**Python First Application:**

http://www.vogella.com/tutorials/Python/article.html

https://docs.python.org/2/tutorial/

https://www.python-course.eu/python3\_course.php

http://engineering.hackerearth.com/2014/08/21/python-requests-module/

**Python links:**

<https://www.techbeamers.com/python-interview-questions-experienced/>

<https://filippo.io/instance-monkey-patching-in-python/>

<http://net-informations.com/python/iq/memory.htm>

<https://www.edureka.co/blog/interview-questions/python-interview-questions/>

<https://pymotw.com/2/threading/>

<http://net-informations.com/python/iq/multi.htm>

<https://www.quantstart.com/articles/Parallelising-Python-with-Threading-and-Multiprocessing>

<https://hackernoon.com/decorators-in-python-8fd0dce93c08>

<https://medium.com/@happymishra66/lambda-map-and-filter-in-python-4935f248593>

<https://www.ploggingdev.com/2017/01/multiprocessing-and-multithreading-in-python-3/>

<https://www.slideshare.net/dabeaz/an-introduction-to-python-concurrency>

<https://www.python-academy.com/courses/specialtopics/python_course_advanced.html>

<http://interactivepython.org/runestone/static/CS152f17/Lists/ConcatenationandRepetition.html>

**Multithreading Multiporcessing**

The threading module uses threads, the multiprocessing module uses processes. The difference is that threads run in the same memory space, while processes have separate memory. This makes it a bit harder to share objects between processes with multiprocessing. Since threads use the same memory, precautions have to be taken or two threads will write to the same memory at the same time. This is what the global interpreter lock is for.

Spawning processes is a bit slower than spawning threads. Once they are running, there is not much difference.

<https://www.ibm.com/developerworks/library/ba-on-demand-data-python-1/index.html>

<https://www.ibm.com/developerworks/library/os-pythondescriptors/index.html>

<https://www.ibm.com/developerworks/linux/library/l-cpdecor/index.html>

Logging Module

<https://www.blog.pythonlibrary.org/2012/08/02/python-101-an-intro-to-logging/>

IMAPlib

<http://www.nevprobusinesssolutions.com/how-to-get-emails-using-python-and-imaplib/>

<https://gist.github.com/robulouski/7441883>

MAP,FILTER,REDUCE,LAMBDA

<http://www.bogotobogo.com/python/python_fncs_map_filter_reduce.php>

Garbase Collection:

<https://www.digi.com/resources/documentation/digidocs/90001537/references/r_python_garbage_coll.htm>

REGEX

<https://www.thegeekstuff.com/2014/07/python-regex-examples>

<http://www.pyregex.com/>

<https://www.geeksforgeeks.org/class-method-vs-static-method-python/>

static variable and static/class method difference

<http://radek.io/2011/07/21/static-variables-and-methods-in-python/>

Inheritance:

<https://www.journaldev.com/14623/python-multiple-inheritance>

Super:

<http://www.pythonforbeginners.com/super/working-python-super-function>

Overloading :

It is not possible in python but achieved using default parametr

Python +Excel:

<http://openpyxl.readthedocs.io/en/default/usage.html>

Interview questions:

<https://intellipaat.com/interview-question/python-interview-questions/>

Pickle:

<https://pythontips.com/2013/08/02/what-is-pickle-in-python/>

**OAuth:**

https://requests-oauthlib.readthedocs.io/en/latest/examples/examples.html

**Networking Links:**

https://www.youtube.com/watch?v=nomyRJehhnM

https://www.youtube.com/watch?v=rYodcvhh7b8

<https://www.practicalnetworking.net/series/>

<https://techdifferences.com/difference-between-arp-and-rarp.html>

<https://www.practicalnetworking.net/series/nat/why-nat/>

<http://www.geeksforgeeks.org/whats-difference-http-https/>

<https://www.practicalnetworking.net/series/packet-traveling/key-players/#router>

<http://www.omnisecu.com/tcpip/dhcp-dynamic-host-configuration-protocol-how-dhcp-works.php>

<http://www.learncodeonline.in/blog/what-is-ssl-and-how-to-install-ssl/>

<https://2buntu.com/articles/1204/ping-how-does-it-work/>

<https://www.verisign.com/en_IN/website-presence/online/how-dns-works/index.xhtml>

**Rest API Video:**

https://www.youtube.com/watch?v=NXcbofCuKTA&list=PLFDB9C9B4670B91D1

**Python,Flask,Rest:**

https://blog.miguelgrinberg.com/post/designing-a-restful-api-with-python-and-flask

**Python and MongoDB Connection along with some mongo basics command:**

http://www.informit.com/articles/article.aspx?p=2246943

https://www.bogotobogo.com/python/MongoDB\_PyMongo/python\_MongoDB\_pyMongo\_tutorial\_installing.php

**MongoCommand Tutorial:**

https://code.tutsplus.com/tutorials/getting-started-with-mongodb-part-1--net-22879

https://code.tutsplus.com/articles/mapping-relational-databases-and-sql-to-mongodb--net-35650

**Difference Between web socket and rest:**

https://www.pubnub.com/blog/2015-01-05-websockets-vs-rest-api-understanding-the-difference/

**Devops:**

<https://www.appdynamics.com/media/uploaded-files/White_Paper_-_An_Intro_to_DevOps.pdf>

<https://www.youtube.com/watch?v=h5E--QSBVBY>

**INstallation of MongoDB,java,tomcat and rabbitmq:**

<http://tecadmin.net/install-mongodb-on-centos-rhel-and-fedora/>

Process to cpy db of other :

copy new mongo zip to var/lib

change its permission using chown -R mongod:mongod mongo

if copy mongo at other location then change path in file /etc/mongod.conf

Installation of Java

<http://tecadmin.net/steps-to-install-java-on-centos-5-6-or-rhel-5-6/>

---------------------------------------------------------------------------------------------------------------

Tomcat installation

<https://oracle-base.com/articles/linux/apache-tomcat-7-installation-on-linux#downloads>

MySQL Installation:

https://www.linode.com/docs/databases/mysql/how-to-install-mysql-on-centos-6/

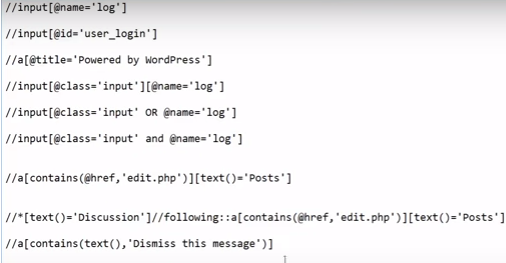
Java,Selenium,Eclipse Setup:

http://www.software-testing-tutorials-automation.com/2013/08/how-to-download-and-install-selenium.html

**CSS vs xPath**

<http://www.way2automation.com/selenium-webdriver-xpath-vs-css.php>

**Xpath Example:**



**VNC Installation:**

https://rbgeek.wordpress.com/2012/06/26/how-to-install-vnc-server-on-centos-6/

yum groupinstall Desktop

yum install tigervnc-server -y

yum install xorg-x11-fonts-Type1 -y

yum install vnc -y

chkconfig vncserver on

vncpasswd

vi /etc/sysconfig/vncservers

VNCSERVERS="1:root"

VNCSERVERARGS[1]="-geometry 1024x600"

service vncserver restart

vi .vnc/xstartup

      #twm &

      exec gnome-session &

service iptables stop

xhost +

**Protractor:**

<http://www.protractortest.org/>

<https://docs.google.com/presentation/d/1QWFnYAur19R7RQ5KkLkLDMOMz5jrzNlBId3XBrwRNs8/edit#slide=id.gf9a5a479_019>

<https://www.3pillarglobal.com/insights/using-the-protractor-automation-tool-to-test-angularjs-applications>

<http://www.guru99.com/protractor-testing.html>

[*https://github.com/angular/protractor/blob/master/docs/locators.md*](https://github.com/angular/protractor/blob/master/docs/locators.md)

[*https://github.com/andresdominguez/elementor*](https://github.com/andresdominguez/elementor)

**SNMP Help Document:**

Simple Network Management Protocol (**SNMP**) is a popular protocol for network management. It is used for collecting information from, and configuring, network devices, such as servers, printers, hubs, switches, and routers on an Internet Protocol (IP)

We need  to have  **net-snmp** rpm package installed on the servers , generally it would come with repository.

<https://www.liquidweb.com/kb/how-to-install-and-configure-snmp-on-centos/>

<https://www.fineconnection.com/how_to_install_and_enable_snmpv3_on_a_linux_system_for_authentication_en_encryption_testing-2/>

<https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/System_Administrators_Guide/sect-System_Monitoring_Tools-Net-SNMP.html>

Elastic Search:

Elasticsearch uses a structure called an inverted index, which is designed to allow very fast full-text searches. An inverted index consists of a list of all the unique words that appear in any document, and for each word, a list of the documents in which it appears.

<https://buildingvts.com/elasticsearch-architectural-overview-a35d3910e515>

<https://stackoverflow.com/questions/7727686/whats-the-difference-between-an-inverted-index-and-a-plain-old-index>

Kafka :

<http://javainuse.com/misc/apache-kafka-hello-world>

<https://www.tutorialspoint.com/apache_kafka/>

Openstack Links:

<https://docs.openstack.org/mitaka/install-guide-ubuntu/launch-instance.html>

<https://github.com/bmangesh/openstack_ubuntu/blob/master/openstack_Glance.sh>

**Adding-a-new-ssh-key-to-your-github-account or bitbucket**

<https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/>

<https://confluence.atlassian.com/bitbucket/set-up-an-ssh-key-728138079.html>

<http://jr0cket.co.uk/2016/05/ssh-or-https-that-is-the-github-question.html>

Normal setup:

1.Create Any Dir and in that create .ssh dir and cd

2.ssh-keygen -t rsa -b 4096

3.Ensure the ssh-agent is running:

eval $(ssh-agent -s)

4.Add private Key to agent

ssh-add createdDir/.ssh/id\_rsa

Adding a new SSH key to your account BitBucket  :

Add key id\_rsa.pub key in bitbucketrepo->setting->AccessKey

and change url to ssh in account

git config --global [user.name](http://user.name/) "YOUR NAME"

git config --global user.email "YOUR EMAIL ADDRESS"

Also change remote url for local repo to ssh using

git remote set-url origin sshpathURL.

git config --global [user.name](http://user.name/) "YOUR NAME"

git config --global user.email "YOUR EMAIL ADDRESS"

try git pull

**Selenium:**

<https://www.softwaretestingmaterial.com/selenium-webdriver-architecture/>

<http://www.aosabook.org/en/selenium.html>

<https://github.com/seleniumhq/selenium-google-code-issue-archive>

<https://www.linkedin.com/pulse/how-does-selenium-webdriver-work-alex-siminiuc/>

<https://www.quora.com/How-does-the-Selenium-WebDriver-work>

<https://saucelabs.com/resources/articles/selenium-tips-css-selectors>

**Java:**

<https://www.geeksforgeeks.org/vector-vs-arraylist-java/>

<https://www.programcreek.com/2013/03/hashset-vs-treeset-vs-linkedhashset/>

**OAuth linkdin:**

<https://gist.github.com/lebedov/8c3f33ebb55a67b732c1>

<https://github.com/linkedin/api-get-started/blob/master/python/tutorial.py>

**sample code**

#!/usr/bin/env python

API\_KEY = "86186n6h8xhk7k"

SECRET\_KEY = "MlW8tQLYdaWxGt5T"

NAME = "0awarerahul0@gmail.com"

PASSWORD = "Bajarang@123"

import re

import ConfigParser as cp

import oauth2

import urlparse

import lxml.html

import mechanize

br = mechanize.Browser()

br.set\_cookiejar(mechanize.CookieJar())

br.set\_handle\_redirect(True)

br.set\_handle\_robots(False)

# Get request token:

consumer = oauth2.Consumer(API\_KEY, SECRET\_KEY)

client = oauth2.Client(consumer)

request\_token\_url = 'https://api.linkedin.com/uas/oauth/requestToken'

response, content = client.request(request\_token\_url, 'POST')

if response['status'] != '200':

raise Exception('Invalid response')

request\_token = dict(urlparse.parse\_qsl(content))

print 'request token: ', request\_token['oauth\_token']

print 'request token secret: ', request\_token['oauth\_token\_secret']

# Use token to redirect to user login:

authorize\_url = 'https://api.linkedin.com/uas/oauth/authorize'

redirect\_url = '%s?oauth\_token=%s' % (authorize\_url, request\_token['oauth\_token'])

# Login with mechanize:

br.open(redirect\_url)

br.select\_form(nr=0)

br.form['session\_key'] = NAME

br.form['session\_password'] = PASSWORD

br.submit()

html = br.response().read()

tree = lxml.html.fromstring(html)

oauth\_verifier = tree.xpath('.//div[@class="access-code"]')[0].text\_content()

# Use PIN to obtain access token:

token = oauth2.Token(request\_token['oauth\_token'],

request\_token['oauth\_token\_secret'])

token.set\_verifier(oauth\_verifier)

access\_token\_url = 'https://api.linkedin.com/uas/oauth/accessToken'

client = oauth2.Client(consumer, token)

response, content = client.request(access\_token\_url, 'POST')

access\_token = dict(urlparse.parse\_qsl(content))

print 'access token: ', access\_token['oauth\_token']

print 'access token secret: ', access\_token['oauth\_token\_secret']

token = oauth2.Token(access\_token['oauth\_token'],

access\_token['oauth\_token\_secret'])

client = oauth2.Client(consumer, token)

profileUrl="http://api.linkedin.com/v1/people/~?format=json"

resp,content =client.request(profileUrl,'GET', headers={})

print resp.status,resp,content

**Vagrant:**

Installation of vagrant on windows :

<https://www.sitepoint.com/getting-started-vagrant-windows/>

 (Install virtual box,vagrant ,create dir ,navigate to dir,vagant init,vagrant box add os\_name,vaagrant up,vagrant ssh\_config)

<https://www.vagrantup.com/intro/getting-started/index.html>

<https://www.taniarascia.com/what-are-vagrant-and-virtualbox-and-how-do-i-use-them/>

**BitBucket API:**

<https://confluence.atlassian.com/bitbucket/use-the-bitbucket-cloud-rest-apis-222724129.html>

<https://developer.atlassian.com/bitbucket/server/docs/latest/how-tos/command-line-rest.html>

**Docker:**

<https://www.javatpoint.com/docker-useful-commands>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>

Docker installation on windows lower than 10.1 versions:

**it** is basically a container engine which uses the Linux Kernel features like namespaces and control groups to create containers on top of an operating system and automates application deployment on the container. ... **Docker** is incredibly fast and it can run on the host with compatible Linux Kernel.

<https://www.docker.com/products/docker-toolbox#/tutorials>

<https://devopscube.com/what-is-docker/>

**Networking :**

<https://docs.docker.com/engine/tutorials/networkingcontainers/#add-containers-to-a-network>

installing Ping:

apt-get install iputils-ping

To installed Net packages :apt-get install net-tools

Installed ssh server

sudo apt-get install openssh-server

passwd

We can install Git without having to add any repositories.

apt-get install git Check :git --version

<https://docs.docker.com/engine/examples/running_ssh_service/>

<https://www.digitalocean.com/community/tutorials/how-to-install-and-manage-supervisor-on-ubuntu-and-debian-vps>

**AngularJs:**

**https://www.youtube.com/watch?v=cUXBsZ3BwBk Angularjs**

**NodeJS:**

<http://blog.thedigitalgroup.com/ujwalap/2015/05/15/introduction-to-how-node-js-works/>

<http://www.journaldev.com/7462/node-js-architecture-single-threaded-event-loop>

<https://dzone.com/articles/quick-introduction-how-nodejs>

<https://www.codementor.io/olatundegaruba/nodejs-restful-apis-in-10-minutes-q0sgsfhbd>

**Gulp:**

<https://julienrenaux.fr/2014/05/25/introduction-to-gulp-js-with-practical-examples/>

<https://hackernoon.com/how-to-automate-all-the-things-with-gulp-b21a3fc96885>

**Git:**

<http://ohshitgit.com/>

<https://stevebennett.me/2012/02/24/10-things-i-hate-about-git/>

<http://blackbe.lt/common-git-problems-and-solutions/>

**Window Application Automation using Selenium+Winium**

<https://www.youtube.com/watch?v=juRKu9cBwQ0>

**JMETER:**

<https://www.testingexcellence.com/jmeter-tutorial-testing-rest-web-services/>

<https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server>

<https://www.testingexcellence.com/http-basics/>

<https://www.blazemeter.com/blog/api-testing-with-jmeter-and-the-json-extractor>

<https://dzone.com/articles/advanced-usage-of-the-json-path-extractor-in-jmete>

**API Testing :**

<http://www.techbeamers.com/rest-api-interview-questions-answers/>

<https://www.tutorialspoint.com/http/http_status_codes.htm>

REGEX PYTHON:

<http://www.thegeekstuff.com/2014/07/python-regex-examples>

<https://www.fullstackpython.com/docker.html>

MYSQL :

<https://www.tutorialspoint.com/python/python_database_access.htm>

**Linux Command:**

****

**Linux:**

Alias in Linux :

echo "alias aliasname='command'" >> ~/.bash\_aliases`  
puts the alias in your bash\_aliases file. Then the  
`source ~/.bash\_aliases`  
command loads the bash\_aliases into the current shell.

SSH to remote server and running command:

<https://www.shellhacks.com/ssh-execute-remote-command-script-linux/>

SCP  without promting password in shell script:

yum install sshpass

sshpass -p "$password" scp /<PATH>/final.txt $username@$Ip:/root/<PATH>

**SonarQube:**

**Install and configure SonarQube for Python code**

In this blog post I will show you how to install and configure SonarQube in order to manage the code quality of your Python project.

**Installing SonarQube**

1. Go to the [SonarQube official website](http://www.sonarqube.org/downloads/" \t "_blank) and download the latest version. At the time of writing this blog post the latest version is 6.0.
2. The downloaded file is in a ZIP format so you have to unzip it in a folder of your choice. I am using Ubuntu so I extracted the zip in */opt/sonarqube*.
3. Now open a terminal/command prompt and go to *$SONAR\_INSTALL\_PATH/bin/$OS*where $OS is your operating system. For me it is *linux-x86-64.*
4. Depending on you operating system you can start the SonarQube server as follows:
   1. For Windows - type *StartSonar.bat* and hit Enter.
   2. For Linux/MacOS - type *./sonar.sh console* and hit Enter.  
      If you have some issues related access denied then run the command as sudo.  
      If you have some issues related to JVM such as "wrapper  | Unable to start JVM: No such file or directory" or similar then go to *$SONAR\_INSTALL\_PATH/conf*and open the *wrapper.conf* file. Edit the *wrapper.java.command* property and set it to an appropriate Java path. In my case it is */opt/java/jdk1.8.0\_101/bin/java*.
5. Open a browser and go to <http://localhost:9000/>. If you see something then everything is ok. Otherwise you should take an eye on the console output and investigate the issue.

**Installing the Python plug-in**

1. Open a browser and go to <http://localhost:9000/>
2. Log in with admin/admin.
3. Go to the  Administration tab -> System -> Update Center (these may vary due to your SonarQube server version).
4. Click on the Available button.
5. Search for the Python plug-in and choose Install.
6. Restart the SonarQube server if needed.

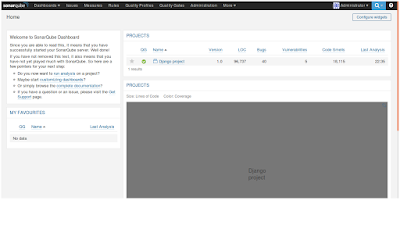
**Installing SonarQube Scanner**

1. Go to the [SonarQube Scanner page](http://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner" \t "_blank) and download the latest version. At the time of writing this blog post the latest version is 2.7.
2. And again the downloaded file is in a ZIP format and you have to unzip it in a folder of your choice. In my case it is in */opt/sonar-scanner.*
3. Optional but a good idea is to add the *$SONAR\_SCANNER\_PATH/bin* to your path variable because the *sonar-scanner* command will be used later.
4. Now is the time to configure some settings of the scanner. Open the *$SONAR\_SCANNER\_PATH/conf/sonar-scanner.properties*file. You can set some properties for your project. The most common and their defaults are:
   1. sonar.host.url=[http://localhost:9000](http://localhost:9000/) - the URL of the SonarQube server
   2. sonar.jdbc.username=sonar - username for the database (if you want an external database)
   3. sonar.jdbc.password=sonar - password for the database
   4. sonar.jdbc.url - the URL string to the database (Oracle, MySQL, Postgre and so on)

**Configuring the project to be scanned**

1. Go to the Python project you want to scan and create a file named "*sonar-project.properties*".
2. Now edit the file and set some properties:
   1. sonar.projectKey=my:project - the project id
   2. sonar.projectName=My project - the project name
   3. sonar.projectVersion=1.0 - the project version  
      These 3 properties are required
   4. sonar.sources=src1,src2 - the directories with sources you want to scan (comma-separated)
   5. sonar.language=py - the programming language (Java, Python, etc.)
   6. sonar.python.pylint=/usr/local/bin/pylint - the path to the pylin, if you want to include some quality rules from it

**Running the scan**

1. Make sure the SonarQube server is up and running.
2. Open a terminal/command prompt and navigate to the project folder where the "*sonar-project.properties*" file is located.
3. Type *sonar-scanner* and hit Enter (if you added the sonar-scanner path to you path variable, otherwise use the absolute path to the command).
4. When the execution is done go to the SonarQube server URL, e.g. [http://localhost:9000](http://localhost:9000/).
5. You should see something like this:  
   [](https://4.bp.blogspot.com/-931bpHivkgc/V9hYhq-fCdI/AAAAAAAABa8/JWbOJPztvyIF6VhfGCodM7_zBk_H5svAwCLcB/s1600/sonar.png)
6. You can log with admin/admin credentials and make some investigation of the features.

**Conclusion**

That was just a brief explanation of how to run a Sonar scan over your Python project and the aim was not a review of the Sonar functionalities.

**Jenkins :**

<https://www.edureka.co/blog/what-is-jenkins/>

public Repo

<https://mohitgoyal.co/2017/02/22/build-github-project-using-jenkins/>

Privare Repo

<https://mohitgoyal.co/2017/02/27/configuring-ssh-authentication-between-github-and-jenkins/>

**Protractor and Protractor Perf**

<http://www.webdriverjs.com/protractor/>

<http://www.protractortest.org/#/faq>

<https://github.com/angular/protractor/blob/master/docs/toc.md>

Protractor Perf

<https://www.linkedin.com/pulse/automating-angular-apps-using-protractor-integrating-continuous>

<https://github.com/axemclion/protractor-perf>

**Chef:**

chef srever login :

<https://api.chef.io/login>

rahulaware/rahul@123

<https://www.edureka.co/blog/install-chef/>

<https://www.edureka.co/blog/chef-tutorial/>

<https://www.digitalocean.com/community/tutorials/how-to-use-roles-and-environments-in-chef-to-control-server-configurations>

**SSL,GIT,Celery with rabbitMQ**

<http://www.learncodeonline.in/blog/what-is-ssl-and-how-to-install-ssl/>

<https://www.ntu.edu.sg/home/ehchua/programming/howto/Git_HowTo.html>

<https://orga.cat/posts/most-useful-git-commands>

<https://tests4geeks.com/python-celery-rabbitmq-tutorial/>

<https://www.systutorials.com/docs/linux/man/1-rabbitmqctl/>

**GitHUB,Devops,sonarquebe:**

<https://www.atlassian.com/git/tutorials/undoing-changes/git-reset>

<https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified>

<https://devopscube.com/devops-courses/>

<https://discuss.devopscube.com/>

SonarQubes:

# [SonarQube – What is it? How to get started? Why do I use it?](https://matthiasgeiger.wordpress.com/2014/02/19/sonarqube-what-is-it-how-to-get-started-why-do-i-use-it/)

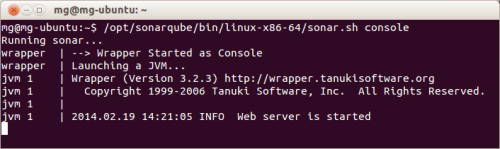
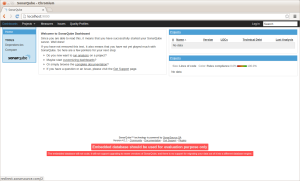
As [promised in my first post](https://matthiasgeiger.wordpress.com/2014/02/18/yet-another-blog/) this starts a small series of tutorials using *[SonarQube](http://www.sonarqube.org/" \t "_blank)* to verify some properties on BPMN process files. In this post I briefly sketch the purpose of SonarQube, describe the basic installation process and how the different parts of SonarQube can be used to perform some first analysis. Finally, I introduce my use case of analysing BPMN process models.

### What is SonarQube?

First of all some words to SonarQube: SonarQube (formerly known as Sonar) is an open source tool suite to measure and analyse to quality of source code. It is written in Java but is able to analyse code in about 20 different programming languages. Code analysis may be started manually by executing a so-called sonar runner but SonarQube’s full is potential is especially revealed when [used in combination with continuous integration](http://docs.codehaus.org/display/SONAR/Continuous+Integration) such as a Jenkins server. The results of an analysis are shown in a fancy web frontend with ‘green’ and ‘red lights’, nice charts, issue lists and an ability to drill down from project level to a single class. A public live demo showing the results for various open source projects is available [here](http://nemo.sonarqube.org/).

### Getting Started: Download, Installation, First Results

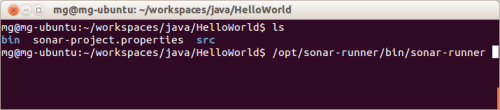
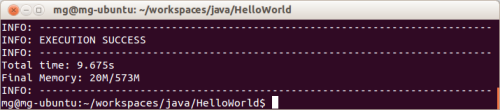
[Following the official documentation](http://docs.codehaus.org/display/SONAR/Setup+and+Upgrade)it is rather simple to get a local SonarQube instance up and running:

* Prerequisite: You need a Java installation on your machine.
* [Download the recent version of the SonarQube Server](http://www.sonarqube.org/downloads/), unzip it to your favored folder and run the OS-specific startup script. On my ubuntu 64bit machine this looks like this:  
  [](https://matthiasgeiger.files.wordpress.com/2014/02/sonarqube-startup-screenshot.png)  
  On Windows machines the `StartSonar.bat` file has to be executed which is located in the `$sonarqubeBase\bin\windows-x86-xx\` folder – where xx stands for 32/64 depending whether your machine has a 32bit or 64bit architecture.
* Using the default setting, the server is now accessible on [http://localhost:9000](http://localhost:9000/). However, there are no results to see as no project has been analyzed by now:  
  [](https://matthiasgeiger.files.wordpress.com/2014/02/sonarqube-first-run.png)

The getting started Tutorial now proposes to analyze a sample project from GitHub – I think it is more exiting to analyze on of your own existing projects. In order to add a project to your SonarQube server 1) you have to write/adjust a sonar-project.properties file and 2) run the analysis using the Sonar Runner.

* So let’s first have a look at an example `sonar-project.properties` file: Each project to be added to SonarQube needs a unique key, a name and a version identifier. Moreover, at least one folder pointing to the sources to be analyzed has to be defined. In my case the sample project is written in Java and encoded in UTF-8.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | # Required metadata  sonar.projectKey=hello-sonar-example  sonar.projectName=My first Project to be analyzed  sonar.projectVersion=1.0    # Comma-separated paths to directories with sources (required)  sonar.sources=src    # Language  sonar.language=java    # Encoding of the source files  sonar.sourceEncoding=UTF-8 |

* Put this file in the root directory of your project.
* To perform the analysis [download the SonarQube Runner](http://www.sonarqube.org/downloads/) and unzip it.
* Open a terminal/console in the root directory of your project to check and run the `sonar-runner`/`sonar-runnter.bat` script:  
  [](https://matthiasgeiger.files.wordpress.com/2014/02/sonar-runner-start.png)  
  The scripts will produce loads of output but at the end the execution should be successful:  
  [](https://matthiasgeiger.files.wordpress.com/2014/02/sonar-runner-success.png)

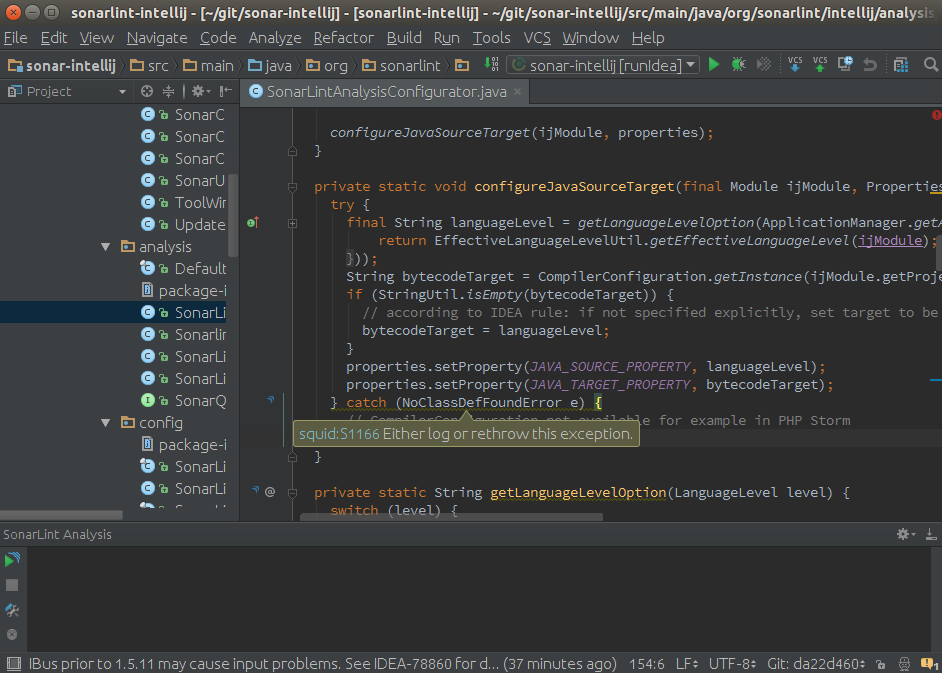
After successful execution of the SonarQube Runner you can find the results in the already mentioned web frontend on [http://localhost:9000](http://localhost:9000/)

Plugin also available :

<https://plugins.jetbrains.com/plugin/7238-sonarqube-community-plugin>

Sonarlint is same like sonarqube but used with IDE

SonarLint is an IntelliJ IDEA plugin that provides on the fly feedback to developers on new bugs and quality issues injected into Java, JavaScript, PHP and Python code. SonarLint supports SonarSource code analyzers (SonarJava, SonarJS, SonarPHP and SonarPython) as well as custom rules that extend these code analyzers. If your project is analyzed on SonarQube (formerly known as "Sonar"), SonarLint can connect to the server to retrieve the appropriate quality profiles and settings for that project. Java 8 is required to run SonarLint.

[](https://plugins.jetbrains.com/files/7973/screenshot_15543.png)

**Node and NPM help:**

# <https://docs.npmjs.com/getting-started/installing-node>

# npm –save or –save-dev. Which one to use?

# Introduction

If you have ever worked in NodeJs, you must have install one or two packages through “npm install <package>” command.  By running this command, the nodeJs will install this package on your working directory, under node\_modules.

To save this packages as your dependencies, under package.json, you have two choices:

* –save-dev
* –save

What is package.json?

All npm packages contain a file, usually in the project root, called package.json - this file holds various metadata relevant to the project. This file is used to give information to npm that allows it to identify the project as well as handle the project's dependencies. It can also contain other metadata such as a project description, the version of the project in a particular distribution, license information and et al.

Let us understand the difference that it can make.

# Detail

Say you have a package.json within your root folder of your project.

If you don't have one then create a package file using **npm init** command.

My package.json looks like this:

{

"name": "TMSPA",

"version": "1.0.0",

"description": "Single page application for TM",

"main": "index.html",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": {

"type": "git",

"url": "<https://github.com/codebased/android-test.git>"

},

"author": "Am",

"license": "ISC",

"bugs": {

"url": "<https://github.com/codebased/android-test/issues>"

},

"homepage": "<https://github.com/codebased/android-test>"

}

Now I want to install some dependencies.

Before I install one, I need to search for the package name. If you know the package that you want to install thats good. Otherwise, you can use npm search command:

npm search bootstrap

or try one of the following search tools:

* <https://www.npmjs.org/>
* <http://eirikb.github.com/nipster/>
* <http://browsenpm.org/>
* <http://node-modules.com/>
* <https://nodejsmodules.org/>
* [https://github.com](http://github.com/)

Once you have identified the right package that you want to install, you can use the mentioned command i.e. npm install <package name>.

Here you have two, actually three, options.

###### 1. use –save-dev

e.g. npm install should --save-dev

You will use this option when you want to download a package for developers , such as grunt, gulp, then use this option. Thus, when you are distributing your code to production, these dependencies will not be available.

As an example, let say, you want to use grunt as your task runner. This package is required for development purpose. Thus, you should use –save-dev here.

npm install grunt --save-dev

The above command will save grunt dependency under **devDependencies**section of your package.json, shown below:

{

"name": "TMSPA",

"version": "1.0.0",

"description": "Single page application for TM",

"main": "index.html",

"scripts": {

..

"author": "Codebased",

..

..,

**"devDependencies": {**

**"gulp": "^3.8.11"**

**}**

}

###### 2. Use –save flag

You will use this option when you want to save a package dependency for distribution. Item such as angularjs, or any other module that is is required at run time by your program, you will use –save switch.

npm install angularjs --save

Now my package.json looks like this:

{

"name": "TMSPA",

"version": "1.0.0",

"description": "Single page application for TM",

...,

**"dependencies":{**

**"angularjs": "^1.4."**

**},**

"devDependencies": {

"gulp": "^3.8.11"

}

}

###### 3. Use nothing

If you call npm install command without any flag then it will install package. However, there is no way the package.json will be updated with your dependencies.

This option is not recommended because there is no way others will get to know about the dependencies that your module has.

###### Conclusion

In conclusion, we understand that the –save-dev, and –save flags are used for limiting the scope of your dependencies.

**Docker Intallation and Protarctor setup on docker**

docker installation on windows using docker toolbox (windows less than 10 version)

<https://docs.docker.com/toolbox/toolbox_install_windows/#step-1-check-your-version>

Maksym@MaksymPC MINGW64 ~

$ docker ps

An error occurred trying to connect: Get http://%2F%2F.%2Fpipe%2Fdocker\_engine/v1.23/containers/json: open //./pipe/docker\_engine: The system cannot find the file specified.

Maksym@MaksymPC MINGW64 ~

$ docker-machine env default

Error checking TLS connection: Error checking and/or regenerating the certs: There was an error validating certificates for host "[192.168.99.100:2376](http://192.168.99.100:2376/)": tls: DialWithDialer timed out

You can attempt to regenerate them using 'docker-machine regenerate-certs [name]'.

Be advised that this will trigger a Docker daemon restart which will stop running containers.

Maksym@MaksymPC MINGW64 ~

$ docker-machine regenerate-certs

Regenerate TLS machine certs? Warning: this is irreversible. (y/n): y

Regenerating TLS certificates

Waiting for SSH to be available...

Detecting the provisioner...

Copying certs to the local machine directory...

Copying certs to the remote machine...

Setting Docker configuration on the remote daemon...

Maksym@MaksymPC MINGW64 ~

$ docker ps

An error occurred trying to connect: Get http://%2F%2F.%2Fpipe%2Fdocker\_engine/v1.23/containers/json: open //./pipe/docker\_engine: The system cannot find the file specified.

Maksym@MaksymPC MINGW64 ~

$ docker-machine env default

export DOCKER\_TLS\_VERIFY="1"

export DOCKER\_HOST="tcp://[192.168.99.100:2376](http://192.168.99.100:2376/)"

export DOCKER\_CERT\_PATH="C:\Users\Maksym\.docker\machine\machines\default"

export DOCKER\_MACHINE\_NAME="default"

# Run this command to configure your shell:

# eval $("C:\Program Files\Docker Toolbox\docker-machine.exe" env default)

<https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-16-04>

Helpful link:

<http://www.tothenew.com/blog/protractor-with-jenkins-and-headless-chrome-xvfb-setup/>

<https://tecadmin.net/install-google-chrome-in-ubuntu/> or

<https://www.howopensource.com/2011/10/install-google-chrome-in-ubuntu-11-10-11-04-10-10-10-04/>

Installation of Node js :

### To install or update nvm, you can use the [install script](https://github.com/creationix/nvm/blob/v0.33.2/install.sh) using cURL:

curl -o- <https://raw.githubusercontent.com/creationix/nvm/v0.33.2/install.sh> | bash

source ~/.bashrc

nvm install v7.5.0

npm install -g protractor

**Installation of java to run webdriver-manager:**

apt-get install default-jre

Add below in ~/.bashrc and then source ~/.bashrc

export JAVA\_HOME=/usr/lib/jvm/java-8-openjdk

export PATH=$PATH:/usr/lib/jvm/java-8-openjdk/bin

verify webdriver-manger is working:

webdriver-manager update and start

Install Chrome and depedency:

wget -q -O - <https://dl-ssl.google.com/linux/linux_signing_key.pub> | sudo apt-key add -

apt-get update

apt-get install libxpm4 libxrender1 libgtk2.0-0 libnss3 libgconf-2-4

wget -q -O - <https://dl-ssl.google.com/linux/linux_signing_key.pub> | apt-key add -

sh -c 'echo "deb <http://dl.google.com/linux/chrome/deb/> stable main" >> /etc/apt/sources.list.d/google-chrome.list'

apt-get update

apt-get install google-chrome-stable

(Help Link for chrome installation:<https://www.howopensource.com/2011/10/install-google-chrome-in-ubuntu-11-10-11-04-10-10-10-04/>)

apt-get install xvfb gtk2-engines-pixbuf

apt-get install xfonts-cyrillic xfonts-100dpi xfonts-75dpi xfonts-base xfonts-scalable apt-get install imagemagick x11-apps dbus-x11

create below shell script and keep it run or export display variable in ~/.bashrc and then source ~/.bashrc and run two process one for xvfb and anothe rfor webdriver-manager :

#!/bin/sh

Xvfb -ac :99 -screen 0 1280x1024x16 &

#disown $1

export DISPLAY=:99

webdriver-manager start /dev/null 2>&1

sh sript\_name.sh

Open Second promt :

create Automation Dir and create two file

// conf.js

exports.config = {

  framework: 'jasmine',

  seleniumAddress: '<http://localhost:4444/wd/hub>',

  specs: ['spec.js'],

  capabilities: {

    'browserName': 'chrome',

    'chromeOptions': {

      'args': ['no-sandbox']

    }

}

}

// spec.js

describe('Protractor Demo App', function() {

  it('should have a title', function() {

    browser.get('<http://juliemr.github.io/protractor-demo/'>);

    expect(browser.getTitle()).toEqual('Super Calculator');

  });

});

run protractor conf.js  (make sure that server is running)

-RAHUL.

# Setting Up the Selenium Server

When working with Protractor, you need to specify how to connect to the browser drivers which will start up and control the browsers you are testing on. You will most likely use the Selenium Server. The server acts as proxy between your test script (written with the WebDriver API) and the browser driver (controlled by the WebDriver protocols).

The server forwards commands from your script to the driver and returns responses from the driver to your script. The server can handle multiple scripts in different languages. The server can startup and manage multiple browsers in different versions and implementations.

[Test Scripts] < ------------ > [Selenium Server] < ------------ > [Browser Drivers]

The [config file](https://github.com/angular/protractor/blob/master/lib/config.ts" \t "_blank) includes several options for the Selenium Server, which are explained in the sections below.

## Standalone Selenium Server

To run the Selenium Server on your local machine, use the standalone Selenium Server.

**JDK**

To run a local Selenium Server, you will need to have the [Java Development Kit (JDK)](http://www.oracle.com/technetwork/java/javase/downloads/index.html) installed. Check this by running java -version from the command line.

**Installing and Starting the Server**

To install and start the standalone Selenium Server manually, use the webdriver-manager command line tool, which comes with Protractor.

1. Run the update command: webdriver-manager update This will install the server and ChromeDriver.
2. Run the start command: webdriver-manager start This will start the server. You will see a lot of output logs, starting with INFO. The last line will be 'Info - Started org.openqa.jetty.jetty.Server'.
3. Leave the server running while you conduct your test sessions.
4. In your config file, set seleniumAddress to the address of the running server. This defaults to<http://localhost:4444/wd/hub>.

**Starting the Server from a Test Script**

To start the standalone Selenium Server from within your test script, set these options in your config file:

* seleniumServerJar - The location of the jar file for the standalone Selenium Server. Specify a file location.
* seleniumPort - The port to use to start the standalone Selenium Server. If not specified, defaults to 4444.
* seleniumArgs - Array of command line options to pass to the server. For a full list, start the server with the -help flag.

**Connecting to a Running Server**

To connect to a running instance of a standalone Selenium Server, set this option:

* seleniumAddress - Connect to a running instance of a standalone Selenium Server. The address will be a URL.

Please note that if you set seleniumAddress, the settings for seleniumServerJar, seleniumPort, seleniumArgs, browserstackUser, browserstackKey, sauceUser and sauceKey will be ignored.

## Remote Selenium Server

To run your tests against a remote Selenium Server, you will need an account with a service that hosts the server (and the browser drivers). Protractor has built in support for [BrowserStack](https://www.browserstack.com/" \t "_blank) and [Sauce Labs](http://www.saucelabs.com/).

**Using BrowserStack as remote Selenium Server**

In your config file, set these options:

* browserstackUser - The username for your BrowserStack account.
* browserstackKey - The key for your BrowserStack account.

Please note that if you set browserstackUser and browserstackKey, the settings for seleniumServerJar, seleniumPort, seleniumArgs, sauceUser and sauceKey will be ignored.

You can optionally set the name property in a capability in order to give the jobs a name on the server. Otherwise they will just be allotted a random hash.

**Using Sauce Labs as remote Selenium Server**

In your config file, set these options:

* sauceUser - The username for your Sauce Labs account.
* sauceKey - The key for your Sauce Labs account.

Please note that if you set sauceUser and sauceKey, the settings for seleniumServerJar, seleniumPort, seleniumArgs, browserstackUser and browserstackKey will be ignored.

You can optionally set the name property in a capability in order to give the jobs a name on the server. Otherwise they will just be called Unnamed Job.

## Connecting Directly to Browser Drivers

Protractor can test directly against Chrome and Firefox without using a Selenium Server. To use this, in your config file set directConnect: true.

* directConnect: true - Your test script communicates directly Chrome Driver or Firefox Driver, bypassing any Selenium Server. If this is true, settings for seleniumAddress and seleniumServerJar will be ignored. If you attempt to use a browser other than Chrome or Firefox an error will be thrown.

The advantage of directly connecting to browser drivers is that your test scripts may start up and run faster.

**Routing Protocol:**

Routing protocols are the family of network protocols that enable computer [routers](https://www.lifewire.com/how-routers-work-816456) to communicate with each other and in turn to intelligently forward traffic between their respective networks. The protocols described below each enable this critical function of routers and computer networking.

### How Routing Protocols Work

Every network routing protocol performs three basic functions:

1. discovery - identify other routers on the network
2. route management - keep track of all the possible destinations (for network messages) along with some data describing the pathway of each
3. path determination - make dynamic decisions for where to send each network message

A few routing protocols(called link state protocols) enable a router to build and track a full map of all network links in a region while others (called distance vector protocols) allow routers to work with less information about the network area.

RIP :

### Routing Information Protocol Version 1 (RIPv1)

• RIPv1 is a [Distance-Vector Routing protocol](http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-distance-vector-routing-protocols.php).

• RIPv1 is a Classful routing protocol. Classful routing protocols support only the networks which are not subnetted. Classful routing protocols do not send subnet mask information with their routing updates. In other words, if you have a subnetted network in your RIPv1 routing domain, RIPv1 will announce that network to other as unsubnetted network.

• RIPv1 does not support [VLSM (Variable Length Subnet Masking)](http://www.omnisecu.com/tcpip/variable-length-subnet-masking-vlsm.php).

• RIPv1 support maximum[metric (hop count)](http://www.omnisecu.com/cisco-certified-network-associate-ccna/what-is-routing-metric-value.php) value of 15. Any router farther than 15 hops away is considered as unreachable.

• RIPv1 send routing updates periodically every 30 seconds as [broadcasts](http://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php) using destination IP address as [limited broadcast IP adddress 255.255.255.255](http://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php). Since the updates are sent using the destination IP address of [limited broadcast IP adddress](http://www.omnisecu.com/tcpip/internet-layer-ip-addresses.php) 255.255.255.255, every router need to process the routing update messages (whether they are running RIPv1 or not).

• RIPv1 does not support authentication of update messages (plain-text or MD5).

### Routing Information Protocol Version 2 (RIPv2)

RIPv2 is a [Hybrid Routing Protocol](http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-hybrid-routing-protocols.php). A [Hybrid Routing Protocol](http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-hybrid-routing-protocols.php) is basically a [Distance-Vector protocol](http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-distance-vector-routing-protocols.php) which some characteristics of [Link State routing protocols](http://www.omnisecu.com/cisco-certified-network-associate-ccna/introduction-to-link-state-routing-protocols.php).  
  
RIPv2 is classless routing, which allows us to use subnetted networks also. RIPv2 has the option for sending network mask in the update to allow classless routing.

• RIPv2 support [VLSM (Variable Length Subnet Masking)](http://www.omnisecu.com/tcpip/variable-length-subnet-masking-vlsm.php).

• RIPv2 support maximum [metric](http://www.omnisecu.com/cisco-certified-network-associate-ccna/what-is-routing-metric-value.php) (hop count) value of 15. Any router farther than 15 hops away is considered as unreachable.

• RIPv2 supports triggered updates.

• RIPv2 routing updates are sent as [Multicast traffic](http://www.omnisecu.com/cisco-certified-network-associate-ccna/unicast-multicast-broadcast.php) at destination multicast address of 224.0.0.9. Multicast updates reduce the network traffic. The Multicast routing updates also helps in reducing routing update message processing overhead in routers which are not running RIPv2. Only the routers running RIPv2 join to the multicast group 224.0.0.9. Other routers which are not running RIPv2 can simply filter the routing update packet at Layer 2.

• RIPv2 support [authentication of RIPv2 update messages](http://www.omnisecu.com/cisco-certified-network-associate-ccna/how-to-configure-rip-authentication-keychain.php) (plain-text or MD5). [Authentication](http://www.omnisecu.com/cisco-certified-network-associate-ccna/how-to-configure-rip-authentication-keychain.php) helps in confirming that the updates are coming from authorized sources.