**Name Umair Nawaz**

**Sap ID 20751**

**What is the test automation framework?**

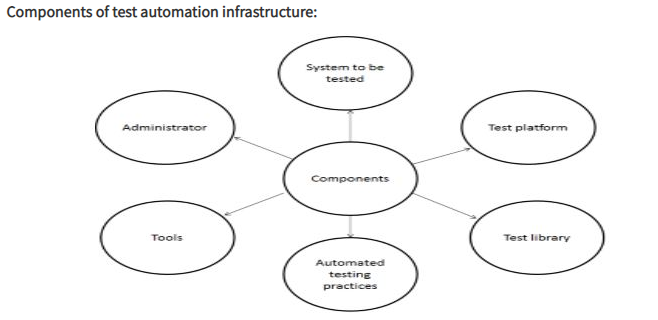
Test automation infrastructure, or framework, consists of test tools, equipment, test scripts, procedures, and people needed to make test automation efficient and effective. The creation and maintenance of test automation framework are key to the success of any test automation project within an organization.

**The idea behind an automation infrastructure is to ensure the following**:

* Different test tools and equipment are coordinated to work together.
* The library of the existing test case scripts can be reused for different test projects, thus minimizing the duplication of development effort.
* Nobody creates test scripts in their own way.
* Consistency is maintained across test scripts.

**Components of test automation:**

* **System to be tested**
* **Test Platform**
* **Test Case Library**
* **Tools**
* **Automated Testing Practices**
* **Administrator**



## Types of Test Automation Frameworks

* Linear Scripting
* The Test Library Architecture Framework.
* The Data Driven Testing Framework.
* The Keyword-Driven or Table-Driven Testing Framework.
* The Hybrid Test Automation Framework.

Test Automation Framework as a set of guidelines for creating and designing test cases. It is a conceptual part of the automated testing that helps testers to use resources more efficiently. A framework is defined as a set of rules or best practices that can be followed in a systematic way that ensures to deliver the desired results. An automation testing framework is a platform developed by integrating various hardware, software resources along with using [various tools for automation testing](https://www.testingxperts.com/blog/software-testing-tools-list#Test%20Automation%20Tools) and web service automation framework, based on a qualified set of assumptions. This framework enables efficient design and development of automated test scripts and ensures reliable analysis of issues or bugs for the system or application under test (AUT).

## **Purpose of a Test Automation Framework**

## Enhances efficiency during the design and development

## Provides a structured development methodology to ensure uniformity of design

## Enables reliable issue and bug detection and delivers proper root-cause

## Reduces dependence on teams by automatically

## Improves utilization of various resources

## What is selenium?

Selenium is an open-source tool that automates web browsers. It provides a single interface that lets you write test scripts in programming languages like Ruby, Java, NodeJS, PHP, Perl, Python, and C#, among others.

### **Selenium WebDriver**

Also known as Selenium 2.0, WebDriver executes test scripts through browser-specific drivers. It consists of:

##### API

##### Library

##### Driver

##### Framework

**How it Works:**

The WebDriver protocol has a local end (‘client’) which sends the commands (test script) to a browser-specific driver. The driver executes these commands on its browser-instance. So, if the test script calls for execution on Chrome and Firefox, the chrome Driver will execute the test on Chrome; the Gecko Driver will do the same on Firefox.

**Why do I need Selenium Automation Testing?**

Imagine that a manual tester has this scenario: Checking whether the web app’s signup page (www.example.com/signup) validates input strings and registers a user successfully in latest versions of Chrome and Firefox, on Windows 7. Assume that the signup page has these input fields—username, email address, and password. The tester will get a Windows 7 desktop and follow these steps, consecutively, on latest versions of Chrome and Firefox:

* Enter the URL in the address bar (www.example.com/signup)
* Enter an invalid string in each input field (email, username, and password)
* Check whether the input strings
* Enter ‘valid’ strings in each input field; click Sign Up
* Check whether “Welcome, ‘{‘username’}’“ page showed up
* Check whether the system database created a new user ID for ‘{‘username’}’
* Mark the test ‘passed’ if it did, ‘failed’ if the signup feature broke anywhere during the test.

Test scripts execute only when the WebDriver’s client and browser/driver are connected.

## Types of Testing can be automated with Selenium?

##### Compatibility Testing

##### Performance Testing

##### Integration Testing

##### System Testing

##### End-to-end Testing

##### Regression Testing:

**The most common tools that are used for configuration management are packer and ansible. You need to concisely compare both of them.**

Software development requires a continuous cycle of update of features or addition of new features and this often translate to rescaling of the product infrastructure by DevOps Engineers to ensure proper deployment and functioning of the product. Configuration and change management are two different but complementary concepts. Configuration Management serves as a version control system for a product, processes, plans, baselines, and takes care of how changes should be handled when scaling up or down. The changes includes the versions and updates that have been applied to installed software packages and the locations and network addresses of hardware devices. Change Management focuses on how any change to the process, or controls should be done.

**Benefits of Configuration and Change Management for Servers:**   
**Reliability:**It ensures that the configuration of a systems is well defined, rather than relying on vague ideas from a DevOps engineer or team.The system ensures clarity of requirements needed to handle changes thereby avoiding problems.

**Organization:**With a documented system, connecting past records of the infrastructure and making informed decisions to address changes in the future becomes pretty easy to come by.

**Cost and Risk Reduction:**A substantial knowledge base of the configuration of the system ensures effective maintenance of the system and prevents wasteful duplication of infrastructure.

**Tools for Configuration Management:**

**Ansible:**This is an open-source automation software, built on the popular Python language, for provisioning, configuration management, and application deployment.  
**Vagrant:**This is an open-source tool that can be used to create and configure reproducible virtual development environments. It built on Ruby and works on top of virtualization products such as Virtual Box, VMware etc.  
There are a host of other configuration management tools which are not mentioned here.

**PACKER:**

Packer is a Hashicorp product. In their words, “Packer is an open source tool for creating identical machine images for multiple platforms from a single source configuration.” We use Packer to take US Government approved Amazon Machine Images (AMIs)  running Red Hat 7 and produce new versions of these AMIs that have all the configuration and software we need to run our application securely in AWS.

Packer supports multiple “provisions,” which handle the actual server configuration. These can be simple shell scripts, or can be a more robust tool like Ansible. Packer handles the creation of the VM and packaging as an AMI, Ansible handles the configuration of the virtual machine.

Reff:

<https://www.ques10.com/p/2293/what-are-the-components-of-test-automation-infra-1/#:~:text=A%20test%20automation%20infrastructure%2C%20or,automation%20project%20within%20an%20organization>.

<https://www.guru99.com/test-automation-framework.html>

<https://www.testingxperts.com/blog/test-automation-frameworks#What%20is%20Test%20Automation%20Framework>?

<https://www.browserstack.com/selenium>

<https://madeintandem.com/blog/packer-ansible-terraform-devops-immutable-servers/>

<https://medium.com/@marcdomain/using-packer-ansible-for-configuration-and-change-management-8ae752fe1b2e>