How to set up a local environment using a Docker container workflow that combines Jenkins, Sonarqube, and Gitlab for a CI development of projects, such as a task management application that uses RESTful API architecture.

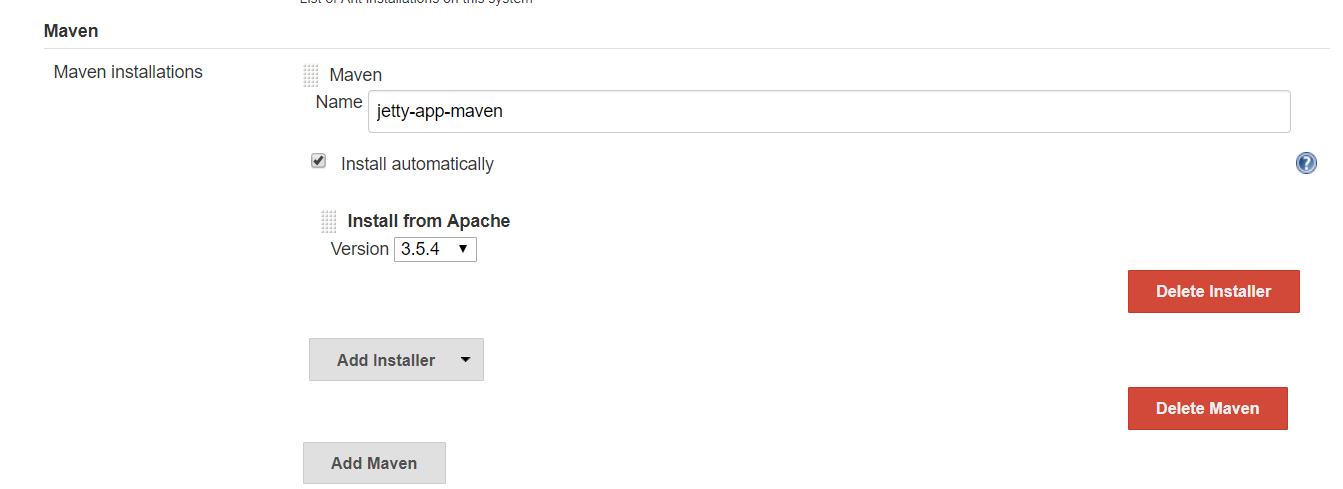
Setup

Project mirror: <https://github.com/SingleDreamer/jetty-app-copy>

(contains both jetty-app and the scripts to set up, should probably separate them? So the setup scripts can be one thing, and the jetty-app can be another)

Dependencies:

This assumes docker and git is installed. In this case, I am using Docker Toolbox on Windows 10. Therefore, the IP addresses of all instances created here will not be on localhost, but the IP address of the default machine docker is using (the IP address that shows up when the Docker Quickstart Terminal first starts).

1. Start the Docker Quickstart Terminal. This is where we will be working in.
2. Clone the project (<https://github.com/SingleDreamer/jetty-app-copy>) and run setup.sh. This will set up, in order, a postgres database for a Sonarqube instance, a Sonarqube instance, a Jenkins container, a Gitlab container, and a local Docker registry.
3. Go to your local Gitlab instance, in this case 192.168.99.100:30080.
   1. Create a new (admin?) user for Gitlab.
   2. Import the project.
4. In Jenkins, login using the initial Admin password.
   1. Install “Gitlab plugin” and restart.
   2. Create new credentials.
      1. Create an access token for Jenkins in Gitlab. (Gitlab -> Account Settings -> Access Tokens; select api), and add to Jenkins credentials
      2. Create a new credential with the Gitlab username and password.
   3. Go to Manage Jenkins -> Configure System -> Gitlab
      1. **Disable** ' Enable authentication for '/project' end-point ‘
      2. add gitlab host URL (in this case <http://192.168.99.100:30080>)
      3. Use the Gitlab API token for the credentials.
      4. If you click ‘Test Connection’, it should succeed.
   4. Go to Manage Jenkins -> Global Tool Config -> Maven
      1. 
   5. Create a new item (pipeline)
      1. In Build Triggers, select ‘Build when a change is pushed to GitLab.’
      2. In Pipeline, Select “Pipeline script from SCM”
         1. SCM: Git
         2. Repo URL: get this from HTTP in the Gitlab, but you may have to edit the address. In this case, it would be: <http://root@192.168.99.100:30080/root/jetty-app.git>. Usually it is something like (http://<user>@<IP>/<user>/<project\_name>.git)
         3. Use the username/password credentials for Gitlab. There should be no errors showing now.
         4. Save
5. In the Gitlab project:
   1. In the project Settings -> Integrations, add the webhook to Jenkins
   2. URL**: http://<jenkins IP address>/project/<project name> (http://192.168.99.100:8080/project/jetty-gitlab-test)**
   3. Select triggers, in this case push events
   4. **Disable** SSL verification
   5. If you select ‘Test’, you should get a “Hook Successfully executed” message
6. Clone the gitlab project onto your local computer. (git clone HTTP….)
7. When the project is pushed, it should trigger a build in Jenkins. It will read and use the Jenkinsfile in project, and build an image of the project based on the Dockerfile, and push it to the local repository.

STUFF TO AUTOMATE/DO:

* Need a (cool) name.
* Create new users for Gitlab and Jenkins automatically?
* Start Jenkins with plugins installed and necessary variables for the plugins (specifically, the maven and gitlab settings), and credentials (access key and user/pass settings), possibly?
* Start Gitlab with project imported, access keys, webhook, etc.
* Clean up the repo?
* Make scripts download file

curl http://192.168.99.100:30080/api/v4/session --data 'login=root&password=\*HnYnKkn4869'

curl --request POST --header "PRIVATE-TOKEN: TqffaQpkuMcxiixSfADs" --data "name=mytoken" --data "expires\_at=2018-08-07" --data "scopes[]=api" http://192.168.99.100:30080/api/v4/users/1/impersonation\_tokens

curl --request POST -H "Content-Type: application/json" --header "PRIVATE-TOKEN: TqffaQpkuMcxiixSfADs" --data '{“user\_id”:”1”, “name”:”test”, “import\_url”:” https://github.com/SingleDreamer/jetty-app-copy.git”} ' <http://192.168.99.100:30080/api/v4/projects>

curl -H "PRIVATE-TOKEN: TqffaQpkuMcxiixSfADs" -H "Content-Type:application/json" --data '{"id":"4","url":"http://192.168.99.100:8080/project/jetty-gitlab-test","enable\_ssl\_verification":false, "push\_events":true}' http://192.168.99.100:30080/api/v4/projects/4/hooks

user\_password

user\_password\_confirmation

powershell (RUN AS ADMIN)

Set-ExecutionPolicy Bypass -Scope Process -Force; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

https://duo.com/decipher/driving-headless-chrome-with-python

python

jq

(pizza) LOCAL DELIVERY – training engineers for rapid software dev

Why containers important/money/time/headaches

Don’t need to install all these tools ourselves, dependencies (version), isolated but can work together

Why dev ops

Why cont deliv

Test driven development

Why more than just writing code

Making sure using the right tools, working efficiently, making sure the code you write is consistent and works with other ppl

Training ppl to work with other ppl

New feature works, only to find it breaks something else

Version deployed

Install docker Cygwin