Test Automation Framework as a Service (TAFaaS) – Scale Test Automation & DevOps Practices with Cloud, Containers, and Microservice.

Sandeep Sivanandan

AbstractTraditional test automation frameworks are tightly coupled with technology stacks, process, and teams. The team relies on their own test framework, which adds cost to migration, scalability, integration, maintenance, and performance. The test infrastructure is cost-oriented - idle time is more, instances are required if we want to run multiple automation runs. Other team members need to build the same framework and infrastructure for automation and DevOps. Dedicated resources are needed to maintain the framework, infrastructure, and operations. Migrating to a new test version of test framework, test tools, libraries become tedious as it is cost-oriented to move up and down. Validating new framework with the same infrastructure and tools takes another hardware instance procurement. As technology changes, test framework changes are hard and cost-effective. Test framework becomes oriented toward, functional automation, performance automation, unit tests, security tests, API testing etc. And most automation scenarios are not exact customer environment based at least on IOT and Cloud [replica of customer setup]. The paper is design for implementing a scalable, mutable and self -learning test automation framework providing test framework as a service.

Testing, Automation Framework, Continuous Integration, Continuous Deployment, DevOps, Cloud, Selenium,Applitools,Jenkins,Microservice, Agile, VM-virtual machine, Containers.

1. INTRODUCTION

Most traditional test automation frameworks are always built in and around the test tools used for automation. The basic elements are added once the tool proves it can be used for the current situation. But in today's world, the technology stacks are different and advanced. Test tools alsoneed to be smart and effective to perform tests. But the problem doesn't get solved by using smarter tools, the problem can be somehow mitigated by using smart, self-learning and adaptive test automation framework for On-premise[Enterprise],Cloud, Data Center, Mobile platforms and as the need arises.

There can be multiple approaches to tackle this problem, and one of the design approach proposed is to build a test framework which is built with Microservice Architecture [2], Containers[6], virtualization and use the power of the cloud.

2. PROBLEM DEFINITION

Traditional automation framework has multiple issues. This paper will focus on a few of those problems which will help to make the test automation for cloud and next-generation applications and products.

The below Figure 1 most of the components are tightly coupled, upgrading one structure can break the other components.

Upgrade of tools can be difficult because test libraries are built around that particular version of tools. The framework cannot support plug and play tooling. Test environments are either virtual machines or hardware or maybe cloud environment. Automation runs for browser applications on a single Virtual machine can have 1 instance of browser or simulator. To scale up execution a whole bunch of Virtual Machines isneeded. Most of the times the Virtual Machines are idle after the run consuming power and resources.

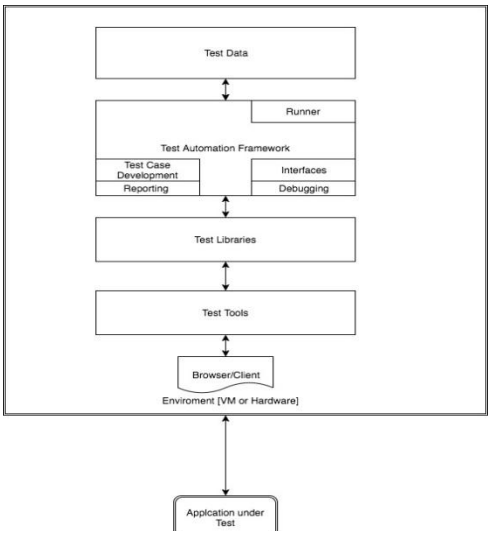

Figure 1: Basic standalone automation framework

3. SOLUTION APPROACH

What is Microservice?

The term "Microservice Architecture[2]" has sprung up over the last few years to describe a particular way of designing software applications as suites of independently deployable services. While there is no precise definition of this architectural style, there are certain common characteristics around organization around business capability, automated deployment, intelligence in the endpoints, and decentralized control of languages and data.

Revised Manuscript Received on May 29, 2019.

Sandeep Sivanandan, VxBlock Engineering Division, DellEMC India Pvt Ltd., Bangalore, Karnataka, India

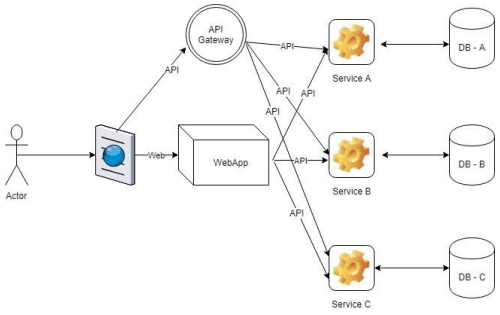

Figure 2: Microservice Architecture

In the same lines mentioned above, a software automation test framework is also a software test solution for minimizing the regression effort of product change.

What is a Container?

In Figure 3: Containers[6] as the name specifies implies in storing your package inside a box and packing it and sending it over to any place for usage. InSoftware, terms reflect the same aspect. Bundle your required software code and dependencies and deploy on any platform and use them at ease. Docker is one of the technologies which uses containerization. Its lightweight, independent and can run anywhere with the application code, libraries, tools and on runtime.

Docker[7] containers[6] are available on platforms - Linux, Windows, Cloud, Serverless etc.

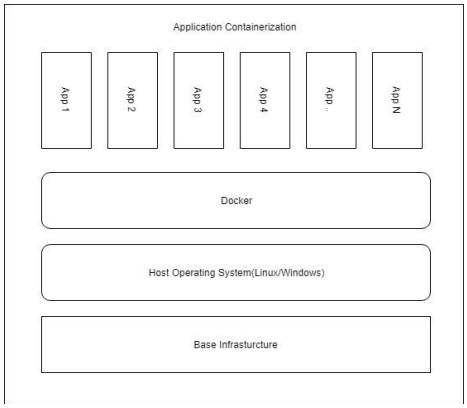

Figure 3: Container Architecture

What is Virtual Machine?

Virtual Machine is an Operating system that is installed on a Software which replicates dedicated hardware. End-userhas the same experience as they are using dedicated hardware and OS/application.

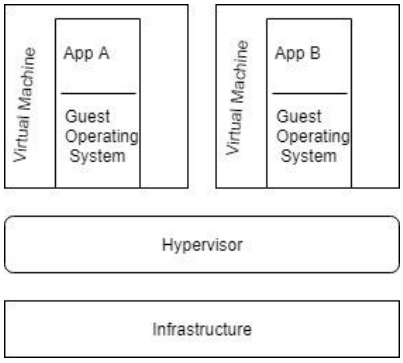

Figure 4: Virtual Machine

Containers and Virtual machines play a vital role when used together. There can be different use cases for the usage of both together or as different entities.

Test Automation Framework Architecture

As shown in below Figure 5, the framework architecture will have multiple components scaled in a cloud infrastructure.

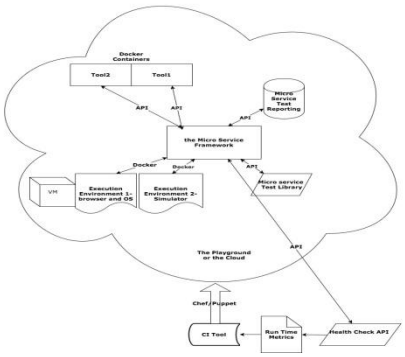

Figure 5: TAFaaS Architecture

Explanation of each element in the architecture:

1. Docker Containers [Tools]: This can be any automation tool that is used for automating the test – Selenium[16],Appium[17],Applitools[19], protractor[18] or any of those.
2. The Microservice Automation Framework: The heart of the playground. All Framework business logic sits here.
3. Microservice Test Libraries: Test Libraries which does the main work of verification and executions during test executions. API[11] test libraries, UI test libraries, command line libraries etc.
4. Virtual Machines: Execution platform. Can be Windows with multiple OS flavors, Linux with OS flavors, Mobile simulators.
   We can bundle the above into docker containers and deploy into single Virtual machine also. Again it will depend on the use case.
5. Microservice Reporting infrastructure: All test reports are exposed as API interfaces, can be written to a database or real-time to Continuous integration tool like jenkins[10]. Extensible according to usage.
6. Framework health check API: The framework processing is written to a logging mechanism and cloud monitoring tools can be leveraged to keep check of how is the playground behaving.
7. Framework Metrics: Reporting of the health check API[11].
8. CI(Continuous Integration) – Client/Customer Side: Jenkins / CloudBees or any CI to trigger the whole on-demandexecution. Can be configured according to needs.
9. Configuration Tools: Chef or puppet to create a configuration for deployment of the framework on the cloud. This can be leveraged on any Cloud platforms.

Example of the above approach can be for an e-commerce application with simple features which takes orders from customers, verifies inventory and available credit, and ships them. The application consists of several components including the Web-UI, which implements a user interface, along with some backend services for checking credit, maintaining inventory and shipping orders. The application consists of a set of services.

Test Development can be done with Behavior Driven or Keyword Driven approach making test case development much faster. Depending on the requirement; Continuous Integration tools will trigger the run on any Cloud:

Sample Steps (This can be enhanced to any extent based on usage):

1. Launch the virtual machines based on the configurations: Example: Windows 10 and Chrome browser for GUI test to run or we can leverage any third-party vendors like Sauce labs[19] for the infra.
2. Trigger the test Framework Components, test libraries, reporting as Microservice.
3. Launch the Tools set as containers and bring them up.
4. Run a Health Check monitoring as a microservice and monitor all the services and Dockers are communicating within and real-time statistics or health of the solution is monitored.
5. Trigger the tests and start publishing the result in real time.

There can be multiple version for framework before it is moved in production. Easy to replicate the framework for more wider usage and scaling. Replication of framework or components can be done. The different framework can integrate to any components of the cloud.

Example: An enterprise java/python test framework can leverage the Reporting infra on cloud and use it for reporting than using its own. If some test libraries need to be shared those libraries can be replicated and used. Adding new tools to the cloud won't be much of hassle, a new set of test libraries needs to be built. If one tool becomes obsolete, it can be taken off.

During older release service upgrade, only the test libraries need to be pulled from Git, the rest of the components are usable. Version controlled or release specific runs can be designed.

4. BENEFITS OF THE TEST AUTOMATION FRAMEWORK AS SERVICE

- Each automation microservice is relatively small
  - Easier for a test developer to understand
  - The test executions start faster, which makes developers more productive, and speeds up testing and deployments
- Each service can be deployed independently of other services - easier to deploy new versions of services especially test tools and test libraries frequently
- Easier to scale development. It enables you to organize the development effort around multiple teams. Each team is the owner and is responsible for one or more single service. Each team can develop, deploy and scale their services independently of all of the other teams.
- Eliminates any long-term commitment to a technology stack. When developing a new service, you can pick a new technology stack. Similarly, when making major changes to an existing service you can rewrite it using a new technology stack.
- Automation teams can leverage the framework extensible helping reduce the consumption for resources.
- The framework can duplicate on need basis not hampering execution for other teams or projects.
- On-Demand – SAAS[12][Software as a Service] Solution support.

5. CONCLUSION AND FUTURE WORK

The objective of this paper is to build some generic, robust, self-learning and analytics supported as a service test framework. Can be extended and used for any kind of application infrastructure, Enterprise, Cloud, IOT,Hardware, Mobile Apps etc. Build High Availability for the frameworks using the cloud advantages.

Future is bright, we would like to work on design much of machine learning capabilities and bots to understand the code and auto-generate test stubs, where the rest of the code can be made to make it testable. Bots to understand the code coverage functional and performance. As the importance of serverless[21] technologies are coming in and major use case is with Custom Automation and Data processing, explore how well this can be leveraged in such an environment

ACKNOWLEDGMENT

The author wishes to thank members of Test and Product; VxBlock DellEMC, Bangalore for their help in completing this work.

REFERENCES

1. Nokia Solutions and Networks, "Robot Framework User Guide, Version 2.8.5", Online:http://robotframework.googlecode.com/hg/doc/userguide/RobotFrameworkUserGuide.html?r=2.8. 5, Section 1 - 4,2014
2. Chris Richardson, Microservice .io http://microservices.io/patterns/microservices.html
3. Sandeep Sivanandan and Yogeesha C B, "Agile Development Cycle: Approach to Design an Effective Model-Based Testing with Behaviour Driven Automation Framework", 20th annual International Conference on Advanced Computing and Communications (ADCOM 2014) at Bangalore, 19th - 22nd September 2014.
4. Docker Selenium : https://github.com/SeleniumHQ/docker-selenium -SeleniumHQ/docker-selenium
5. Mark Hughes – Docker Protractor Online: https://github.com/mrsheepuk/docker- protractor - mrsheepuk/docker-protractor
6. Linux Containers:Online: https://opensource.com/resources/what-are-linux-containers
7. Docker Containers, Online: https://www.docker.com

8. WebDriver, Online: http://webdriver.io/guide.html
9. What are Virtual machines, https://searchservervirtualization.techtarget.com/definition/virtual-machine
10. What isJenkins, Online:https://www.jenkins.org
11. What is API,Online: https://www.api.org/
12. What is as a Service, Online:https://en.wikipedia.org/wiki/As_a_service
13. What is SaaS ?, Online:https://azure.microsoft.com/enus/overview/what-is-saas/
14. What is Software Framework, Online:https://en.wikipedia.org/wiki/Software_framework
15. Appium Docker Containers,Online:https://github.com/appium/appium-docker-android
16. Sauce Labs, for online automation testing,Online: https://saucelabs.com/
17. Serverless Architecture,Online: https://serverless.com/framework

Sandeep Sivanandan is an M.Tech from SAM Higginbottom Institute, Allahabad. Over 15 years of rich experience in Testing, Test Automation Architecting, DevOps, Agile, Virtualization, Security and Performance Engineering. Currently working as Principal Software Engineer, DellEMC India Pvt. Ltd. for VXBlock Team, Bangalore from 2018.