DevOps UNIT-V

Introduction

* Testing is very important for software quality

* We will Concern ourselves with the following topics:

- How to make manual testing easier and less error Prone.

- Various types of testing, Such as unit testing, how to perform them in practice.

- Automated System integration testing.

* Manual testing will always be an important part of Software development.

* Test automation has larger be an important part of Software development.

* Each Organization is different, so it is not possible to give generally useful advice in this area other than the KISS rule: "keep it simple, stupid".

Automation of testing Pros and cons:

* Software testing is Completely necessary for aprogram to work reliably in the real world.

* Manual testing is too slow to achieve Continuous Delivery.

* So, we need test automation to achieve Continuous Delivery.

* The following are Some of the areas, need focus to improve situations:

- cheap tests have lower value.
- It is diffcult to Create text cradles that are relevant to automated integration testing
- The functionality of programs very over time and tests must be adjusted accordingly, which takes time are effort.
- It is diffcult to write robust tests that work reliably in many different build scenarios.
- It is just hard to write good automated tests.

Selenium - Introduction:

* Selenium is one of the most widely used open Source web UI automation testing Suite.

* It was originally developed by Jason Huggings in 2004 as an interval tool at ThoughtWorks.

* Selenium Supports automation across different browsers, platforms and programming language.

*Selenium Supports a variety of programming languages through the use of drivers specific to each languages.

* Selenium web driver is most popular with Java. and C#.

*Selenium Can be used to automate functional test and can be integrated with automation tools like. Tenkins and Docker to achieve Continuous testing.

Selenium Features:

*Selenium is an open Source. and portable web testing Framework.

* Selenium IDE provides a playback and record feature for authoring tests.

*If Can be Considered as the leading cloud-based testing platform.

*Selenium Supports various operating systems, browsers and programming languages.

Operating Systems: windows, Linux, Mac, Android, ios...

Browsers: Google chrome, mozilla ~~For~~ Firefox, Internet Explorer, Opera, Safari

Programming Languages: C++, Java, Python, PHP, Ruby, Perl Javascript.

* It also Supports parallel test execution which reduces time and efficiency of tests

* Selenium requires fewer resources as Compared to Other automation test tools.

* Selenium Supports Functional and Regression testing of web applications.

JavaScript testing:

* JavaScript testing framework is used to test web applications:

- Karma is a test runner for unit tests in the JavaScript language.

- Jasmine is a Cucumber like behaviour testing framework

- protractor is used for AngularJS.

* Protractor is similar Selenium in Scope but optimized for AngularJs.

* It uses the Selenium web driver implementation under the hood.

* You Can use JavaScript for writing test Cases for Selenium.

Testing backend integration Points:

* Automated testing of backend functionality such as SOAP and REST endpoints is normally quite Cost effective.

* The tests are easy to write the tools suchas SoapUI, which Can be used to write and execute tests.

* These tests Can also be run from the Command line and with Maven.

* The SoapUI is agood example of a tool that appears to Several different roles.

* Testers who builds test Cases get a fairly well-structured environment for writing tests and running than interactively.

* Developers Can integrate test Cases in their builds without necessarily using the GUI. There are Maven. plugins and Command-line runners.

* The SoapUI User Interface provides, atree view listing test Cases on the test. It is possible to Select single test or entire test Suiter and run them..

* The results are presented in the area on the right.

* Test Cases are defined in XML. This makes it possible to manage them as Code in the Source Code repository.

~~* Test Cases~~

Test-driven development:

* Test-drive development (TDD) has an added focus on test automation.

* TDD is usually described as asequence ofevents as follows:

- Implement the test:

* → First write the test and write the Code afterwards

→ To be able to write the test, the developer must find all relevent requirements specifications, Use Cases and user stones.

- Verify that the new test fails:

→ The newly added test should fail because there is nothing is implement the behaviour properly yet

- Write Code that implements the tested feature:

→ The Code we write does not yet have to be particularly elegant or efficient.

→ Initially, we just want to make the new best pass.

- Verify that the new test passes together with the Old tests:

→ When the new test passes, We know that we have implemented the new feature Correctly.

→ Since the old tests also pass we haven't broken existing functionality.

- Refactor the Code:

→ The word "refactor" meaning in programming is Cleaning up the Code and make it easier to understand and maintain.

→ We need to refactor since we cheated abit earlier in the development

*TDD is a style of development that fits well with DevOps.

REPL - driven development:

* REPL - driven development ~~is very Common when~~ isn't a widely recognised term.

* This style of development is very Common when working with interpreted languages, Such as Lisp, Python, Ruby and JavaScript.

* When you work with REPL(Read Evaluate Print Loop) you write Small functions that are independent.

* This style of development differs a bit from TDD.

* The focus is on writing Small functions with no Or very few side effects.

* You Can Combine this style of development with Unit testing

Why are the So many deployment System ?

* There are many options regarding the installed of packages and Configuring them on actual Servers.

* Let us first examine the basic of the problem we are trying to solve

* We have a typical enterprise application, with a number of different high-level Components

* In our Scenario, we have:

- web Server

- An Application Server

- A Database Server.

* If we only have a single physical Server physical server and these few Components to worry about that get released once ayear orso.

* It is more likely that alarge organization has hundreds of servers and applications and that they are all deployed differently with different requirement

* Managing all the Complexity that the real world displays is hard, So it starts to make Sense that there are alot of different Solutions.

* The following are the challeges to deal with:

- Configuring the base os.

- the popular method today is to provide bas operating Sytem images that Can be reused between machines.

- when you ask to cloud system for a new virtual machine, it is Created using an existing image asabase.

* ~~Cont~~

- Container Systems Such as Docker also workin asimilar way

• Describing clusters

- puppet allows machines to have different roles.

- Ansible and Salt have Same as well

- cloud System Such as AWS also have methods and descriptors for cluster deployments

• Delivering packages to asystem.

- much of an application Can be installed as Packages, which are installed uncodified on the target system by the Configuration management System.

- package delivery is usually done with operating System facilities, but Sometimes by Configuration management System.

<u>Virtualization</u> <u>Stacks</u>:

* You Can use Virtualization techniques to simulate entirely different hardware than the One you have physically

*This is Commonly referred to ~~as~~ an emulation.

Executing Code on the Client:

* Several of the Configuration management System describe here allow you to reuse the node descriptions to execute Code on matching nodes.

*In the puppet ecosystem, this Command execution System is Called marionette Collective or MCollective for short.

* It is easy to tryout the various deployment systems using Docker.

*We will first try each of the different deployment System that are usually possible in the local deployment nodes.

* keep in mind that actually deployment System in Production will require attention to security and other details than what we discussed here.

Kubernetes:

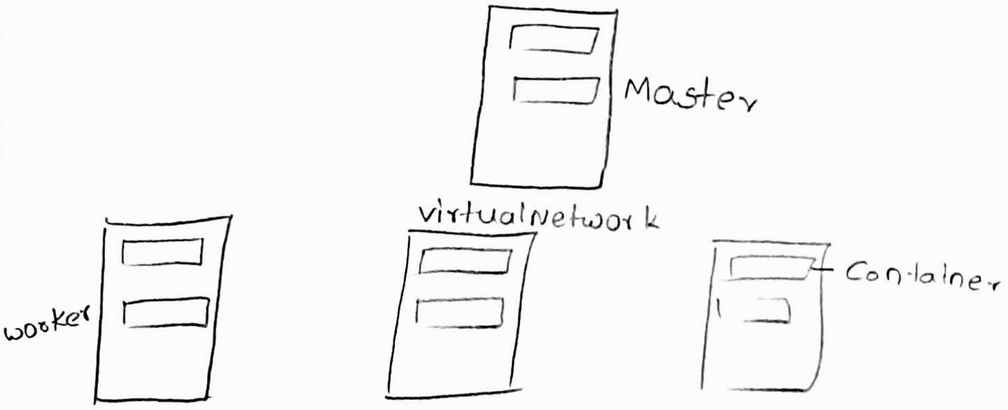
* It is an open Service " Container Orchestration tool", developed by Google.

*It helps you to manage Containerized applications in different deployment environments.

*The following are the main features of kubernetes:

- High Availability (No downtime)
- Scalability (High performance)
- Disaster Recovery (Backup and restore)

*Kubernetes Basic Architecture

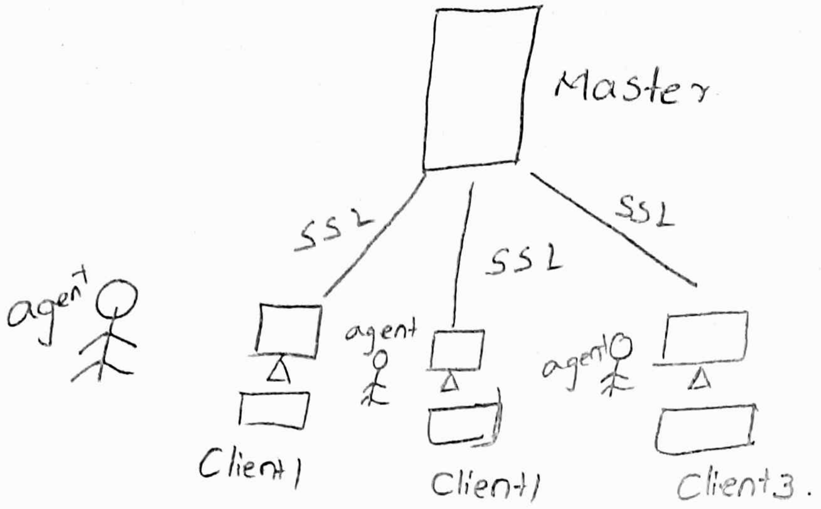


The Puppet master and puppet agents:

* puppet is a deployment solution that is very popular in larger organizations and is one of the first systems of this kind.

* puppet Consists of a client/server Solution, where the client nodes check in regularly with the puppet Server to See if anything need to be updated in the local Configuration.

* puppet Architecture:



* Master and client are Communicated through SSL CSecure Sockets layer)

Ansible:

* Ansible is a simple deployment solution.

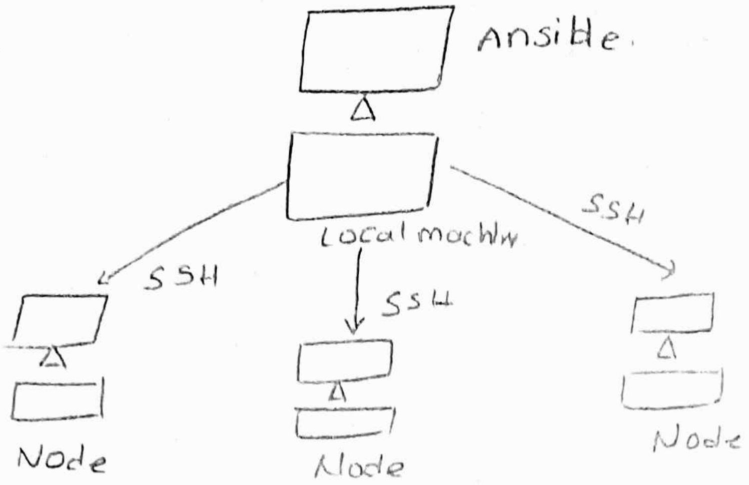
* The Ansible architecture is agentless.

* The Ansible Server logs into the Ansible node and issues Commands over SSH CSecure SHell) in order to install the required Configuration.

* The Core of Ansible, playbooks are written in YAMLCYet Another Markup Language)

* Ansible works well with environments where the focus is on getting the Servers up and running quickly

* Ansible Architecture:



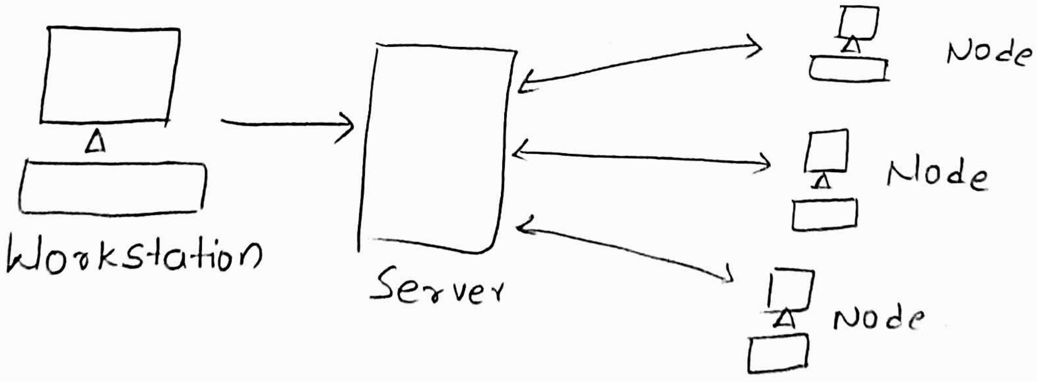
* performance Speed is often less than other tools.

Chef:

* chef is a Ruby-based deployment System from OpsCode.

* It is best Suited for organizations that have a heterogenous infrastructure and are looking for mature Solutions.

* Chef Architecture



*chef integrates well with Git which provides astrong Version Control.

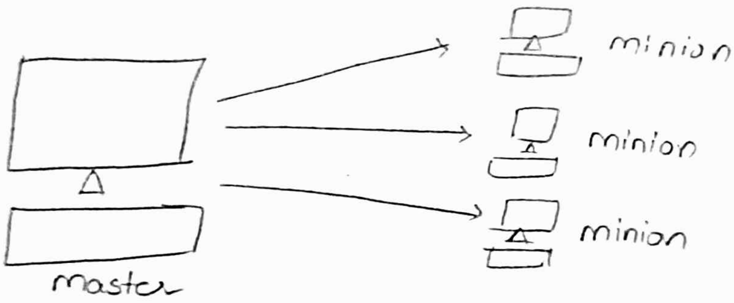
*A Considerable amount of learning time is required if One is not Comfortable with Ruby

SaltStack:

* SaltStack is apython-based deployment ~~slo~~ solution.

*SaltStack is perfect for an environment with scalability and resilience as its priority.

* Salt stack Architecture



*Agood reporting mechanism that allows one to easily view all operations

*Setup phase is slightly tougher.

Differences between chef, Puppet, Ansible and Salt stack.

Tool metrics	Chef	puppet	Ansible	Saltstack
Architecture	Client/Server	Client /Server	Client only	Client/Server
Ease of Setup	~~pro~~ moderate	moderate	Easy	moderate.
management	moderate	moderate	Easy	Easy
Scalability	Scalable	Scalable	Scalable	Scalable