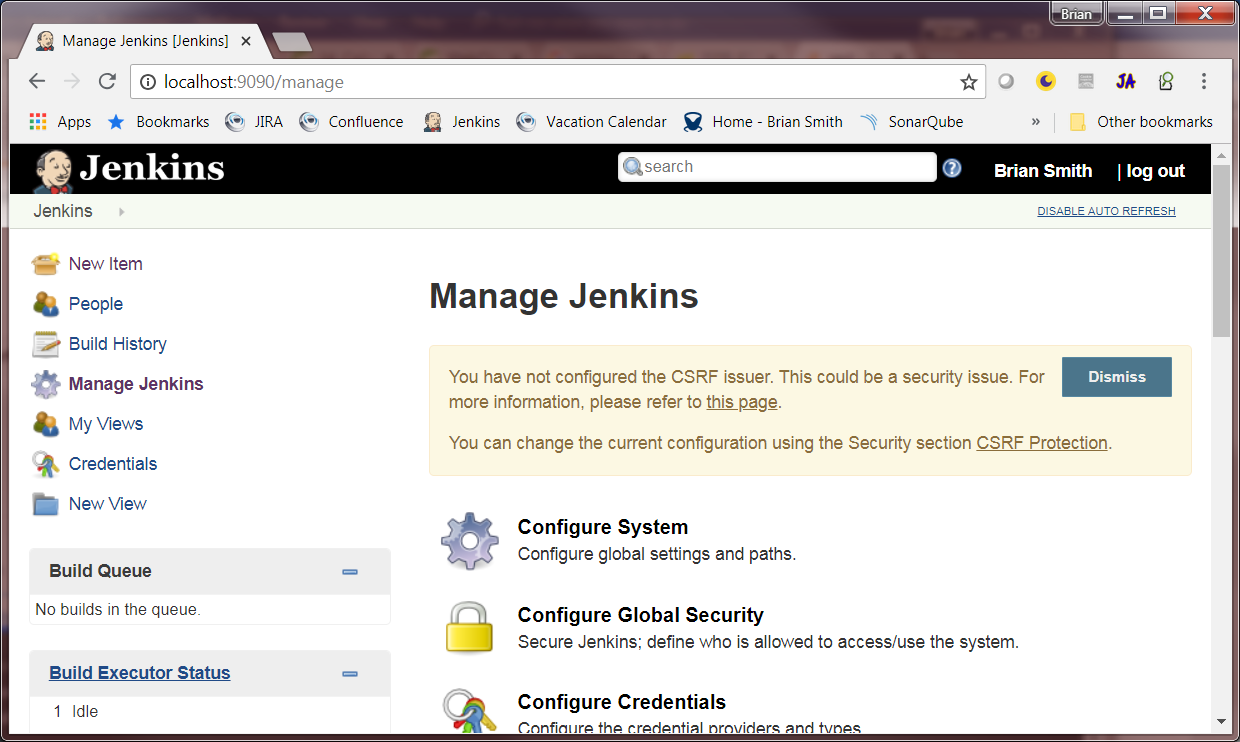
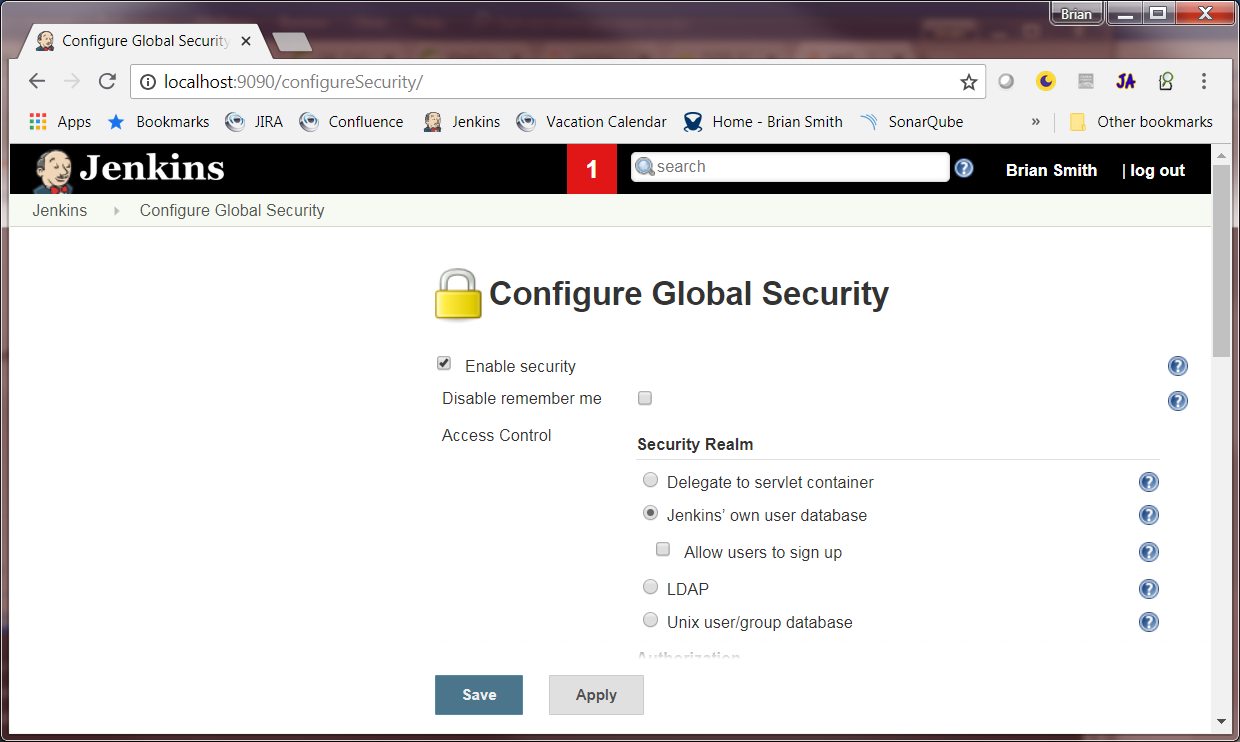
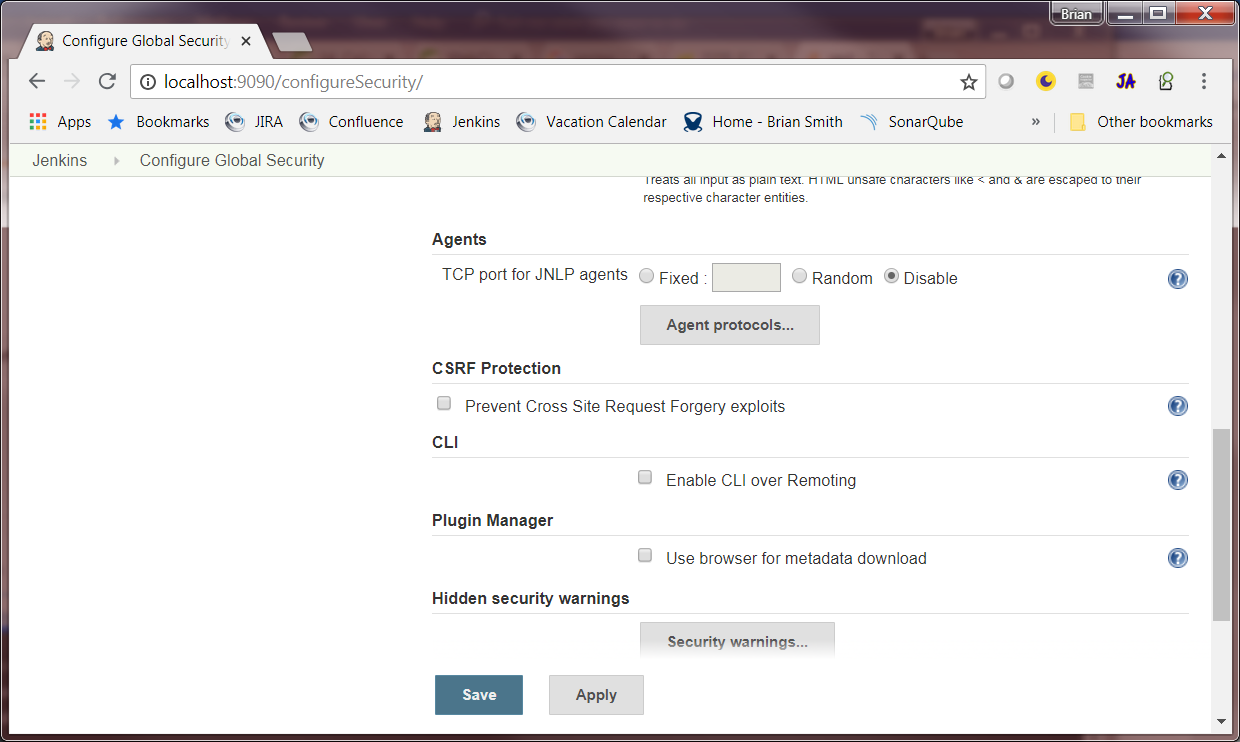
# Needed Code

The Jenkins Folder from the <https://github.com/xpanxion/xhire-spring-bootcamp> github repo

# Setup

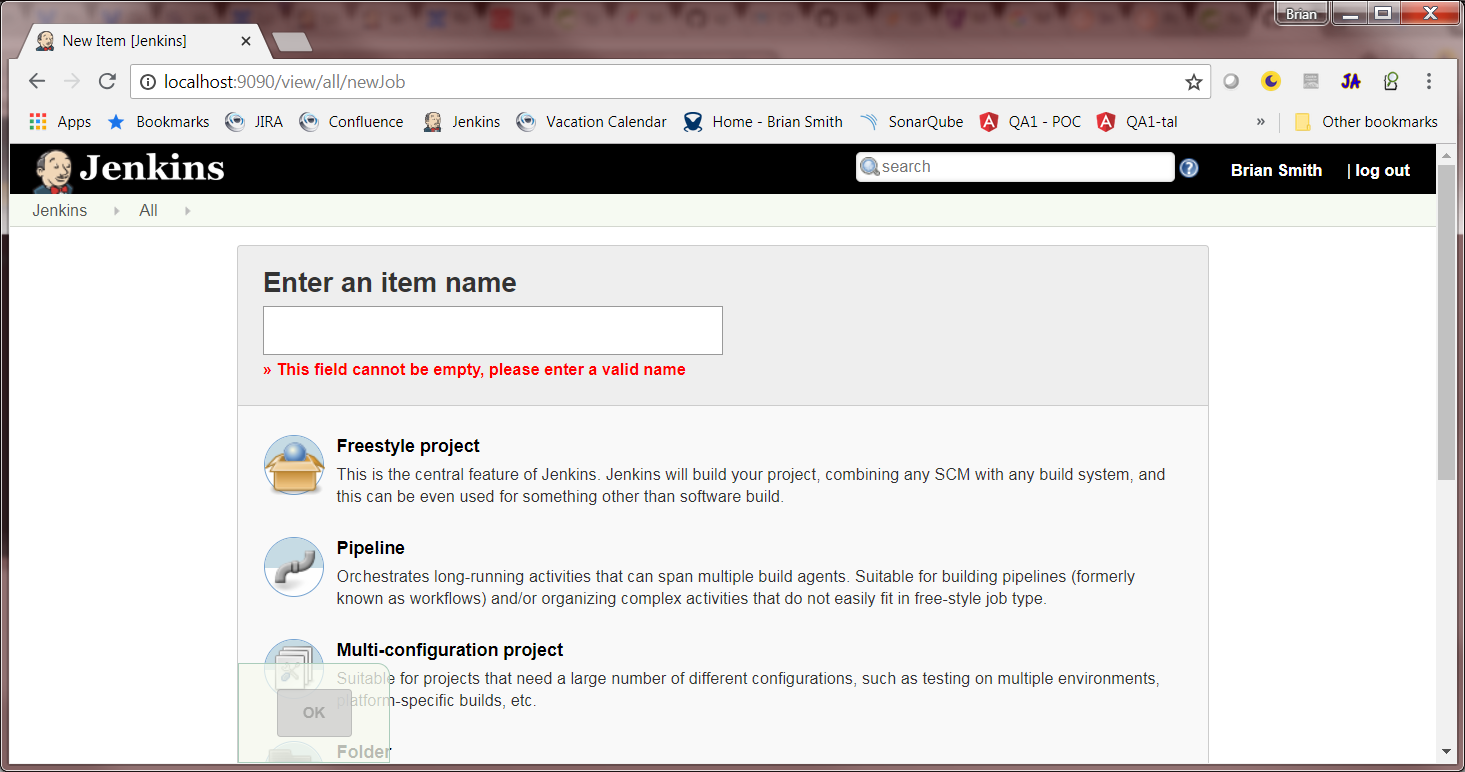
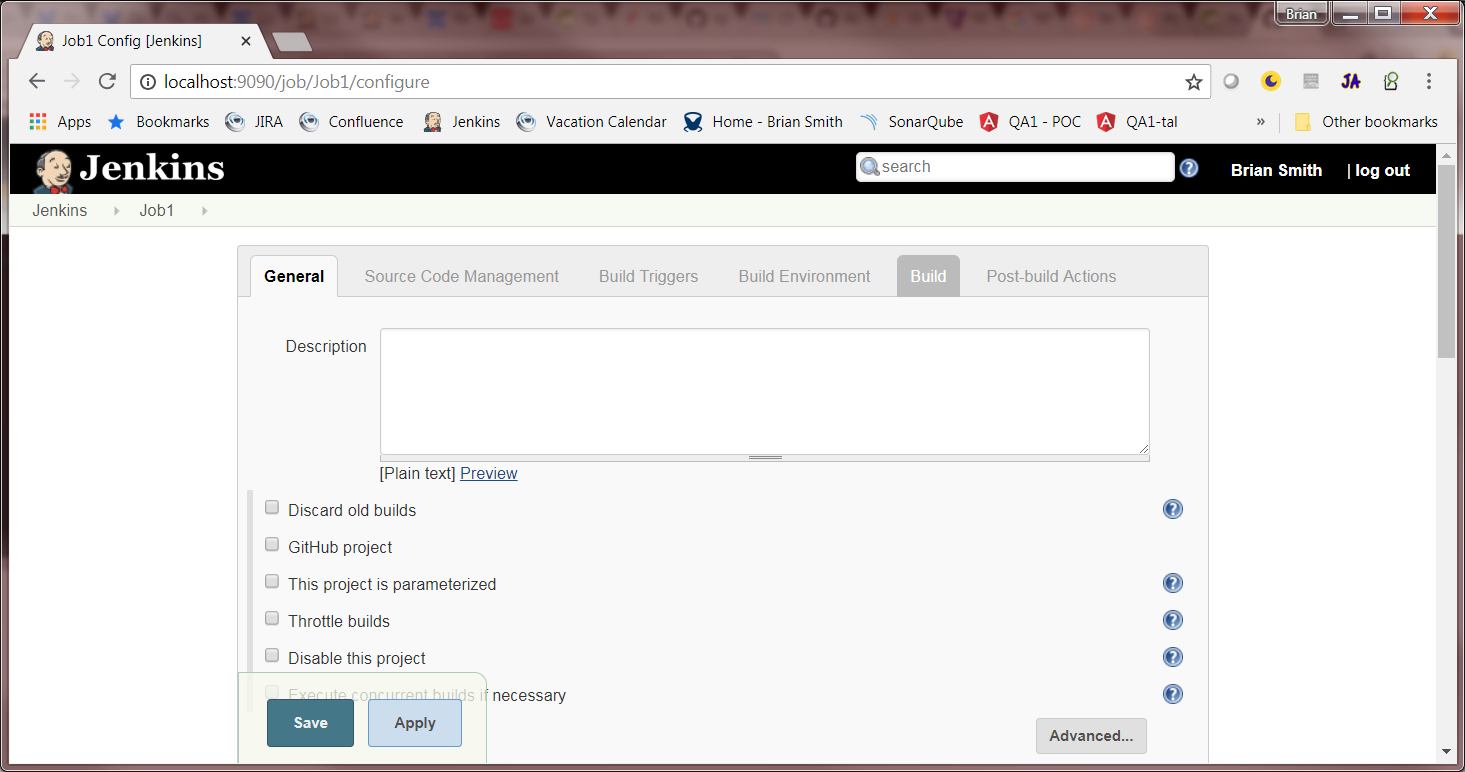
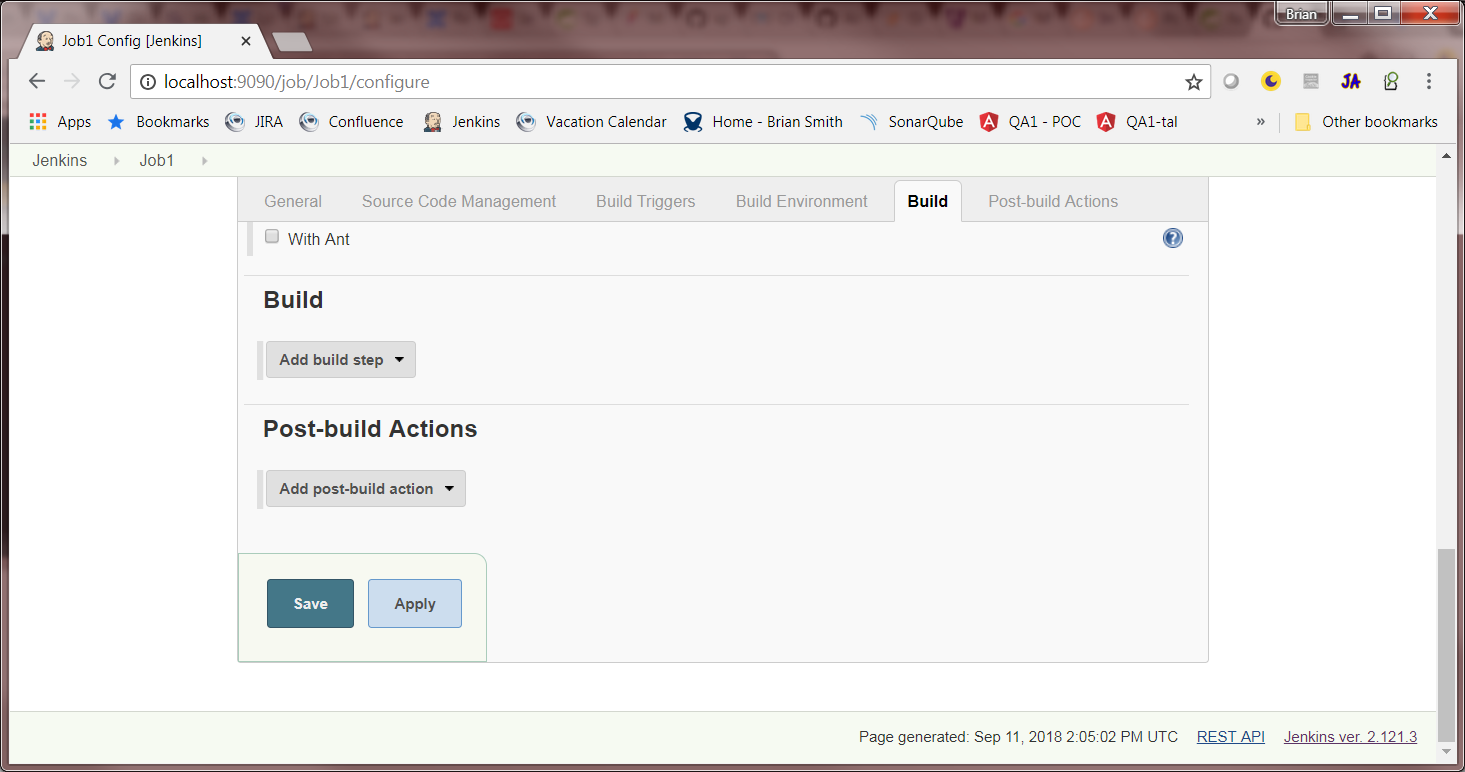
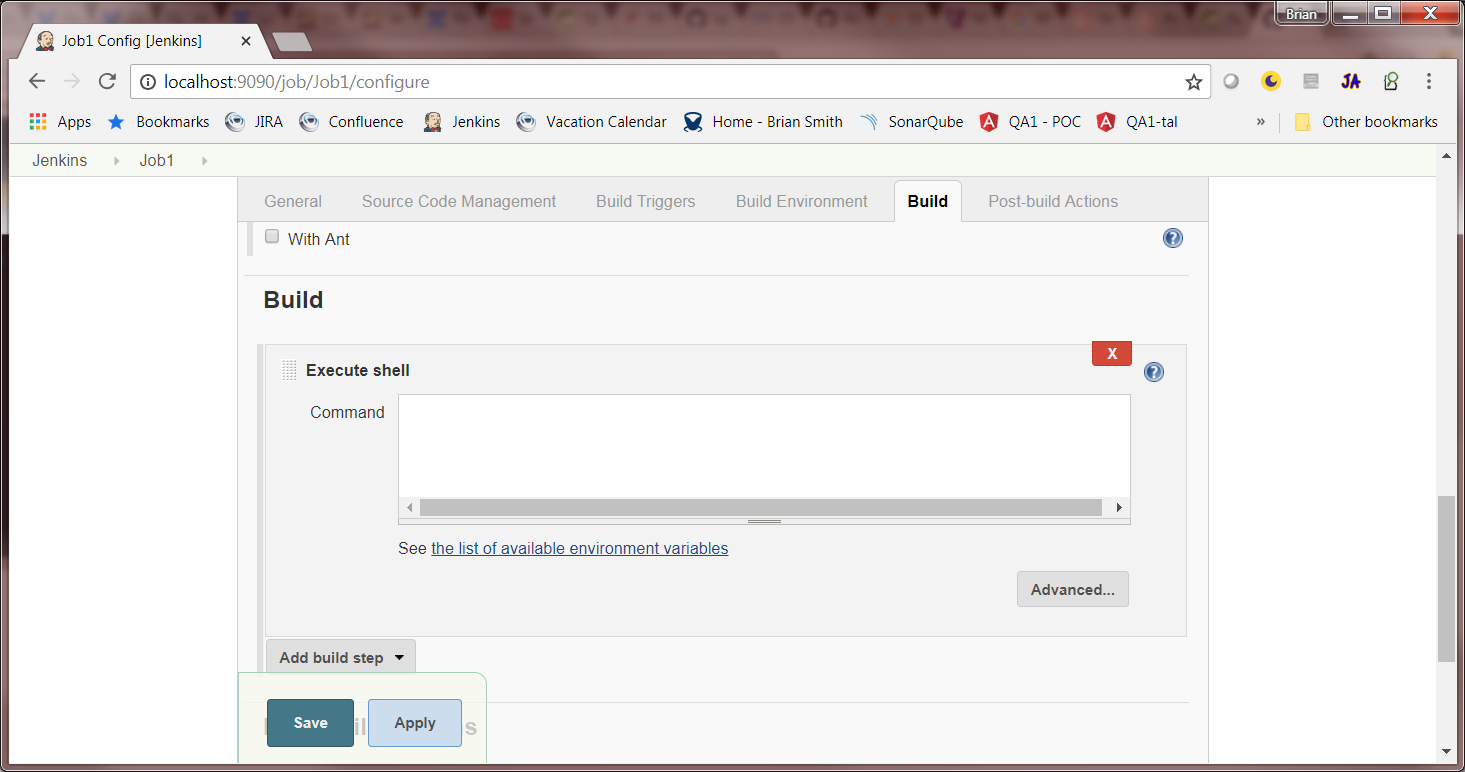
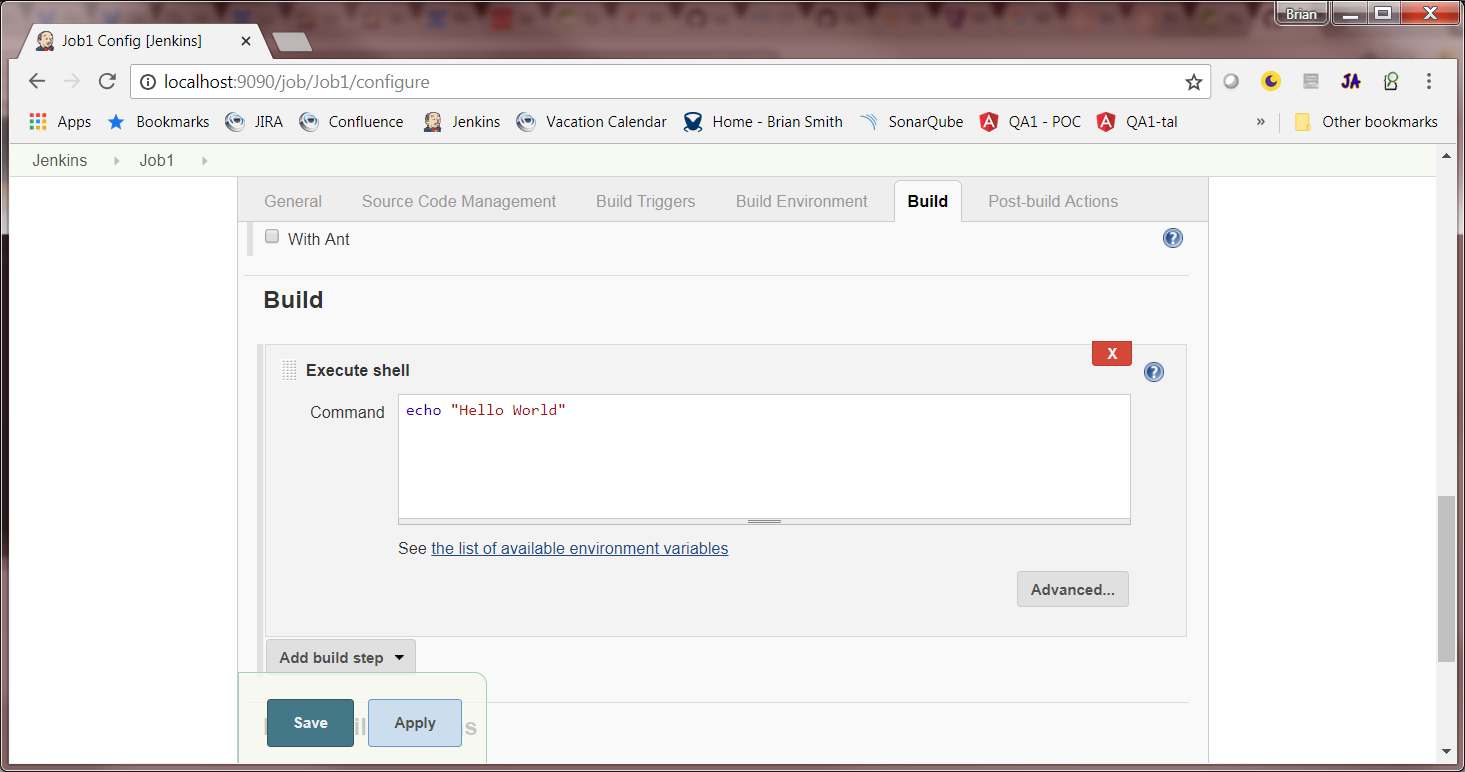
We will use vagrant to create a VM that has Jenkins running on it. Then we will setup a basic configuration, create a user and generate a couple of simple builds that we will use our Micro Service to access.

## Run Vagrant

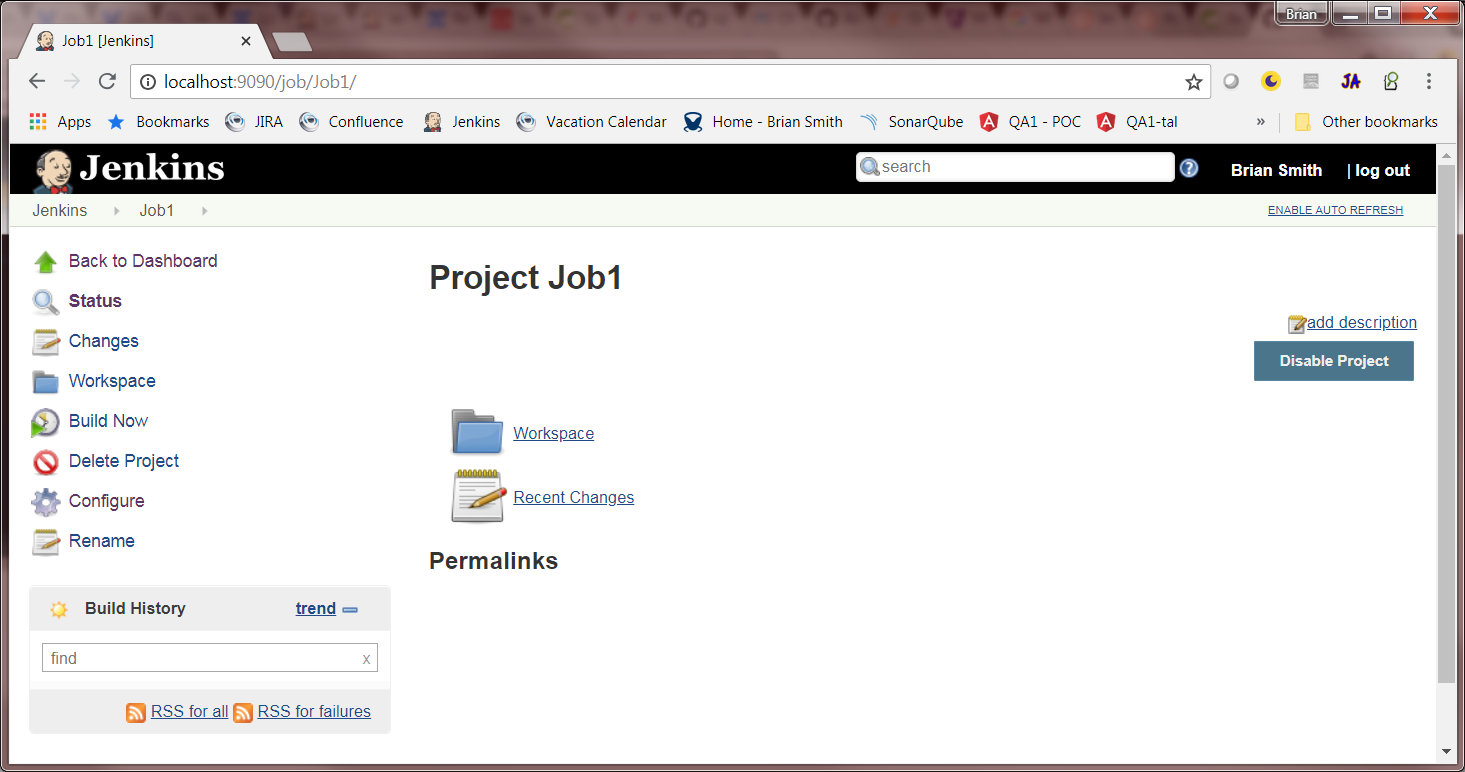
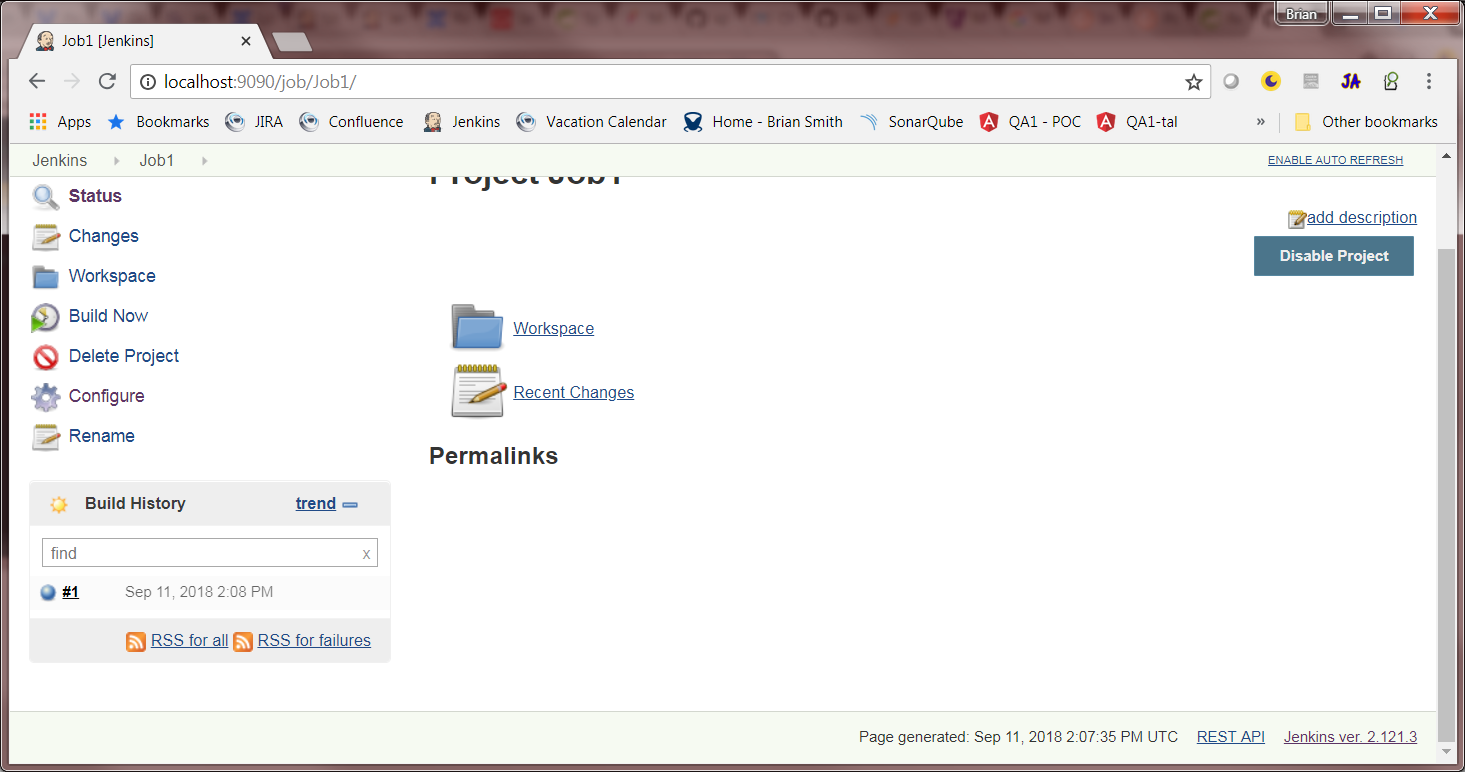
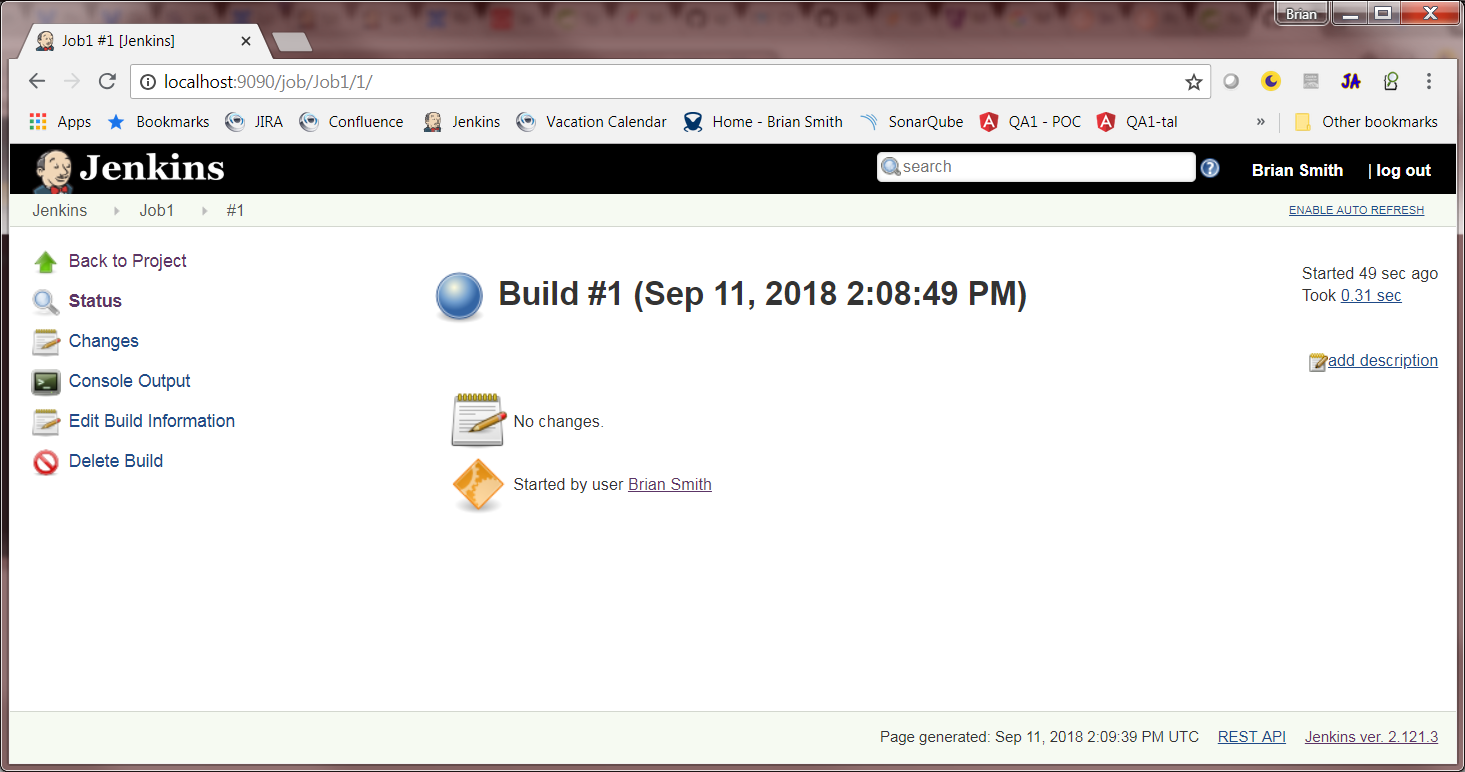
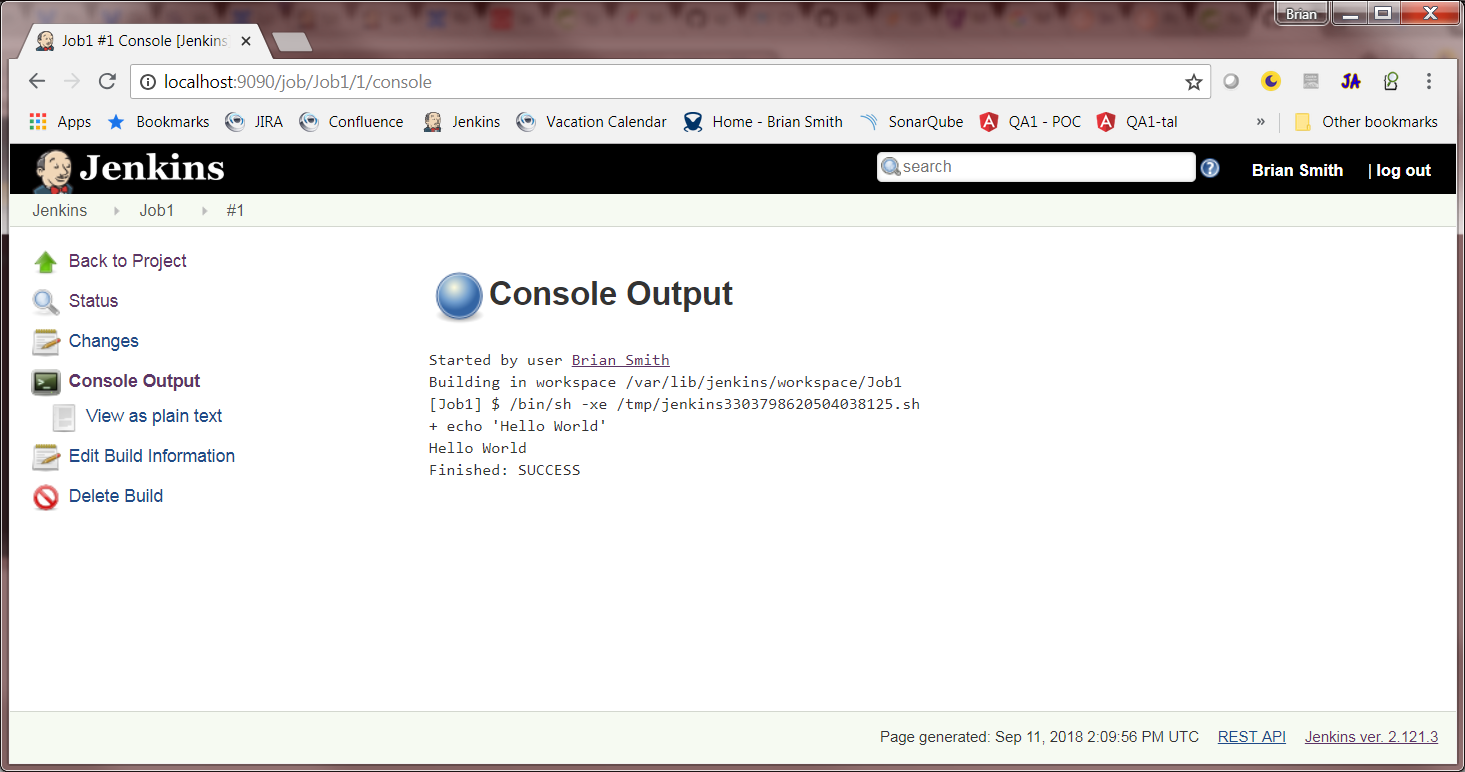
1. Open a command line and navigate to the Jenkins folder of the git repo.
2. Run vagrant up in your command line and wait for that to finish. It will create a new VM with Cent OS and install Jenkins on the VM. Jenkins will be accessible from <http://localhost:9090> in your browser.
3. You may see a notification initially that Jenkins is getting ready and then you will something as shown in the screen shot below where we need to unlock the Jenkins instance.  
   
4. In the command line run vagrant ssh -c "sudo cat /var/lib/jenkins/secrets/initialAdminPassword" This will output the key to the console  
   
5. Copy the key and put it in the Administrator Password field in the Jenkins window and press continue.   
   
6. Here click Install Suggested plugins and wait for all of the plugins to be installed once completed it will ask you to create your first user.   
   
7. Fill in your user information just make sure you are able to remember it for later and press Save and Continue.
8. The next screen will ask to setup the url. The default that it has in there should be fine so you can click save and finish.   
   
9. And then click Start Using Jenkins  
   
10. Click Manage Jenkins on the left links  
    
11. Click Configure Global Security  
    
12. Scroll down to the CSRF Protection section and uncheck “Prevent Cross Site Request Forgery exploits” then click save  
    

At this point your Jenkins instance is ready to setup and go. Next we will create a couple of simple jobs so that we can use to hook our micro service into.

## Create a Simple Job

1. On the left menu in Jenkins click the New Item link.   
   
2. Enter in a name for your job and click Freestyle project, and then click OK at the bottom of the screen.   
   
3. Click the build tab at the top of the page  
   
4. Click the Add Build Step drop down and select Execute Shell  
   
5. For the command enter echo "Hello World"  
   
6. Click Save at the bottom of the screen

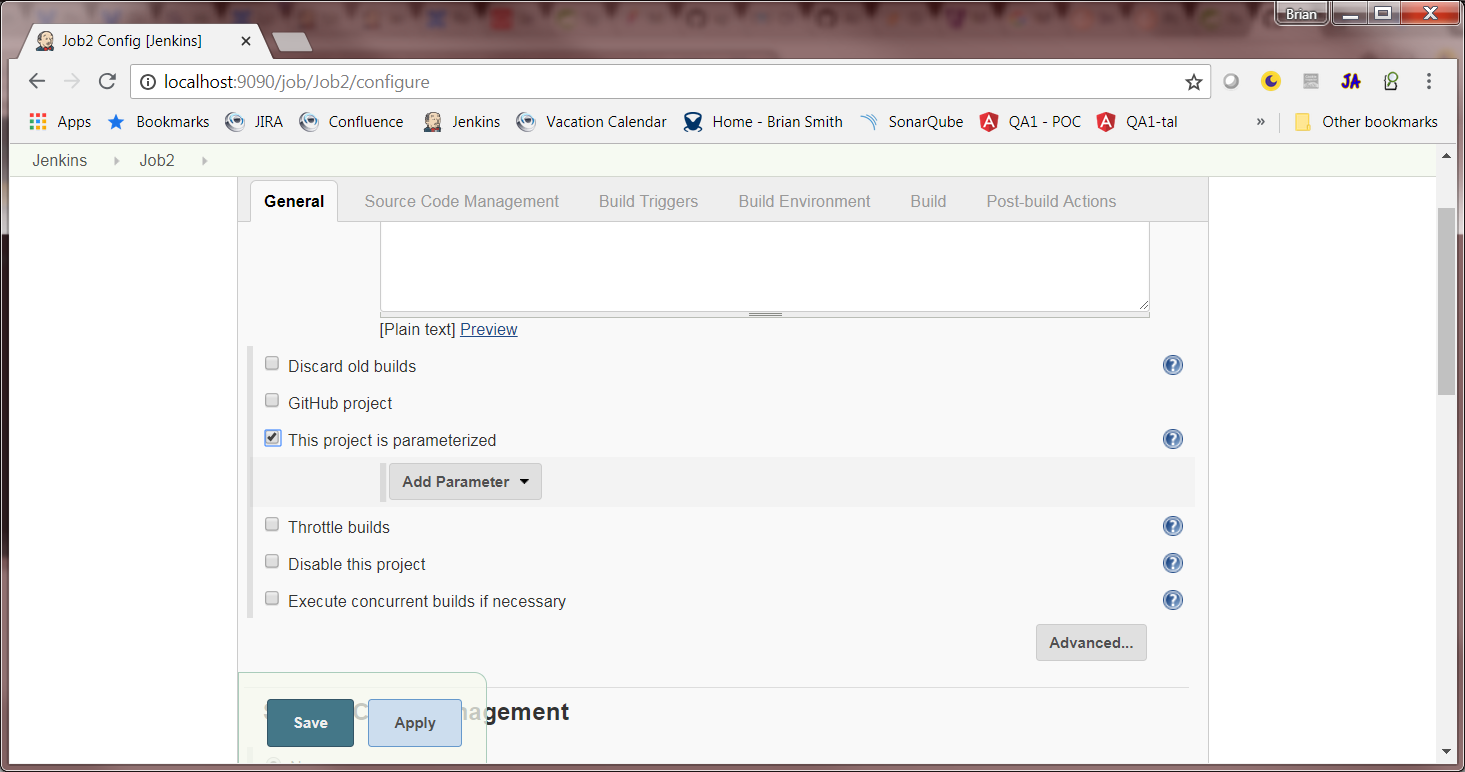
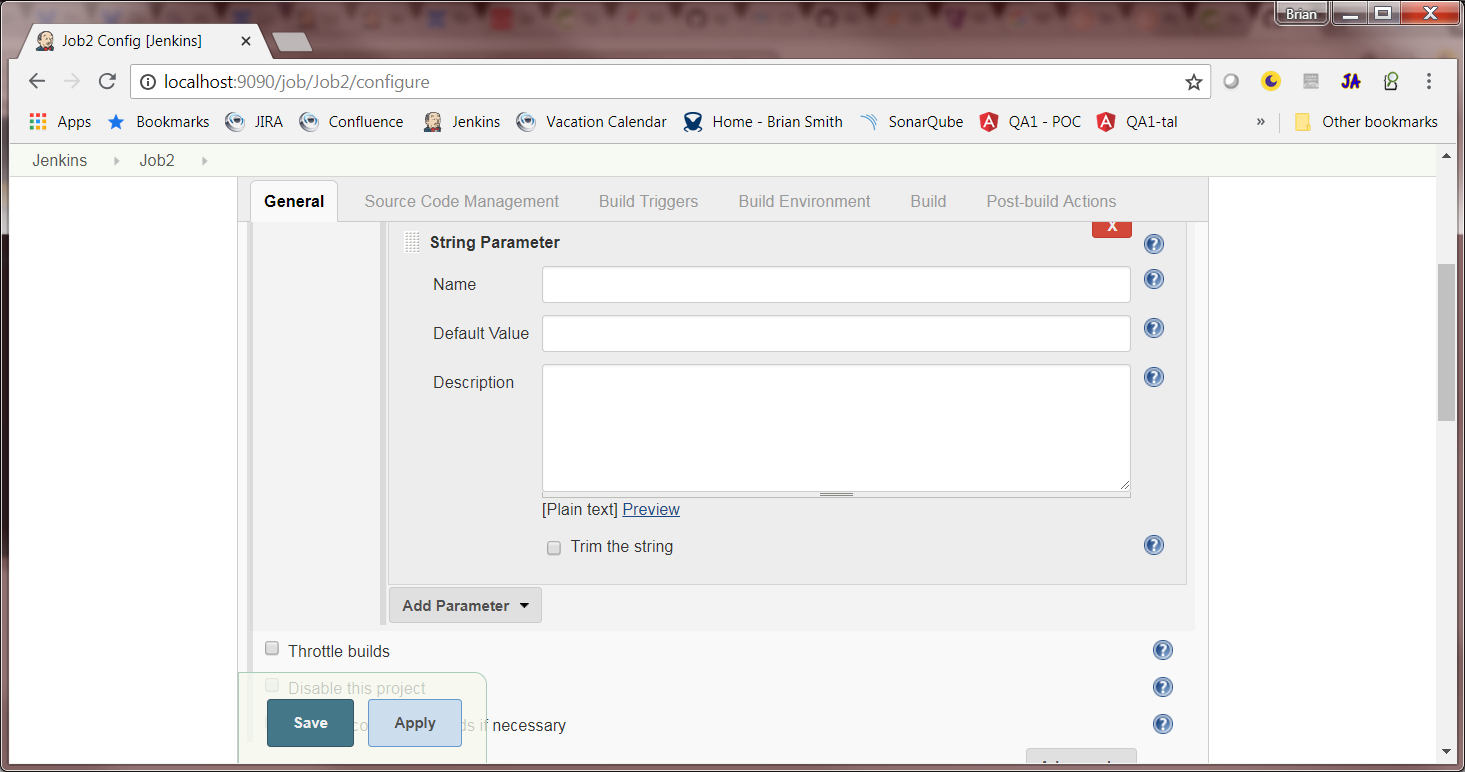
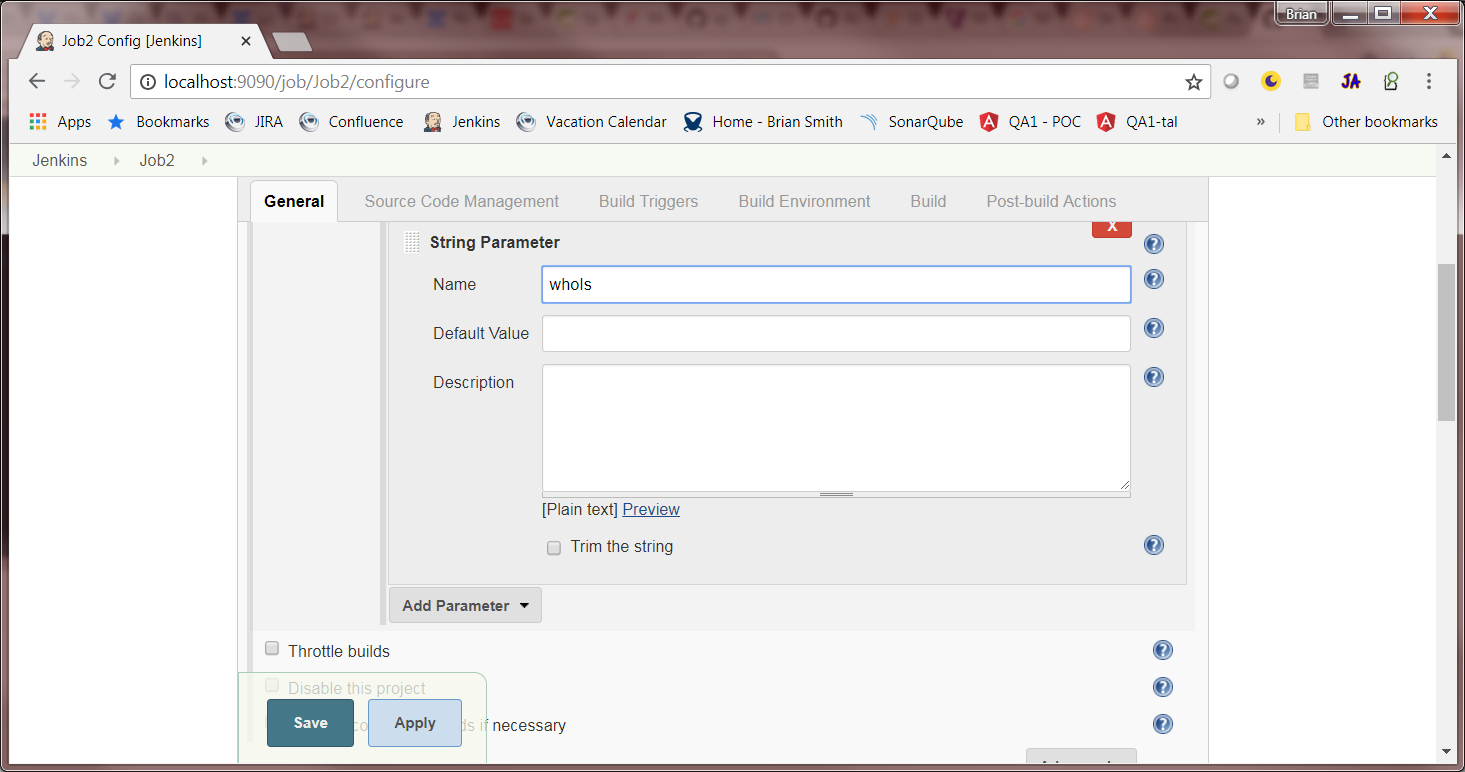
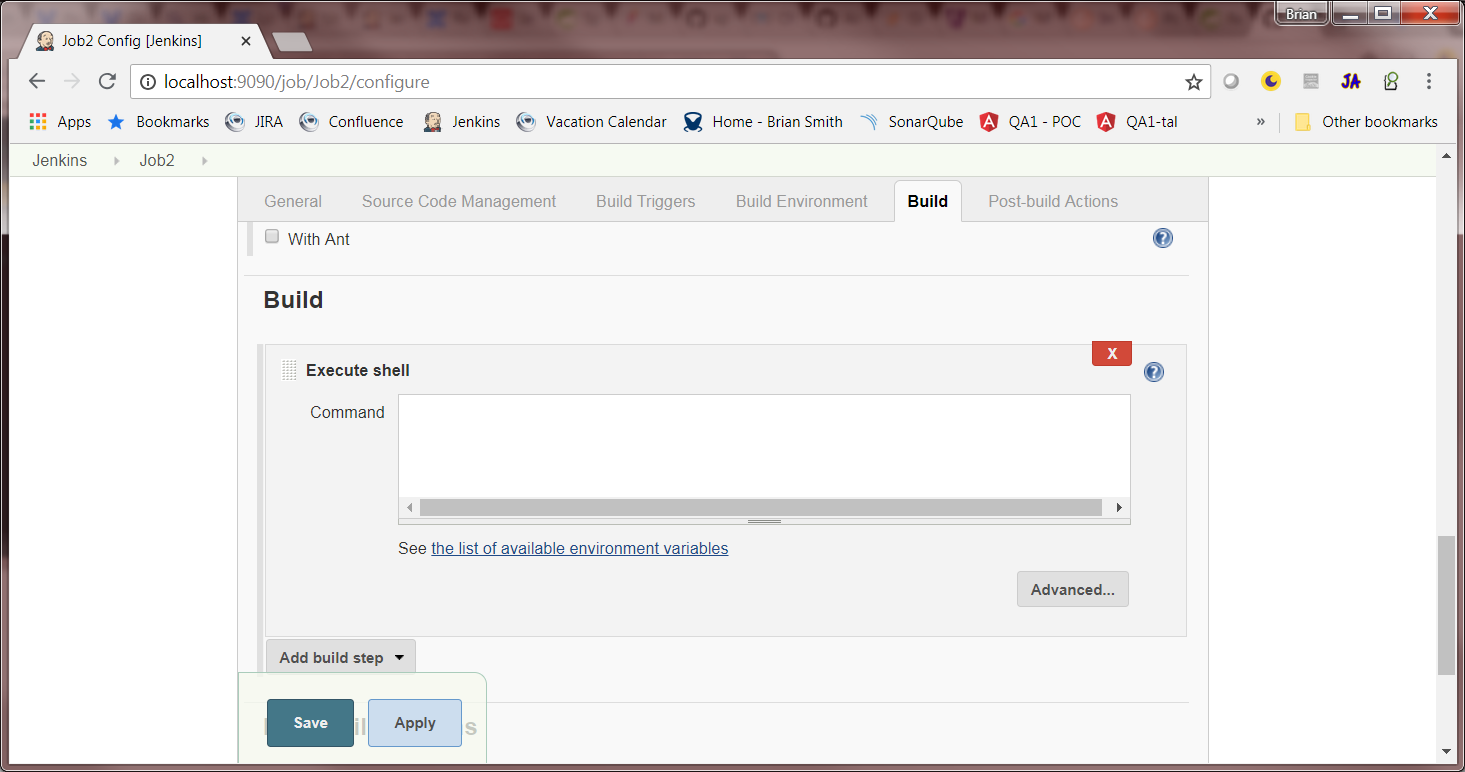
Now Let’s run the job and verify that the text Hello World gets printed to the console.

1. You should be looking at a page that looks like this  
   
2. On the left click the Build Now link. After a couple of seconds a build will show up in the build history section in the bottom left.   
   
3. Click that build in the lower left.   
   
4. Click Console Output  
   

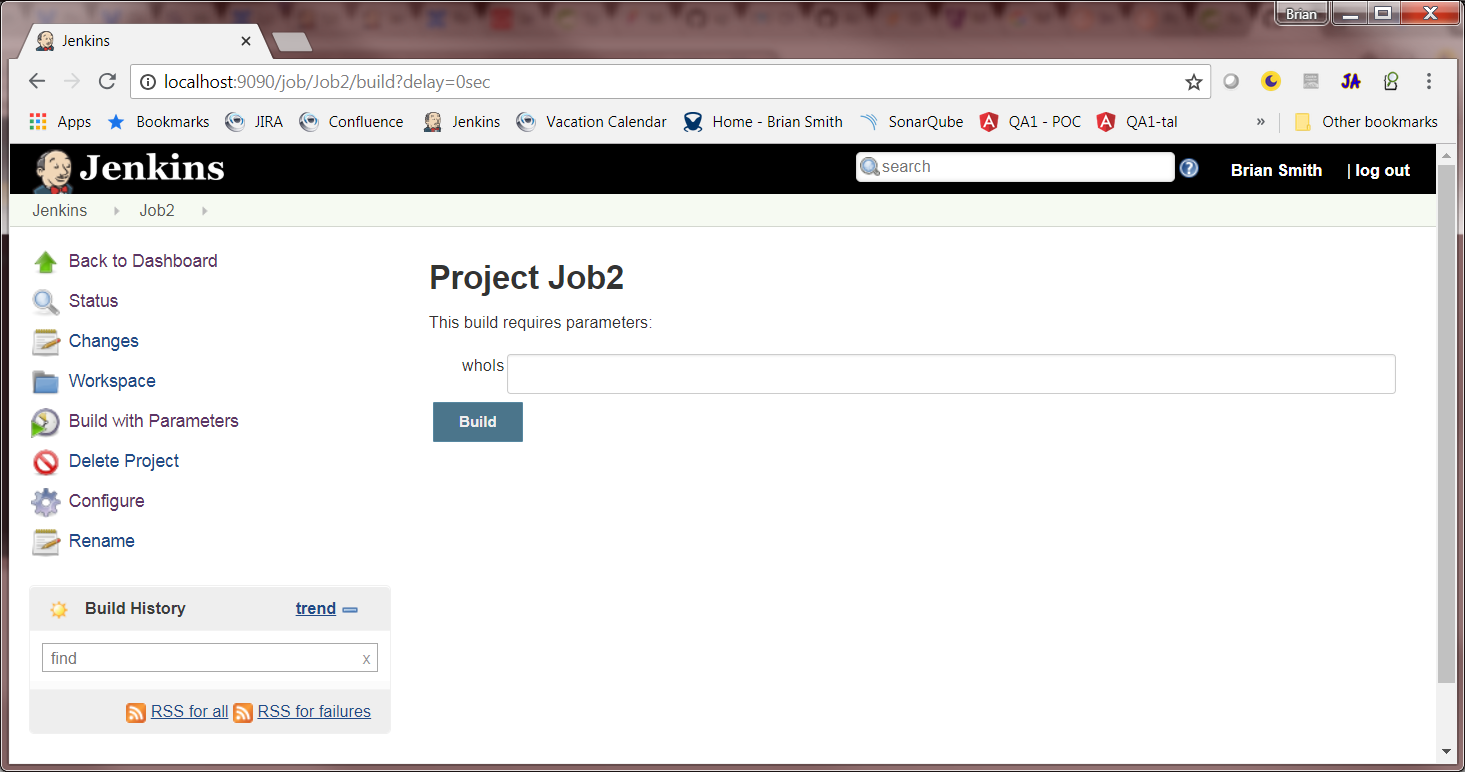
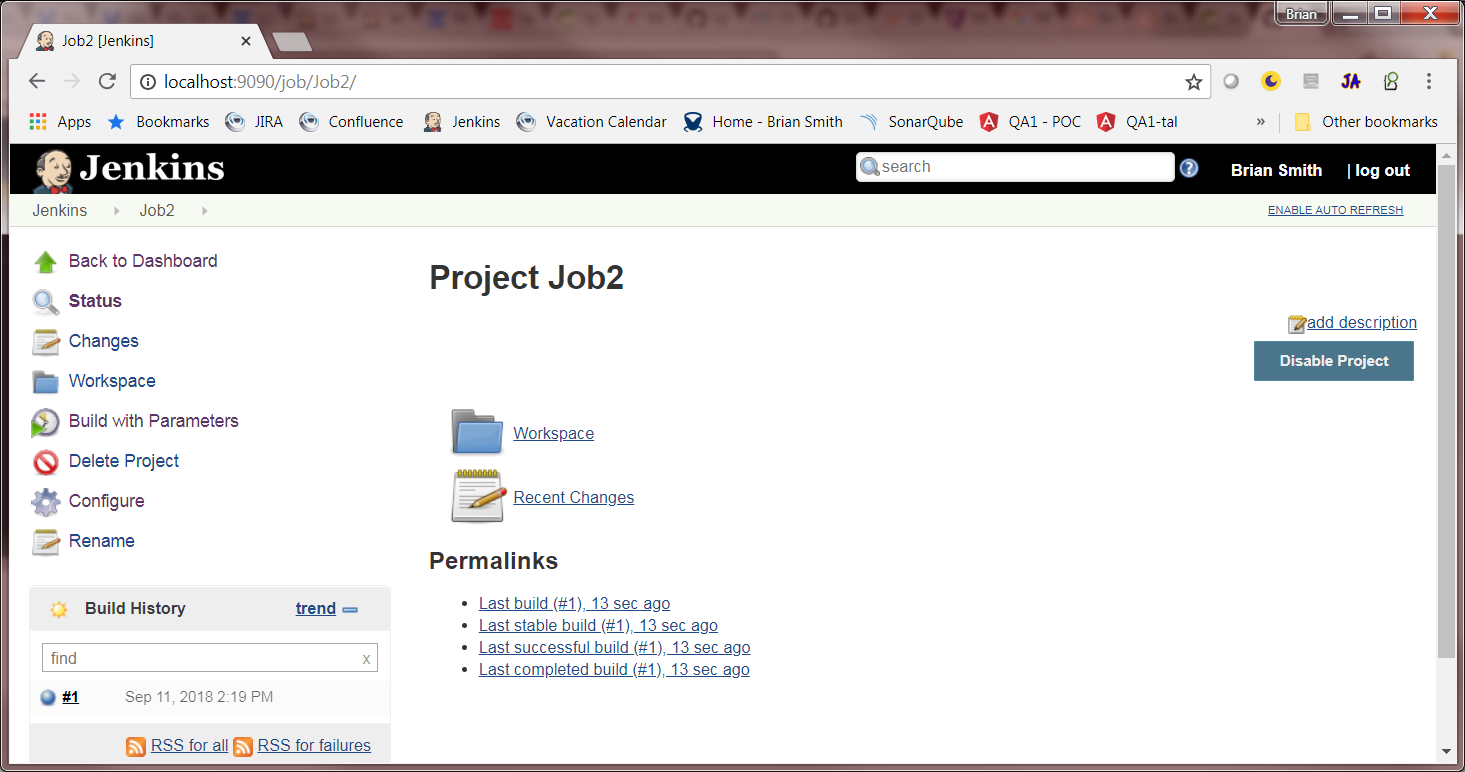
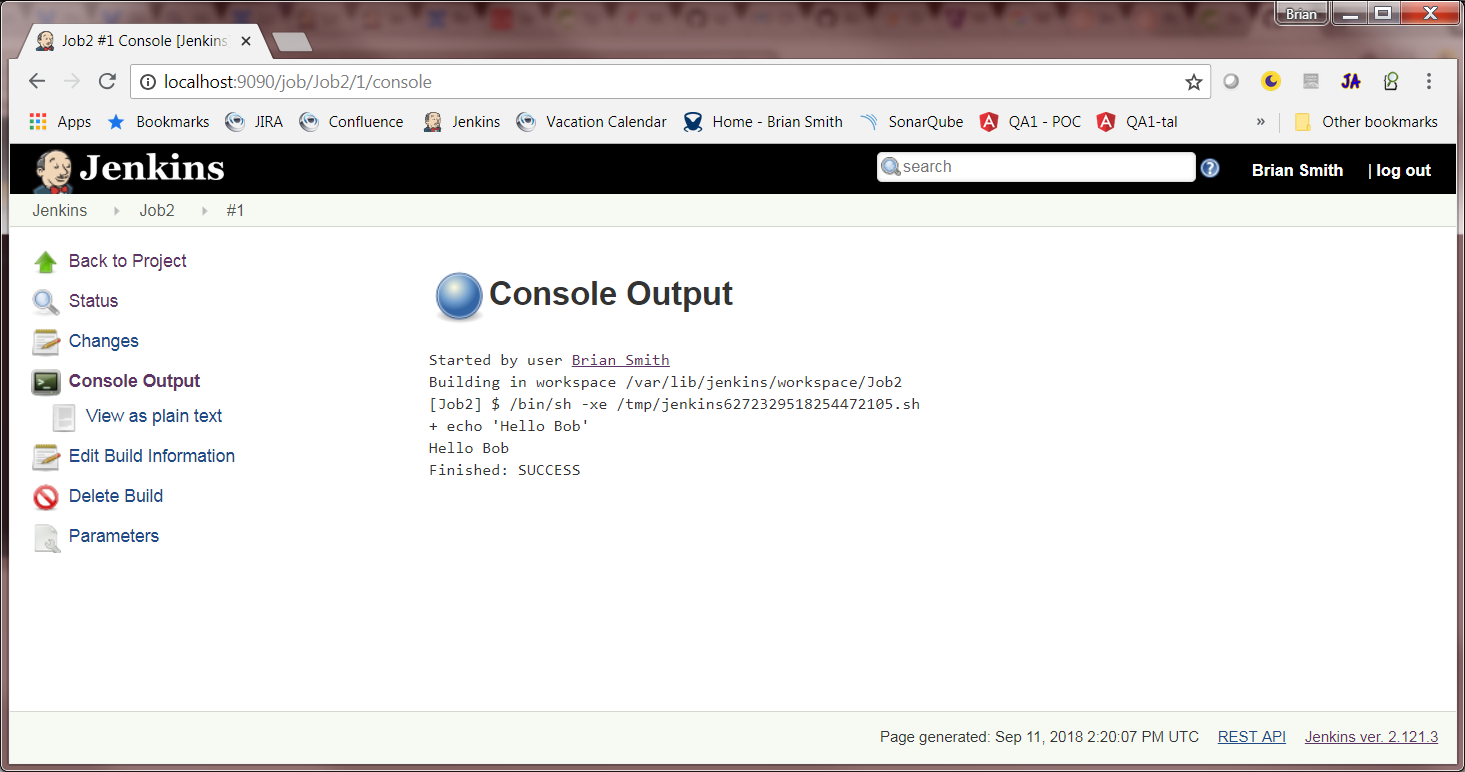
You will see in the console that it outputs our command and then the result of the command.

## Create A Simple Parameterized Job.

We can also create jobs that take in a parameter and use that in the build. Lets create one of those as well so we can invoke it and query it in our micro service.

1. Follow Steps 1 & 2 from creating the first job above.
2. In the general tab near the top there is a check box for “This Project is Parameterized” check that  
   
3. Click the Add Parameter button and in the drop down select String Parameter  
   
4. Give your parameter a name, the other items are optional leave them blank for now.   
   
5. Click the build tab at the top and add a n Execute Shell action as before.   
   
6. This time enter echo "Hello $parameter" parameter is the name you gave our parameter above.
7. Click Save

Running this job is going to be similar to before with a couple of differences.

1. This time our build link is “Build with Parameters” go ahead and click that  
   
2. Now we have to fill in our parameters before we build go ahead and put something in ad click the build button   
   
3. Once the build is done open the console output as before, but we can see Jenkins replace the $paramterName with the parameter value   
   

# Micro Service Requirements Part One.

1. The micro service should be able to kick of a non-parameterized build.
   1. The build should be configurable via property file.
   2. The service should include a URL parameter that indicates if the user wants a response or not
   3. If the user wants a response it should look like this
      1. { success: true}
2. The microservice should be able to kick off a parameterized build
   1. The build should be configurable via property file.
   2. The service should include a URL parameter that indicates if the user wants a response or not
   3. If the user wants a response it should look like this
      1. { success: true, value=”the value”}
   4. The parameter for the build should be passed in via JSON as a POST
3. The service should supply an endpoint that returns the build numbers for both of configured builds.
   1. The build numbers should be returned in this format
      1. { numbers: [1, 2, 3]}
4. The service should return the param for a given build
   1. The response should resemble
      1. { number:1, param: “value”}
5. Your Code should be properly unit tested and checked against sonar with any major or critical bugs addressed.

# Micro Service requirements part two.

Uh oh, here comes the client with some additional requirements we should update the existing application to handle them

1. The user should be able to pass in the job name so they can access any job on the server.

# Completed Code

A completed example of this project is available in the source/Jenkins folder of the get hub repo.