**UNIT** **-** **V**

# Testing Tools and automation:

* Various types of testing,
* Automation of testing Pros and cons,

# Selenium

* Introduction,
* Selenium features,
* JavaScript testing,
* Testing backend integration points,
* Test-driven development,
* REPL-driven development

# Deployment of the system:

* Deployment systems,
* Virtualization stacks,
* code execution at the client,
* Puppet master and agents,
* Ansible,

# Deployment tools:

-Chef, Salt Stack and Docker.

# Testing Tools and automation:

Software testing is a software engineering activity to check whether the actual results of the software being developed matches with those of the expected results and to ensure that the software system is defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. Software testing is not a new field and it has appeared many years earlier, in software, we need to create a good and efficient software product or services, but this can be difficult because software must be tested before by stakeholders. A simple definition of software testing is the process of investigating software to check if it satisfies the requirement and detects errors that can happen in any software. “Another definition of software testing is the process of testing, verifying and validating the user's requirements”

Testing in a DevOps environment looks like:

* Testing is a continuous and automated process that enables continuous and faster delivery of software.
* Testing spans every stage of the software development lifecycle (SDLC).
* Each step of the SDLC involves different forms of testing. This minimizes backtracking in the case that you've detected an error.
* Testing is no longer the responsibility of one particular team. Shared testing responsibilities allow everyone to understand the impacts behind each change.

# Various types of testing

The complexity of software testing in the DevOps world of fast-paced deployment demands flexible thinking about deployment of test resources. Risk reduction is paramount and crowd testing can be a critical component in that effort. Testing may include:

* Integration testing

# Functional testing

* API testing
* Exploratory testing
* Regression testing

# Compatibility testing across platforms

* Security testing
* Acceptance testing
* Deployment testing

# Automation of testing Pros and cons

Automated testing is a powerful tool that can help you and your team with various tasks, including writing better code to simplify the process of regression. Unfortunately, automated testing can be misunderstood by some developers who don't see any value in it.

Automated testing is a method in which software tests and other sets of repeatable tasks can be performed without human interaction. Furthermore, these tests can run frequently to ensure that your application performs as expected continuously. This happens typically whenever the source code is updated.

Many people tend to confuse automated testing with automatic (or robotic) testing, a form of automated testing that uses automation tools to execute tests without human intervention. In this article, however, we will be focusing on the more common definition of automated testing.

# The pros of automated testing

1. **Increased** **accuracy**

One of the main benefits of automated testing is that it can increase accuracy. Indeed, automated testing is less likely to be affected by human error.

When tests are automated, they run more frequently and with greater consistency than when running tests manually. This can be beneficial when dealing with a large codebase or when new features are added. In addition, automation testing helps ensure that any errors or defects in the code are identified and fixed as quickly as possible.

# Faster execution

Automated testing can also lead to faster execution of tests. This is because the tests will run concurrently instead of serially. Running tests concurrently means more tests run in a shorter amount of time.

# Reduced costs

Automated testing can also lead to reduced costs. When tests are automated, the need for manual testers is reduced. In addition, the time needed to execute tests is reduced, leading to savings in terms of both time and money.

Moreover, automated tests can help reduce the cost of software development by detecting and fixing errors earlier in the process. They can also help reduce the cost of supporting your application, as automated tests will need less time to find and fix bugs.

# More trustworthy results

Another benefit of automated testing is that it can lead to more reliable results. This comes as a result of the fact that tests are run automatically and with greater frequency. Automated software testing helps you quickly identify any issues or regressions on your application, making it easier for you and your team to address these problems as soon as they arise.

# Increased efficiency

Automated testing can help improve developer productivity by automating tasks that would otherwise have to be done manually.

For example, you can configure your continuous integration (CI) system to automatically execute and monitor the results of your automated tests each time a new feature or change is introduced into your application. This will help ensure that any issues in the recent changes are identified and fixed as quickly as possible.

# a. Increased collaboration between developers

Automated testing can help to improve collaboration between developers. When you have a suite of computerized tests, other developers on your team can rely on them when implementing new changes or features. This ensures that a high level of code coverage is in place and reduces the likelihood of bugs in newly added code.

# F. Improved scalability

Automated tests can be used on many devices and configurations, making it easier to test more things at once.

For example, automated tests can be written to measure the performance of your application on different devices or browsers. This allows you to more easily test the different variations in which your application is being served and ensure that these are running as expected across a variety of end-user devices.

# The cons of automated testing

1. **Complexity**

Automated tests can take longer to develop than manual tests, especially if they are not well designed. They can also be more challenging to implement into your development workflow.

If your tests are complex or hard to maintain, it could lead to a reduction in the quality of your test suite. This can have negative consequences for achieving continuous testing throughout the application lifecycle.

That is why we developed a scripting language that is close to natural language for our automated testing tool, UIlicious.

# High initial costs

One of the main drawbacks of automated testing is that it initially takes a significant amount of time and money to implement. However, this investment can often be recouped very quickly in terms of improved developer productivity and more trustworthy results.

Moreover, UIlicious offers affordable premium plans that anyone can afford. Our objective is to make automation available to the most people on the web!

# It needs to be rewritten for every new environment.

When you make a change in one environment, your automated tests will need to be updated in order for the results to pass. Unfortunately, this means that you will have to rewrite your automated test scripts in many different locations in your local development environment, CI system, and production environments to ensure that they work as expected.

This is the reason why we made UIlicious able to recognize web page elements based on their labels, not only on their XPath or CSS. With our automation tool, you can change your code as you want, if the user flow has not changed, you will not need to adapt your test scripts.

# Generates false positives and negatives

Automated tests can sometimes fail even when there is no actual issue present. For example, this can be the case if the test contains an error or is not comprehensive enough to cover all of its intended use cases. Similarly, your tests may generate false negatives if they are designed only to verify that something exists and not that it works as expected.

# Difficult to design tests that are both reliable and maintainable

Designing a comprehensive suite of automated tests is no small task. They need to be reliable enough that they can be run frequently and consistently without giving you false positives or negatives. On the other hand, your test scripts must be maintainable enough to adapt to changes in your application. This requires a high level of developer expertise and careful design and implementation.

# Cannot be used on GUI elements (e.g., graphics, sound files}

While automated tests can be used to test most functionality of your application, they are not suited to testing things like graphics or sound files. This is because computerized tests typically use textual descriptions to verify the output. Therefore, if you try using an automated test on a graphic or audio file, it will likely fail every time, even if the content appears correct.

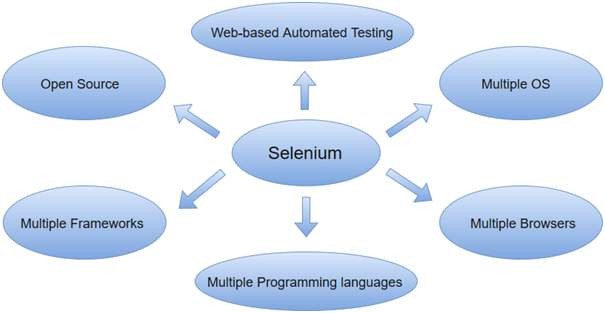
# Selenium

**-** **Introduction**

Selenium is one of the most widely used open source Web UI (User Interface) automation testing suite.It was originally developed by Jason Huggins in n004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android.

Selenium supports a variety of programming languages through the use of drivers specific to each language.Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby.Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers. Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari.



Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports.

# Selenium features,

* Selenium is an open source and portable Web testing Framework.
* Selenium IDE provides a playback and record feature for authoring tests without the need to learn a test scripting language.
* It can be considered as the leading cloud-based testing platform which helps testers to record their actions and export them as a reusable script with a simple-to-understand and easy-to-use interface.
* Selenium supports various operating systems, browsers and programming languages. Following is the list:
* Programming Languages: C#, Java, Python, PHP, Ruby, Perl, and JavaScript
* Operating Systems: Android, iOS, Windows, Linux, Mac, Solaris.
* Browsers: Google Chrome, Mozilla Firefox, Internet Explorer, Edge, Opera, Safari, etc.
* It also supports parallel test execution which reduces time and increases the efficiency of tests.
* Selenium can be integrated with frameworks like Ant and Maven for source code compilation.
* Selenium can also be integrated with testing frameworks like TestNG for application testing and generating reports.
* Selenium requires fewer resources as compared to other automation test tools.
* WebDriver API has been indulged in selenium whichis one of the most important modifications done to selenium.
* Selenium web driver does not require server installation, test scripts interact directly with the browser.
* Selenium commands are categorized in terms of different classes which make it easier to understand and implement.
* Selenium Remote Control (RC) in conjunction with WebDriver API is known as Selenium n.0. This version was built to support the vibrant web pages and Ajax.

# JavaScript testing

JavaScript Unit Testing is a method in which JavaScript test code is written for a web page or application module.

It is then combined with HTML as an inline event handler and executed in the browser to test if all functionalities work as desired. These unit tests are then organized in the test suite.

The following JavaScript Testing Frameworks are helpful for unit testing in JavaScript. They are as follows:

# Unit.js

An assertion library for Javascript runs on Node.js and the browser. It works with any test runner and unit testing framework like Mocha, Jasmine, 》arma, protractor (EnE test framework for Angular apps), QUnit, etc.

# Mocha

Mocha is a test framework running both in Node.js and in the browser. This framework makes asynchronous testing simple by running tests serially. Mocha tests run serially, allowing for flexible and accurate reporting while mapping uncaught exceptions to the correct test case. It provides support for all browsers, including the headless Chrome library and is convenient for the developers to write test cases

# Jest

It is an open-source testing framework built on JavaScript, designed majorly to work with React and React Native-based web applications. Often, unit tests are not very useful when run on the front end of any software. This is mostly because unit tests for the front end require extensive, time-consuming configuration. This complexity can be reduced to a great extent with the Jest framework.

# Jasmine

Jasmine is a popular JavaScript behavior-driven development framework for unit testing JavaScript applications. It provides utilities that run automated tests for both synchronous and asynchronous code. It is also highly beneficial for front- end testing.

# Karma

》arma is a node-based test tool allowing you to test your JavaScript codes across multiple browsers. It makes test-driven development fast, fun, and easy and is termed as a test-runner technically.

# a. Cypress

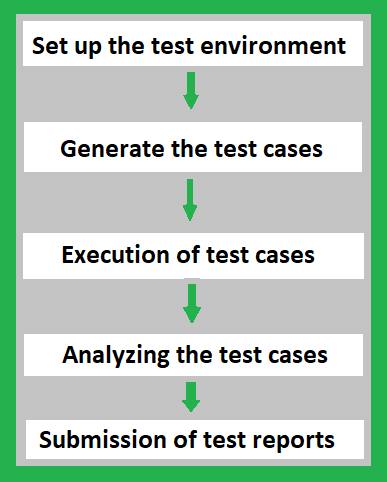
Cypress framework is a JavaScript-based end-to-end testing framework built on top of Mocha — a feature-rich JavaScript test framework running on and in the browser, making asynchronous testing simple and convenient Unit tests in Cypress are executed without even having to run a web server. That makes Cypress the ideal tool for testing a JS/TS library meant to be used in the browser.

# F. NightwatchJS

Nightwatch.js framework is a Selenium-based test automation framework written in Node.js and uses the W3C WebDriver API (formerly Selenium WebDriver). It communicates over a restful HTTP API with a WebDriver server (such as ChromeDriver or Selenium Server). The protocol is defined by the W3C WebDriver spec, which is derived from the JSON Wire protocol.

# - Testing backend integration points

**What** **is** **Backend** **Testing?**

****

Backend Testing is a testing method that checks the database or server-side of the web application. The main purpose of backend testing is to check the application layer and the database layer. It will find an error or bug in the database or server- side.

For implementing backend testing, the backend test engineer should also have some knowledge about that particular server-side or database language. It is also known as Database Testing.

# Importance of Backend Testing:

Backend testing is a must because anything wrong or error happens at the server-side, it will not further proceed with that task or the output will get differed or sometimes it will also cause problems such as data loss, deadlock, etc.,

# Integration Testing

Integration Testing is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as ‘I & T' (Integration and Testing), ‘String Testing' and sometimes ‘Thread Testing'.

# Types of Integration Testing

Software Engineering defines variety of strategies to execute Integration testing, viz.

 Big Bang Approach :

Incremental Approach: which is further divided into the following  Top Down Approach

 Bottom Up Approach

 Sandwich Approach — Combination of Top Down and Bottom Up

# How to do Integration Testing?

The Integration test procedure irrespective of the Software testing strategies:  Prepare the Integration Tests Plan

 Design the Test Scenarios, Cases, and Scripts.

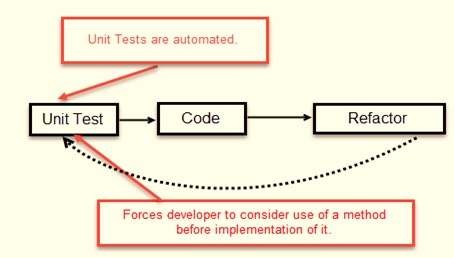
 Executing the test Cases followed by reporting the defects.  Tracking & re-testing the defects.

 Steps 3 and 4 are repeated until the completion of Integration is successful.

# Test-driven development

Test Driven Development (TDD) is software development approach in which test cases are developed to specify and validate what the code will do. In simple terms, test cases for each functionality are created and tested first and if the test fails then the new code is written in order to pass the test and making code simple and bug-free.

Test-Driven Development starts with designing and developing tests for every small functionality of an application. TDD framework instructs developers to write new code only if an automated test has failed. This avoids duplication of code. The TDD full form is Test-driven development.

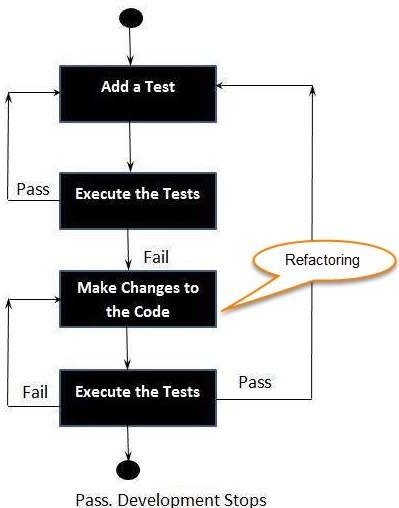


The simple concept of TDD is to write and correct the failed tests before writing new code (before development). This helps to avoid duplication of code as we write a small amount of code at a time in order to pass tests. (Tests are nothing but requirement conditions that we need to test to fulfill them).

Test-Driven development is a process of developing and running automated test before actual development of the application. Hence, TDD sometimes also called as Test First Development.

# Following steps define how to perform TDD test,

* **Add** **a** **test.**
* **Run** **all** **tests** **and** **see** **if** **any** **new** **test** **fails.**
* **Write** **some** **code.**
* **Run** **tests** **and** **Refactor** **code.**
* **Repeat**

****

* **REPL-driven** **development**

REPL is short for read-evaluate-print-loop. It means an interactive terminal such as your Bash shell or DOS command line interface, where you type a command and see an immediate response. A command or expression is read and then evaluated. The result is then printed to screen.

With REPL based development, testing and coding is more of a merged interactive task. The way you go about it is not strictly defined the way TDD is. It is more of a loosely defined practice and philosophy.

With a REPL approach you are continously running code and looking at outputs. Every time you are typing a line of code you get to verify if you are doing a sensible thing. If you get the wrong results, you can quickly bring the previous line of code back from history with the  arrow key and modify it.

# Deployment of the system:

* **Deployment** **systems**

Deployment in DevOps is a process that enables you to retrieve important codes from version control so that they can be made readily available to the public and they can use the application in a ready-to-use and automated fashion. Deployment tools DevOps comes into play when the developers of a particular application are working on certain features that they need to build and implement in the application. It is a very effective, reliable, and efficient means of testing and deploying organizational work.

Continuous deployment tools in DevOps simply mean updating the required codes on a particular server. There can be multiple servers and you need the required amount of tools to continuously update the codes and refresh the website. The functionality of the DevOps continuous deployment tools can be explained as follows:

* In the first phase of testing, the DevOps codes are merged for internal testing.
* The next phase is staging where the client's test takes place as per their requirements.
* Last but not least the production phase makes sure that any other feature does not get impacted because of the updating these codes on the server.

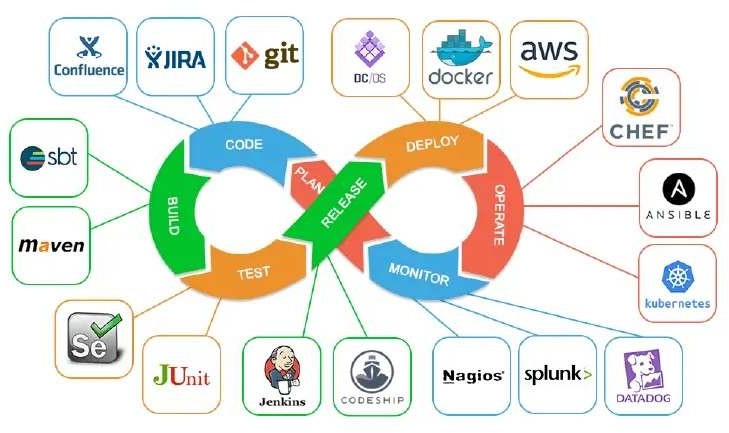
DevOps deployment tools make the functionality of the servers very convenient and easy for the users. It is different from the traditional way of dealing with the applications and the improvement has given positive results to all the companies as well as to all the users.

# What are DevOps Deployment Tools?

DevOps tools make it convenient and easier for companies to reduce the probability of errors and maintain continuous integration in operations. It addresses the key aspects of a company. DevOps tools automate the whole process and automatically build, test, and deploy the features.

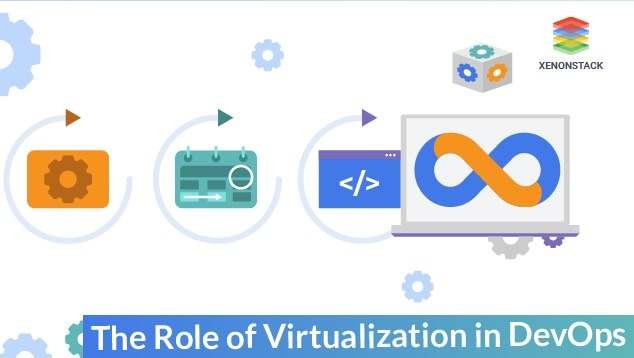
DevOps tools make the whole deployment process and easy going one and they can help you with the following aspects:

* Increased development.
* Improvement in operational efficiency.
* Faster release.
* Non-stop delivery.
* Quicker rate of innovation.
* Improvement in collaboration.
* Seamless flow in the process chain.



# Virtualization stack

It is the process of creating and running a virtual instance of something. In most of the cases, there is a layer of abstraction between the actual hardware and the virtual instance. That way, we can increase the capabilities of a system.



# The various types

There are different types of virtualization:

# Hardware Virtualization

It may be considered as the most common type these days. The best example of it is a Virtual Machine. A virtual machine works and looks like a real system with the same or a different operating system.

# Network Virtualization

It is a process in which a combination of software and hardware network resources form a single software network, which is commonly known as Virtual Network. Also, the available bandwidth is divided into several independent channels, which can be used by real devices and servers.

# Desktop Virtualization

In the case of it, the logical or virtual desktop is separate from the physical desktop. Here, instead of accessing the desktop using the computer hardware like keyboard, mouse of the system, the desktop is located remotely from another system by using a network connection. The network can be a wired/ wireless LAN or the internet. So, the user can access their files from any system without physically operating the order that contains the data.

# Storage Virtualization

In this case, a combination of several storage disks forms a storage pool or group. These groups are virtual storage units. These can then be assigned to servers for use. Logical volumes are one of the examples of it, which represent the storage as a coherent unit rather than a physical unit.

# Application Virtualization

In this, applications are virtualized and encapsulated. Virtual applications are not installed like traditional applications but are used as they are installed.

# a. Server Virtualization

This type comes in handy when we need to run a single physical server on multiple operating systems simultaneously. With this process, the performance, capacity and efficiency of the server are increased, while managing costs and complexity are reduced.

# Role of Virtualization in DevOps

It plays a vital role in devops. It, automates various software development processes, including testing and delivery. With the help of it, the its teams can develop and test within virtual and simulated environments using similar devices and systems to the end-users. This way, the development and testing become more efficient and less time-consuming. Virtual live environments can also be provided to test the software at the deployment level. This helps in real- time testing, as the team can check the effect of every new change made to the software. By doing these tasks in virtualized environments, the amount of computing resources is reduced. This real-time testing helps in increasing the quality of the product. Working with a virtual environment reduces the time for retesting and rebuilding the software for production. Thus, it reduces the extra efforts for the devops team, while ensuring faster and reliable delivery.

# What are the benefits?

There are many perks of it, are below listed:

# The workload is reduced

The providers of it continuously update the hardware and software used for it, so there is no need to do these updates locally. The IT staff of a company can focus on other important things and save time and cost for the organization.

# Testing Environment

With the help of it, we can set up a local testing environment. This environment can be used for various kinds of testing for software. Even if a server crashes, there won't be any data loss. So, the reliability is increased, and the software can be tested on this virtual environment until it is ready for live deployment.

# Energy-saving

It saves energy as instead of using local software or servers; the virtualization takes place with the help of virtual machines, which lowers the power or energy utilization. By saving this energy, the cost is reduced, and this saved money can be used for other useful operations.

# Improving Hardware utilization

With it, the need for physical systems decreases. Thus, maintenance costs and power utilization is reduced. The use of CPU and memory is improved.

# What are the challenges?

Despite having several perks, virtualization in it also have some challenges or limitations.

# Time consumption

Even if the development and testing time is saved, but it still consumes much time, as its configuration and application need time.

# Security risk

There is a big chance of data breach with the process of it as the remote accessibility and virtualizing desktop or applications is not a very secure option.

# Infrastructure knowledge

To work with it, the IT staff should have expertise in virtualization. Hence, either the existing employees can be trained, or new employees are required for an organization if they want to start working with it and devops. It involves much time and costs much money.

# code execution at the client,

* + **Puppet** **master** **and** **agents**

Puppet is a configuration management technology to manage the infrastructure on physical or virtual machines. It is an open-source software configuration management tool developed using Ruby which helps in managing complex infrastructure on the fly.

Puppet is a configuration management tool developed by Puppet Labs in order to automate infrastructure management and configuration. Puppet is a very powerful tool which helps in the concept of Infrastructure as code. This tool is

written in Ruby DSL language that helps in converting a complete infrastructure in code format, which can be easily managed and configured.

Puppet follows the client-server model, where one machine in any cluster acts as the server, known as puppet master and the other acts as a client known as a slave on nodes. Puppet has the capability to manage any system from scratch, starting from initial configuration till the end-of-life of any particular machine.

# Features of Puppet System

Following are the most important features of Puppet.

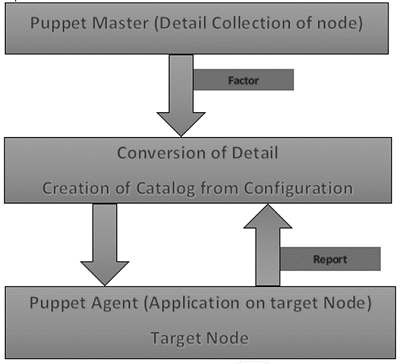
# Idempotency

Puppet supports Idempotency which makes it unique. Similar to Chef, in Puppet, one can safely run the same set of configuration multiple times on the same machine. In this flow, Puppet checks for the current status of the target machine and will only make changes when there is any specific change in the configuration.

Idempotency helps in managing any particular machine throughout its lifecycle starting from the creation of machine, configurational changes in the machine, till the end-of-life. Puppet Idempotency feature is very helpful in keeping the machine updated for years rather than rebuilding the same machine multiple times, when there is any configurational change.

# Cross-platform

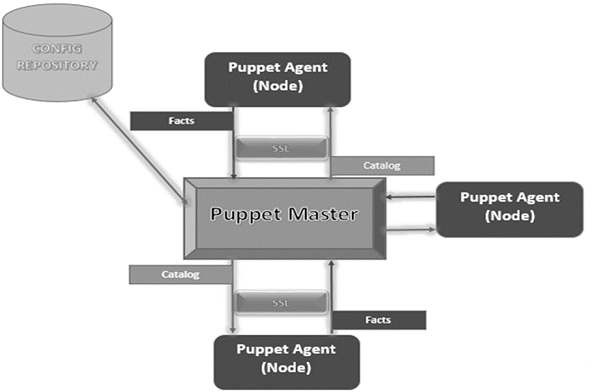
In Puppet, with the help of Resource Abstraction Layer (RAL) which uses Puppet resources, one can target the specified configuration of system without worrying about the implementation details and how the configuration command will work inside the system, which are defined in the underlying configuration file.



Puppet uses the following workflow to apply configuration on the system.

* + In Puppet, the first thing what the Puppet master does is to collect the details of the target machine. Using the factor which is present on all Puppet nodes (similar to Ohai in Chef) it gets all the machine level configuration details. These details are collected and sent back to the Puppet master.
  + Then the puppet master compares the retrieved configuration with defined configuration details, and with the defined configuration it creates a catalog and sends it to the targeted Puppet agents.
  + The Puppet agent then applies those configurations to get the system into a desired state.
  + Finally, once one has the target node in a desired state, it sends a report back to the Puppet master, which helps the Puppet master in understanding where the current state of the system is, as defined in the catalog.

# Puppet Archietecture

****

**Puppet** **Master**

Puppet Master is the key mechanism which handles all the configuration related stuff. It applies the configuration to nodes using the Puppet agent.

# Puppet Agent

Puppet Agents are the actual working machines which are managed by the Puppet master. They have the Puppet agent daemon service running inside them.

# Config Repository

This is the repo where all nodes and server-related configurations are saved and pulled when required.

# Facts

Facts are the details related to the node or the master machine, which are basically used for analyzing the current status of any node. On the basis of facts, changes are done on any target machine. There are pre-defined and custom facts in Puppet.

# Catalog

All the manifest files or configuration which are written in Puppet are first converted to a compiled format called catalog and later those catalogs are applied on the target machine.

**-** **Ansible**

Ansible is simple open source IT engine which automates application deployment, intra service orchestration, cloud provisioning and many other IT tools.

Ansible is easy to deploy because it does not use any agents or custom security infrastructure.

Ansible uses playbook to describe automation jobs, and playbook uses very simple language i.e. YAML (It's a human-readable data serialization language & is commonly used for configuration files, but could be used in many applications where data is being stored)which is very easy for humans to understand, read and write. Hence the advantage is that even the IT infrastructure support guys can read and understand the playbook and debug if needed (YAML — It is in human readable form).

Ansible is designed for multi-tier deployment. Ansible does not manage one system at time, it models IT infrastructure by describing all of your systems are interrelated. Ansible is completely agentless which means Ansible works by connecting your nodes through ssh(by default). But if you want other method for connection like 》erberos, Ansible gives that option to you.

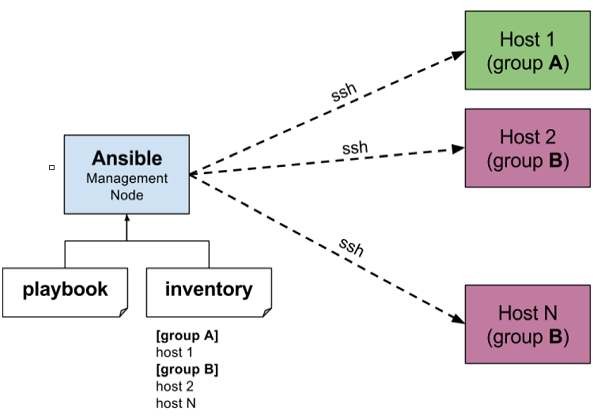
After connecting to your nodes, Ansible pushes small programs called as “Ansible Modules”. Ansible runs that modules on your nodes and removes them when finished. Ansible manages your inventory in simple text files (These are the hosts file). Ansible uses the hosts file where one can group the hosts and can control the actions on a specific group in the playbooks.

# How Ansible Works?

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. Ansible then executes these modules (over SSH by default), and removes them when finished. Your library of modules can

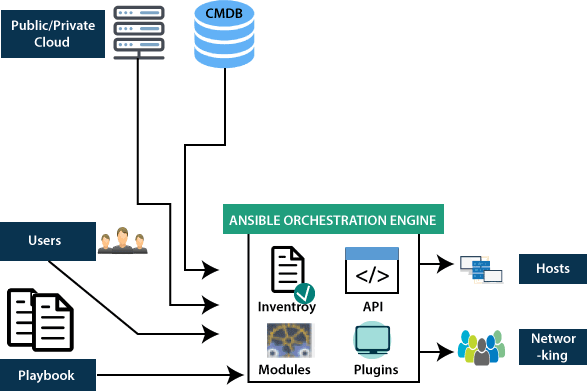
reside on any machine, and there are no servers, daemons, or databases required.

# The picture given below shows the working of Ansible.

****

The management node in the above picture is the controlling node (managing node) which controls the entire execution of the playbook. It's the node from which you are running the installation. The inventory file provides the list of hosts where the Ansible modules needs to be run and the management node does a SSH connection and executes the small modules on the hosts machine and installs the product/software.

Beauty of Ansible is that it removes the modules once those are installed so effectively it connects to host machine , executes the instructions and if it's successfully installed removes the code which was copied on the host machine which was executed.



# Deployment tools:

**-Chef,** **Salt** **Stack** **and** **Docker.**

Chef is an open source technology developed by Opscode. Adam Jacob, co- founder of Opscode is known as the founder of Chef. This technology uses Ruby encoding to develop basic building blocks like recipe and cookbooks. Chef is used in infrastructure automation and helps in reducing manual and repetitive tasks for infrastructure management.

Chef have got its own convention for different building blocks, which are required to manage and automate infrastructure.

# Why Chef?

Chef is a configuration management technology used to automate the infrastructure provisioning. It is developed on the basis of Ruby DSL language. It is used to streamline the task of configuration and managing the company's server. It has the capability to get integrated with any of the cloud technology.

In DevOps, we use Chef to deploy and manage servers and applications in- house and on the cloud.

# Features of Chef

Following are the most prominent features of Chef −

* Chef uses popular Ruby language to create a domain-specific language.
* Chef does not make assumptions on the current status of a node. It uses its mechanisms to get the current status of machine.
* Chef is ideal for deploying and managing the cloud server, storage, and software.

# Advantages of Chef

Chef offers the following advantages −

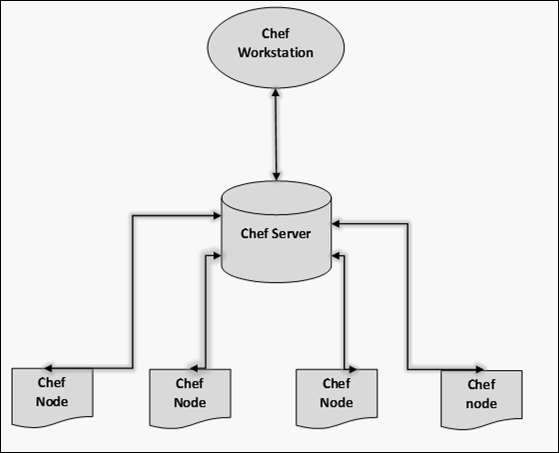
* Lower barrier for entry − As Chef uses native Ruby language for configuration, a standard configuration language it can be easily picked up by anyone having some development experience.
* Excellent integration with cloud − Using the knife utility, it can be easily integrated with any of the cloud technologies. It is the best tool for an organization that wishes to distribute its infrastructure on multi-cloud environment.

# Disadvantages of Chef

* Some of the major drawbacks of Chef are as follows −
* One of the huge disadvantages of Chef is the way cookbooks are controlled. It needs constant babying so that people who are working should not mess up with others cookbooks.
* Only Chef solo is available.
* In the current situation, it is only a good fit for AWS cloud.
* It is not very easy to learn if the person is not familiar with Ruby.
* Documentation is still lacking.

# Architecture

Chef works on a three-tier client server model wherein the working units such as cookbooks are developed on the Chef workstation. From the command line



utilities such as knife, they are uploaded to the Chef server and all the nodes which are present in the architecture are registered with the Chef server.

In order to get the working Chef infrastructure in place, we need to set up multiple things in sequence.

The following components.

# Chef Workstation

This is the location where all the configurations are developed. Chef workstation is installed on the local machine. Detailed configuration structure is discussed in the later chapters of this tutorial.

# Chef Server

This works as a centralized working unit of Chef setup, where all the configuration files are uploaded post development. There are different kinds of Chef server, some are hosted Chef server whereas some are built-in premise.

# Chef Nodes

They are the actual machines which are going to be managed by the Chef server. All the nodes can have different kinds of setup as per requirement. Chef client is the key component of all the nodes, which helps in setting up the communication between the Chef server and Chef node. The other components of Chef node is Ohai, which helps in getting the current state of any node at a given point of time.

# Saltstack

Salt is a very powerful automation framework. Salt architecture is based on the idea of executing commands remotely. All networking is designed around some aspect of remote execution. This could be as simple as asking a Remote Web Server to display a static Web page, or as complex as using a shell session to interactively issue commands against a remote server. Salt is an example of one of the more complex types of remote execution.

Salt is designed to allow users to explicitly target and issue commands to multiple machines directly. Salt is based around the idea of a Master, which controls one or more Minions. Commands are normally issued from the Master to a target group of Minions, which then execute the tasks specified in the commands and then return the resulting data back to the Master. Communications between a master and minions occur over the ZeroMQ message bus.

SaltStack modules communicate with the supported minion operating systems. The Salt Master runs on Linux by default, but any operating system can be a minion, and currently Windows, VMware vSphere and BSD Unix variants are well supported. The Salt Master and the minions use keys to communicate. When a minion connects to a master for the first time, it automatically stores keys on the master. SaltStack also offers Salt SSH, which provides an “agent less” systems management.

# Need for SaltStack

SaltStack is built for speed and scale. This is why it is used to manage large infrastructures with tens of thousands of servers at LinkedIn, WikiMedia and Google.

Imagine that you have multiple servers and want to do things to those servers. You would need to login to each one and do those things one at a time on each one and then you might want to do complicated things like installing software and then configuring that software based on some specific criteria.

Let us assume you have ten or maybe even l00 servers. Imagine logging in one at a time to each server individually, issuing the same commands on those l00 machines and then editing the configuration files on all l00 machines becomes very tedious task. To overcome those issues, you would love to update all your servers at once, just by typing one single command. SaltStack provides you exactly the solution for all such problems.

# Features of SaltStack

SaltStack is an open-source configuration management software and remote execution engine. Salt is a command-line tool. While written in Python, SaltStack configuration management is language agnostic and simple. Salt platform uses the push model for executing commands via the SSH protocol. The default configuration system is YAML and Jinja templates. Salt is primarily competing with Puppet, Chef and Ansible.

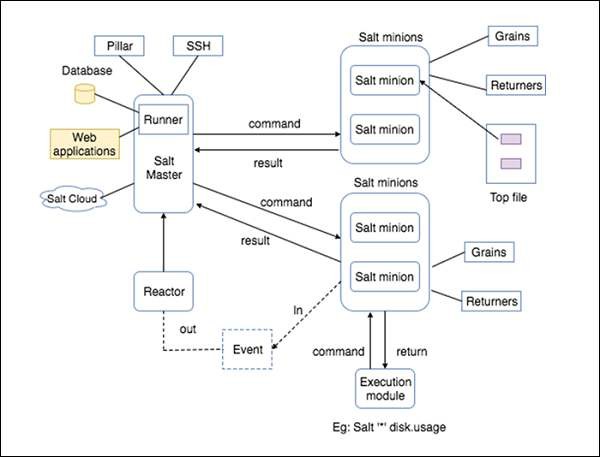
Salt provides many features when compared to other competing tools. Some of these important features are listed below.

* **Fault** **tolerance** − Salt minions can connect to multiple masters at one time by configuring the master configuration parameter as a YAML list of all the available masters. Any master can direct commands to the Salt infrastructure.
* **Flexible** − The entire management approach of Salt is very flexible. It can be implemented to follow the most popular systems management models such as Agent and Server, Agent-only, Server-only or all of the above in the same environment.
* **Scalable** **Configuration** **Management** − SaltStack is designed to handle ten thousand minions per master.
* **Parallel** **Execution** **model** − Salt can enable commands to execute remote systems in a parallel manner.
* **Python** **API** − Salt provides a simple programming interface and it was designed to be modular and easily extensible, to make it easy to mold to diverse applications.
* **Easy** **to** **Setup** − Salt is easy to setup and provides a single remote execution architecture that can manage the diverse requirements of any number of servers.
* **Language** **Agnostic** − Salt state configuration files, templating engine or file type supports any type of language.

# Benefits of SaltStack

Being simple as well as a feature-rich system, Salt provides many benefits and they can be summarized as below −

* **Robust** − Salt is powerful and robust configuration management framework and works around tens of thousands of systems.
* **Authentication** − Salt manages simple SSH key pairs for authentication.
* **Secure** − Salt manages secure data using an encrypted protocol.
* **Fast** − Salt is very fast, lightweight communication bus to provide the foundation for a remote execution engine.
* **Virtual** **Machine** **Automation** − The Salt Virt Cloud Controller capability is used for automation.
* **Infrastructure** **as** **data,** **not** **code** − Salt provides a simple deployment, model driven configuration management and command execution framework.



# Arhietechture

* **SaltMaster** − SaltMaster is the master daemon. A SaltMaster is used to send commands and configurations to the Salt slaves. A single master can manage multiple masters.
* **SaltMinions** − SaltMinion is the slave daemon. A Salt minion receives commands and configuration from the SaltMaster.
* **Execution** − Modules and Adhoc commands executed from the command line against one or more minions. It performs Real-time Monitoring.
* **Formulas** − Formulas are pre-written Salt States. They are as open-ended as Salt States themselves and can be used for tasks such as installing a package, configuring and starting a service, setting up users or permissions and many other common tasks.
* **Grains** − Grains is an interface that provides information specific to a minion. The information available through the grains interface is static. Grains get loaded when the Salt minion starts. This means that the information in grains is unchanging. Therefore, grains information could be about the running kernel or the operating system. It is case insensitive.
* **Pillar** − A pillar is an interface that generates and stores highly sensitive data specific to a particular minion, such as cryptographic keys and passwords. It stores data in a key/value pair and the data is managed in a similar way as the Salt State Tree.
* **Top** **File** − Matches Salt states and pillar data to Salt minions.
* **Runners** − It is a module located inside the SaltMaster and performs tasks such as job status, connection status, read data from external APIs, query connected salt minions and more.
* **Returners** − Returns data from Salt minions to another system.
* **Reactor** − It is responsible for triggering reactions when events occur in your SaltStack environment.
* **SaltCloud** − Salt Cloud provides a powerful interface to interact with cloud hosts.
* **SaltSSH** − Run Salt commands over SSH on systems without using Salt minion.