REVIEW ON SINGLE RADIO ACCESS NETWORK AND CICD INTEGRATION TOOLS

Manasa Chandrashekar[1], Prof. Smitha G.R[2]

[1]*PG student, Department of Information Science and Engineering, RV College of Engineering® Bengaluru, India*
[2]*Assistant Professor, Department of Information Science and Engineering, RV College of Engineering® Bengaluru, India*

Abstract : The idea of a Single Radio Access Network arose from the desire to operate several radio technologies on an unified platform (SRAN). For GSM, WCDMA, LTE and 5G technologies, a single RAN comprises of multifunctional hardware and shared software. Continuous procedures, such as continuous integration, delivery, and deployment, are multinational technology company approaches that enable businesses to release new features and products on a regular and consistent basis. Continuous Integration (CI) is primarily concerned with development process automation and build/code integration/test automation. Saving resources such as time is a big difficulty for emerging firms, thus automation in integration and deployment utilizing Jenkins and Robot framework saves a lot of time, and it can also be readily updated if there are frequent changes that need to be made in the project. Jenkins integration is simple because it is open source and has various plug-in options. Robot Framework is a keyword-driven, general automated test framework that has seen widespread application in automated testing. This work provides an on overview of Single RAN and demonstrates how the Jenkins and Robot frameworks make it easy to build up a Continuous Integration (CI) or Continuous Delivery (CD) environment with pipelines for nearly any combination of languages and source code repositories, as well as automate other common development operations**.**

Index Terms **-** Single RAN, BTS, CICD, Jenkins, Robot Framework

I. INTRODUCTION

Single Radio Access Network (SRAN) is a development of traditional Radio Frequency (RF) sharing, which shared only radio modules, to having a single network entity with common operability and transport platform, sharing both system (baseband) and radio modules between multiple radio access technologies (RATs) [1]. A multi-RAT Network Element with common operability and maintenance functionalities, the Single RAN hardware and software solution simplifies network management. SRAN comprises the SBTS entity, the existing Nokia GSM and WCDMA controllers and the WCDMA OMS. The SBTS, along with the traditional GSM and WCDMA BTSs share the same controllers (BSC and RNC). SRAN offers high performance by enabling simultaneous co-ordination and co-operation of different radio access technologies. The SRAN product presents Common Operability and Common Transport solutions for all RATs.

In today's world, a product is developed utilizing AGILE approach, which allows for continuous development and testing throughout the project's software development lifecycle [4]. Testing is also important in this case because it is required in less time. When opposed to manual testing, automation has a number of advantages. The automated testing framework has been a popular testing framework in recent years because it is reusable, low-cost, and easy to maintain [10]. It's a framework that's driven by keywords. This framework is a Python-based framework for test automation designed by Nokia Siemens. This framework is scalable and may be used for both keyword and data searches. It's also used for software vulnerability scanning. Users can utilize current Robot Framework keywords or create new relatively high keywords from existing ones by utilizing the same syntax that is used to build test scenarios along this hierarchy. Robot Framework [3] is a keyword-driven generic test automation framework commonly used in automated testing. It parses test cases, then invokes the basic services system, the Test Library function, and the internal resources library, before concluding the data interchange during the test. It's possible to incorporate test control logic inside the test object's implementation. Based on the test object's test results, it may produce HTML test reports. The framework might be the first step, followed by the test data file, and lastly the test cases.

Jenkins is an open-source, self-sufficient automation server that may be used to automate a wide range of operations related to software development, testing, and delivery or deployment [9]. Jenkins may be installed through native system packages, Docker, or even run solo on any computer that has the Java Runtime Environment (JRE).

II. SCOPE

Nokia's SRAN concept is straightforward: it combines many radio technologies on a common multifunctional hardware platform. Single RAN will, in its most advanced form, consist of a single radio installation with a common transport, operational, and management system, as well as integrated unified security across radio access technologies (RATs) [1]. Furthermore, it allows the unified coordination and operation of several RATs, as well as the ability to employ current RATs to get the greatest results by collaborating their benefits.

The focus behind CICD is more towards faster time to market and early faults finding to R&D teams. Devops process provides the new pathway for better quality products with end-to-end automation [2]. CICD approach removes the legacy way building, testing and deployment from software products (mostly manual ways) to completely automated by adapting Robot Framework and Jenkins.

III. TECHNOLOGIES AND WORKFLOW

This section covers the detailed explanation of Single RAN product and workflow of CICD Integration automation tools.

3.1 Single RAN Base Station

SRAN is mostly about hardware that may be used for several purposes, as well as common software, Management and Operation, transportation, and network performance optimization and configuration deciding tasks. More advanced SRAN competences will enhance these collaborating possibilities, making network management easier and providing mobile broadband operators with greater flexibility, scalability, and robustness.

It offers the following benefits to the operator:

- SRAN allows the operator to flexibly and efficiently evolve and reframe their network in accordance to user demand shifting from 2G and 3G to LTE.
- Simplified network architecture.
- Flexible and efficient RF spectrum usage and reframing of network based on user demand from 2G and 3G to LTE
- Efficient shared hardware usage
- Reduced energy consumption and site size through BTS hardware resource sharing
- Common Transport and Operability for all RATs
- SRAN site visualization managed as one network element in NetAct
- One SBTS Web-based Element Manager (WebEM) with site management covering all RATs
- Converged planning, operations and management
- System Module sharing and RF sharing

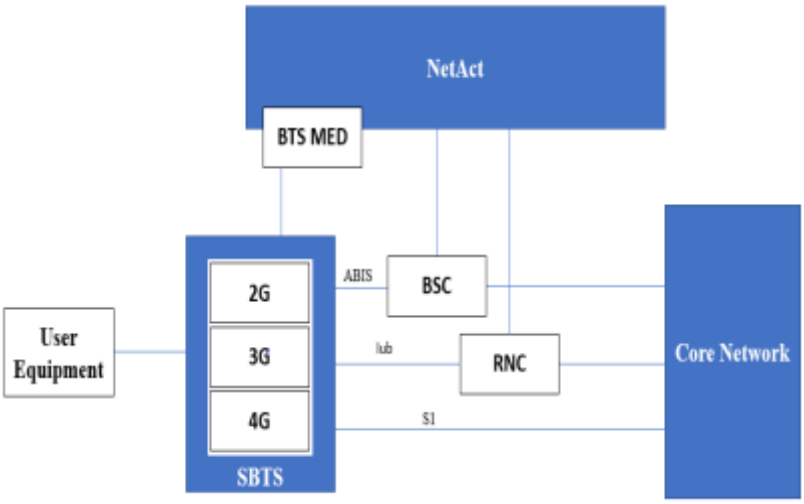

Figure. 1: Overview of Single RAN system architecture

Figure 1 shows Single RAN brings evolution to the network, management, and operations structure. SBTS (Single RAN BTS) provides three main features: common transport, common operability, and system and radio module sharing across shared RF fibers. Within the same network, SBTS and specialized RAT BTSs can coexist. It provides a wide range of sharing options, including baseband, RF, backhaul and fronthaul transport, network, and spectrum, as well as end-to-end security and a single common Operations and Maintenance (O&M) functionality.

NetAct: In multi-technology radio and core networks, it is a Network element manager that provides capabilities for configuration management with common procedures and data storage. From element management to operational management, it enables centralised end-to-end network monitoring on a single screen for all technologies.

BTS Mediator: It acts as a mediator between the BTS M-Plane and the EMS at the destination (Element Management System). Data collection, harmonization, conversion, and distribution are all part of the mediation functionality.

Web-based Element Manager (WebEM): Separate BTS Site Managers for different RATs are replaced with a single user interface. SBTS Element Manager is a Web application which directly loads from the BTS. It can be used locally as well as remotely, without the need to install any additional software on the operator's side.

3.2 CICD Integration

Jenkins makes it simple to set up Continuous Delivery (CD) and Continuous Integration (CI) as well as automating other common development jobs. It uses pipelines to create an environment for practically any combination of languages and source code repositories [6]. Jenkins does not eliminate the need for individual stage files, but it does make integrating the full chain of builds, tests, and deployment tools faster and more reliable. Jenkins needs to be connected with automated test cases so that every build executes critical tests and reports on the build's stability.

Robot Framework is a general open-source test automation framework [3]. It's open and extendable, and it can work with almost any other tool to build powerful and adaptable automation solutions. It uses human-readable terms and has a simple syntax. Libraries written in Python can be used to expand its capabilities. New keywords may be defined using pre-existing keywords since keywords are composable. By supplying keywords, libraries offer Robot Framework with the actual automation and testing capabilities. The framework comes with a number of standard libraries, plus there are a plethora of individually built external libraries that may be installed.

CICD Automation Testing: Continuous Delivery (CD) and Continuous Integration (CI) are two key DevOps Software Development strategies that help organizations produce applications faster and more consistently [2]. Continuous Integration allows developers to automatically merge their code changes into a single repository on a regular basis, followed by automated builds and testing. As a consequence, developers can quickly discover and repair problems, resulting in a higher-quality output.

CICD Automated Testing with Jenkins and Robot Framework

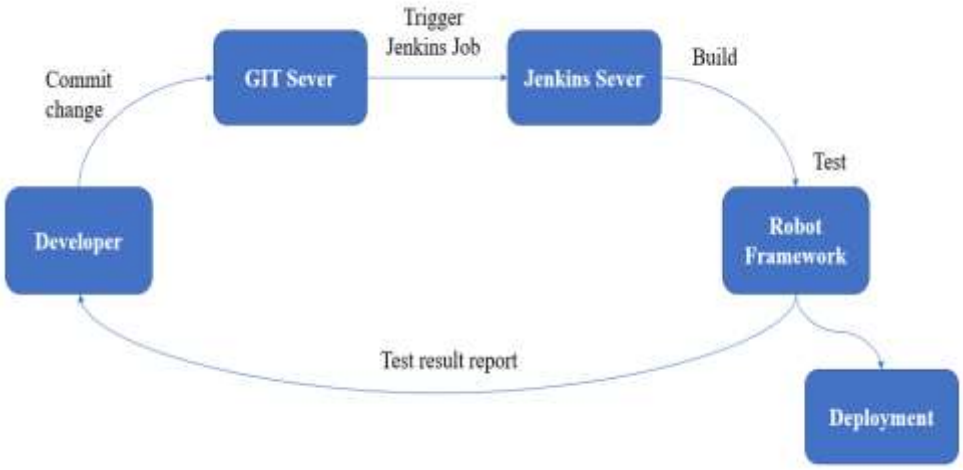

Figure. 2: Workflow of Continuous Integration Testing

Figure 2 represents steps followed in CICD pipeline. The testing process shown in Figure.2 is a conceptual representation of how software would go through the different phases of its lifecycle before being given to the customer or becoming live in production. Everything begins with the development of a testcase file in Robot format in a text editor, complete with the requisite key words, variables, tags, and suite configuration [5]. When a developer finishes coding, the change is committed to a central repository Git. It then moves on to the build step by activating a job in Jenkins, which offers us with a variety of interfaces and tools to help us automate the process. Jenkins pulls the code from Git and moves it forward to the commit step, where it is perpetrated from all branches. We compile the code in the build step. Because it is Python code in Robot format, Jenkins is used to build it and then compile it so that it can be deployed to perform a set of tests. Jenkins is once again in charge of these test cases. The code then moves on to the testing step when the build step is completed.
Steps:

- Login into the machine with Jenkins user
- Install Robot Framework Jenkins Plugin on Jenkins Server
- Jenkins Job Configurations includes the following steps,

  - Login to Jenkins Web application
  - Specification of job Name and Select Freestyle Project Type
  - Adding description to project
  - Setting up git to get automation code from a repository
  - Configuration of the build trigger in accordance with the requirements
  - Setting up the environment prior to the construction
  - Using Shell scripts to configure Robot test scripts
  - Robot framework test result analysis configuration for executed tests

- Making use of the Robot Framework Plugin as a Post-Build Action

- Jenkins Execution- By initiating a build, test cases may be run above Jenkins jobs. On the Jenkins page [6], the detailed test results are projected. Within Jenkins, the Robot Report may be accessed as a log file or as an HTML report.

IV. BENEFITS OF TECHNOLOGIES

Single RAN: A single RAN consists of multifunctional hardware and common software for GSM, WCDMA, LTE, and 5G technologies. Supporting a wide range of hardware sharing choices – radio, fronthaul, baseband, backhaul, OAM, spectrum – while also reducing complexity and increasing cost efficiency – all while providing a future-proof end-to-end solution with enhanced security and a single operating and administration interface.

CICD Automated Testing: Automation will generate outcomes more quickly than a human could manually. In order to release software on a consistent basis, speed and the avoidance of delays are critical. Test automation aids consistency by removing the possibility of human mistake during application testing. Agility is also aided [2].

Jenkins for Automation Testing: Jenkins will enable the user to schedule and run test automation scripts at a given time. It shows a graph of the test result trend [9]. Test failures from the previous build will also be included in this graph. Jenkins assists a tester in determining the cause of a test failure. It will show information such as an error message. In the post-build activity, it enables the Email Notification functionality.

Robot Framework for Automation Testing: RFW includes trend graphs for test suites and cases, environment variable expansion for build paths in configuration, output files for parsing and archiving multiple Robot result files, the option of evaluating only critical tests against pass thresholds, and a list view column in project listing and duration trend to show overall passed/failed tests.

V. CONCLUSIONS

This paper provides an overview of the Single RAN system, comprises of a installation of a single radio with a shared transport, management and operational system, as well as integrated comprehensive protection across radio access technologies, in order to get the greatest performance by coordinating their benefits. Also, by showcasing the benefits of software tools like Robot Framework and Jenkins, it focuses on replacing archaic ways of developing, testing, and deploying software products in favour of entirely automated CICD for SBTS construction.

VI. Acknowledgment

I take this opportunity to thank my guide, Prof. Smitha G.R, Assistant Professor, Department of Information Science & Engineering, RV COLLEGE OF ENGINEERING, Bengaluru for her valuable suggestions, support, regular source of encouragement and assistance throughout this project. I also thank senior authorities whose constant encouragement made this possible.

REFERENCES

[1] Nokia Solutions and Networks, “Single RAN Advanced Evolution: The future just got simpler”, white paper, June 2014.
[2] M. Shahin, M. Ali Babar and L. Zhu, "Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices," in IEEE Access, vol. 5, pp. 3909-3943, 2017.
[3] Q. Na and D. Huaichang, "Extension based on robot framework and application on Linux server," 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), pp. 293-296, 2017.
[4] S. A. I. B. S. Arachchi and I. Perera, "Continuous Integration and Continuous Delivery Pipeline Automation for Agile Software Project Management," Moratuwa Engineering Research Conference (MERCon), pp. 156-161, 2018.
[5] E. H. Kim, J. C. Na and S. M. Ryoo, "Test Automation Framework for Implementing Continuous Integration," Sixth International Conference on Information Technology: New Generations, pp. 784-789, 2009.
[6] S. Mysari and V. Bejgam, "Continuous Integration and Continuous Deployment Pipeline Automation Using Jenkins Ansible", International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pp. 1-4, 2020.
[7] M. A. Habibi, M. Nasimi, B. Han and H. D. Schotten, "A Comprehensive Survey of RAN Architectures Toward 5G Mobile Communication System," in IEEE Access, vol. 7, pp. 70371-70421, 2019.
[8] V. Armenise, "Continuous Delivery with Jenkins: Jenkins Solutions to Implement Continuous Delivery," IEEE/ACM 3rd International Workshop on Release Engineering, pp. 24-27, 2015.
[9] N. Seth and R. Khare, "ACI (automated Continuous Integration) using Jenkins: Key for successful embedded Software development," 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS), pp. 1-6, 2015.
[10] S. 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), pp. 22-25, 2014.