**Solum, Jenkins & gitlab Integration**

1) Bring up solum devstack.

**Jenkins Installation**

1. Using the following document to install jenkins on ubuntu

<https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu>

2) Install openjdk-jre v7

sudo apt-get install openjdk-7-jre

3)

**Installation**

wget -q -O - https://jenkins-ci.org/debian/jenkins-ci.org.key | sudo apt-key add -

sudo sh -c 'echo deb http://pkg.jenkins-ci.org/debian binary/ > /etc/apt/sources.list.d/jenkins.list'

sudo apt-get update

sudo apt-get install jenkins

this will fail to start jenkins because of port conflicts

4) Edit /etc/default/jenkins and modify the

HTTP\_PORT=8888

5) sudo service jenkins start

6) access the jenkins console using http://192.168.76.2:8888

**Installing Gitlab**

Follow the document(<http://doc.gitlab.com/omnibus/docker/>)

1) docker pull gitlab/gitlab-ce

2) Start Git lab using the following commands

sudo docker run --detach --hostname gitlab.example.com --publish 444:443 --publish 800:80 --publish 222:22 --name gitlab --restart always --volume /srv/gitlab/config:/etc/gitlab --volume /srv/gitlab/logs:/var/log/gitlab --volume /srv/gitlab/data:/var/opt/gitlab gitlab/gitlab-ce:latest

3) Launch the gitlab web ui with the url → http://<ip>:800/

<http://doc.gitlab.com/omnibus/docker/#after-starting-a-container>

4) Once done follow the following 2 links to establish a trust between jenkins and gitlab

<http://doc.gitlab.com/ee/integration/jenkins.html#jenkins-ci-integration>

<https://github.com/jenkinsci/gitlab-plugin#using-it-with-a-job>

5) Host the following project in gitlab

<https://github.com/ashishjain14/solum-python-sample-app>

6) In project settings add a hook to our glue code which can be done by setting the following url http://localhost:7000/v1/japi

**Pre-requisites to integrate jenkins and solum**

1) Jenkins is setup with all the plugins

- Envinject

- logstash

- docker-build-step

- git

- docker-build-publish

- docker-common

2) OpenStack is setup with all the components

3) Gitlab is set up with the project which has to be built

4) A custom component be built which will listen to a post request from gitlab and thereafter which will create all jenkins jobs. This code will basically create all the jobs in jenkins and fire the first job

Following workflow diagram depicts the complete flow

Git Lab(2)

Config file for custom code

User Checks in the Code change (1)

OpenStack(5)

Jenkins(4)

Custom Code(3)

**Fig 1. Workflow for the Pipeline**

Here is the explanation of the complete set of events

1. User checks in the code change
2. Git lab fires an event as soon as a change is received
3. The custom code receives the code change and populates Jenkins job configuration files(xml) and fires a post request to Jenkins. This will create a complete job pipeline in Jenkins. The custom code on successfully creating the jobs in Jenkins fires the first job.
4. Jenkins will start executing the jobs, with first job being downloading the code from gitlab for the said project. It will execute all the tests and finally upload an image to glance
5. OpenStack executes the deployment of the app on solum

**Few more points to note here**

1. The code repo in gitlab should also host a Dockerfile which has details on building an image which could be used for running the test as well as deployment
2. It will be one docker image to which during runtime commands will be passed to run the test case and finally the deploy command during deployment
3. This custom code will act as a receiver of git web hook call. This code will have the following functionalities
   1. Act as a wsgi server to receive a POST api call from git lab
   2. Parse the parameters in the git lab call to get user details, any token details and any other details
   3. This code will read the config file, this config file will have parameters which will required for all the different jobs for example credentials for OpenStack, api endpoints for openstack etc.
   4. This code will than set all these variables into Jenkins job XML files
   5. This code will than fire a post request to Jenkins to upload the XML file and create a complete pipeline
   6. Later this code will fire the first job in the pipeline of jobs
4. Base Image or language pack will be pre-created and will be kept in docker registry or glance repository(we need to discuss this up)

**Setting up Docker Registry**

1) Download the registry image from docker hub

sudo docker pull registry

2) Run the registry using the following command

docker run -d -p 5001:5000 --restart=always --name registry -v `pwd`/data:/var/lib/registry registry:2

3) Set the following docker\_opts in /etc/default/docker

DOCKER\_OPTS='-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock --insecure-registry localhost:5001'

restart the docker daemon using the command “sudo service docker restart”

**Configuring docker api endpoint in jenkins**

1) <http://192.168.76.2:8888/configure>

2) In the Docker Builder section add Docker URL as “[http://127.0.0.1:2375](http://127.0.0.1:2375/)”

3) Configure OS for jenkins user to accept sudo w/o password

http://stackoverflow.com/questions/17414533/best-practices-to-avoid-jenkins-error-sudo-no-tty-present-and-no-askpass-progr

%jenkins ALL=NOPASSWD: ALL

**Configure logstash plugin**

**Setting up solum\_jenkins glue code**

1) ssh into devstack

2) Clone the repo https://github.com/ashishjain14/jenkins\_solum

3) pip install -r requirements.txt

4) Configure the following parameters etc/config/js.conf

job\_template\_dir

docker\_daemon\_uri

5) In file jenkins/git/handler handler.py. **Please note this step will be changed**

modify config\_file = "/home/vagrant/jenkins\_glue/etc/config/js.conf"

6) python setup.py install

7) To start run "japi --config-file etc/config/js.conf

**Testing the code**

1. Make a change to the code hosted in gitlab and checkout the successful execution of complete pipeline