**First tool of the devoops:**

**Version controlling**: each developer uploads his code in to a server which is called version controller.

The version controller accepts the project uploads from multiple developers and creates an integrated project.

The process of uploading into version controller is called **checkin.**

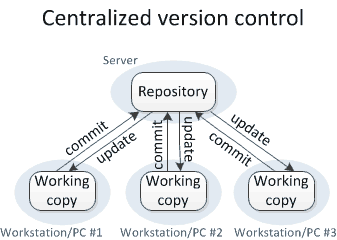
Downloading is called **checkout.**

Version controlling is characterized in 2 ways:

1. **Centralised version controlling-SVN [sub version]**
2. **Distributed VC-GIT**

**Centralised version controlling-SVN [sub version]**

Into centralized version controlling the repository is maintained on a remote machine all the developer and testers should upload their code in to the remote controller.

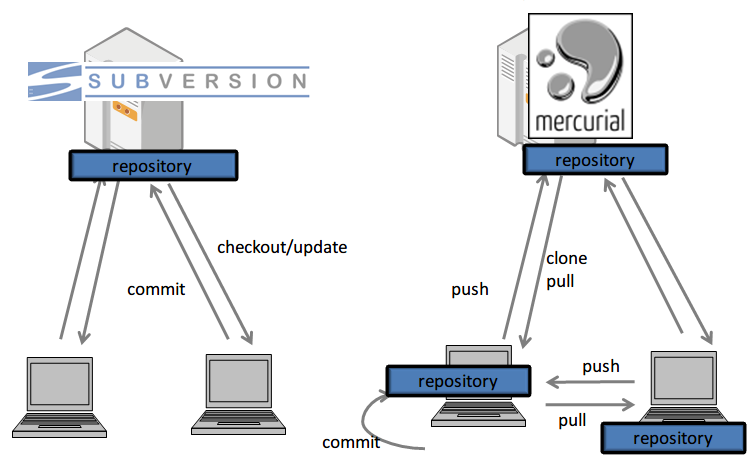


**Distributed version controlling:**

In distributed version controlling repository, local repository is maintained on every developer’s machine. The developer first commits the code into the local repository completely version controlling can be done on the local repository.

From the local repository the code is **pushed** into the remote repository.

The remote repository accepts the project uploads from multiple developers and integrates them



**GIT:**

Stp1: download and install git from

[**http://git-scm.com/downloads**](http://git-scm.com/downloads)

Once git is install we will get an application called git bash: this is command prompt of GIT.

This GIT repository install is an every developer machine and we should configure there username and emailID

**Command for configuring username**

$ git config --global user.name “younis”

**To configure emailID:**

**$** git config --global user.email “[younis.413@gmail.com](mailto:younis.413@gmail.com)”

To see the list of all the configurations on our git account

$ git config --list

**Starting or initializing git**

**$ git init**

This will initialize into git repository. It will create hidden folder called git

We can see all the files and folders

Using

**$ ls –a**

Git uses 3 sections for storing and migrating data.

1. **Working directory:** Working directory is the actual folder where the code is created by developer or testers
2. **Staging area:** Staging area is temporary zone where indexing is done
3. **Local repository:** From the staging area data goes in to local repository.

23/08/2017

**$ pwd**

**$ cd desktop**

**$ cd gitfolder**

To find the status of the git repository

**$ git status**

To send files in to the staging area

**$ git add filename**

To send multiple files into the staging area

**$ git add file1 file2 file3**

After adding some files go to staging area give **git status** command

This will display the untraced files red colour. The files present in staging area in green colour.

To send the file from staging area to git local repository

**$ git commit --m “some massages”**

With the above command all the files in the staging area will be get saved in the git local repository

To send all files and folders including sub folders in to the staging area

**$ git add . [. Means present working directory]**

To send the files and folders including sub folders from staging to git local repository

**$ git commit --m “add the total files”**

The files send in to the staging area can be taken back using the reset command

**$ git reset filename**

We can also use rm command for remove files staging area

**$ git rm --cached filename**

To remove the command from the local repository and bring them in to the staging area we can use

**$ git reset - -soft HEAD^**

The data present LR should be sent in to the remote repository we can do this by creating an account with git hub.

**Note:** when we add a new file in to working directory which is in untracked stage.

When we modified the existing file in staging area, Local repository which is come in tracked stage only.

**WORKING ON GIT REMOTE REPOSITORY:**

1. Open github.com

2. Create a free account

3. Login using that account

4. Click on + down arrow –

5. Click on new repository

6. Give some name to repository

7. Click on public radio button

8. Create repository

9. Go to “Push and existing repository form the command line”

10. Copy the first command paste it in git bash

11. Copy the 2nd cmd and paste it in git bash

12. It will ask for username and password of the GitHub account

24/08/2017

**$ git log - -stat [ total status at the files]**

**$ git log**

**$ git push url.**

Downloading from remote repository:

This can be done in 2 ways:

1. **Clone-in cloning the complete project gets downloaded into our machine: $ git clone**
2. **Pull: in pulling we will download only the updates. Generally these updates can be by other team members: $ git pull**

**Branching it GIT**

We can create multiple branches and assign them to induvial developers each developer will work on his induvial branch and this work can be latter merged with the master branch.

The default branch is master branch

To find the list of all the branches

**$ git branch**

To create new branch

**$ git branch branchname**

**NOTE:** The branch with asterisk symbol is the one on which the control of git is present

To move into a branch

**$ git checkout branchname**

To merge a branch with another branch

1. Checkout into the master branch $ git checkout master
2. $ git merge [branch\_to\_be\_merged] webapp

**Ignoring files:**

Files in current working directory can be ignored.

That is GIT cannot track them or untrack them.

This can be done by using a file called .gitingnore

1. **cat > .gitignore**

**\*.log**

Ctrl +c [come out of the]

Now all the files that are added in the .gitingnore cannot be accessed by git.

28/08/2017

**Deleting branches from local repository**

**$ git branch - -d [the\_local\_branch\_name]**

To delete branches from the remote repository

**$ git push origin --delete the\_remote\_branchname**

**Build process:**

**Ant and maven:**

Are used as build tools for building the java code

**Configuring apache Ant:**

1. download and install JDK 1.7 or JDK 1.8
2. open c drive – program files- java- open JDK folder-copy its path
3. right click THIS PC – click on properties – advanced system settings- advanced-environments variables – go to system variable’s- new
4. create a variable JAVA\_HOME and paste the path of the JDK
5. High let the path variable click on edit

**Win 10**

Click on new to existing path add; paste path of JDK/bin

**Win 7**

Paste path of JDK/bin

Open [**https://ant.apache.org/bindownload.cgi**](https://ant.apache.org/bindownload.cgi)

And download apache ant bin.zip version.

Extract it open it we will find another apache ant folder

Open that folder copy its path

Click on environment variables

Click on new

Create a variable ANT\_HOME

Paste the path of the apache ant – click on ok

High let path variable click on edit-

**Win 10**

Click on new paste path of apache ANT/bin

**Win 7**

To existing path add; paste path of apache ANT/bin

Checking whether ant is configure or not

Open command prompt

C:\ant –version

Ant performance it’s a build actives using file called build.xml

**Sample structure of build.xml file**

**<!..name of the ANT project and default target-->**

**<project name=”My ANT Project” default=”target3”>**

**<!--name of the target-->**

**<target name=”target”>**

**<!--ANT task to be performed-->**

**<echo>This is 1st ant statement</echo>**

**</target>**

**<target name=”target2” depends=”target1”>**

**<echo>This is 2nd ant statement</echo>**

**</target>**

**<target name=”target3” depends=”target2”>**

**<echo>This is 3rd ant statement</echo>**

**</target>**

**Stages in ANT build process:**

**CLEAN:**

In the clean stage ANT will delete the pervious builds.

**INIT:** in the init stage ANT will recreate the folder structure

**COMPILE:** in compile stage .java files are converted into .class files

This .class files are also called as Byte code.

They can be execute any o/s

**Build:**

In this stage .class files archive in to a single jar file or war file

30/08/2017

**Maven**

It is a build tool of java and it runs on community servers which are maintained by Apache all the open source community upload their updated jars in to this maven global servers.

We connect to the maven global server download the jars are necessary for our project and store them on separate machine called maven local server.

All the developers and tester connect their projects to this maven local server and those jar files perform the development and testing activities.

In this way the code is stores separately on git and jar files required for development are store and separate server this will help in protecting our code presenting in git from any virus that might be present in these open source jars similarly it becomes easy to upgrade or downgrade the existing jars without disturbing code in git.

31/08/2017

**Configure the Maven**

1. download and install JDK
2. open cdrive – program files- open java-open JDK folder – copy its path
3. click on environment variables click on new- create a variable JAVA\_HOME and paste the path of JDK
4. High let the path variable click on edit- click new – paste the path JDK/bin
5. Open – **https://maven .apche.org/download.cgi**
6. And download bin zip version
7. Extract it and it open and copy its path
8. Click on environment variables – click on new- create a variable M2\_HOME and paste the path of apache maven
9. High let the path variable click on edit-click on new- paste the path of MAVEN/bin
10. Checking whether maven configured or not
11. Open cmd – mvn - -version

**Creating a MAVEN project:**

**These project are also called as artefact**

And they are create from cmd prompt.

Open cmd prompt – **mvn archetype: generate**

Choose a number or apply filter (format :…..): 1027: enter

Choose a number: 6: enter

Define value for groupId: com.inte

Give the name of package in which developer

Define value artifactID: give some name for the project

Define value for version: just hit enter

Define value for package: enter

Confirm the above details: y

The maven project created in cmd prompt can we accessed from various java IDEs like [eclipse, intalliq…] to convert the project so that it can be accessed from eclipse

**cd project name-press enter**

**Mvn eclipse: eclipse**

This we create two file called .class path .project

If these files are create the maven project can be imported in to eclipse

Importing a maven project in to eclipse

Open eclipse- file menu- import – expand general – click on existing place –click on next – browse the path of our project.

**Adding third part**y API into the

Expand the maven project in eclipse click on pam.xml

Go to dependency node and in the dependency node we should add the third party APIs

Open search.maven.org

And search for the API that we require click on the latest version – go to dependency information copy the xml code and paste in pom.xml file.

Navigate to users and open .m2 folder – open the repository folder copy the its path – right click on our project on eclipse – click properties click on – java build path- click on libraries – click on add variable – click on configure variables – click on new-create variable M2\_REPO

Path: paste the path of the repository folder – click on ok- close eclipse- open cmd- mvn compile- this will download all the dependency that we added in to pom. Folder in to .M2 folder

**Mvn eclipse:ecplise**

This will update changes into .class and .project files.

After updating the APIS we need to compile the project.

**Mvn compile**

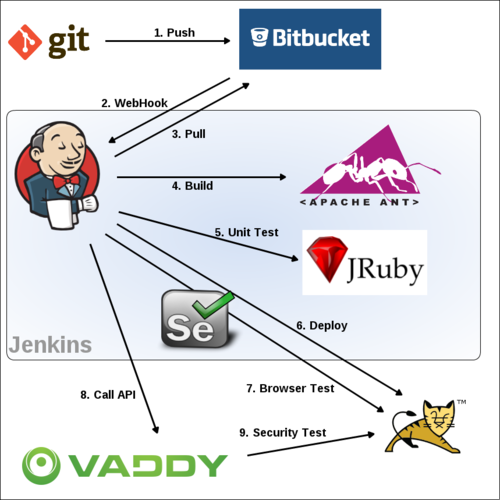
**1/09/2017**

**Jenkins**

This is an open source tool for implementing CI.

Jenkin can manage all the activities that should be done on the code until it goes for the deployment.

1. Whenever code is committed into git. Jenkin detects that and it will initiate the build process.
2. Jenkin will take the help of plugins like Ant, Maven and perform the build process that is converted the code into a war file or jar file etc.….
3. Jenkins will deploy this war file in the QA environment and it will pick the automation testing programs which are present in git and run those programs on the application under test / desk.
4. Jenkins will generate reports of after the testing and it can send automated mails to the team members.
5. Jenkins will now deploy the code in to a pre-production environment and it will run the **user acceptance test** [UAT] scripts which might be creating using cucumber, fitness etc... Finally Jenkins will deploy this code in to the pre-production environment where it goes like…



**Download and installing Jenkins**

1. JDK (latest) must be installed
2. Open Jenkins.io/download
3. Go to long-term support
4. Download version corresponding to our o/s
5. Install
6. Jenkins run on a port no: 8080 to
7. To open the dashboard of Jenkins
8. Launch any browser
9. Give localhost:8080 in **url**
10. First time, an unlock screen will open, where it will show a location where the admin password is present open the file, copy password and unlock the Jenkins
11. In the customised, Jenkin page click on install suggested plugins
12. To create the first admin user by entering the user name as admin and other details
13. Save and finish
14. Click on start using Jenkins

**[Unlock the Jenkins]:**

In the unlock screen it will show the location where the admin password open that file copy the password and unlock Jenkins continuous.

In the customize Jenkins page click on install suggested plugins.

**Create first Admin user**

Username: younis

Password: younis

Confirm password: younis

Email: younis.41333333@gmail.com

Click on save + finish

Click on start using Jenkins finish

**Creating a jobs in Jenkins:**

**Jobs is nothing but tasks.**

**NOTE:** jobs are created for different activities like pulling the code from the git, performing the build deploying into a specify environments etc.

1. Open dashboard of Jenkins
2. Localhost:8080
3. Click on new item give some item name
4. Click on free style project
5. Click on ok

Go to build section click on add build step click on execute windows bash command

Echo “same statement”

Apply and saved

Go build to the dashboard of Jenkins and here we will find all the jobs click on {icon} schedule build icon and build get executed

To view these output of the build click on the job that we created go to build history and click on the build number click on console output.

2/09/2017

**Step1:**

The code which is committed into the GIT repository by the developer should be downloaded by Jenkins. This is called cloning or fetching.

**Download and install GIT and configuring into Jenkins**

1. Open <http://git.scm.com./downlaod>.
2. Install.it
3. Now go to C:/programfiles/git/bin/copy the path of git.exe
4. Open the dashboard of Jenkins
5. Click on mange Jenkins
6. Go and click on global tool configuring
7. Go to git section and give some name to it.
8. Paste the path of git.exe – apply
9. Open c:/programfiles/java/Jdkfolder/copy path
10. Open the dashboard of Jenkins click on manage Jenkins click on global tool configuration.
11. Go to java section and give some name
12. Paste the path of jdk and click apply and save
13. Go back to dashboard lo Jenkins
14. Go to the job we created click on downward facing arrow- click on configure
15. Go to source code management, click on git ration button and paste the url of repository provided by developer like.
16. <http://github.com/selenium-saikrishna/maven.git>
17. Apply and save
18. Got to the dashboard of Jenkins
19. Go to job we have created and click on schedule job
20. Go to name section and click on job name that we have given.
21. Go to build history, click on latest build no..
22. Click on console o/p. here it will display that Jenkins is cloning the project from git. It will also show in which located the project is download.

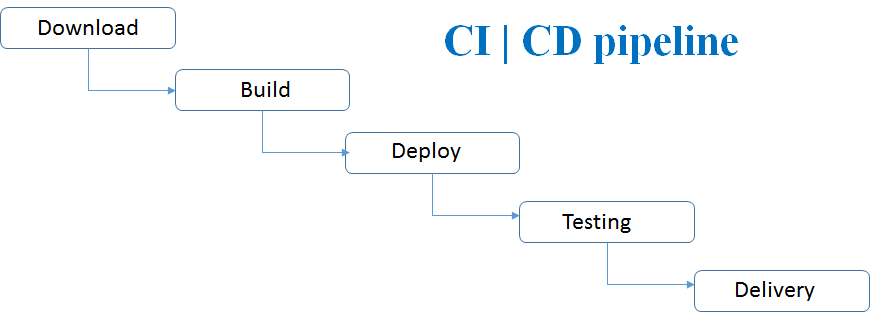
**Step2:**

1. Performing a build of the downloaded / cloned a project. In previous step.
2. Here Jenkins will created a war file with the help of maven.
3. Download and install maven in your **m/c** form apache.maven.org
4. Extract it and open it copy its path.
5. Open dashboard of Jenkins
6. Click on manage Jenkins, global tool configuring
7. Go to maven installation section
8. Given some name and paste the path of apache maven.
9. Go back to the dashboard of Jenkin, go to the job we created, click on the downward facing arrow.
10. Click on configure
11. Go to build section
12. Click on add build step-invoke top level maven tag target – go to maven version and select the name of maven that we have created earlier.
13. Go to “goal section and give “clean package”

Note: clean will delete the previously created. code and, package will convert the source code in to a war file. Click on apply and click on save.

04/09/2017

**SENDING AUTOMATED MAILS VIA JENKINS**



* Open dashboard of Jenkins[localhlost:8080]
* Go to the job we created click on the downward arrow
* Click on configure
* Go to post build actions
* Click on email notification and enter emails IDs of the our team members
* Click on apply & save
* Go back to the dashboard to the Jenkins click on manage Jenkins
* Click on configure system
* Go to email notification
* SMTP serves smtp.gmail.com
* Click on advanced
* Click use SMTP Automation
* Enter Gmail for username and password
* Check use SSL [ click on the check box right]
* Enter SMTP port 465
* Check the box of test configuration by sending test e-mail [younis.413@gmial.com](mailto:younis.413@gmial.com)
* Click on test configuration
* Check for the msg in email was successfully sent
* Click apply and save.

**Scheduling the build for particular date and time**

* Open the dashboard of Jenkins
* Go to the job we created click on the downward arrow click on configure
* Go to build triggers
* Click on check build periodically schedule the date and time
* Apply and save

**Step 3:**

The code which was built in the previous step was built in the form of war file. This was file should now deploy in to the QA environment so that tester using can start now it.

Jenkins can automated this deployment process using a plug in called **deploy to container**

* Open the dashboard Jenkins
* Click on manage Jenkins
* Click on manage plugins
* Click on Available
* Search for deploy
* Check to deploy to container plugins
* Click on download now install after Restart
* We required apache tomcat[glassfish, jboss] to servs on a container to deploy our war file
* <https://tomcat.apache.org>
* Download tomcat 7 are 8 else 9 also
* Extract it. We will find a folder called webapps – in this webapps folder the war file should be pasted.

06/09/2017

Tomcat also starts on port no .8080 server on this machine we are running Jenkins in the save port no: it will create a conflict so we should start tomcat as different port no:

* Go back to the dashboard off jenkins
* Go to the job we created click on downward arrow-click on configure
* Go to post build actions click on add port build actions – click on deploy war/ear to container
* WAR/EAR files: “\*\*/\*war”
* Context path : webapps
* Go to containers –click on add container select tomcat 7&8
* Go to credentials :admin/\*\*\*\*\* - add
* Here we should enter the tomcat credentials. This we can find in the tomcat folder that we downloaded
* Open tomcat folder- open conf folder- open tomcat-using and copy the user name and password
* **<user username="admin" password="admin" roles="manager-script"/>**
* Go to tomcat url and enter the url of tomcat click on apply and save.
* Go to the dashboard of Jenkins
* Go to the job we created click on schedule build icon
* Now it will perform all the three steps that is downloading from git, creating a war file deploying into QA environment.
* Jenkins can also we directly deployed on a tomcat server rather than installing it.
* We required Jenkins .war file
  + Open Jenkins.io – download
  + Go to LTS- Generic java package download
  + We will get file called Jenkins.war
  + Copy this war file and paste it in the webapps folder of tomcat
  + Restart the tomcat…
  + Lunch any browser – navigate to the localhost:9090/Jenkins.

07/09/2017

**Step4:**

Once the code is deployed into QA env Jenkins should fire the selenium programs created by the testers and perform functional testing.

1. Open the dashboard of Jenkins
2. Click on new item and created a new job for testing
3. Click on free style project click on ok.
4. Go to add build step click on execute windows batch command
5. And give the path of the jar file provided by the QA team command : java –jar c:\users\desktop sample.jar
6. Apply and save
7. Go to the dash board of Jenkins
8. Go to the job we created – click on configure
9. Go to post build actions – click on add post build action.
10. Click on build other projects and give the name of the QA job we created.
11. Click on apply and save
12. Go to the dash board of Jenkins and run the main job that we created
13. This job will download the code from git perform a build using maven deploy into the QA env. And now the 2nd job starts that is the testing programs gets executed.

**Step5:**

Once the testing activity is finished the war file should deployed into QA env.

For this we required apply in called copy artefacts plug ins.

1. Open the dash board of Jenkins
2. Click on manage Jenkins
3. Click on manage plug ins
4. Click on available - search for copy artefacts plugin and install it.
5. [need some matter ]
6. [need some matter ]

**08/09/2017**

**Creating Jenkins setup on Linux machine**

1. We will create 3 vms. 1st for installing Jenkins server 2nd as QA server where the testing will be done and 3rd one is production server
2. Vms can be created using vagrant.
   1. Download install oracle virtual box.
   2. <http://www.virtualbox.org/wiki/downloads>
   3. Download and install vagrant <http://www.vagrant>
   4. Copy the vagrant file into some folder open command prompt change the dir to that folder vagrant up
   5. This will create 3 vms
   6. Open oracle virtual box and jenkins vms- server enter username as a: vagrant Password: vagrant
   7. To connect to the Linux vms we can use applications called putty.
   8. Open putty.org and install it
   9. Open putty desktop app enter the IPaddress of the jenkins server – enter the username and password as a vagrant and vagrant
   10. To install jenkins we should switch in to root action su – enter the root password as vagrant
   11. Root password : vagrant

**Update the java repository:**

1. # add –apt –repository ppa:webupd8team/java
2. Click on enter
3. Update the database
4. # apt-get update
5. Install oracle java 8.
6. # apt-get install oracle-java8-installer
7. y/n:y
8. adding the Jenkins keys by downloading
9. wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add –
10. sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
11. # apt-get using installing applications
12. # apt-get update
13. Install Jenkins
14. # apt-get install Jenkins
15. Uname –a

* To open the dashboard of Jenkins
* Open any browser in windows
* Give the ipaddress of jenkins server :8080
* 192.168.60.40:8080

**Installing Tomcat in QAserver and production server.**

1. Open putty give the iPad dress of the QA server – enter username and password as vagrant, vagrant
2. Switch to root user
3. Su – enter root password as vagrant
4. # apt-get update
5. # apt-get install tomcat
6. # apt-get install tomcat7-admin
7. y/n:y
8. tomcat have dir will be in /etc/tomcat7
9. change the dir # cd /etc/tomcat7 # ls
10. here we should edit tomcat-user.xml file
11. opne the file # vim tomcat-user.xml
12. press i insert mode
13. here……add the last in line
14. **username=”tomcat-admin”password=”tomcat-adim”role=”manager-scripts”**

**Installing maven, git on Jenkins server**

1. apt-get update
2. apt-get install git maven
3. we should specify the path of JDK git and maven
   1. open the Jenkins server machine type the command # which java
   2. copy the path which is displayed # readlink –f path-of-java
   3. copy the path of the folder upto jre /user/lib/jvm/java-8-oracle
   4. open the dash board of jenkins-manage jenkins click on global configuration- click on JDK installation.
   5. JDK myjava JAVA\_HOME /usr/lib/jvm/java-8-oracle
   6. Go to git and give this details git name as **mygit** path of git **git**
   7. Go to maven –maven installation
   8. Maven mymaven path /usr/share/maven

**Implementing pipeline in Jenkins**

1. Pipelines are used for displaying the flow of jobs on jenkins in a graphical user format
2. We should install pipeline plugin
   1. Open the dashboard of Jenkins click on manage Jenkins – manage plugin- available-search for pipeline install build pipeline plugin click install.
   2. Go to the dash board of jenkins
   3. Click on new view icon [+]
   4. Select build pipeline view new to the pipeline click on
   5. Go to pipeline flow go to ustream/downstream config…
   6. Select the password job that should applied. Click on apply and save

**Jenkins – job – configuration tools**

**They are 6 types:**

1. **General**
2. **Source code management**
3. **Build Triggers**
4. **Build environment**
5. **Build**
6. **Post-Build**

**General:**

In this filed we have multiple options and they are using for particular job.

1. **Project name:**

Description:

2. **Discard old builds:**

Strategy: Log rotation

Days to keep builds:

Max # of builds to keep:

3. **GitHub old project**

Project url:

4. This project is parameterized:

Add parameter:

1. Boolean parameter
2. Choice parameter
3. Credentials parameter
4. File parameter
5. List subversion tags (and more)
6. Multi-line string parameter
7. Password parameter
8. Run parameter
9. String parameter

5. Throttle builds

Number of builds:

Time period:

6. Disable this project:

7. Execute concurrent builds if necessary

**Source code management:**

**1. None**

**2. GIT:**

Repositories:

Repository URL:......................................

Credentials:

Here we can multiple repositories: add repositories

Branches to build:

Branch specifier (blank for ‘any’): add branch name

Repository browser:

* Auto
* Assembalweb
* FishEye
* Kiln
* Microsoft Team Foundation Server/visual studio team services
* Bitbucketweb
* Cgit
* Gitblit
* Githubweb
* Gitiles
* Gitlab
* Gitlist
* Gitoriousweb
* Gitweb
* Gogs
* Phabrictator
* Redmineweb
* Rhodecode
* Stash
* Viewgit

**Additional Behaviours:**

**Advanced checkout behaviours**

**Advance clone behaviours**

**Advanced sub-modules behaviours**

**Calculate changelog against a specific**

**…etc**

**So many additional behaviours is available**

**3. Subversion:**

Modules:

* + - Repository URL:
    - Credentials:
    - Local module directory:
    - Repository depth:
    - Ignore externals:

Add module

Additional Credentials:

Realm:

Credentials

Check-out strategy:

* + - Use ‘svn update as much as possible
    - Always check out a fresh copy
    - Do not touch working copy. It is updated by other script
    - Emulate clean checkout by first deleting unversioned/ignored files, then “svn update’
    - Use ‘svn update ‘ as mouch as possible, with ‘svn revert’ before update

Repository browser:

* + - (Auto)
    - Assembla
    - CollaNet
    - FidhEye
    - Phabricator
    - SVN::Web
    - Sventon 1x,2.x
    - ViewSVN
    - WebSVN

**Build Triggers:**

**11/09/2017**

**Docker**

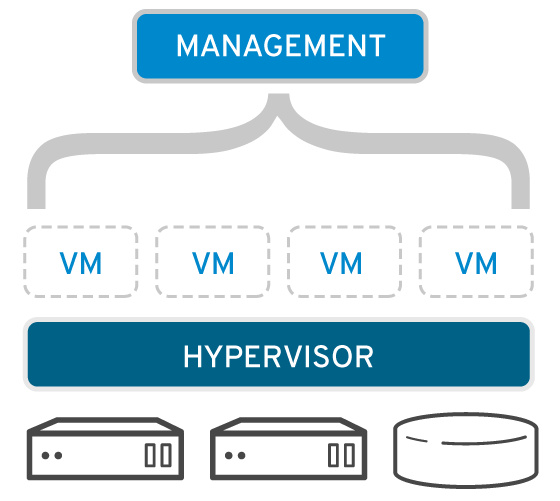
This is a containerization platform which can be used for creating the development environment, testing environment and production environment….

Docker use a concept called containers. This is the next step in virtualization.

**Virtualization:**

In virtualization we can create multiple VM’s on one host o/s. This is done by using a software called hypervisor.

The problem with VM is the application running on ………. Fast through multiple layer to access the multiple resources this will decrease the performance on the application.



**Containerization:**

We have a docker engine and on this docker engine the container run on separate process.

These containers not to be allocated and fixed amount of H/W resources.

During the runtime based on the usage of each container docker engine automatically assign the necessary amount H/W resources.

Note: containers are not to be an individual process. Which is run in user spaces.

**Installing docker on Linux:**

Open get.docker.com

Copy those two commands paste them in the Linux terminal

**This script is meant for quick & easy install via:**

$ curl -fsSL get.docker.com -o get-docker.sh

$ sh get-docker.sh

Get.docker.com is a shell script in which is docker community has created the code for the installing docker on different version on Linux.

The shell will available which flour of Linux. These are using accordingly it will installing docker.

**Windows docker installing:**

On windows docker can install on windows **10 and enterprise edition on 2016server.**

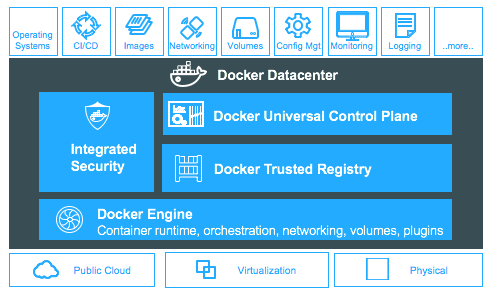
Open <http://docs.docker.com/engine/insttalation/#supported-platforms>.

Go to supported platform search and download the windows related version installing it.

**Note:** docker when installed on windows activates on applicant called HYPER-V.

It is the virtualization of Microsoft once this Hyper-V is active it does not allows other virtualization to run on our machine

Go to control panel check the box of Hyper-V.



**Docker information commands:**

1) --version

# docker --version

This will displays version of docker installed on our machine

2) # docker info

This will display the information about docker engine and also the system on which it is running.

3) # docker help

This will give the list of all the commands that can be used in docker

The help can also be used for finding information about application docker commands

Eg: # docker search --help

**Images and containers:**

**Image:**

Creating all the binaries and libraries which are required for a particular process it is a snapshot of the S/W.

**Container:**

Running instances of on image is called “**container”.**

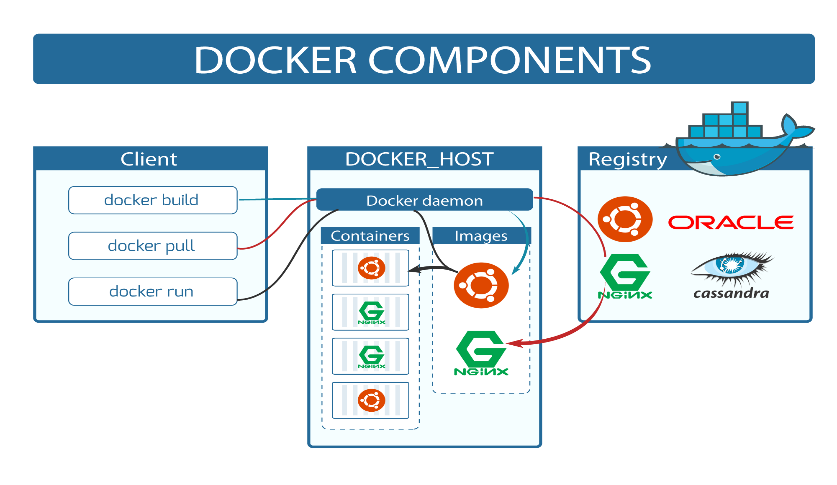
All the docker images present in a site called hub.docker.com

We can download the images & customize them according to our requirement.

**Components of docker**

Docker contains 3 main components

1. Docker client [DC]
2. Docker daemon
3. Docker registry (or) hub



DC: It is a location where we execute docker commands.

DD: It contains is docker engine and it also contains the containers and images.

All the commands we execute through docker client will be received by docker engine.

The docker engine sends this commands to respective images or containers and perform the action.

DH (or) DR: this is the cloud site of docker where docker community has uploaded different images.

The docker engine can download the image from this registry.

**Important docker commands:**

**Section-I:** (Build commands)

1) It is a command to build docker image and give some name to it.

# docker build –t image\_name

2) To show list of all images in docker engine

# docker images ls

Or

# docker image

3) Delete images from the local images

# docker rmi image\_name

**Section-II(ship commands)**

1) pull on image from registry

# docker pull image\_name

2) login to the docker registry

# docker login my\_registry.com.8080

3) Rename a local image

# docker tag original image\_name new image\_name

4) push on image to docker registry

# docker push name\_of\_our\_repository/image\_name

**Section-III (Run commands)**

1) To start a container

# docker run image\_name

**Run commands uses is multiple options**

* + 1. –rm : remove the containers after we exit from it.
    2. –it: comments the containers to an interactive terminal whole. We can execute Linux commands.
    3. --name: to give customized name to our containers.
    4. –p: -p 5000:5000 this will expose the containers internal port 5000 to host machine port 5000.
    5. –v : this is used for mounting volumes
    6. –d : to run detached mode –detach
    7. –p: publish the port numbers
    8. –e : to pass environment variables to contains (settings)
    9. –a: to attach the container to STDIN or STDOUT

2) To stop the container

# docker stop container-name/Id

Or

# docker kill container-name

3) To restart the container

# docker restart container-name

4) To restart after 20 seconds

# docker restart –t 20 container-name

5) To get into a container which is already running

# docker container exe –it container-name bash

6) To see the ports of a containers

# docker container port container\_name

7) To see all running container

# docker container ls

8) To find all available networks

# docker network ls

9) To inspect

# docker network inspect network-name

10) To inspect a specific container

# docker container inspect container-name

11) To create a new N/W

# docker network create my\_network

12) To connect to a N/W

# docker network connect network\_id container\_id

13) to disconnect from N/W

# docker network disconnect n/w\_id container\_id

14 ) to display logs container-name

# docker logs container\_name

15 ) To display last 100 logs of a container

# docker logs –tail 100 container\_name

16 ) docker information commands

# docker –version

# docker info.

15/09/2017

**Use cases**

1. **Run nginx on external port 80: and internal port 80 name it as webserver**

* **# docker container run –p 80:80 –d –name webserver nginx**
* The above command will start nginx. to see the home page of the nginx
* Open another session or terminal give the command
* **# enlinks** [**http://localhost:80**](http://localhost:80)
* Note: in the above command run initial checks for that image in the local image cache. If the image is present here it will started
* If it is not present it will download from docker hub and start the container.
* It creates a virtual ip on the docker private network in the docker engine.
* Opens port 80 on the host machine and communicates with port 80 on the docker container.

2) **Run httpd in detached mode on eternal port 8080 and internal port 80 name it apache webserver**

* # docker container run –d –p 8080:80 –name apachewebserver httpd
* # elinks <http://localhost:8080>

**3) Run MySQL on external port 3307 and internal port 3306 … pass environment variable MYSQL\_ROOT\_PASSWORD=yes**

* **# docker container run –d –p 3307:3306 –e MYSQL\_ROOT\_PASSWORD=yes –name mydb mysql**
* **# elinks http://localhost:3307**

**4) Run Ubuntu container and install git on it. Commit the image. Run the image with a new name and check if git is still present**

* **# Docker container run -it –name myubuntu Ubuntu**
* **In Ubuntu shell execute the below commands**
* **# apt-get update**
* **# apt-get install git**
* **# exit**
* **Open hub.docker.com -> create a free account**
* **To commit the image**
* **# docker commit container\_id[name] repository\_id/image\_name[syntax]**
* **# docker commit myubuntu intelliqit/git-ubuntu**
* **# docker run –it intelliqit/git-ubuntu**
* **We will find git installed in it**

**5) Push the above image to docker hub**

* The docker image created in the previous step is present with in the docker daemon it can be uploaded into the docker hub from where any can download or it can be uploaded into the local registry
* To upload into docker hub
  + Create an account in hub.docker.com
  + Docker login
  + Enter username and password
  + # docker push image\_name

**Committing images to the local registry**

* Local registry is similar to docker hub but it is specific for particular organization.
* which ever images are committed in to the local registry can be access with in that network
* To create local registry we should download and run the registry image.

**6) Download alpine Linux and push into the local registry**

* # docker pull alpine
* # docker container run –d –p 5000:5000 –-name local registry
* The above will download an image called registry run it on external and internal port 5000
* To commit any image in to the local registry we should first tag it [tag mean giving a name]
* # docker tag tag\_name localhost:5000/itmage\_name[syntax]
* # docker tag alpine localhost:5000/alpine
* # dicker push localhost:5000/alpine

**Image layers**

* Docker images are always downloaded in form of layers the same layer will not get downloaded multiple times that is if we download Ubuntu docker image
* Next we download Ubuntu with tomcat install on it
* One more Ubuntu with git install
* Another tomcat with web application deployed on it.’
* It will download them one time

18/09/2017

**Data volumes**

Whenever we exit from a container the data that has been created with in the container will be losses.

To preserve the data that has been created by container we can use volumes.

They are two types.

1. Data volume container
2. Data volumes container

**Note:** volumes can be mounted using the docker run command –v options.

**Use cases**:

**7.** **Run Ubuntu container with name myUbuntu... Mount it on a data volume called ‘data’ and create some files in this in this ‘data’ folder …check if the files remain intact after restart of the container.**

* $ docker container run –it –name myubuntu –v /data Ubuntu
* In the Ubuntu container.
* $ cd /data
* $ touch file{1..3}
* $ exit
* $ docker restart myubuntu
* $ docker attach myubuntu
* $ cd data
* We will find all the previously created files

**Data volume containers:**

The volume which is used be one container can be shared with other containers using data volumes containers.

**8. Create a Ubuntu container and name it container1 mount a volume called /data in this container create some files in this /data folder come out of the container without exiting share this container volume with container2 and share container2 volumes with container3.**

* $ docker container run –it –name container1 –v /data Ubuntu
* In the container cd to data volume
* Create files using $touch file{1..5}
* Come out without existing (ctrl+p+q)
* $docker container run –it –volumes-from container1 –name container2 Ubuntu
* $docker container run –it –volumes-from container2 –name container3 Ubuntu
* Stopping multiple container
* # docker stop $(docker ps –aq)
* Remove the containers
* # docker rm $(docker ps –aq)

**Linking’s**

**Multiple can take can be using –-link option**

**This will enable us to allow commutation b/w containers**

**9. Run a busy box container and name it source run another busy box container and name it target link target with source and check if we can ping b/w the containers**

* # docker container run –it –name source busybox
* Come out of container using ctrl+p+q
* # docker container run –it –link source:source\_alias –name target busybox. --
* **Here source is source container name**
* **Source\_alias is alias name**

**Note: the network establish using –link option will work only for communicating b/w containers it can’t be used for communicating for the host machine.**

**19/09/2017**

**Use case**

10) Run mysql container and name it intelliq-mysql run wordpress container and name it intelliq-wordpress and link to the my sql bd container

* # docker run –name intelliq-mysql –e MYSQL\_ROOT\_PASSWORD=password1 –d mysql
* # docker run –name intelliq-wordpress –link inteqlliq-mysql:mysql\_alias –p 8080:80 –d wordpress
* Open any browser and navigate to localhost:8080
* Create a wordpresss website.

**20/09/2017**

**Docker compose**

* This is tool of docker which is used for executing multiple docker commands from one point of control.
* This file is created using yamil.
* Yamil always takes the data in the format of key and values.

**---**

**intelliq:**

**- traniers:**

**- sai:**

**- devops: 12000**

**- selenium: 5000**

**- sheshi:**

**- devops: 12000**

**- aws: free**

**- receptionist**

**- shailaja**

**...**

To validated our yamil syntax <http://www.yamllint.com>

* To install docker compose on Linux
* <https://docs.docker.com/compose/install/#install-compose>’
* sudo curl -L https://github.com/docker/compose/releases/download/1.16.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose
* sudo chmod +x /usr/local/bin/docker-compose
* Docker-compose –version

21/09/2017

**Use case**

**1) Create a docker compose file for starting a word press container and mysql container**

* Create a file called docker-compose.yml
* # vim docker-compose.yml
* Go into insertion mode by pressing “I”

Version: ‘3’

Services:

Mysql:

Image: mysql

Environment:

MYSQL\_ROOT\_PASSWORD: mypassword

Wordpress:

Images: wordpress

Ports:

- 8080:80

* **Save and quit**
* **Escape : wq enter**
* **To run the above file**
* **# docker-compose up**
* **To stop the services using our compose file**
* **# docker-compose down**
* **# docker –compose –f filename up**

**Building docker images using docker file**

Docker file is a simple test file which uses specific commands using which it is possible to create our own images.

It uses the following keywords to create or modify images

* + FROM
  + MAINTAINER
  + CMD
  + ENTERYPOINT
  + RUN
  + COPY
  + EXPOSE
  + ADD
  + USER
  + VOLUME
  + WORKDIR
  + ENV
  + LABEL

To create images via docker file we should perform the below two steps

1. Create the docker file with above commands
2. Build an image using that file

**Use case**

1) Create a docker file using the base image Ubuntu and specify the name of the author

* # vim dockerfile
* Go into insert mode by pressing ‘i ‘

FROM ubuntu

MAINTAINER Sai

* Save and quit(:wq)

To build an image using the above t…..

# docker build –t newdocker .

Note: -t is used for specifying a name for our image. represents current working directories i.e it will build an image based on the docker file present in our working directory.

2) Create a docker file based on alpine Linux image and executes some Linux commands in it

# vim dockerfile

Go into insert mode by pressing ‘I’

FROM alpine

MAINTAINER Sai

CMD [“date”]

CMD [“ls”, “-la”]

:wq

# docker build –t newalpne .

**ENTEYPOINT**

This command is used for taking which comes from CMD as arguments

**Use case**

Create a docker file from busybox image set the ENTEYPOINT as cat command and open a file called /etc/passwd”.

# vim dockerfile

FROM busybox

MAINTAINER Sai

ENTERYPOINT [“/bin/cat”]

CMD [“/etc/passwd”]

# docker build -t newbusybox .

# docker run newbusybox

22/09/2017

**Use case**

1. Download Ubuntu image and then install git and maven ping and curl init.

Perform the above action thorough a docker files.

# vim dockerfile

FROM ubuntu

RUN apt-get update && apt-get install –y git \

Maven \

Oputils\* \

Curl \

:wq

# docker build –t myUbuntu .

Run the myUbuntu image created using the docker file

# docker run –it myUbuntu

# git –version

# mvn –version

2. Create 5 docker images using the same dockerfile that we have created in the previous use case.

* Create a shell script with the name ‘myscript.sh’
* # vim myscript.sh
* Go Into insert mode by pressing ‘I’
* For I in {1..5}
* do
* docker build –t myUbuntu$i .
* :wq
* Give execute permissions on the above shell script
* # chmod u+x myscript.sh
* Run the shell script using
* ./myscripts.sh

03/10/2017

**Docker Networking**

To see the list of networks available

# docker network ls

To create new network

# docker network create network\_name

To remove/delete a network

# docker network rm network\_name

To find the information about the network

# docker network inspect network\_name

To attach a container to a network

# docker network connect network\_name container\_name

To disconnect the network to containere

# docker network disconnect network\_name container\_name

**Use case:**

Create a network called intelliq1 another network called intelliq2

Create 3 busy box containers container1 container2 container3

Start container1 and container2 on intelliq1 network and check if they are pining not

Start container3 is intelliq2 network and check that it cannot ping to containere1 and container2

Now container1 should be able to communicate with container2 but cannot communicate with container3 similarly container3 should be able to communicate with container2 but not with container1

# docker network create intelliq1

# docker network create intelliq2

# docker run –itd –name container1 –network intelliq1 **busybox**

**# docker run –it –name container2 –network intelliq1 busybox in container2**

**Ping contaienr1: it should successfully ping**

**Ctrl+p+q to come out of container2**

**# docker run –it --name container3 –network intelliq2 busybox**

**In container3**

**Ping container1**

**Ping container2**

**It should not be able to ping because contaiener1 and containere2 are running intelliq1 network**

**# docker network connect intelliq2 container2**

**The above command will attach container2 to intelliq2 network…container2 is now running on both intelliq1 and intelliq2 network**

**Docker container attach container3**

**Ping container2: it will ping successfully**

**Ping container1: it cannot ping**

05/10/2017

**Configuration management**

**Ansible**

This is the process of systematically handling the changes to a system or a group of systems so that we can maintain integrity over a period of time

Using configuration management tools it is possible to automate the process of provisioning a server.

**Advantage of configuration management**

* Deployment of servers:
* Quick recovery from critical events:
* Handling snowflake servers:
* Replicating environment:
* Idempotent Behaviour:
* System facts:
* Templating:

1. Deployment of servers: multiple servers present at different geo graphical location can be provisioned from one point of controlled.
2. Quick recovery from critical events: in case a server goes down due to some unknown reason auditing has to be done it to find RCA[root cause analyse] this auditing can take hour at time using cm tools we can quickly deploy replacement server which temporarily behaves has the original servers
3. Handling snowflake servers: a single data centre has multiple servers and each server can be having different software’s or different version of the software’s maintaining 100 of severs with different service running on them becomes complicated if it is done manually cm tools can use simple automated programs for handling such scenarios
4. Replicating environment: cm tools make it very easy to create duplicate environment with the same set of software’s and configurations this helps us to effectively create a multi stage development, testing and production environment.
5. Idempotent Behaviour: cm tools always first check the stage of the server that it is going to configure. If the desired stage it already present cm tools will not disturb the server.

Ex: if there are 1000 servers on w which we want install git and already on 500 servers git has been installed cm tools will not disturb these 500 servers

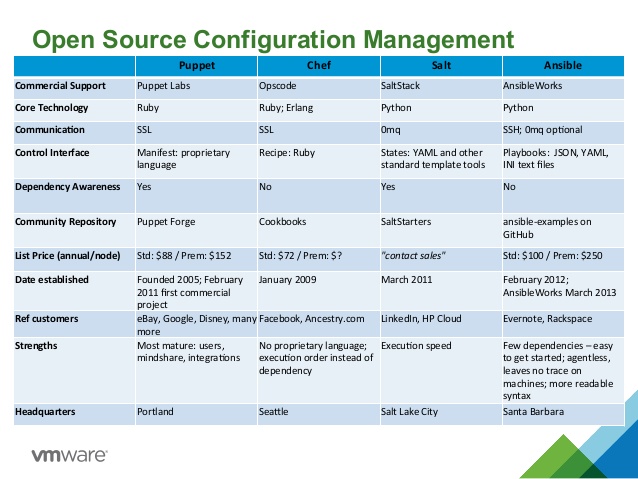
1. System facts: cm tools when automating any server capture that server’s configuration and store them in same variables these variables are called system facts. This data can be later we used for provisioning other servers with the same set of configuration
2. Templating: cm tools can create automation scripts which are reusable that is the same scripts can be used for installing git in one scenario and Jenkins in another scenario.

Popular CM tools

1. Chef
2. Puppet
3. Salt
4. Ansible

06/10/2017

Compression in b/w ansible puppet and chef



**Creating Infrastructure of ansible:**

1. Download and install oracle virtual box
2. Download and install vagrant
3. To check if vagrant is installed or not open cmd prompt
4. Vagrant –version
5. Copy the vagrant file into any empty folder
6. Open cmd prompt
7. Cd path\_of\_folder\_where\_we\_saved\_vagrant\_file
8. Vagrant up

Ansible

Ansible is an open source configuration management and orchestration utility.

It can automate the configuration of remote hosts are virtual machines

Using ansible it is possible to lunches or shutdown multi-tiered a

Due to this reason ansible is used for performing rolling updates with 0% downtime.

Instead of creating shell scripts for management remote servers ansible **plays a play is collection of task that should be performed on remote host a file which contains these plays are called as ansible playbooks.**

Ansible uses agent less architecture i.e. ansible pushes its tasks via Ssh so it does not require any agent to be installed on the manage hosts.

**Note:**

Ansible requires python to be installed on all the machines

**Controller:** This is the centre on which we install ansible

**Managed Hosts or managed Nodes:** these are the remote machines which will be controlled by ansible.

**Disadvantages of Ansible:**

1. It cannot perform installation of the basic o/s.
2. It cannot monitor the changes done on the remote host.

**Installing Ansible:**

1. Open oracle virtual box
2. Start the centre vm (username: vagrant, password: vagrant)
3. sudo apt-get update
4. $ sudo apt-get install software-properties-common
5. $ sudo apt-add-repository ppa:ansible/ansible
6. $ sudo apt-get update
7. $ sudo apt-get install ansible

07/10/2017

**Agile scrum**

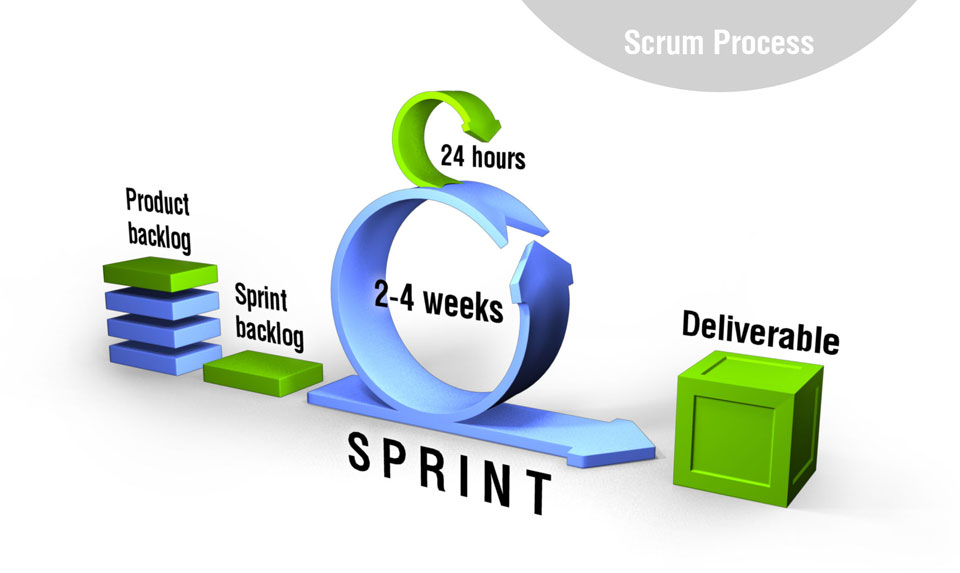
Scrum is an incremental and iterative model which promises more return on investments.

In scrum the complete software is divided into modules each module is developed, tested and delivered to the client with in a time span of 2 to 4 weeks.

This is called as sprint. At the end of the sprint the one module of s/w to client in swapped ready state. Then it is client can start using this module.

This sprint is farther divided into smaller sub sprints as small sprint as 24hours

Continuous monitoring of the client will be present on all the sprints.



**Scrum team**

**A typical scrum team contains the following people**

Product owner, business owner - 1

Scrum master – 1

System testers 1 - 2

Developer – 1 - 2

Testers – 1 – 2

Devops admin 1 - 2

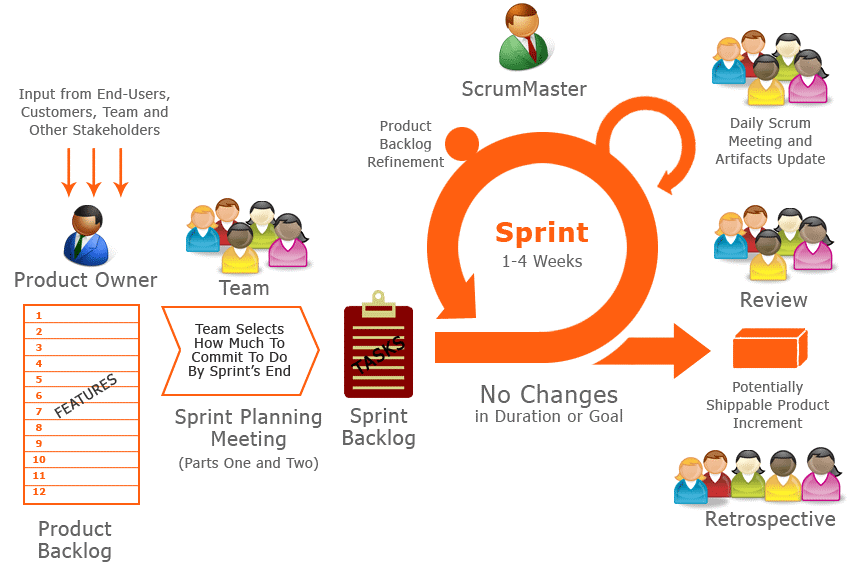
The entire scrum team discusser with product owner to understand his requirements and these requirements are created in the form of “user stories”.

The collection of all these user stories is called as product Backlog”

**Syntax of user story**

**As a [role] I want [features]**

**So, that [benefit to client]**



Once the product backlog is finalised the scrum team prioritizes the user stories.

Highest priority user stories will go with first sprint and so on…

This priority can be done in **MoSCoW** principle.

Mo – must have

S – Should have

Co – couldn’t have

W – Wouldn’t have

Any user story with Mo priority is considered as highest priority and should be delivered in first sprint

User stories with s priority go with second sprint

User stories with Co priority go with 3rd sprint

Wouldn’t assign to user stories which are eliminated

**Agile ceremonies**

1. Sprint plan meeting: this is the one day meeting conducted by scrum master and here work allocation is done from all team members.
2. Scrum meeting or standard : this is conducted every day where is scrum team members should discourse with product owner what work done on yesterday and what we work plan today suggestions from the product owner should be implemented
3. Sprint retrospective: this meeting conducted is end of every sprint and here scrum team members will analyse the drawback they faced in previous sprint and suggest solution overcome.

**Work estimation for a sprint:**

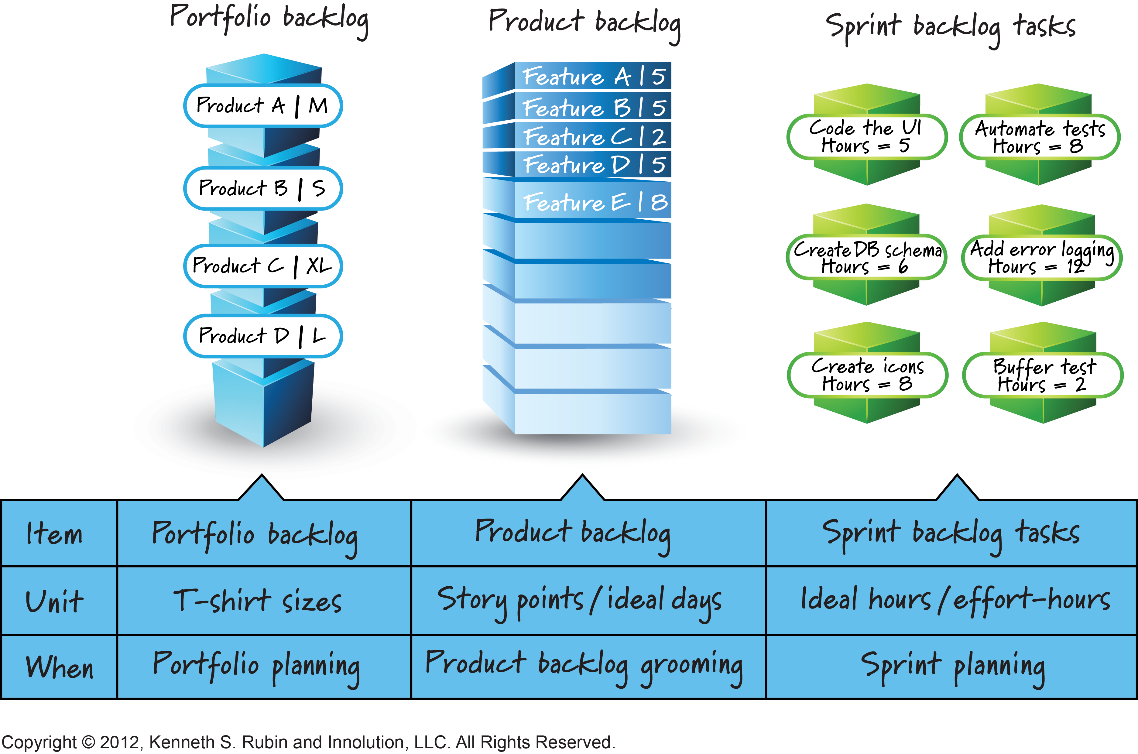
Work estimation is done using Fibonacci’s numbers

1, 1, 2, 3, 5, 8, 21 …

Each user story assigned a Fibonacci number the smaller Fibonacci number means user story in less complex and it require in amount of time.

Bigger Fibonacci number assigned for a complex number.

Any user story which has Fibonacci number greater than 20 is consider as very complex user story and it is spilt into small user stories.



Scrum Board: it is present in the conference room where scrum meeting are conducted. It contains 3 sessions

1. To do
2. In progress
3. done



Initially all the user stories posted in to do section in the form sticky notes as the sprint progresses it moves into in program and done section this gives transparency to all the scrum team members and they will clearly know how the sprint is progressing.

09/10/2017

Ansible uses ssh for perfuming remote access of the servers and it should be password less ssh

To do this we should generate ssh keys on the controller machine and copy the public key into the remote machines.

To generate ssh keys

# ssh-keygen

This will generate to files in /home/vagrant/.ssh

1. id\_rsa
2. id\_rsa.pub

The public key should be copied into the managed hosts

# ssh-copy-id username@ipadress\_of\_managed\_host

This will copy the content of id\_rsa.pub into the destination machine into a file “authorisedkeys”

Hence forth we can do # ssh username@ipaddress

Inventory file:

The remote hosts that are managed via ansible are specified in file called inventory file by default this inventory file is present in /etc/ansible/ folder

The name of the file is hosts.

/etc/ansible/hosts

Grouping of multiple servers can be done by giving the group name is square brackets.

[Webserver]

192.168.60.3

192.168.60.2

[Dbserver]

192.168.60.5

192.168.60.4

[Virginia:children]

Webserver

Dbserver

Children can be grouped based on group name’s for this we can use the keyword children

10/10/2017

***Modules in ansible***

Ansible performs remote configuration of servers by using in built modules. These have been created using python. Each module is deigns for performing a specific task

***Important modules:***

**Command**: This is the module of ansible and it is used for executing Linux commands on the managed nodes.

**Shell module:** This is used to execute shell Linux shell based commands [eg: redirection and piping)

**Copy:** this is used to copy files or directories into managed nodes

**Fetch:** this is used to capture files from managed nodes into control node

**File:** this is used performing file retailed operations on the managed nodes

**apt:** this is used for packaged management on Ubuntu based machines

**yum:** this is used for packaged management on Redhat Linux, centos, fedora, oracle enterprise Linux etc…

**service:** this is used for starting and stopping service on remote machine

**debug:** this performs remote debugging on managed nodes

**ping:** this is used for checking whether the managed nodes or pin gable node

**url:** to check whether remote url access or not

To find the list of all the modules that are available.

$ ansible-doc –l [ it will display the all the modules]

$ ansible-doc copy

**Syntax:**

$ ansible ipadress/group\_name/all –m module –a ‘arguments’

$ ansible 192.168.60.2 –m module –a ‘ls’

**Command module:**

Ansible command to find the list all the files present on remote machine.

$ ansible 192.168.60.2 –m command –a ‘ls –a’

$ ansible groupmate –m command –a ‘ls –a’

$ ansible all –m command –a ‘ls –a’

$ ansible 192.168.60.2 –a ‘ls –a’

In this module whether we give –m are not it will give the output.

**Shell module:**

Redirection and piping related commands can be executed shell module

To copy the output of ls –l in to file1 on remote machine

$ ansible 192.168.60.2 –m shell –a ‘ls –l > file1’

11/10/2017

12/10/2017

**Shell module:**

This module is used for executing shell based commands the commands which generally involves redirection and piping.

Ansible command to capture the output of ls –la command into file1 on all the manage nodes

$ ansible all –m shell –a ‘ls –la > file1’

To redirect the error massage of data command into file1

$ ansible all –m shell –a ‘data 2> file1’

To redirect both error and output to file on the managed node

$ ansible all –m shell –a ‘(data;ls) &> file1

Command capture the 5th line from the /etc/passwd and store it file1 on the manage node

$ ansible all –m shell –a ‘head –d /etc/passwd | tail –l > file1’

Use case

Carputer all the sentence which have word bin b/w line number 5 to 15 from /etc/passwd and store it in file1 on the manage nodes

$ ansible all –m shell –a ‘head -15 /etc/passwd | tail -1 > file1 | grep bin > file1

**Package management**

This can be done using apt module and yum module

**apt module :**

This is used for package management on Ubuntu Debain etc…….

13/10/2017

To install git on all the manage nodes

$ ansible all –m apt –a ‘name=git state=present’ –b

To update the apt repository and then install git

$ ansible all –m apt –a ‘name=git update\_cache=yes state=present’ –b

To uninstall git from all the manage nodes we can give state=absent

$ ansible all –m apt –a ‘name=git state=absent’ –b [-b is called as become it give all the privileges]

I would like to install the all the manage nodes

$ ansible all –m apt –a ‘name=apache2 update\_cache=yes state=present’ –b

**Copy module:**

This is used for copying files from the control machine to managed nodes

To copy file1 from controller machine to all managed nodes

$ ansible all –m copy –a ‘src=file1 dest=/home/vagrant’

To copy entire dir along with sub directories and files

$ ansible all –m copy –a ‘sro=dir1 dest=/home/vagrant’

To copy a file to all the manage nodes and change the ownership and group ownership to root on the manage nodes

$ ansible all –m copy –a sro=file1 dest=/home/vagrant owner=root group=root’ –b

To change the permission of the file that is copped into the managed nodes we can use mode options

$ ansible all –m copy –a ‘sro=file1 dest=/home/vagrant mode=777’ –b

To directly send some data into file we can use content argument

$ ansible all –m copy –a ‘content=Ansible session\n is done\n dest=/home/vagrant/file3’ –b

16/10/2017

Using copy module we can copy a file present on managed node to some other location on the same manged node using on argument remote\_src

$ ansible all –m copy –a ‘remote\_src=yes src=/hme/vagrant/file1 dest=/temp/file1’ –b

**File module:**

This used for creating files are directories on the manage nodes ansible command to create file on the manage nodes

$ ansible all –m file –a ‘name=file1213 state=touch’

We can also create directories and subdirectories in it

$ ansible all –m file –a ‘ name=d1/d2/d3 state=directory’

Command to create a file using file module give only execute permission and change the ownership to root, groupship to root

$ ansible all –m file –a ‘name=file23 state=touch mode=111 owner=root group=root’ –b

$ ansible all –m file –a ‘name=folder1/folder2 state=directory mode=111 owner=root group=root’ -b

To recursively change the permissions for folder and sub folders we can recurs=yes

$ ansible all –m file –a ‘name=folder1/folder2/folder3 state=directory mode=111recurse=yes’ –b

**User module:**

This is the use for the creating user accounts on manage nodes it is also used for setting the passwords specifying the default home directory default shell etc…..

Create user called shanu and assign the password shanu on all the manage nodes

$ ansible all –m user –a ‘name=shanu password=shanu’ –b

Create a user called ram assign the password and assign the default home directory and shell

$ ansible all –m user –a ‘name=ram password=intelliq home=/home/vagrant/ram shell=/bin/bash’ –b

17/10/2017

**Services module:**

This is used for starting or stopping services

Command to start apache2 on all the manage nodes

$ ansible all –m service –a ‘name=apache2 state=present’ –b

Ansible command to install apache2 on all manage node and stop the apache2 service.

$ ansible all -m service –a ‘name=apache2 state=stopped’ -b

**URI module:**

This is used to check if a particular url is accessible from a manage a node if it is accessible it will return the status has 200 otherwise it will return the status has -1 or 040

To check whether google is accessible form the all manage nodes

$ ansible all –m uri –a ‘url=”<http://google.com>” status=200’

Whenever we configure apache2 in /var folder it will create the sub folder called www within which there is another folder html this folder contains file index.html this is the basic present in apache2 to check whether this page is reachable or not

$ ansible 192.168.60.101-m uri –a ‘url=”hhtp://192.168.60.101” status=200’

**Setup module**

This is used to capture system facts that is system information.

$ ansible all –m setup

To find information about one variable we can give that variable name as on argument

Ext; to find the kernel on remote machine is running

$ ansible 192.168.60.101 –m steup –a ‘filter=facter\_kernel’

**Fetch module:**

To fetch a file from manage nodes into the controller machine we can use fetch module.

$ ansible all –m fetch –a ‘src=/home/vagrant/file1 dest=/home/vagrant’

**Ansible playbook:**

The ansible commands that we have execute till now are called has ad-hoc commands and they are useful for only performing single operation and they can work only on a single setup arguments they cannot be used for complex configuration activities.

Ad-hoc commands can execute only one module at time to work on multiple modules we can use playbooks.

Playbooks are created in yamil format

Playbooks are powerful and flexible for performing cm

Using playbooks we can change lengthy and complex at administrative activities in to repeatable routines

Ansible playbooks are combination of plays each play defines set of operation that should be performed on manage nodes these operation are called as tasks and managed nodes called as hosts.

Each task execute specific modules the modules are executed in the order

In which we present in the playbooks.

**Use case1:**

Create an ansible playbook for creating dir on all manage nodes and copying a file from the controller machine into the newly create dir on the manage nodes

Note: to execute this with ansible ad-hoc commands we should execute the following two commands

$ ansible all –m file –a ‘name=mydir state=directory’

$ ansible all -m copy –a ‘src=/home/vagrant/myfile dest=/home/vagrant/mydir’

The alternative these two commands I will go with playbook

---

- name: For cratering directories and copying files

hosts: all

tasks:

- name: creating directory

file:

name: mydir

state: directory

- name: copying files

copy:

src: /home/vagrant/myfile

dest: /home/vagrant/myfile

…

$ ansible-playbook –-syntax—check

21/10/2017

Use case3:

Create an ansible playbook for going to the manage nodes and fetching the all the users who are using /bin/bash shell

---

- name: for capturing user eiht /bin/bash/ shell

hosts: all

tasks:

- name: Capturing info from /etc/password

shell: grep /bin/bash /etc/passwd > file1

- name: Fethcing files

fetch:

src: /home/vagrant/file1

dest: /home/vagrant

....

Use case4:

Create an ansible playbook for creating users on all the manage nodes capture there’s users names and home directories.

---

- name: creating users and capturing username and home dir

hosts: all

become: yes

tasks:

- name: creating users

users:

name: intelliq

home: /home/vagrant/intelliq

- name: capturing user info

shell: grep intelliq /etc/passwd | cut -d ":" -f 1,6 > file1

- name: Fetching files

fetch:

src: /home/vagrant/file1

dest: /home/vagrant

....

Use case:

Install apche2 on one manage node start apache2 service and check the url response.

---

- name: creating users and capturing username and home dir

hosts: 192.168.60.2

become: yes

tasks:

- name: update apt repo and install apache2

apt:

name: apache2

state: present

update\_cache: yes

- name: starting apache service

service:

name: apache2

state: started

- name: checking url response

uri:

url: http://192.168.60.101

status: 200

...

23/10/2017

Create a ansible playbook installing apache2 one more mcahine

Note: the playbooks we have created till now are working only on group hosts that is the entire paly is getting executed on that group of hosts. To perform different activity’s on different group hosts we can create multiple plays in playbook.

---

- name: installing apache2

hosts: 192.168.60.1

tasks:

- name: installing apache2

apt:

name: apache2

state: present

- name: installing git

hosts: 192.168.60.2

tasks:

- name: installing git

apt:

name: git

state: present

...

**Variables and ansible:**

These are classified into 3 types

1) Global scope: these variables are defined from the command prompt and they can affect the complete playbook.

2) Host scope: these variables are defied with respect the hosts and it can affect all the plays with in the

3) Play scope: These are defined at the level of induvial play and they can effect only that particular play.

Create an ansible playbook which can be used either for installing or uninstalling packages similarly it should also be used either for updating or non-updating the apt repository.

---

- name: installing/uninstalling packages

hosts: all

tasks:

- name: installing/uninstalling packages

apt:

name: "{{a}}"

state: "{{b}}"

update\_chache="{{c}}"

...

To run this playbook command prompt for installing maven without updating the repository:

$ ansible-playbook playbook12.yml --extra-vars “a=maven b=present c=no” -b

Similarly we can use the same the playbook for uninstalling git after updating apt repository.

$ansible-playbook playbook12.yml – extra-vars “a=git b=absent c=yes” –b

Usecase:

Create an ansible playbook for creating files or directories on the manage nodes and also for controlling the ownership groupowenership and permissions.

---

- name: creating files and directories

hosts: all

tasks: file creating/ dir creation and controlling ownership

file:

name: "{{a}}"

state: "{{b}}"

owner: "{{c}}"

group: "{{d}}"

mode: "{{e}}"

...

---

- name: creating files and directories

hosts: all

tasks: file creating/ dir creation and controlling ownership

file:

name: "{{name}}"

state: "{{state}}"

owner: "{{owner}}"

group: "{{group}}"

mode: "{{mode}}"

...

$ ansible-playbook playbook13.yml - - extra-vars “a=file1 b=touch c=root d=sai e=111” -b

$ ansible-playbook playbook13.yml - - extra-vars “name=dir1 state=directory owner=vagrant group=sai mode=777” -b

using this play book we can create directory and controller the ownership and group ownership

Note: global scope have the highest presidency compare host scope and play scope.

24/10/2017

**Play scope:**

These variables can effect only that particular play and they should be defined with in the playbook

**Use case:**

Create an ansible playbook for installing apache2 on all manage nodes after updating the apt repository

---

- name: package installation/uninstallation

hosts: all

vars:

- a: apache2

- b: present

- c: yes

tasks:

- name: installing/uninstallation

apt:

name: "{{a}}"

state: "{{b}}"

update\_chache: "{{c}}"

...

The above playbook can we template

Using which we can install r uninstall by passing global scope variables from the command prompt.

Since global scope variables have more priority then local scope variables the playbook work on global scope variables.

Run above playbook for installing the git.

$ ansible-playbook playbook16.yml –extra-var “a=git b=absent c=no” –b

Host scope variables:

These are character into 2 types

Based on groups

Based on induvial hosts.

For creating group based variables we should first create a folder group\_var

In the same folder we the palybooks are present

Mkdir group\_vars

Change directory into this folder

Cd group\_vars

Create a files

Whos name is same as that of the groups present in ansible inventory file

Defined the variables in this files

Cat > webserver

User: ravi

Password: intelliq

Home: /home/vagrant/ravi

Shell: /bin/bash

Change the directory to the folder where the playbooks are present

Which will create a user’s on webserver group assign home dir etc…..

---

- name: create users

hosts: webserver

vars:

tasks:

- name: create user and assign homedir, passwd etc

apt:

name: "{{user}}"

passwrod: "{{password}}"

home: "{{home}}"

shell: "{{shell}}"

...