A project report on

RF CI/CD FRAMEWORK

Submitted in partial fulfillment for the award of the degree of

MTech Integrated Software Engineering

by

KOYYADA VISHNUVARDHAN (18MIS1098)

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

May,2023

RF CI/CD FRAMEWORK

Submitted in partial fulfillment for the award of the degree of

MTech Integrated Software Engineering

by

KOYYADA VISHNUVARDHAN (18MIS1098)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

May,2023

DECLARATION

I here by declare that the thesis entitled "**RF CI/CD FRAMEWORK**" submitted by me, for the award of the degree of Specify the name of the degree VIT is a record of bonafide work carried out by me under the supervision of

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place:chennai
Date: 31/05/2022

k.vishnuvardhan
Signature of the Candidate

INTERNSHIP COMPLETION CERTIFICATE

ABSTRACT

Nokia Networks is a multi – national telecommunication company headquartered at Espoo, Finland. Initially, the company started as a rubber and pulp manufacturing company, later extended its branches to wired communications and then shifted to wireless telecommunication. Nokia company has the pride of inventing the first wireless GSM (2G) voice call through Nokia's mobile phone. Nokia is currently working on 4G and 5G networks and is responsible to provide the eNodeB components required for 5G communication. For internship, the student trainee position was offered for Mobile Networks business group, in R&D department. The team is Software Configuration Management Radio Frequency (SCM RF).

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to M D SUNILKUMAR, TEAM MANAGER, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavor. My association with him / her is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of devops.

I would like to express my gratitude to **Dr.G.Viswanathan, Dr. Sekar Viswanathan**, **Dr.Rambabu Kodali**, **Dr. Kanchana Bhaaskaran V. S**, and **Dr. Ganesan R, School of Computer Science and Engineering**, for providing with an environment to work in and for his inspiration during thetenureofthecourse.

In jubilant mood I express ingeniously my whole-hearted thanks to **Dr. Nisha V.M, Program Chair and Associate Professor,** all teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project

Place: Chennai **Koyyada Vishnuvardhan**

Date: 31/05/2023 Name of the student

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

RF	Radio frequency
CI/CD	Continuous integration/continuous deployment
SCM	Software configuration management

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF NOKIA

Nokia Networks is a multi – national data networking and telecommunications equipment company headquartered in Espoo, Finland. It started on 12th May 1865 in Tampere, Grand Duchy of Finland, Russian Empire by Fredrik Idestam, Leo Mechilin and Eduard Polon as a joint venture between Nokia of Finland and Siemens of Germany known as Nokia Siemens Networks. Nokia Networks has operations in around 120 countries. In India Nokia has been connecting people since 1995, the first Global System for Mobile Communication (GSM) call was made on a Nokia handset over the Nokia network. From enabling growth of the 2G technology, bringing high quality 3G services, pioneering 4G to now steering India towards the 5G revolution. Every single call in India, touches a Nokia network element in the complete call flow and 1/3rd of overall mobile subscribers in India are served by Nokia Radio. Currently, 4 telecom operators rely on Nokia to provide high quality and far-reaching telecommunication. Nokia also supplies telecom infrastructure to the Indian Defence and Indian railways including the Kolkata Metro. Nokia is a strong supplier and service provider to the leading public and private operators, large enterprises, utilities companies etc.

At Nokia, the purpose is to create technology that helps the world act together. Nokia's commitment is to deliver critical networks through technology leadership and trusted partnerships. Nokia's essentials are guiding principles for the ways of working with and for Nokia are being open, fearless and empowered. Open – in mindset, to opportunity, through/with transparency. Fearless – bringing authenticity, sharing ideas and opinions, embracing collaboration. Empowered – to make decisions, to act with clear accountability. Currently Nokia Networks is head by Pekka Lundmark, as the President and CEO of the company. Nokia targets three customer segments with hardware, software and services portfolio: Communications service providers, enterprise verticals and hyper – scalers. India has approximately16,000 employees working at Nokia. The 5 key locations include Gurgaon, Noida, Mumbai, Bangalore, Chennai, project offices across 26 cities and one Innovation center: Technology Center in Bangalore.

Nokia Networks company located at Manyata Embassy, Bangalore. Bangalore is one of the major branches of Nokia located in India. Nokia has 7000+ employees working at Bangalore. R&D department is only located and Bangalore and Bell Labs is located at Chennai. Nokia Bell Labs, originally named Bell Telephone Laboratories (1925– 1984), then AT&T Bell Laboratories (1984–1996) and Bell Labs Innovations (1996–2007), is an American industrial research and scientific development company owned by Finnish company Nokia. With headquarters located in Murray Hill, New Jersey, the company operates several laboratories in the United States and around the world.

Researchers working at Bell Laboratories are credited with the development of radio astronomy, the transistor, the laser, the photovoltaic cell, the charge-coupled device (CCD), information theory, the Unix operating system, and the programming languages B, C, C++, S, SNOBOL, AWK, AMPL, and others. Nine

Nobel Prizes have been awarded for work completed at Bell Laboratories
Through Nokia Bell Labs and the company – wide investment in R&D, the company is committed to deliver networks and technology solutions that push the limits of science. A century of pioneering science and telecommunications leadership has enabled to create foundational technology in the electronics industry, the Internet and networking, optics, mobile and fixed communications industries. For more than a century, Nokia's unrivalled history of innovation has formed the foundation and shaped the future of technology across the spectrum of telecommunications.
From the transistor to the laser and solar cells to satellite communications Nokia's inventions have created positive industry disruption. Recognized as leading the technology evolutions that matter, Nokia is committed to delivering networks at the edge of science across mobile, infrastructure, cloud and enabling technologies and to nurturing diversity that makes Nokia a more vibrant and exciting place to work. Nokia has been recognized as the 2022 Microsoft Media and Communications Partner of the Year for achievements in cloudification, analytics, security, automation, and digitalized operations to enable our customers' digital transformation. Together with their standardization partners, Nokia won a NATAS Technology & Engineering Emmy Award for the standardization of HTTP encapsulated protocols, which are used for streaming media over the Internet. Nokia has been honoured six times by the Ethisphere Institute as one of the World's most Ethical Companies.
Ethisphere selected 136 companies representing 45 industries from 22 countries as the 2022 winners of this prestigious award. Nokia is one of five winners in the telecommunications industry and the only Finnish company to be honoured. Nokia was listed in Bloomberg's Gender Equality Index (GEI) for the fourth consecutive year. The Bloomberg GEI tracks the performance of public companies committed to supporting gender equality through policy development, representation and transparency

1.2 Organization Structure

Nokia's renewed operating model is designed to enable the delivery of strategic ambitions, with a lean corporate center enabling fully accountable business groups. Nokia has four business groups with each business group aiming to become a technology and market leader in their respective sector. Fig. 1.1 provides the different business groups structured at Nokia.

Fig. 1.1.1 Organization Structure – Business Groups

Mobile Networks: Mobile Networks (MN) provides products and services for radio access networks covering technologies from 2G to 5G, and microwave radio links for transport networks. Cloud and **Network Services**: Cloud and Network Services enables Communication service Provider (CSPs) and enterprises to deploy and monetize 5G, cloud-native software and as-a-Service delivery models.

Network Infrastructure: Network Infrastructure provides fiber, copper, fixed wireless access technologies, IP routing, data center, subsea and terrestrial optical networks – along with related services – to customers including communications service providers, webscales (including hyperscalers), digital industries and governments

Nokia Technologies: Nokia Technologies is responsible for managing Nokia's patent portfolio and monetizing Nokia's intellectual property including patents, technologies and the Nokia brand.

1.3 Products and services

okia is a leading vendor in the network and IP infrastructure, software, and the related services market. They provide a broad range of products, from the hardware components of networks used by communication service providers and increasingly by customers in other select verticals, to software solutions, as well as services to plan, optimize, implement, run and upgrade networks. Products and services of Nokia is broadly classified into different domains. They are Business Support System/ Operator Support System (BSS/OSS), Core Networks, Data Center, Fixed Networks, IP Networks, Mobile Networks, Optical Networks, Private Networks, Security, solutions for industry. The brief descriptions of Nokia Products and Services are as below.

1.3.1 BSS/OSS

Software Solutions to Telcos AVA helps to secure data, automate operations and monetize assets. AVA infuses Nokia's OSS, BSS and security software with "Intelligence Everywhere" through Artificial Intelligence (AI) and machine learning. Products include AVA – Network data analytics function, AVA open analytics framework, Deep – field Cloud Intelligence, Service management platform, Flowone, NetAct, Nokia Assurance center, etc. Services include Worldwide IoT network Grid (WING), Software Support Services, AI for network efficiency, etc.

1.3.2 Core Networks

Core networks differentiate business by enabling to grow, with the flexibility to choose the path that best fits technology needs, optimizes investments, enables a smooth evolution of legacy services and delivers extreme new services for enterprises and consumers. The products include Cloud Mobile Gateway, Cloud Mobility manager, Nokia Registers – Home Subscriber Service (HSS), One – Network Directory Server (NDS), Cloud Signaling Director, Network Exposure Function, Nokia Policy Controller, Session Border Controller, etc. Services include Cloud Deployment, Cloud Support Services, Core Engineered Systems, Service lifecycle automation for 5G core, Core Engineered Systems, etc.

1.3.3 Data Center

Data center fabric solution provides openness, extensibility and unparalleled visibility to any cloud while providing built – in intent – based automation capabilities that embrace the cloud native era. The products include Interconnect Router for data center fabrics, Edge network controller, Fabric services system, Network services platform, Service Router (SR) Linux, Airframe data center manager, Airframe data center fronthaul gateway, Airframe open edge server, etc. The services include Adaptive Cloud Networking, Data Center Fabric, Airframe data center, etc

Nokia is a leading vendor in the network and IP infrastructure, software, and the related services market. It provides a broad range of products, from the hardware components of networks used by communication service providers and increasingly by customers in other select verticals, to software solutions, as well as services to plan, optimize, implement, run and upgrade networks.

Core Networks

Nokia core networks differentiate your business by enabling you to grow, with the flexibility to choose the path that best fits your technology needs, optimizes your investments, enables a smooth

evolution of legacy services and delivers extreme new services for enterprises and consumers. Evolve now with clarity, remove the limitations and explore new opportunities. Envision new horizons.

Products: 7750 Service Router Mobile Gateway, Cloud Mobile Gateway,
Cloud Mobility Manager, Home Subscriber Server, One-NDS, Cloud Signaling Director, Network Exposure Function, Nokia Policy Controller, Session Border Controller, CFX-5000 SIP Session Control, Telecom Application Server.

Data Center

Nokia data center fabric solution provides openness, extensibility, and unparalleled visibility

to any cloud while providing built-in intent-based automation capabilities that embrace the cloud native era. In addition to the networking capabilities, They provide scalable computing from a single server to hyper scale for every use case with lean operations. The Cloud computing capabilities offers the flexibility that is needed to enable modern, agile business opportunities with secured and high availability services.

Products: 7220 Interconnect Router for Data Center Fabrics, 7250 Interconnect Router for data center fabrics, Fabric Services System, Network Services Platform, SR Linux, AirFrame Data Center, AirFrame Data Center Manager, AirFrame Fronthaul Gateway, AirFrame Open Edge Server, AirFrame Rackmount server, Edge Automation Tool, Service Enablement Platform.

Fixed Networks

Brings ultra-broadband faster and at the right cost. Ultra-broadband enhances the lives of everyone it reaches. With the extensive portfolio of fixed network services and solutions spanning copper, cable, fiber and wireless technologies, you can bring ultra-broadband services to more people, more quickly, at the right cost for your business.

Products: ISAM FD (ETSI), Lightspan DF fiber access nodes, Lightspan FX, Lightspan MF fiber access nodes, Lightspan SF-8M Sealed Fiber access Node, Lightspan SX/DX, Altiplano Access Controller, Fiber ONT.

Internet of Things

Nokia IoT innovations enable service providers, enterprises and governments to make sense of the massive volumes of data produced by connected sensors, devices and systems. They can help you capture the data that matters, manage it securely and use it to generate insights that create value and improve business outcomes.

They support the full IoT value chain with capabilities that include connectivity, analytics, security and platforms for device management and data collection. Their portfolio gives you the flexibility to choose solutions, products, and services that address your unique needs. It provides all the building blocks you need to unleash the full potential of the IoT.

Products: Service Management Platform, iSIM Secure Connect, NetGuard Endpoint Detection and Response.

IP Networks

Nokia IP networks portfolio lets you combine massive capacity with unmatched performance, security and efficiency. It is powered by breakthrough silicon and software innovations, along with intent-based network automation. With the products and solutions, you can build an IP network that you can trust, which accelerates time to market, reduces operating expenses and drives sustainable growth. Master the unexpected and embrace new cloud, 5G, broadband, mission-critical network and Industry 4.0 opportunities with Nokia IP networks.

Products: 7210 Service Access System, 7250 Interconnect Router, Virtualized Service Router, Deepfield Cloud Intelligence, Deepfield Video Analytics, Edge Network Controller, SR OS Connect, IPsec Security Gateway.

Mobile Networks

Nokia create technology that helps the world act together. The solutions enable service providers, industries and the public sector to create the critical networks that bring together the world's people, machines and devices. The mobile network portfolio includes RAN and MWR products,

associated network management solutions, as well as network planning and optimization, network deployment and technical support services as shown in Fig 1.1.
Products: AirScale Active Antennas, Smart Node Femtocells, AirFrame Data Center Manager, Edge Automation Tool, Automation and Analytics, Compact Mobility Unit, EdgeNet SON.

Optical Networks

Nokia optical network products and solutions enable scaling without limits to enable the foundational connectivity for your networked communications. They provide the solutions to scale from the network edge and across long-haul/core and subsea links, while simplifying network operations to let you build smarter, more automated networks that streamline service delivery and lower network TCO. By enabling new services such as 5G and packet services, while enabling more efficient network architectures, They help stay competitive and turn your optical networks into platforms that create value for your business and your customers.
Products: 1830 Photonic Service Interconnect-Modular (PSI-M), 1830 Optical Network Extender (ONE), WaveLite Enterprise Optical Networking, Optical sub-systems, Network Services Platform.

Private Networks

Digitalization is essential to achieve higher operational efficiency, safety and sustainability. Technological advances under the umbrella of Industry 4.0 are enabling companies to become data-driven and to adopt zero touch automation to transform their operations. Nokia One Platform for Industrial Digitalization is designed to simplify digital transformation for industries and deliver secure, reliable and high-performing wireless infrastructure.
Products: Compact Mobility Unit, Nokia Digital Automation Cloud Wi-Fi.

Security

The network is facing growing attacks both on the network itself and on the data traveling over the network. Proliferating IoT devices increase the number of attack openings. We offer security products and services to address a variety of security concerns. Learn how you can identify threats more quickly, stop them automatically and take fast remediation actions when needed. Protect the network from degradation and deliver the security protection guarantees in your service level agreements.
Products: NetGuard Audit Compliance Manager, Deepfield Defender, iSIM Secure Connect, Threat Intelligence.

Services

Maximizes the potential of your network. Delivers superior end user experiences, seize new revenue opportunities and optimize network and operational performance through our comprehensive services portfolio. Nokia services teams offer expertise and global reach that will enable you to realize the potential of technology.

Cloud Network Services

Nokia Cloud and Network Services can smooth the path to embracing disruption and developing new business models. They can advise on how to properly plan for these disruptions using our unique knowledge of 5G technology and business, cloud networking experience, analytics and security insights. They are your trusted partner to help you deploy, integrate and scale our industry-leading telco software solutions and all generations of core networks in multi-vendor environments. the global multi-vendor, multi-network experience

can help you execute with confidence, optimize performance and operations and achieve your business results.
Services: Advanced Telecom Consulting Services, Cloud Deployment, Cloud Support Services.

Fixed Network Services

Brings ultra-broadband faster and at the right cost. Ultra-broadband enhances the lives of everyone

it reaches. With the extensive portfolio of fixed network services and solutions spanning copper, cable, fiber and wireless technologies, you can bring ultra-broadband services to more people, more quickly, at the right cost for your business. To create your broadband network of the future. Accelerate new broadband revenues. Network optimization and care. Network transformation.
Services: Cloud Acceleration Services, ONT Easy Start, Network Assessment and Health Check, PSTN Smart Transform, Multivendor Care, Predictive Care, Outside Plant.

Services for industry and the public sector
Industrial and public sector customers need solutions to their business problems, not technologies. Nokia use case driven approach helps our customers to maximize the business value from their digital transformation investments.
The underlying basis for digitalization is the Digital Network Architecture which allows us to transform the service delivery across the entire lifecycle of design, build, operate and maintain. Digitalization of manual procedures drives efficiency, enhances quality and enables enterprises to benefit from business agility. Customers can track the status and progress of deployments and operations in real time through a single pane of glass.
Services: Cloud Acceleration Services, Managed Security Services, Managed operations services, Services for private wireless.

Services for mobile networks

With the increasing adoption of 5G, the network complexity is growing, which requires intelligent solutions to keep the subscribers connected and happy. Investing in the best radio

products is the first step. To make a significant impact on your business outcome, an equally important step is actively driving the entire lifecycle of your radio network. We have designed our Services for Mobile Networks to support you all the way.
Services: Digital deployment services, Network Planning and Optimization, Technical support services.

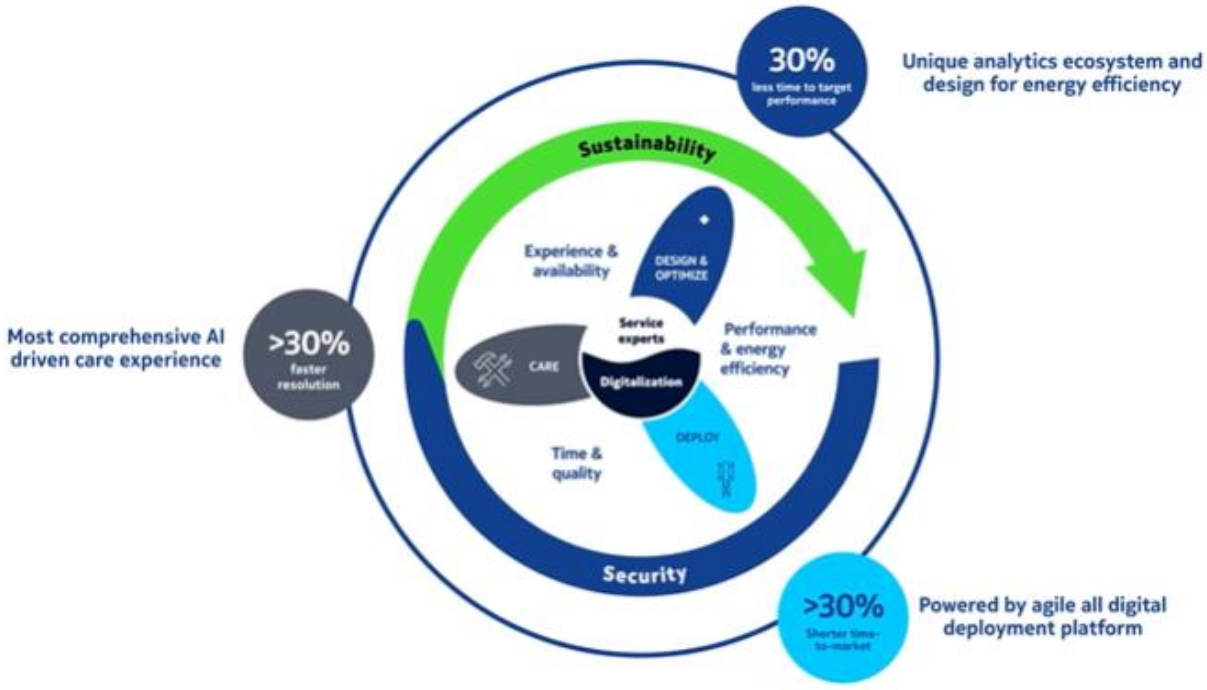

Fig 1.3.1: Business value of the mobile network

Managed Operations

These are just some of the immediate business outcomes Nokia Managed Operations has achieved for a growing global customer base of communications service providers (CSPs) and enterprises. Managing and optimizing multi-vendor network operations across mobile, fixed, and IP/Optical networks is the daily business. No two CSPs' 5G journeys are exactly alike. The modular portfolio of targeted operations services and business models gives you support for just the issues you need to fix to achieve your specific operational and digital transformation targets. Nokia has three managed service offerings designed to address these issues. Each service type consists of a modular catalog of operations services with laser focus on tangible outcomes.

Services: Managed Services for Public Safety, Managed Services for Rail, Services for Utilities.

Manpower

Nokia has employed approximately 87927 people across over 100 countries did business in more than 130 countries.SCM team has around eighty members out of this four members in the rf team

Professional Practices :

Nokia Solutions and Networks provides a healthy working environment for its employees and staff, with a clean and hygienic workplace. Smart lighting systems have been adopted, in order to conserve energy in the workplace.

Nokia Solutions and Networks senior leader provides intensive training sessions for mental health wellbeing of employees. The sessions mainly concentrate on building good qualities so that they can positively handle stressful situations in life. The sessions are also conducted on time management, emotional intelligence and effective communication skills. Sessions are very authoritative as they are conducted by experienced and senior leaders of the organization. Regular trainings are conducted to enhance the technical skills of the employees.

1.4 DATA CENTER

Nokia had an acceleration in its financial stability. Nokia follows a quarterly analysis. The summary of year 2022 is described here. Q4 net sales grew 11% y-o-y in constant currency (16% reported). Full year net sales grew 6% (12% reported). In Q4 Network Infrastructure grew net sales 14% in constant currency with all units contributing. MN grew 3% with a meaningful shift in regional mix in the quarter while Cloud and Network Services grew 5%. Nokia Technologies grew 82% as a long-term licensee exercised an option leading to higher revenue recognition in Q4. Enterprise net sales grew 49% y-o-y in constant currency in Q4 (55% reported); 21% in full year 2022 (27% reported). Nokia expects 2023 full year net sales 8 of between EUR 24.9bn to 26.5bn, comparable operating margin between 11.5 to 14.0% and Free Cash Flow conversion from comparable operating profit of 20 to 50%

1.5 Societal Concerns

Nokia's Corporate Social Responsibility (CSR) programs actively seek to improve overall quality of life and have a positive impact on society. Nokia's activities are spread across 11 states in India. It also works globally with Governments, Non – Governmental Organization (NGOs), customers and vendors to drive access for all. The 5 key pillars are – education, healthcare, e – governance, financial inclusion, livelihood. Nokia also takes projects with respect to societal concerns. Two major projects are described below.

1.5.1 Smartpur

Digital Village Eco-system Nokia has developed its flagship project called Smartpuras a means of driving rural India to its digital future. It transcends the idea of a 'smart' village as simply technologically equipped to a model that seamlessly integrates such technology into daily lives of

all. Also, it supports education for out-of-school children from migrants by providing remedial classes. A range of activities have been designed, developed and implemented at various Smartpur locations across the country. The five pillars of health, education, governance, livelihoods and financial inclusion have been developed specifically taking into consideration the demography, geography, economic conditions of the area as shown in fig. 1.2.

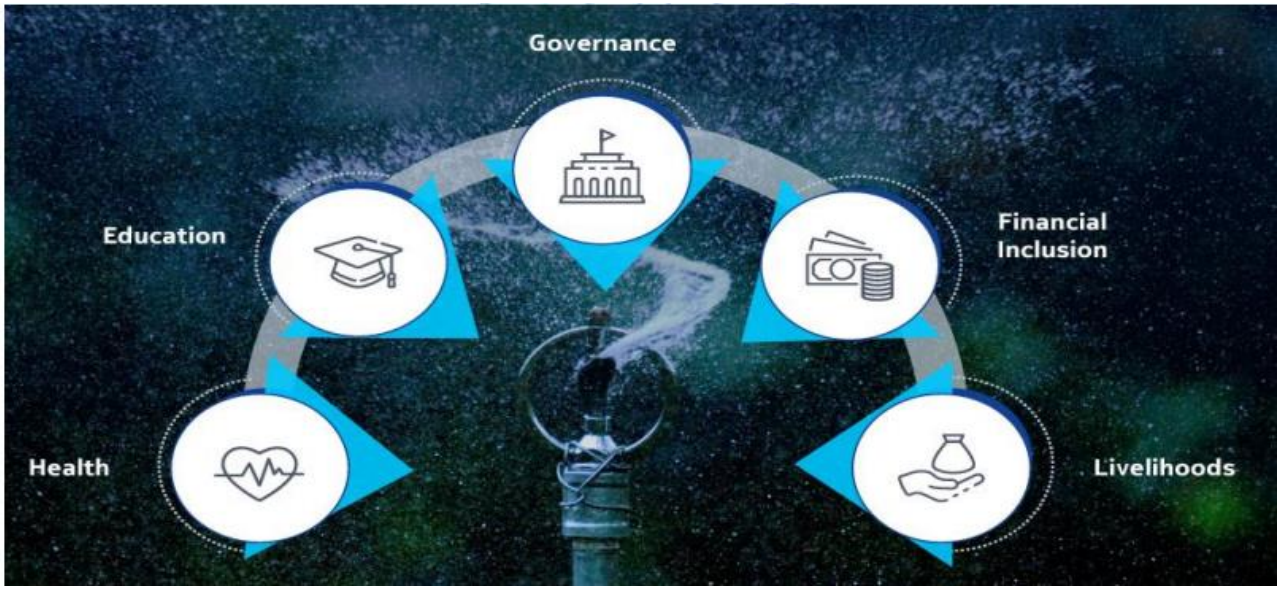

Fig. 1.2 The Five Pillars of Nokia Smartpur Initiative

1.5.2 Covid – 19

Nokia assisted in humanitarian response through provision of essentials such as PPE kits, face masks and sanitation material. Also, it collaborated with various hospitals and supported in providing medical equipment including oxygen plants

1.6 Environment and Sustainability

Nokia aims to be the leader in energy efficiency building on its Silicon, software, and systems. To optimize across the network with energy orchestration and green operations. It also aims to improve energy efficiency in 5G-Advanced and 6G through early engagement in standardization and ecosystem development. Nokia focuses on hardware circularity, both in the use of recycled material in its own products and the refurbishment and recycling of products when removed from customers networks. Nokia's approach to sustainability centers around creating technology that helps world act together. It believes the positive social impactof digitalization and enhanced connectivity – its handprint, far outweighs its potential negative social impact – its footprint.

Currently, 789 million people around the world are without access to any electricity, and millions more lack a reliable energy supply. Surging fuel prices exacerbate this problem, leading millions of people into energy poverty. Many households are facing a choice between heating and eating. At the same time, the world faces another crisis, carbon emissions are warming the atmosphere rapidly, and burning fossil fuels for energy is among the main contributors. As a result, an increasing number of people are considering off – grid living

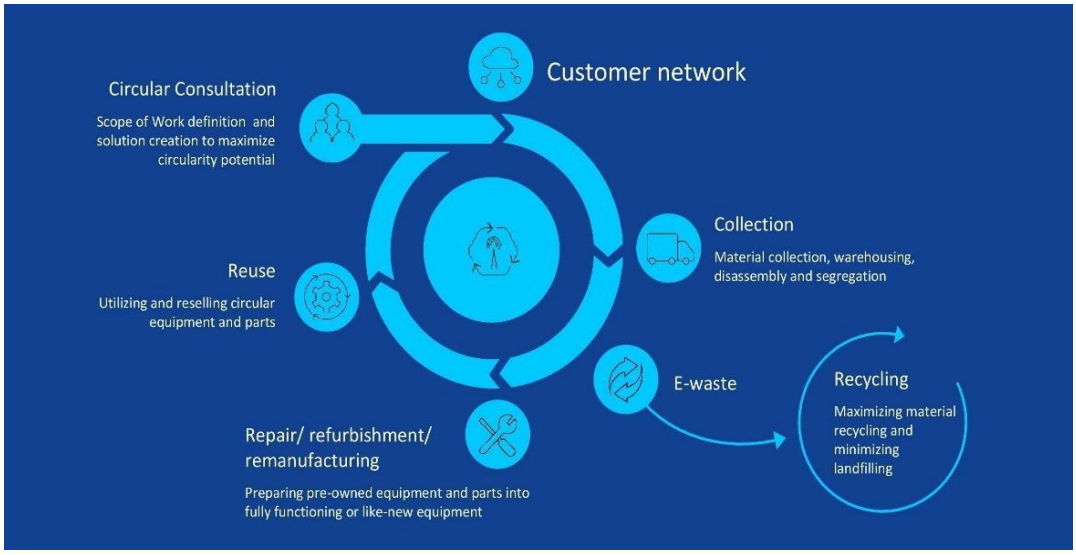

Fig. 1.3 Nokia's Environmental Concern

1.7 Organization of Report

The report is organized as follows. Chapter 1 gives the details of profile of the organization. Chapter 2 explains the overview of departmental structure and its activities carried out. Chapter 3 gives the implementation procedure of the tasks performed during internship tenure.

1.8 ACTIVITIES OF THE DEPARTMENT

Mobile Networks COO COSI Software Control Management is the working team where the task of tracking and controlling changes in the software is performed.Also, continuous integration and continuous deployment is performed. CI/CD
bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications
Mobile Networks COO COSI Software Control Management is the working team where the task of tracking and controlling changes in the software is performed. Also, continuous integration and continuous deployment is performed. CI/CD
bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications.

Mobile Networks COO COSI

Mobile Networks Chief Operating Office unit aspires to elevate the new MN to a world-class level in overall operations, including R&D operation, sourcing, supply chain, quality management, capacity management, site strategy and new organization integration, in addition to acting as the real estate and IT interface for MN.
Mission: To take the new Mobile Networks to world-class level in common services and infrastructure. Enable MN to reach its short-term and long-term profitability targets. Develop cost-efficient, digital and automated business infrastructure with tools, labs, common services
(Central SCM, security, customer documentation), asset/CAPEX management, Real Estate, and IT.
The key responsibilities of COSI (Common Services and Infrastructure) unit include:

- MN CAPEX + associated expensive lab/test equipment TCO management
- MN central labs (including R&D labs, Service Labs and Shared Reference Network SRN)

• MN central clouds (telco cloud, NESC, Amazon/Azure/Google public cloud internal use)
• MN common tools and their harmonized usage
• MN-IT interlock + associated MN level cost management
• MN-P&P (Real Estate) interlock + associated MN level cost management
• MN R&D central SW build (SCM) infrastructure and services
• MN Security & Privacy
• MN Customer Documentation

Software Configuration Management

Software Configuration Management (SCM) is a task of tracking and controlling changes in the software. It also takes care of integrating changes in smoother fashion across different units.

CI/CD is the combined practices of continuous integration and either continuous delivery or continuous deployment. It bridges the gaps between development and operation activities and teams by enforcing automation in building, testing and deployment of applications. CI/CD services compile the incremental code changes made by developers, then link and package them into software deliverables.

Continuous integration

➢ The main aim of CI is to prevent integration problems.

➢ CI is intended to be used in combination with automated unit tests written through the practices of Test-Driven Development

➢ This mainly makes use of build servers, which automatically runs the unit test periodically or even after every commit

➢ This also helps Quality Analysts to for Quality Analysis process

➢ Continuous Delivery process extends CI by making sure that the software checked in on the mainline is always in a state that can be deployed to users, and thus makes the deployment process fast.

Workflow for continuous integration (CI)

➢ When embarking on a change, a developer takes a copy of the current code base on which to work. As other developers submit changed code to the source code repository, this copy gradually ceases to reflect the repository code.

➢ The longer a branch of code remains checked out, the greater the risk of multiple integration conflicts and failures when the developer branch is reintegrated into the mainline.

➢ When developers submit code to the repository, they must first update their code to reflect the changes in the repository since they took their copy.

About Sack

SACK – is a collection of components containing basic types, constants, definitions used in BTS software (C/C++ source code/header files), common definitions across products. faults,counters and messages/interfaces definition which are commonly used/referred to by different components.

Functions of Sack

SACK Team creates and maintains the CI of these components. Usually, the contents of the components' repository are NOT SACK Team's responsibility. Responsibility of the content is on the developers/contributors of that component. SACK team's responsibility mainly lies in identifying a change and delivering the tag and other artifacts corresponding to that change, for each component. SACK component CIs deliver the tag to WFT and

bitbake file(s) to the integration repository. The tag delivered to WFT is added to the ECL file in the ECL_SACK_BASE repository (by Integration Managers, NOT SACK team) so that the tags can be used to define the CI/build environment in all dependent components. Hence common definitions that are used across multiple sub-components are usually delivered via SACK as common components. Exceptions to this are due to historical reasons and the exceptions going forward will be well documented with reasons in the FOT documents and SACK documents for that component.

CHAPTER 2

TECHNOLOGIES LEARNT

2.1 GIT

A version control system, or VCS, tracks the history of changes as people and teams collaborate on projects together. When a developer makes changes to a project, any previous version of the project can be reverted at any time.
Developers can review project history to find out:
• What changes have been made?
• Who made the change?
• When was the change made?
• Why is the change necessary?
VCSs provide a unified and consistent view of the ongoing work for each contributing project. Seeing a clear history of changes, who made them and how they contributed to the project helps team members stay on track while working independently.

In a distributed version control system, each developer has a complete copy of the project and project history. Unlike the once popular centralized version control systems, DVCSs do not require a permanent connection to a central repository. Git is the most popular distributed version control system. Git is commonly used for open source and commercial software development, with significant benefits for individuals, teams, and businesses.
• Git allows developers to see the changes, decisions, and progress of any project at once. From the moment they enter the project's history, developers have all the context to understand it and start contributing to it.
• Creators always work in the same time zone. With DVCS like Git, collaboration can happen at any time while maintaining the integrity of the source code. Using a fork, developers can safely propose changes to the production code.
• Businesses using Git can eliminate communication barriers between teams and focus on the best work. In addition, Git allows you to align experts in the project to collaborate on large projects.

Some basic git commands
To use Git, developers use special commands to copy, create, modify, and merge code. These commands can be executed directly from the command line or using a program such as GitHub Desktop. Some common commands for using Git:
• git init starts a new Git repository and starts tracking existing directories. Add a hidden folder inside the existing directory that contains the internal data structure needed for version control.
• git clone Create a local copy of a remote project. A clone contains all files, history, and project branches.
• add changes to go. Git tracks changes to a developer's codebase, but requires staging and capturing changes to add to the project's history. This command performs staging, the first part of a two-step process. Any staged changes will be part of the next snapshot and project history. Separate staging and execution gives developers complete control over their project history without changing their code or functionality.
• git commit commits the image to the project history and completes the change tracking process. In short, commitment is like photography. Anything staged with git will be part of git's capture and git.
• git status indicates the status of unmonitored, modified, or staged changes.
• git branch shows the local running branch.
• git merge merge development trends. This command is usually used to merge changes made in two separate branches. For example, developers will merge when they want to merge into the

master branch to propagate changes from the feature branch.
• git pull updates the local development pipeline with updates from remote peers. Developers use this command if a colleague has committed to a remote branch and they want to reflect those changes in their local environment.
• git push updates the remote repository with the commits associated with the new branch.

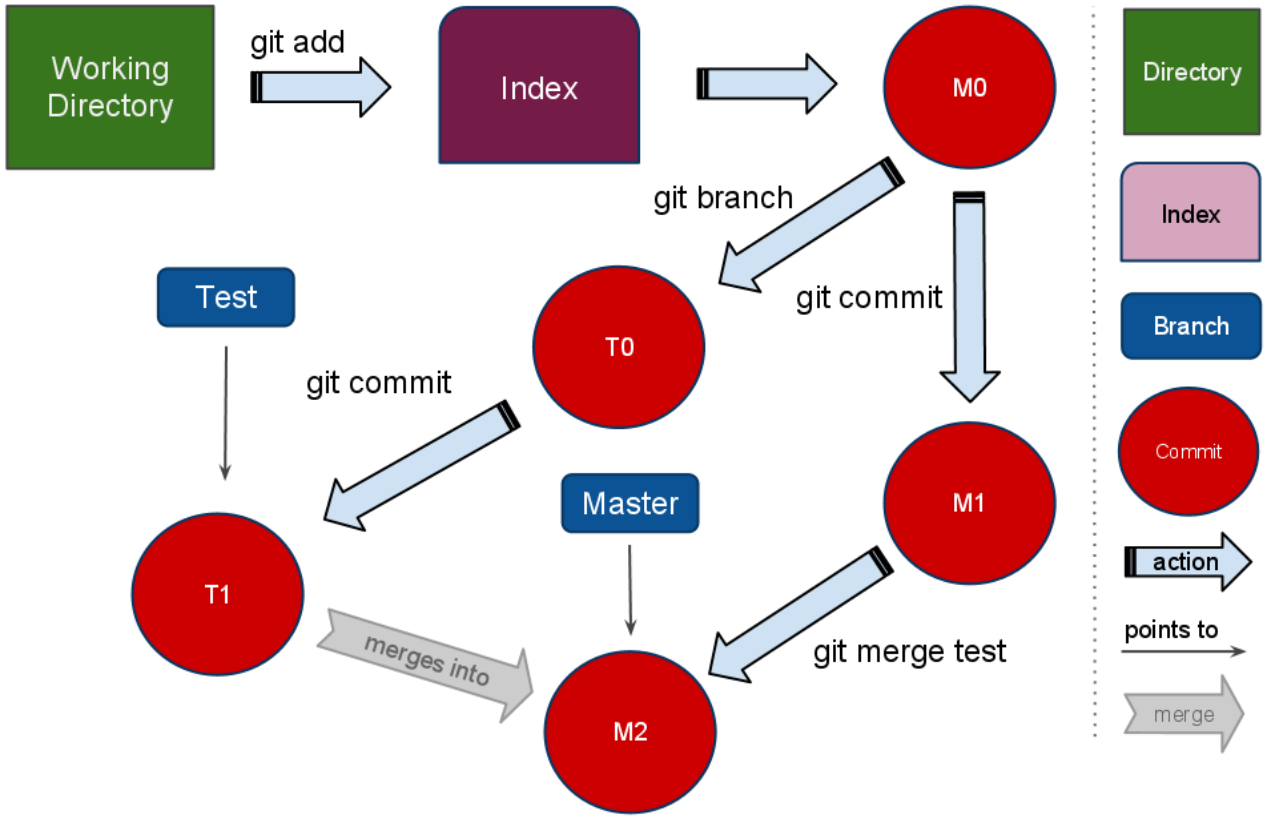

Fig 2.1.1 git working flow

2.2 JENKINS

Jenkins is a self-contained open-source automation server that can be used to automate all kinds of tasks, including building, testing, and deploying software. Jenkins native packages can be built using Docker, or run independently by any machine with the Java Runtime Environment installed. Jenkins Pipeline is a set of plugins that support the implementation and integration of continuous delivery pipelines into Jenkins. Pipeline provides an extensive toolset for simple modeling to complex "as code" delivery pipelines.
A Jenkinsfile is a text file that contains Jenkins pipeline definitions and is checked into source control. This is basically "Plug-code"; Maintenance is part of the continuous delivery pipeline that requires versions and reviews, just like any other code. Creating a Jenkinsfile provides several immediate benefits:
• Automatically generate pipelines for all branches and draw requirements
• Review / repeat the code in the debugger
• inspection route for plumbers
• A single source of truth for the Pipeline that can be viewed and edited by multiple project members.
Although the syntax to define a pipe is the same as in the web UI or Jenkinsfile, it is generally considered the best practice to define a Jenkinsfile pipe and check for source control.

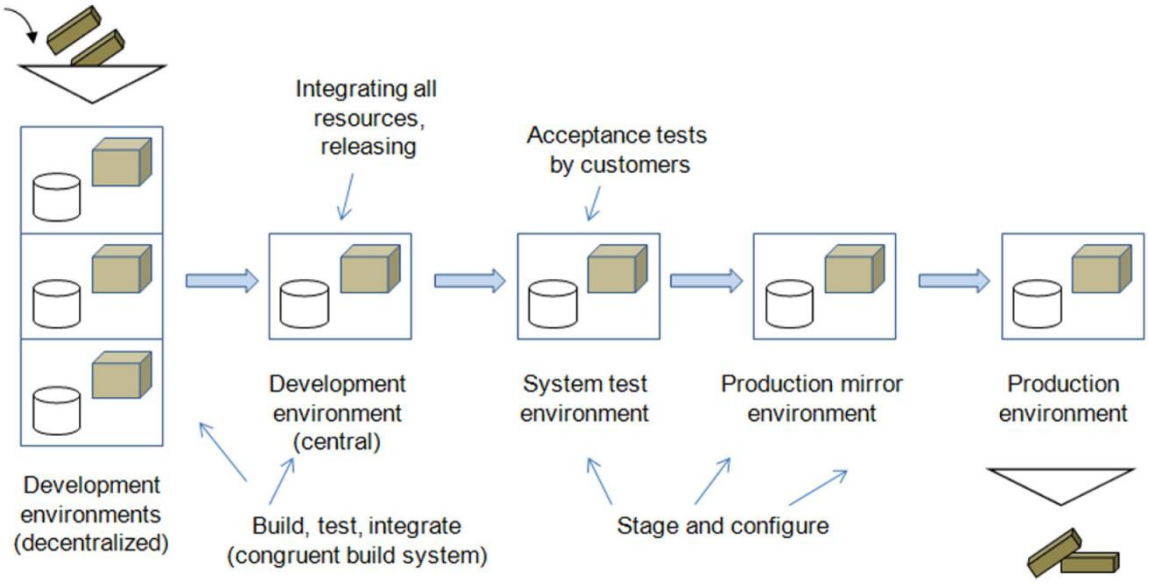

Fig 2.2.1 jenkins work flow

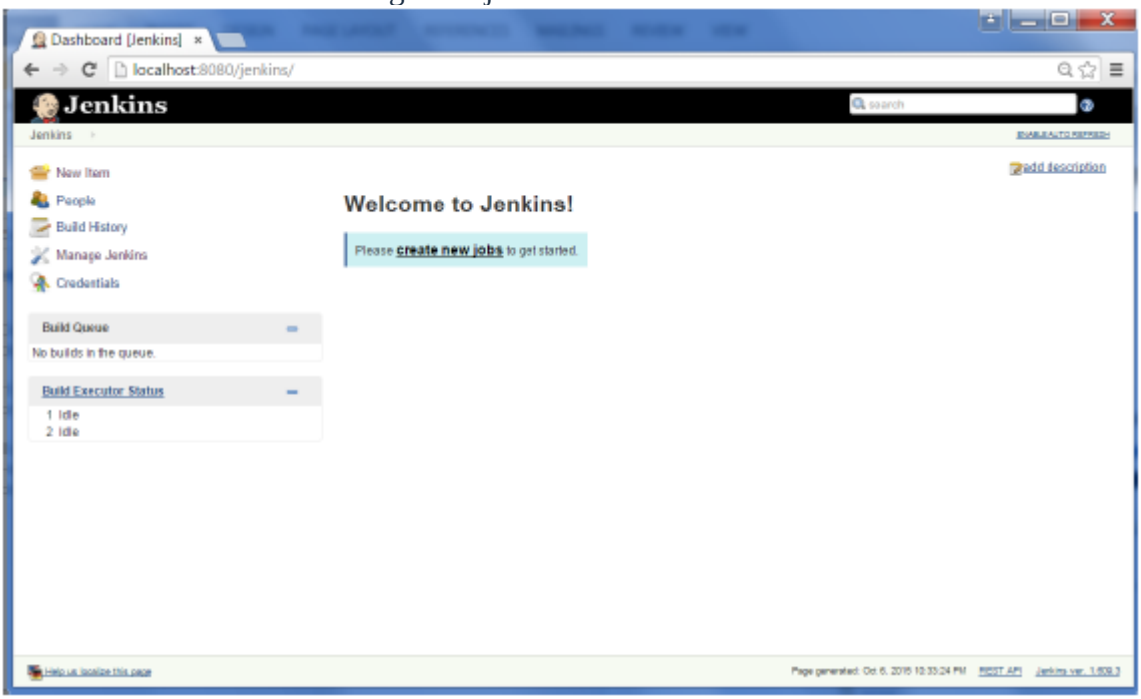

Fig 2.2.2 Home page of Jenkins.

Jenkins Pipeline

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a set of plugins that support the implementation and integration of continuous delivery pipelines into Jenkins.

Continuous delivery (CD) is the automated expression of your process for getting software from version control directly to your users and customers. Every change to your software (committed under source control) goes through a complex process on its way to release. This process involves creating software in a reliable and repeatable way, as well as progressing the created software (called a "build") through several stages of testing and deployment.

Pipeline provides an extensible set of tools for modeling simple to complex delivery channels "as

code" through the Pipeline Domain Specific Language (DSL) syntax.

The Jenkins Pipeline definition is written to a text file (called a Jenkinsfile), which in turn can be committed to the project's source control repository. [2] This is the basis of "Pipeline-as-code"; treating the CD channel as part of the application to be versioned and checked in like any other code.

Creating a Jenkinsfile and committing it to source control provides a number of immediate benefits:
Automatically creates a Pipeline build process for all branches and pull requests.
Code review/iteration on Pipeline (along with remaining source code).
Audit trail for pipelines.
A single source of truth for Pipeline that can be viewed and edited by multiple project members.
While the syntax for defining a Pipeline, whether in the Web UI or using a Jenkinsfile, is the same, it is generally considered best practice to define a Pipeline in a Jenkinsfile and check it in source control.

Declarative versus scripted pipeline syntax

A Jenkinsfile can be written using two types of syntax - declarative and scripted.
Declarative and scripted channels are constructed fundamentally differently. Declarative Pipeline is a newer feature of Jenkins Pipeline that:
provides richer syntactic features over the Scripted Pipeline syntax andis designed to make Pipeline code easier to write and read.However, many of the individual syntactic components (or "steps") written to the Jenkinsfile are common to both the declarative and scripted pipelines. Read more about how these two types of syntax differ in the Pipeline Concepts and Pipeline Syntax Overview sections below.

Why Pipeline

Jenkins is essentially an automation engine that supports a number of automation patterns. Pipeline adds a powerful set of automation tools to Jenkins that support use cases that range from simple continuous integration to complex CD pipelines. By modeling a series of related tasks, users can take advantage of Pipeline's many features:
Code: Pipelines are implemented in code and typically reviewed in source control, allowing teams to modify, review, and iterate on their delivery pipeline.
Resilient: Pipelines can survive scheduled and unscheduled restarts of the Jenkins driver.
Pause: The pipeline can optionally pause and wait for human input or consent before continuing to run the pipeline.
Versatile: Pipelines support complex real-world CD requirements, including the ability to branch/merge, loop, and execute work in parallel.
Extensibility: The Pipeline plugin supports custom extensions to its DSL [1] and multiple integration options with other plugins.
While Jenkins has always allowed basic forms of chaining Freestyle Jobs together to perform sequential tasks, Pipeline makes this concept a first-class citizen in Jenkins.

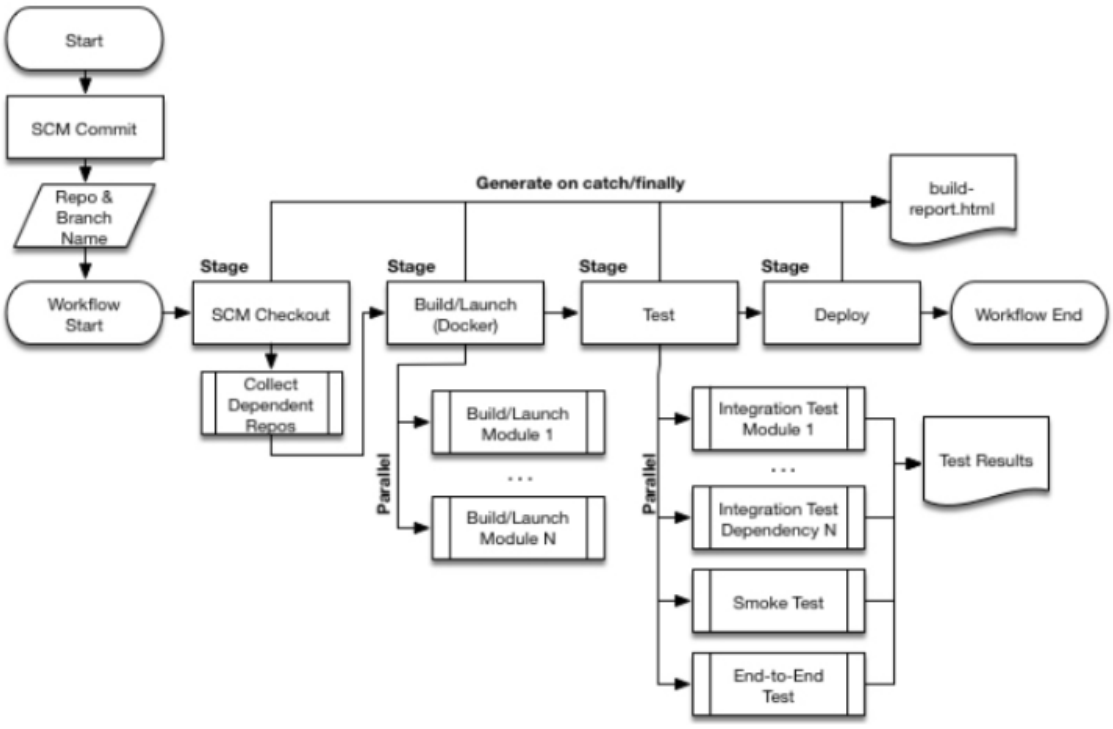

Fig 2.2.1 CD scenario easily modeled in Jenkins Pipeline

Pipeline concepts

The following concepts are key aspects of the Jenkins Pipeline that are closely related to the Pipeline syntax (see overview below).

Conduit

A pipeline is a user-defined CD pipeline model. Pipeline code defines your entire build process, which typically includes the stages of building your application, testing it, and then shipping it. The pipe block is also a key part of the declarative pipe syntax.

Node: A node is a machine that is part of the Jenkins environment and is capable of running a Pipeline. Also, the node block is a key part of the Scripted Pipeline syntax.

Phase: A phase block defines a conceptually distinct subset of tasks performed throughout the pipeline (eg the "Build", "Test" and "Deploy" phases) used by many plugins to visualize or present the state/progress of the Jenkins Pipeline. [6]

Step

A single task. Basically, a step tells Jenkins what to do at a certain point in time (aka a "step" in the process). For example, to run a shell command, use the sh step: sh 'make'. When a plugin extends a Pipeline DSL, it usually means that the plugin has implemented a new step.

Overview of pipeline syntax

The following pipe code skeletons illustrate the basic differences between declarative pipe syntax and scripted pipe syntax.

Note that both phases and steps (above) are common elements of both declarative and scripted pipeline syntax.

Basics of declarative piping

In declarative pipeline syntax, a pipeline block defines all the work done in your entire pipeline.

```
Jenkinsfile (declarative pipeline)
pipeline {
   any agent
   phase {
      phase('Build') {
```

```
        steps {
            //
        }
    }
    phase('Test') {
        steps {
            //
        }
    }
    stage('Deploy') {
        steps {
            //
        }
    }
  }
}
```

Run this pipeline or any of its stages on any available agent.
Defines the "Build" phase.
Perform some steps related to the "Build" phase.
Defines the "Test" phase.
Perform some steps related to the "Test" phase.
Defines the "Deploy" phase.
Perform some steps related to the "Deploy" phase.

Scripted pipeline basics

In Scripted Pipeline syntax, one or more node blocks perform the basic work in the entire pipeline. Although not a mandatory requirement of the Scripted Pipeline syntax, constraining the work of your pipeline inside a node block does two things:Schedules the steps contained in the block to run by adding an entry to the Jenkins queue. As soon as the executor is free on the node, the steps will take place.Creates a workspace (a directory specific to that particular channel) where files borrowed from the source control can be worked on.

Note: Depending on your Jenkins configuration, some workspaces may not automatically clean up after a period of inactivity. For more information, see the tickets and discussion linked from

Jenkinsfile (scripted pipeline)

```
node {
  phase('Build') {
    //
  }
  phase('Test') {
    //
  }
  phase('Deploy') {
    //
  }
}
```

Run this pipeline or any of its stages on any available agent.
Defines the "Build" phase. phase blocks are optional in Scripted Pipeline syntax. However, implementing phase blocks in a scripted pipeline provides a clearer visualization of the subset of tasks/steps of each phase in the Jenkins UI.
Perform some steps related to the "Build" phase.
Defines the "Test" phase.
Perform some steps related to the "Test" phase.

Defines the "Deploy" phase.
Perform some steps related to the "Deploy" phase.
An example of a pipeline
Here is an example of a Jenkinsfile using the declarative pipeline syntax - its scripted syntax equivalent can be obtained by clicking the Switch to Scripted Pipeline link below:

```
Jenkinsfile (declarative pipeline)
pipeline {
   any agent
   options {
      skipStagesAfterUnstable()
   }
   phase {
      phase('Build') {
         steps {
            sh 'make'
         }
      }
      phase('Test'){
         steps {
            sh 'do check'
            junit 'reports/**/*.xml'
         }
      }
      phase('Deploy') {
         steps {
            sh 'publish'
         }
      }
   }
}
```

Switch Scripted Channel (Advanced)
pipeline is a pipeline-specific declarative syntax that defines a "block" containing all the content and instructions for running the entire pipeline.
agent is a declarative pipeline-specific syntax that instructs Jenkins to allocate triggers (on a node) and workspace for the entire pipeline.
stage is a block of syntax that describes the stage of this pipeline. Read more about phase blocks in declarative pipeline syntax on the Pipeline Syntax page. As mentioned above, phase blocks are optional in Scripted Pipeline syntax.
Steps is a syntax specific to a declarative pipeline that describes the steps to run in that stage.
sh is a pipeline step (provided by the Pipeline: Nodes and Processes plugin) that executes a given shell command.
junit is another Pipeline step (provided by the JUnit plugin) for aggregating test reports.
sh is a pipeline step (provided by the Pipeline: Nodes and Processes plugin) that executes a given

MANAGING JENKINS:
Most standard administrative tasks can be performed from the screens in the Manage Jenkins section of the dashboard. In this chapter we will look at these screens and explain how to use them. The tiles displayed on the Manage Jenkins page are grouped logically. Here we discuss the pages that are part of the standard installation. Plugins can add pages to this screen.
The top of the Manage Jenkins screen may include "Monitors" to notify you when a new version of Jenkins or a security update is available. Each monitor includes a description of the problem it is

reporting and links to more information about the problem Inline help is available on most Manage Jenkins pages:To access help, select ? icon to the right of each field.
Click on ? hide the help text again.

System Configuration group

Resource configuration screens for your Jenkins instance.
System : Configure the global settings and paths for the Jenkins instance
Tools : Configure tools, their locations, and automatic installers
Plugins: Add, update, remove, disable/enable plugins that extend Jenkins functionality.
Knots and clouds : Add, remove, manage, and monitor nodes used for agents running build jobs.
Configuration as code
Configure your Jenkins instance using a human-readable YAML file instead of a UI. This is an optional feature that will only appear in this group if the plugin is installed on your controller.
Security group:Screens for configuring security features for your Jenkins instance. For general information about managing Jenkins security, see Jenkins Security.
Safety: Set configuration parameters to secure your Jenkins instance.
Manage credentials:Configure credentials that provide secure access to third-party websites and applications that communicate with Jenkins.
Credential Providers:Configure providers and credential types
Users:Manage users defined in the Jenkins user database. This is not used if you are using another security realm such as LDAP or AD.
Status Information group:System information
Displays information about the Jenkins environment.
System log
A Jenkins log that contains all Jenkins-related java.util.logging output.
Load statistics
Displays information about resource usage in your Jenkins instance.
About Jenkins
Provides version and license information for your Jenkins instance.
Troubleshooting group
Old data management
Remove configuration information related to plugins that were removed from the instance.
Tools and actions group
Screens for common administrative tasks and administrative tools that allow you to perform administrative tasks without using a user interface.
Reload the configuration from disk
Clear all data loaded in memory and reload everything from the file system. This is useful when editing configuration files directly on disk.
Jenkins CLI
How to use the Jenkins CLI from a shell or script.
Scripting console
Run an Apache Groovy script for administration, troubleshooting, and diagnostics.
Prepare to shut down
Prevents new builds from running so that the system can be safely shut down. It displays a red banner with a custom message to let users know what's going to happen.

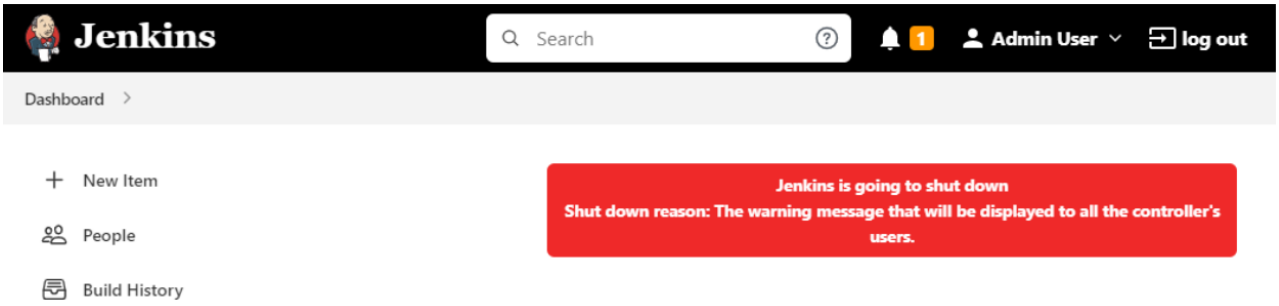

Fig 2.2.3 jenkins shutdown warning

JENKINS AS A SOURCE CODE:

Setting up Jenkins is a complex process as both Jenkins and its plugins require some tuning and configuration with dozens of parameters that can be set in the admin section of the web UI. Advanced Jenkins users rely on groovy init scripts to customize Jenkins and force the desired state. These scripts directly call the Jenkins API and as such can do anything (at your own risk). But they also require you to know Jenkins internals and be confident in writing great scripts on top of the Jenkins API.

The Configuration as Code plugin is an illustrative way to configure Jenkins based on human-readable declarative configuration files. Writing such a file should be doable without being a Jenkins expert, just code the configuration process used in the web UI.

The configuration file below contains root entries for various components of your primary Jenkins installation. One jenkins is for the root Jenkins object and the others are for various global configuration elements.

Jenkins file:

```
jenkins:
  systemMessage: "Jenkins configured automatically by Jenkins Configuration as Code plugin\n\n"
  globalNodeProperties:
  - envVars:
      env:
      - key: VARIABLE1
        value: foo
      - key: VARIABLE2
        value: bar
  securityRealm:
    ldap:
      configurations:
        - groupMembershipStrategy:
            fromUserRecord:
              attributeName: "memberOf"
          inhibitInferRootDN: false
          rootDN: "dc=acme,dc=org"
          server: "ldaps://ldap.acme.org:1636"

  nodes:
    - permanent:
        name: "static-agent"
        remoteFS: "/home/jenkins"
        launcher:
          inbound:
            workDirSettings:
              disabled: true
```

```
            failIfWorkDirIsMissing: false
            internalDir: "remoting"
            workDirPath: "/tmp"

  slaveAgentPort: 50000
  agentProtocols:
    - "jnlp2"

tool:
  git:
    installations:
      - name: git
        home: /usr/local/bin/git

credentials:
  system:
    domainCredentials:
      - credentials:
          - basicSSHUserPrivateKey:
              scope: SYSTEM
              id: ssh_with_passphrase_provided
              username: ssh_root
              passphrase: ${SSH_KEY_PASSWORD}
              description: "SSH passphrase with private key file. Private key provided"
              privateKeySource:
                directEntry:
                  privateKey: ${SSH_PRIVATE_KEY}
```

JENKINS SCALING:
This chapter covers topics related to using and managing large Jenkins configurations: large numbers of users, nodes, agents, folders, projects, concurrent jobs, job results and logs, and even large numbers of Jenkins controllers.
The audience for this chapter is experienced Jenkins users, administrators, and those planning large-scale installations.If you are a Jenkins administrator and want to learn more about managing Jenkins nodes and instances, see

2.3 ELK

ELK Stack is a collection of three open source products - Elasticsearch, Logstash, and Kibana. The ELK stack provides centralized logging to identify problems with servers or applications. It allows you to search all logs in one place. It also helps you find problems on multiple servers by consolidating logs over time.

• E stands for ElasticSearch: used to store logs

• L stands for LogStash: used for shipping as well as processing and storing logs

• K stands for Kibana: A visualization tool (web interface) hosted via Nginx or Apache.

ElasticSearch, LogStash, and Kibana are all developed, managed, and maintained by a company called Elastic.

ELK Stack enables users to ingest data from any source, in any format, and search, analyze, and visualize that data in real-time.

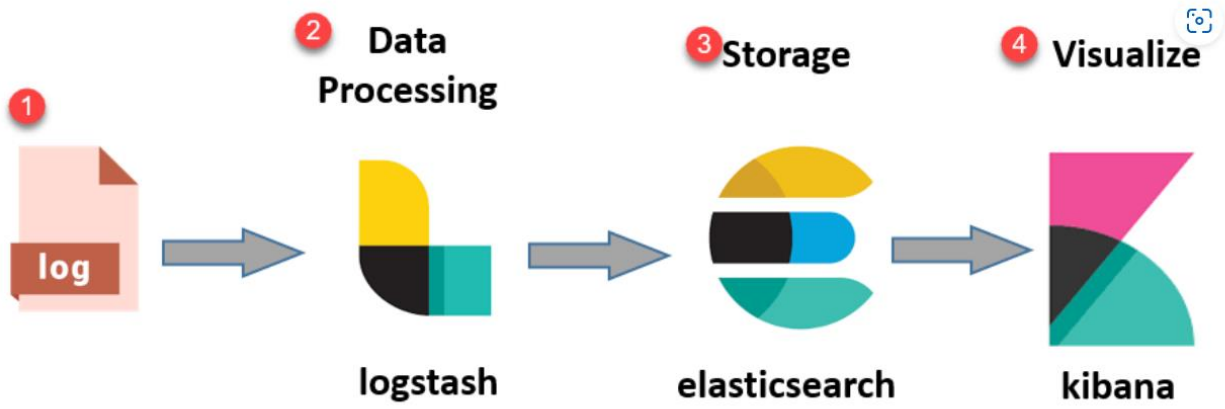

Fig 2.3.1 Architecture of elk stack.

Log: Specifies the server log to check

Logstash: collects logs and event data. It even analyzes and transforms data

ElasticSearch: Store, Search, and Index data converted from Logstash.

Kibana: Kibana uses Elasticsearch DB for search, visualization and sharing.

Elasticsearch

Elasticsearch is a NoSQL database. It is based on the Lucene search engine and is built with a RESTful API. It offers simple deployment, maximum reliability and easy management. It also provides all data for complete analysis and keeps all data centralized. Useful for quickly searching documents.

Elasticsearch allows you to store, search, and analyze large amounts of data. It is often used as a primary engine for power applications that have met the search requirements. Adopted in search engine platform for modern web and mobile applications. Apart from fast search, the tool offers advanced analytics and many advanced features.

Advantages of Elasticsearch:

• An open source search server written using Java

• Used to display all kinds of homogeneous information

• Has a REST API web interface with JSON output

• Full text search

• In Real Time (NRT) search

• A shared, reproducible, JSON document store

• Schema free, REST & JSON-based distributed document store

• Multilingual and geolocation support

Advantages of Elasticsearch

• Schema Save some data and create a schema for your data

• Record your data records using several documentation APIs

• Filter and query your data for insights

• Apache is based on Lucene and provides a RESTful API

• Provide scalability, reliability and versatility using real-time indexing for fast search

• Help to measure vertically and horizontally

Logstash

Logstash is a data collection path. Collect input data and feed it to Elasticsearch. It collects all kinds of data from various sources and makes it available for further use.

Logstash can combine data from different sources and organize data where you want. It allows you to clean and democratize all your data for use case analytics and visualization.It consists of three parts:

• Input: Transfer the log to a machine-readable format

• Filter: A set of conditions to perform a specific action or event

• Output: A decision maker for processed events or logs

Features of Logstash Now let's learn about LogStash features in this LogStash guide:

• Events flow through each stage using an internal queue

• Allow different logins for your log

• filter/analyze your logs

Advantages of Logstash

• Centralizes data processing

• Analyze a variety of structured / unstructured data and events

• ELK LogStash offers plugins to connect to various logging sources and platforms

Kibana

Kibana is a data visualization that complements the ELK suite. This tool is used to visualize Elasticsearch documentation and help developers quickly understand it. Kibana Dashboard offers a variety of interactive graphics, geospatial data, and graphics to visualize complex requests.

Elasticsearch can be used to search, view, and interact with data stored in directories. Kibana helps you perform advanced data analysis and visualize your data in various tables, charts, and maps.

There are several ways to find your data in Kibana.

Advantages of Kibana:

• powerful front-end control panel capable of visualizing index data from an elastic cluster

• Enable real-time search index data

• You can search, view and interact with data stored in Elasticsearch

• Perform data queries and visualize the results in charts, tables, and graphs

• customizable dashboard for logging and truncation of logstash log in elastic search

• Graphics, graphics, etc. able to provide historical information in the form

• Easily customizable real-time dashboards

• Kibana ElasticSearch enables real-time indexed data searches

Kiba's strengths and weaknesses

• Easy to see

• fully integrated with Elasticsearch

• Visualization tools

• Offers real-time analysis, charting, summarization and correction capabilities

• Provides an intuitive and user-friendly interface

• Allows you to share photos of your favorite channels

• Allows you to save dashboards and manage multiple dashboards

2.4 GERRIT

Gerrit is a web based review system so whenever a change is made to the repository and it pushed to the repository it is directly pushed to the repository there is no proper checking verification and validation system.as part of this there is no proper studied code and the chances for the failure for

the code. So we adopted a web based reviewing system for the quality of the code so whenever you push changes to the code it will be not pushed to the repository directly instead it will create a a review there will be reviewers who will be reviewing the code The range for the code review will be starting from -2 to +2 The reviewer gives the score and also comments regarding the code, which will be visible to the developer. If the reviewer gives +2 score and +1 verified, the code automatically gets merged to the main repository. If changes made are to be reverted, they can be reverted and abandoned also. It is always preferred to give at least one +1 score from a reviewer, so that other reviewers can also verify the code, instead of directly merging the code on the basis of a single reviewer

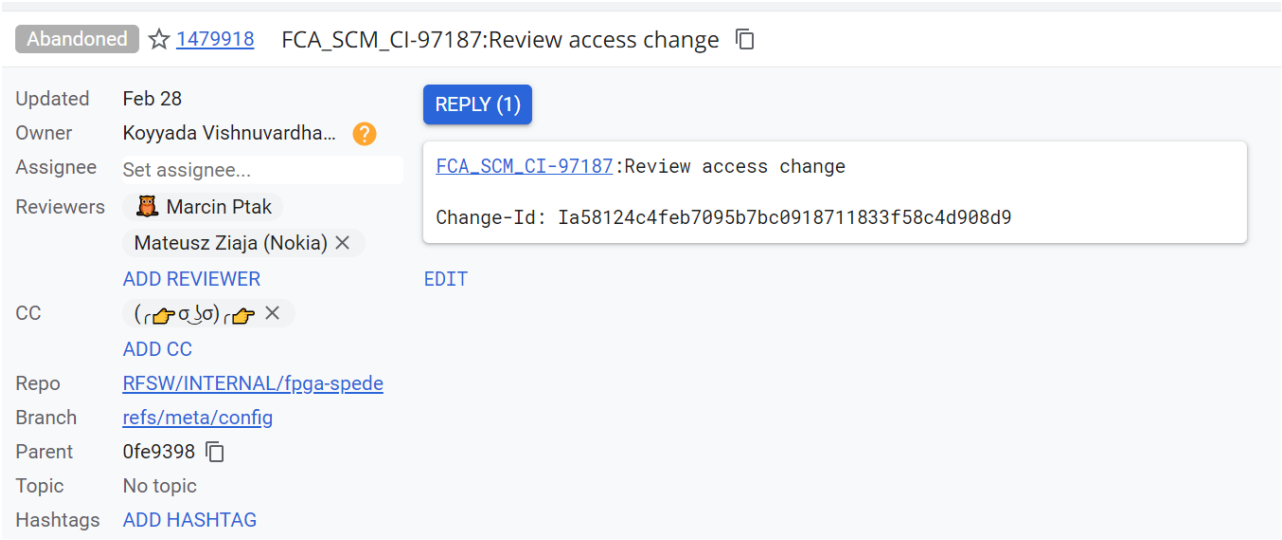

fig:2.4.1 General overview of Gerrit

Gerrit is a web-based code review tool which is integrated with Git and built on top of Git version control system (helps developers to work together and maintain the history of their work). It allows to merge changes to Git repository when you are done with the code reviews. Gerrit was developed by Shawn Pearce at Google, which is written in Java, Servlet, GWT (Google Web Toolkit) as shown in Fig 3.3.

Features of Gerrit

- Gerrit is a free and an open-source Git version control system.
- The user interface of Gerrit is formed on GWT.
- It is a lightweight framework for reviewing every commit.
- Gerrit acts as a repository, which allows pushing the code and creates the review for your commit.

Advantages of Gerrit

- Gerrit provides access control for Git repositories and web frontend for code review.
- You can push the code without using additional command line tools.
- Gerrit can allow or decline the permission on the repository level and down to the branch level.

Shell Scripting

A shell script is a computer program designed to be run by the Unix/Linux shell which could be one of the following:

•The Bourne Shell
•The C Shell
•The Korn Shell
The GNU Bourne-Again Shell

A shell is a command-line interpreter and typical operations performed by shell scripts include file manipulation, program execution, and printing text.
Extended Shell Scripts Shell scripts have several required constructs that tell the shell environment what to
The shell is, after all, a real programming language, complete with variables, control structures, and so forth. No matter how complicated a script gets, it is still just a list of commands executed sequentially.
The following script uses the read command which takes the input from the keyboard and assigns it as the value of the variable PERSON and finally prints it on STDOUT.

```
#! /bin/sh
echo "What is your name?"

read PERSON
echo "Hello, $PERSON"
```

The basic concept of a shell script is a list of commands, which are listed in the order of execution. A good shell script will have comments, preceded by # sign, describing the steps.
There are conditional tests, such as value A is greater than value B, loops allowing us to go through massive amounts of data, files to read and store data, and variables to read and store data, and the script may include functions.
It would be a simple text file in which we would put all our commands and several other required constructs that tell the shell environment what to do and when to do it. Shell scripts and functions are both interpreted. This means they are not compiled.
Assume we create a test.sh script. Note all the scripts would have the .sh extension. Before you add anything else to your script, you need to alert the system that a shell script is being started. This is done using the shebang construct.
For example −
#! /bin/sh This tells the system that the commands that follow are to be executed by the Bourne shell. It's called a shebang because the # symbol is called a hash, and the! symbol is called a bang.
To create a script containing these commands, you put the shebang line first and then add the commands −

```
#! /bin/bash
pwd
ls
```

Save the above content and make the script executable −

```
$chmod +x test.sh
```

The shell script is now ready to be executed − ./test.sh

Review Categories:

Gerrit supports different categories (also known as labels) for review feedback. In its default configuration, it supports the Code-Review category. The Verified category typically means you were able to build and test the change introduced with the Gerrit change. Typically, this is done by an automated process such as a Jenkins / Hudson build server. The "Code Review" category is typically used to vote on the quality of the implementation, code style, code conformity. It also indicates that the change is following the standards desired by the project.

Voting in Gerrit:

The rules for voting in Gerrit are:

➢ Highest vote (+2 in Code-Review and +1 in Verified) enables submitting

➢ Lowest vote (-2 in Code-Review) is a veto blocking that the change can be submitted and can't be overruled by other reviewers The changes made in the code are fixed in gating. It is an intelligent mechanism. The changes can be fixed and tested together. Tool used is Zuul. After gating the code is built and packed together with the help of Yocto. Next integration flow is a phase where the built and packed code is tested again. This step can be called as test phase.
Different tests performed are:

➢ Unit Test

➢ Module Test

➢ Functionality test

Levels of test vary from components to component. Once the test is passed, the generated binary is stored in antifactory. After storage it is updated in workflow tool that package is ready.

2.5 YOCTO

Project Yocto is an open source collaborative project that helps developers create custom Linux-based systems for embedded products, regardless of hardware architecture. The project provides a toolkit and a flexible space for sharing technologies, software, configurations, and best practices that can be used to create custom Linux images for embedded devices.
The project provides standards to provide hardware and software support that allows for interchangeability of software configuration and structure. The tool allows users to build and maintain customizations for multiple hardware platforms and software suites in a stable and scalable manner.
Project Yocto integrates, maintains, and validates three (3) key development elements.
1. A comprehensive set of tools for successful on-premise Linux deployments, including tools for automated installation and testing, processes for board support and license implementation, and component information for Linux-based on-premise operating systems.
2. Referred internal distribution (called Poky)
3. OpenEmbedded is an OpenEmbedded build system supported by the project

Basic terminology in Octoto:
• Configuration file: A file that stores global user definitions of variables, user-defined variables, and hardware configuration information. They tell the build system what to build and configure to support a particular platform.
• Recipe: The most common form of metadata. The recipe will contain a list of settings and tasks (inspection) to install the packages used to build the binary image. The recipe tells you where to get the source code and which patch to use. Recipes describe dependencies, configurations, and configuration options for libraries or other recipes. Stored in layers.
• Layer: A collection of related recipes. Layers allow you to combine related metadata to customize your structure and isolate data for multiple architectural structures. Layers are hierarchical in their ability to override previous specifications. You can add any number of layers from the Yocto project and then add your own layers and adjust the textures. Relational indexes are searchable by the Yocto project layer.
• Metadata: A key element of the octocto Project is the metadata used to build the Linux distribution, which is contained in the parsed files of the build system when the image is built. Metadata generally includes recipes, configuration files, and other information related to build instructions, as well as information used to control and influence the construction of objects. Metadata includes instructions and information used to indicate which version of the software is used and from where they were obtained, as well as changes or additions to the software itself (patches or support files) used to fix or fix bugs. software for use in certain situations. OpenEmbedded Core is an essential set of validated metadata.
• OpenEmbedded-Core: metadata consisting of oe-core, recipes, classes, and related files must be shared between many different OpenEmbedded-derivative systems, including the octo Project. It is

part of a unique set of repositories developed by the OpenEmbedded community, which is transformed into a smaller core set of regularly validated recipes, leading to a tightly controlled and quality-assured core set of recipes.
• Poky: 1) a built-in reference distribution and reference test configuration 1) provides distro basic-level functions that can be used to demonstrate how to configure the distribution, 2) used to validate the Poky Yocto Project to test components of the octocto project; , and 3) as a vehicle for users to download Yocto projects. It's not a pure production-level distro, but a good starting point for customization. "Poky" is an integration layer on top of the "oe" core.
• Build System - "Bitbake": a scheduler and execution engine that parses instructions (recipes) and configuration information. Then it creates a dependency tree to order the assembly, plans the compilation of the added code, and finally executes to build a standard Linux Ubuntu image (distribution) shown. BitBake is a similar tool. The BitBake recipe shows how to build a custom package. It includes all package dependencies, source code locations, configuration, setup, installation, and uninstall commands. The recipe also stores metadata for the package in standard variables. Relevant recipes are grouped together in one layer. Dependencies are tracked during the build process, and local or cross-package execution is implemented. As a first step in building the framework, it will try to create a suitable toolchain (Extensible SDK) for the framework's target platform.
• packaging: The output of the build system is used to create your final image.
• expand

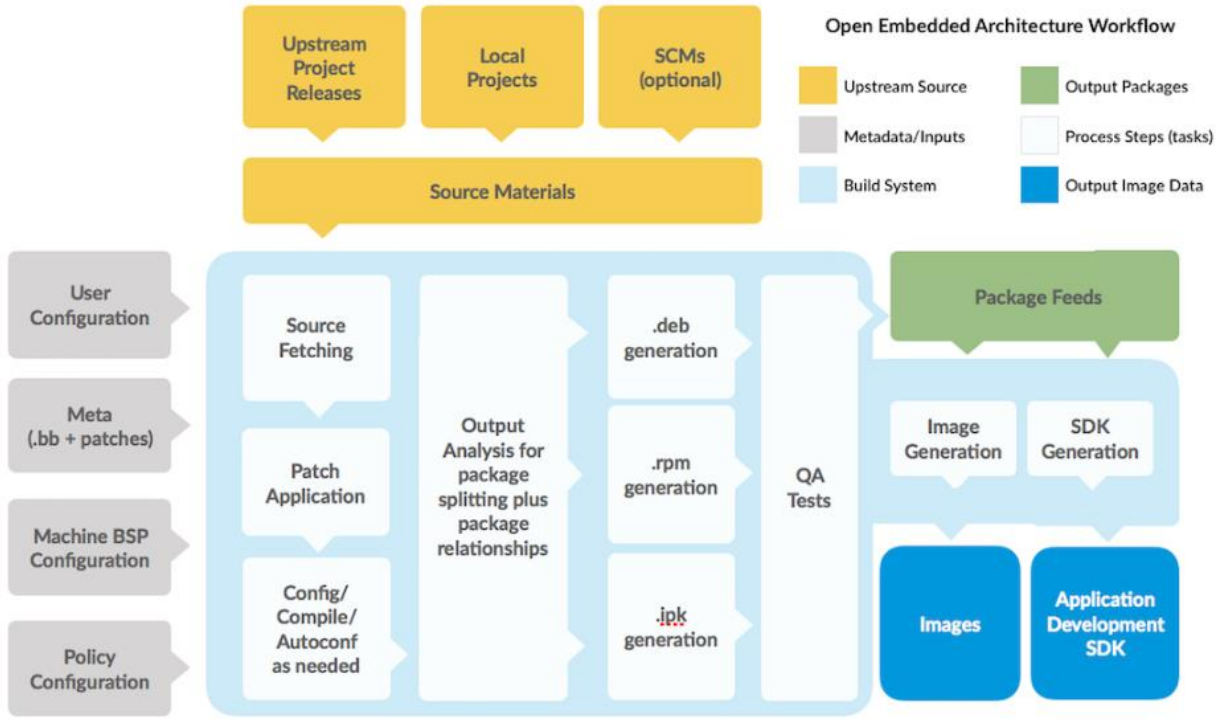

fig 2.5.1 The General flow of Yocto

2.6 Shell Scripting

The shell scripts, when executed in Ubuntu (Linux OS) environment, the command prompt while executions respectively. The scripts depicts the while loop execution and string operations respectively.

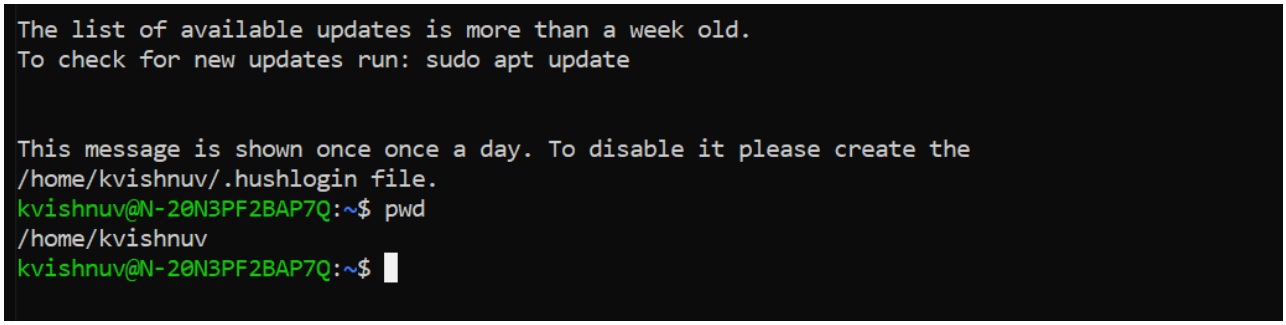

Fig 2.6.1 simple over view of shell scripting

Environment variables:
Contains environment configuration
○ Usually for the shell, but other programs can
set your own.
Created automatically at login.
Scope is global
○ Other programs can read/use them to recognize them
how to behave.
Type "env" to see the full list.

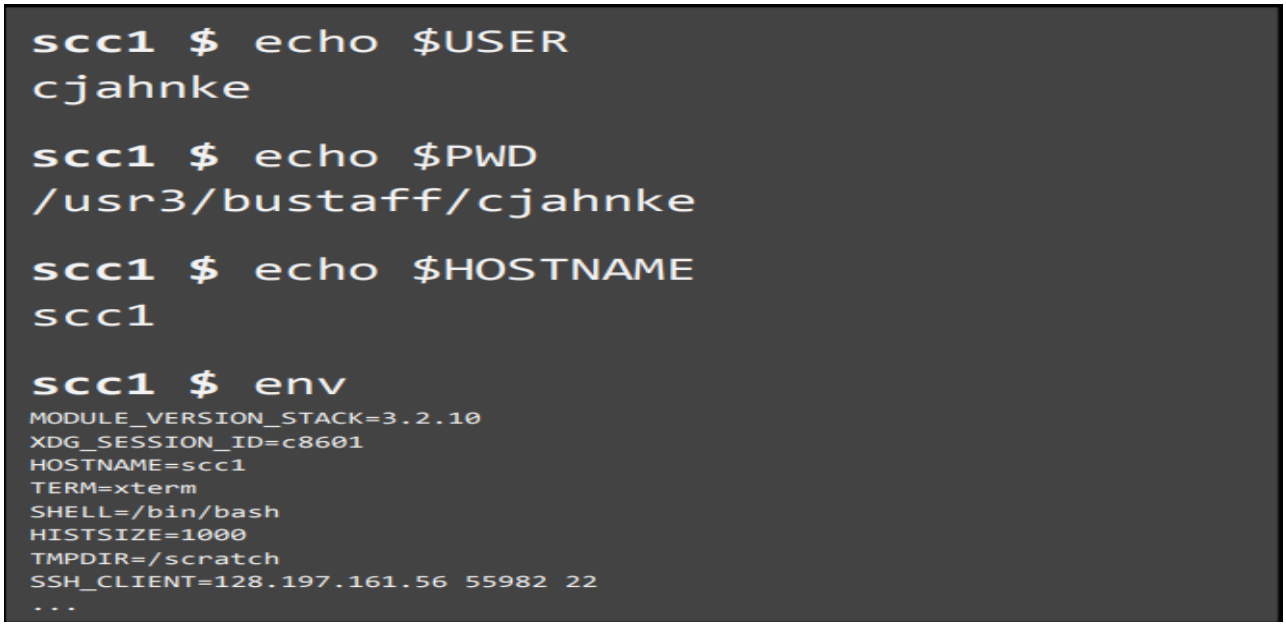

Fig 2.6.1 Environment Variables

Command line arguments in Bash:
The command used to run the bash script passes command information to
script as variables when it is run. This information is accessible through numbered
variable, where "#" is the index of the information.
- $0 → Name of the script
- $1 → The first argument after the script name
- $2 → The second argument after the script name
- ...

Note: Only 9 arguments are captured; after that you have to be creative.

```
#!/bin/bash

# $0 is the script itself
echo '$0' is "$0"
# $1 is the first argument
echo '$1' is "$1"
# $2 is the second argument
echo '$2' is "$2"
```

```
scc1 $

scc1 $ ./cli_arg.sh arg1 “2 items” 3rd
$0 is ./cli_arg.sh
$1 is arg1
$2 is 2 items
```

Fig 6.1.2 Example of command line arguments

Jumping into Standard I/O :

There are 3 types of i/o

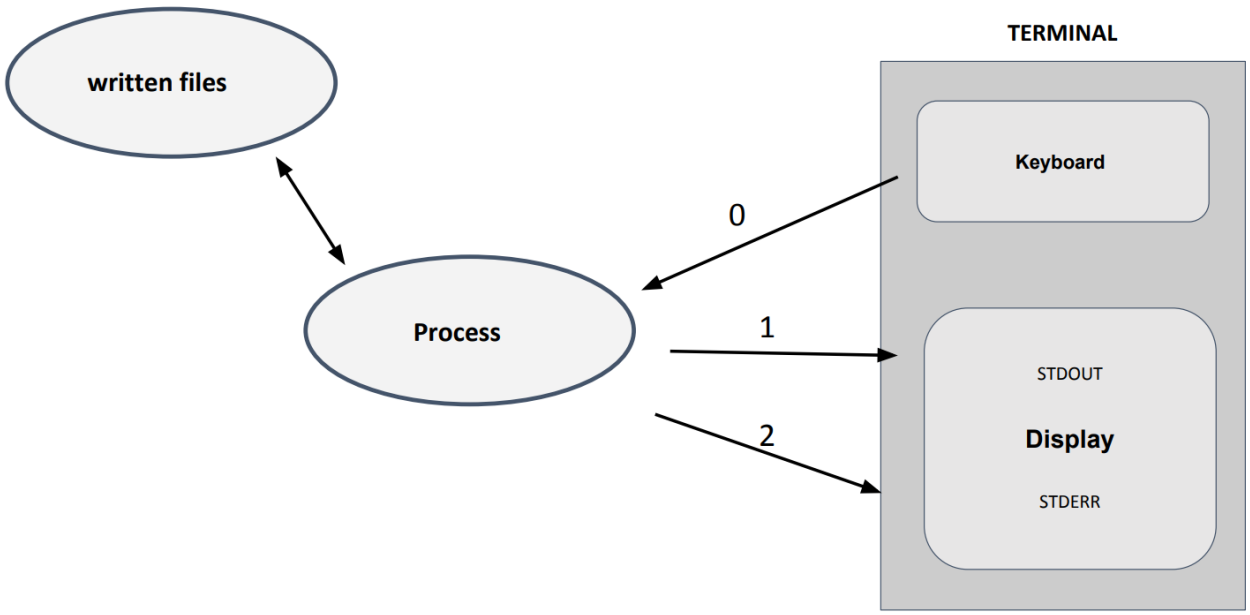

Fig 6.1.3 Standarad i/o

Pipes :

The main use of pipe is to redirect the output of the previous operation to the next operation.
| is the symbol for the pipe.

- Example:

```
[cjahnke@scc1 ~]$ cat sample.vcf | cut -f1,2,7 | sort -k3
```

cat sample.vcf

```
#CHROM  POS      ID        REF  ...
3       14370    rs6054257 G    ...
2       17330    .         T    ...
1       1110696  rs6040355 A    ...
3       1230237  .         T    ...
6       1234567  microsat1 GTCT ...
```

| cut -f1,2,7 | sort -k3

```
#CHROM POS     FILTER
3      14370   PASS
2      17330   q10
1      1110696 PASS
3      1230237 PASS
6      1234567 PASS
```

```
#CHROM POS     FILTER
1      1110696 PASS
3      1230237 PASS
3      14370   PASS
6      1234567 PASS
2      17330   q10
```

Fig 1.6.4 Example execution of pipe command.

Redirection :

Redirection is the function where it writes the standard output of the command to the file directly.

Redirection	Description
COMMAND < filename	Input - Directs a file ★
COMMAND << stream	Input - Directs a stream literal
COMMAND <<< string	Input - Directs a string
COMMAND > filename	Output - Writes output to file (will "clobber") ★
COMMAND >> filename	Output - Appends output to file ★

Example:

```
[cjahnke@scc1 ~]$ cat sample.vcf | cut -f1,2,7 | sort -k3 > sorted.txt
```

Fig 1.6.5 Example of redirection

Control Structures :

LOOPS :

There are 3 types of loops used widely in scripting

1.For :

2.While :

3.Until :

For loop :

It takes opereation written in for loop each time and perform action according to it.

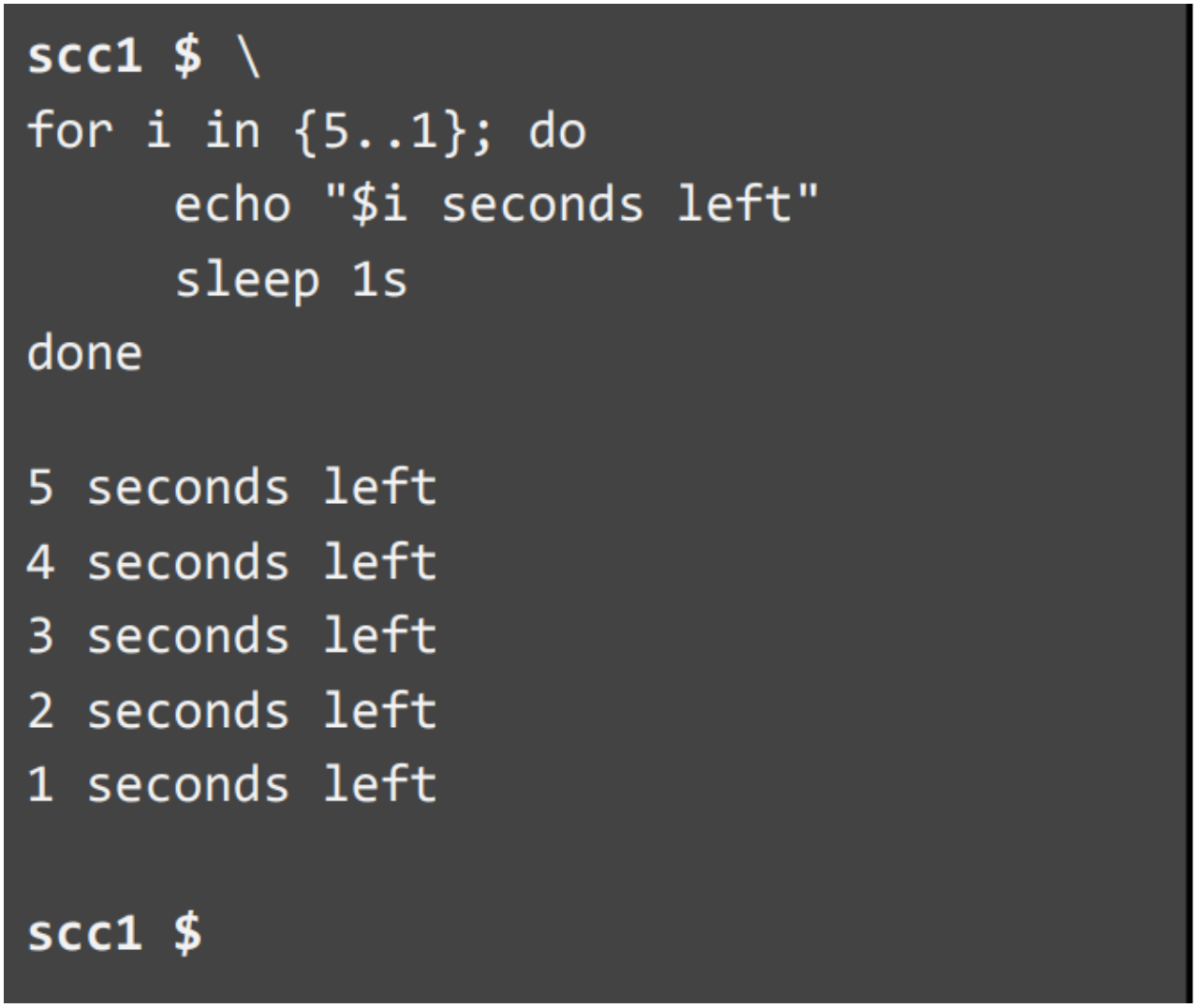

Fig 1.6.6 Example of for loop

Conditional constructions:

test "[[..]]"

○ Evaluates the expression in parentheses and returns 0 (TRUE) or 1 (FALSE)

● if

○ Executes commands according to conditional logic.

● the case

○ Selectively execute commands matching pattern matching.

○ Like if/then statements, but typically used for input parsing and flow determination.

● select

○ Used to create user inputs/optional menus, execute commands on selection.

● Arithmetic "((..))"

○ Performs arithmetic. Be careful, accuracy can be tricky

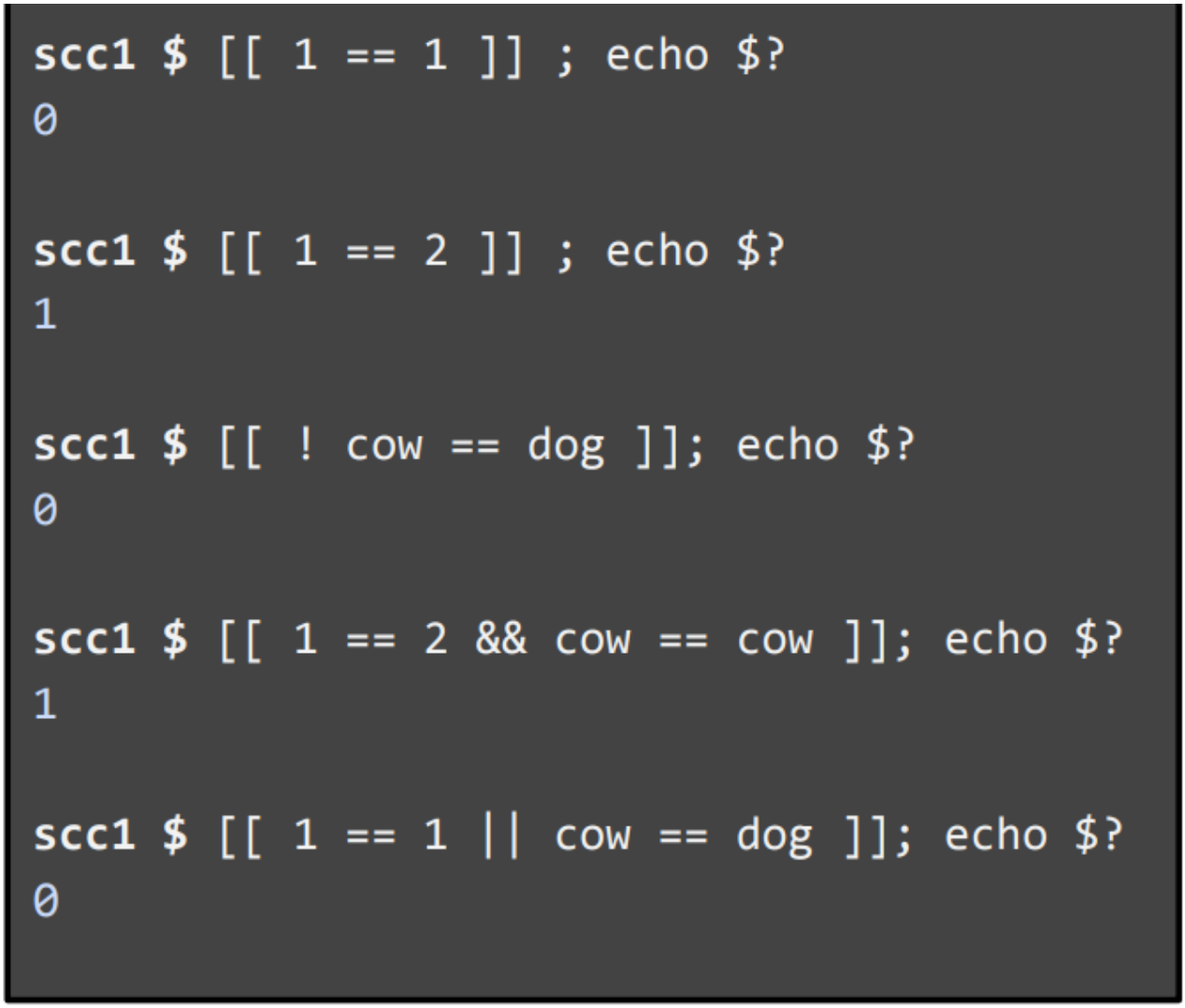

Fig: 1.6.7 Working example of Tests.

IF condition:

If statement (simple)

● The "if" statement executes statements
based on conditional tests.

● The "then" keyword begins with k statements
execute if the condition is true.

● The "elif" keyword can expand if
statement for multiple conditions.

○ Tests are performed in sequence.
Only the first real test will run.

● The universal keyword "else" is used.
execute commands if no conditions are met.

● The "fi" keyword closes a statement

```
scc1 $ ls
TheJungleBook.txt  d  newfile.sh  test.qsub

scc1 $ \
for contents in *; do
    if [[ -f "$contents" ]] ; then
        echo "$contents" is a file
    elif [[ -d "$contents" ]]; then
        echo $contents is a dir
    else
        echo "not identified"
    fi
done

TheJungleBook.txt is a file
d is a dir
newfile.sh is a file
test.qsub is a file
```

Fig 1.6.8 Working example of if condition

2.7 DOCKER AND USAGE :

1.First you need to create a virtual environment if you have already installed python then you can run the command directly or else you need to install the python version that you want to create a virtual environment.

NOTE: myenv is my environment name.

cmd: python -m venv myenv

myenv is the name of the virtual env.

→ if you have n number of versions of python versions so that you want to create a particular versions from that

cmd: python -m venv --python=/usr/bin/python3.9 myenv

/usr/bin/python3.9 path to the location

2.Activate the Virtual Environment

for windows:

cmd : myenv\Scripts\activate

for linux:

cmd: source myenv/bin/activate

3:Install Dependencies/packages

you can install through pip

cmd: pip install package1

if you know the dependencies properly how many packages you want to add in your environment and with exact versions. then keep all packages name with exact version in one file

you can add this in Dockerfile

RUN pip install --no-cache-dir -r requirements.txt

4: Create Dockerfile:

Create the Dockerfile with no extensions and the important is to create the file in the location where you have created the python virtual environment.

5)Build and Run the Docker Container:

cmd: docker build -t myapp .

cmd: docker run myapp

2.8 Agile Methodologies

Agile software development refers to a group of software development methodologies based on iterative development, where requirements and solutions evolve through collaboration between self-organizing cross-functional teams. Agile methods generally promote a disciplined project management process that encourages frequent inspection and adaptation, a leadership philosophy that encourages teamwork, self-organization and accountability, a set of engineering best practices intended to allow for rapid delivery of high-quality software, and a business approach that aligns development with customer needs and company goals.

Benefits of Agile

- Ability to manage changing priorities
- Increased team productivity
- Improved project visibility
- Increased team motivation
- Better delivery productivity

Scrum Overview

In the scrum methodology, the entire team collaborate to deliver the goal iteratively to the customer.

Transparency in scrum

The term transparency in scrum is the accurate and timely disclosure of information. In the scrum methodology, everyone knows about progress.

Definition of Done in scrum
Set of activities that the team undertakes to potentially shippable product increments.
Product backlog
A product backlog is the prioritized list of work (customer requirements/feature/user story). It is derived from the business plan/vision statement
Sprint backlog
In this one, the feature took from the product backlog is placed here. Then that is split into tasks. It provides transparency of the work done during each sprint.

- **Scrum meetings**
- **Sprint planning meeting**
    - In this meeting the requirements are clarified, and the team selects the features from the product backlog as much as it can deliver by the end of the sprint.
    - The features that are selected from the product backlog are placed in sprint backlog and divided into tasks, and its needed time is also estimated.
    - At the end of the sprint planning meeting, the team contacts the product owner and informs the features that can commit in this sprint
- **Daily Standup**

All team members were required to be present in the standup. We had to inform our manager what tasks are completed the before day

- **Sprint review**

At the end of the sprint, the sprint review meetings were held in which the completed product is checked with the predefined requirements. And feedback is given about the product.

- **Sprint retrospective**

In this meeting, feedback about the previous sprint process is collected and plan
is decided to improve the process.

- **Scrum roles**
- **Product owner**

Collects and defines the features from the customer, prioritizes those features and places them in the product backlog. Joints the sprint plan and review

- **Scrum master**

Ensure the team is fully functional and the process is followed. Conduct the daily standup meetings.

- **Team**

Cross functional, selects iterative goals and work to achieve those goals.

Non-Technical Outcomes

The things that I learned through my internship which I want to apply throughout my life.

- Act with Courage
- Leadership quality
- Teamwork
- Time Management
- Own your destiny
- Work Ethics
- Responsibility
- Adaptability Skills
- Communication Skills

2.9 Work Done

The new project must be created for a new repository to migrate to a Gerrit. In Gerrit Login page create a new project. Every organization has its requirement for executing a new repository, first, developer need to access the self-service portal followed by a particular team name, respective team having the requirement to create and has to give the project name, description and if owner want initial commit then can proceed with that else it is fine without initial commit also once the program is executed the repository will automatically create if any issues while creating a repository there must be access problem to the developer, need to contact to the service portal wait for the access to be granted then proceed with all the requirements to create a new project.

Creating a project user needs to enter a respective team name, project name (repository name),

and description for the project (testing purpose or production environment). On the server-side, a computer program store is regularly overseen by source control or store supervisors. A few of the store directors permit to total other store areas into one URL (Uniform Asset Locator) and give a caching intermediary.When doing nonstop builds numerous artifacts are delivered and frequently centrally put away, so naturally erasing the ones those are not discharged is critical. Once the store is effectively made. At that point continue with the movement portion.

System Architecture

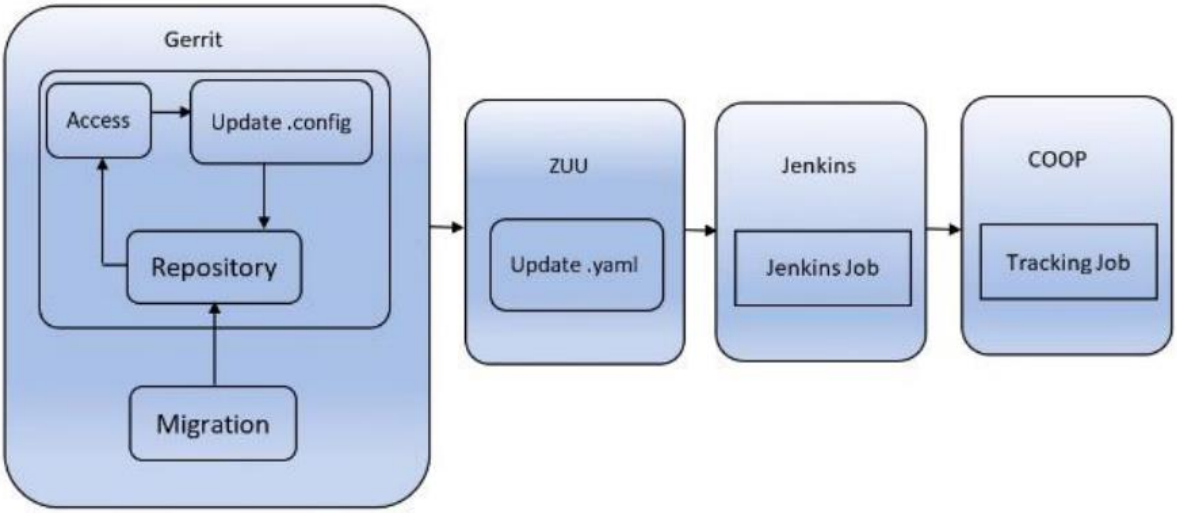

Fig 2.8.1: Block Diagram of Gerrit Migration

It has following sections 1. Create a new repository in Gerrit. 2. Giving access to the repository. 3. Updating the Project. config file. 4. Zuul yaml file updating. and 5. Migrating the contents from svn to Gerrit.

Gerrit

Gerrit is an open source tool and web-based collaboration tool, it is also used for code review purpose and to track the commits to the repo. Gerrit includes git enabled ssh and https server compactable with all git clients.

Create a new repository in Gerrit

Every organization has its requirement for executing a new repository, first, developer need to access the self-service portal followed by a particular team name, respective team having the requirement to create and has to give the project name, description and if developer wants

initial commit then developer can proceed with that, else it is fine without initial commit also. Once the program is executed, the repository will automatically create.

Giving access to the repository

The Code-Review name is arranged upon the creation of a Gerrit occasion. The code esteem ranges from -2 to +2. Directors have all scores (-2 to +2) accessible, -3 is set as the default. Venture Proprietors have constrained scores (-2 to +2) accessible, -2 is set as the default.

Enrolled Clients have restricted scores (-1 to +1) accessible, -1 is set as the default. The code -2 is so unpleasantly incorrect/buggy/broken that it must not be submitted to this venture or this department. This esteem is substantial over all fix sets within the same alter, i.e. the analyst must effectively alter his/her survey to something else some time recently the alter is submittable. This might not be consolidated. The code -1 doesn't see right or might be done in an unexpected way, but the analyst is willing to live with it as-is in case another commentator acknowledges it, maybe because it is way better than what is as of now within the project.
Regularly typically moreover utilized by contributors who do not just like the alter but too aren't dependable for the extend long-term and hence do not have the ultimate say on alter accommodation.

The code says Didn’t attempt to perform the code audit errand or looked over it but don’t have an educated conclusion, however. The code +1 looks right to this analyst, but the commentator doesn’t have get to the +2value for this category. Regularly usually utilized by supporters to a extend who were able to review the alter and like what it is doing, but don’t have last endorsement over what gets submitted. The code +2 Looks great, affirmed. Essentially, the same as +1, but for those who have the ultimate say over how the extend will create.

Updating the Project.config file

Clone the required repo to the local repository by using a command “git clone ”, Go to that folder where the repository is cloned. In the current directory update the config file by using any editor like either in visual code or VI editor. Then check the modified file by command “git status”, update the file by a command “git add.” , make a new developing branch and

switch to that branch. Commit the file with a command -- git commit -m--added the FILE. Yaml file". This is the staging area after this stage Push the file to main repository by the command --git push origin master-- with the developing branch.

Migrating the ontents from SVN to Gerrit

Creating a authors list for the respective repository from the SVN server. Once the author list is created then need to create temporary folder where svn contents are cloned. Change the current directory to that temporary folder, with in the temporary folder Clone the SVN repo into as a Git. If repository is having large number of xml files which requires more storage then it needs artifactory support, it means for cloning and pushing the contents which consists of more storage like in terms of GB. Like 50GB for these types of repositories it is time consuming and data also required more and requires more storage capacity. So, to overcome this git lfs support is needed.

Follow these commands to migrate the SVN contents to Gerrit with the help of git lfs support. Install the git support in the GitHub server “Install the git lfs”. After installation it will pop up with the message “git LFS initialized”. To track the XML files “git lfs track "*.xml" “. After tracking the files, it will pop up the message -- Tracking "*.xml" –. If folder is different, then

it will pop up the message “Not a git repository”. To Config the git lfs file “Config the lfs”, Adding. git attributes to git lfs support. and to “cat. git attributes”, It will pop up with the message -- *.xml filter=lfs diff=lfs merge=lfs -text. --. To track the modified files.

“git status”, Add the files to move to new repository by moving to staging area use the command “git add”. Commit the changes that have made to push to “git commit -m “”, Add the Git Hub repository origin where files have to move “git remote origin”. To check the server is added or not “git remote – v”, if it is existing It will pop up with the message: “fatal: remote origin already exists”. If it is newly added it will give newly added repository server name like
Origin (fetch) and origin (push). Push the contents to repository to remote GitHub “git push origin master”.

Moving SVN branches and tags to the Git repository. List the tags and branches in the SVN repo, “git tag” and “git branch < branch name>”, Push the tags and branches to local Git by a command “git push -all”. SVN has trunk, branches and tags in similar way GitHub also having Master, branches, and tags. The difference is the code which is migrated in trunk is goes to master in GitHub. Here master is the main branch where the SVN contents in the trunk get moved to master in Gerrit server repository.

Zuul

Zuul may be a CI gating system. It may be a program that's utilized to door the source code store of a extend so that changes are as it were combined on the off chance that commit pass tests. The most component of Zuul is the scheduler. It gets occasions related to propose

changes, triggers tests based on those occasions, and reports back. ZUUL configuration is written in yaml file. File extension should be in yaml only.

Zuul .yaml file updating

Clone the required repo to the local repository by using a command “git clone”. Go to that folder where the repository is cloned. In the current directory update the .yaml file in any editor like either in visual code or VI editor. Check the modified file by command “git status”, if no

changes added to commit it will pop up the message use "git add" and/or "git commit -a", then update the file by a command “git add.” , make a new branch and switch to the new branch. Utilize the command 'git push' to send all the neighborhood commits to the inaccessible stores by the command, “git thrust beginning HEAD: refs/for/master” with the creating branch.

Jenkins CI Build

Jenkins is an open-source computerization apparatus composed in Java programming dialect that permits nonstop integration. Jenkins builds and tests the computer program ventures and persistently making it simpler for engineers to coordinated changes to the venture and making it less demanding for clients to get a new construct.

Jenkins Job

The scripts which are in Domain specific language (DSL) are written in .groovy and Cb_funtions, which are written in shell scripts. These are adapted to git version control system replacing with svn. To add the details like getting variable from svn to git. Getting the tags from the svn to git,

updating the internal ECL (Environmental code lines) file and header files, release recipes this information should be pushed to Gerrit by running a Jenkins jobs through a continuous integration. In Jenkins it has pre-CI and Post CI jobs. During the pre-CI it will validate the code, whether all the indentation is correct or not and program structure. It is an independent job. During the pre-CI one more stage where manual intervention takes place to give code review +2 to -2 that how code is efficient to move to post stage, which is a dependent job which checks the jobs in parallel. During a post check It will Post the modified content to GitHub. Once the Jenkins job is success code gets merge to master.

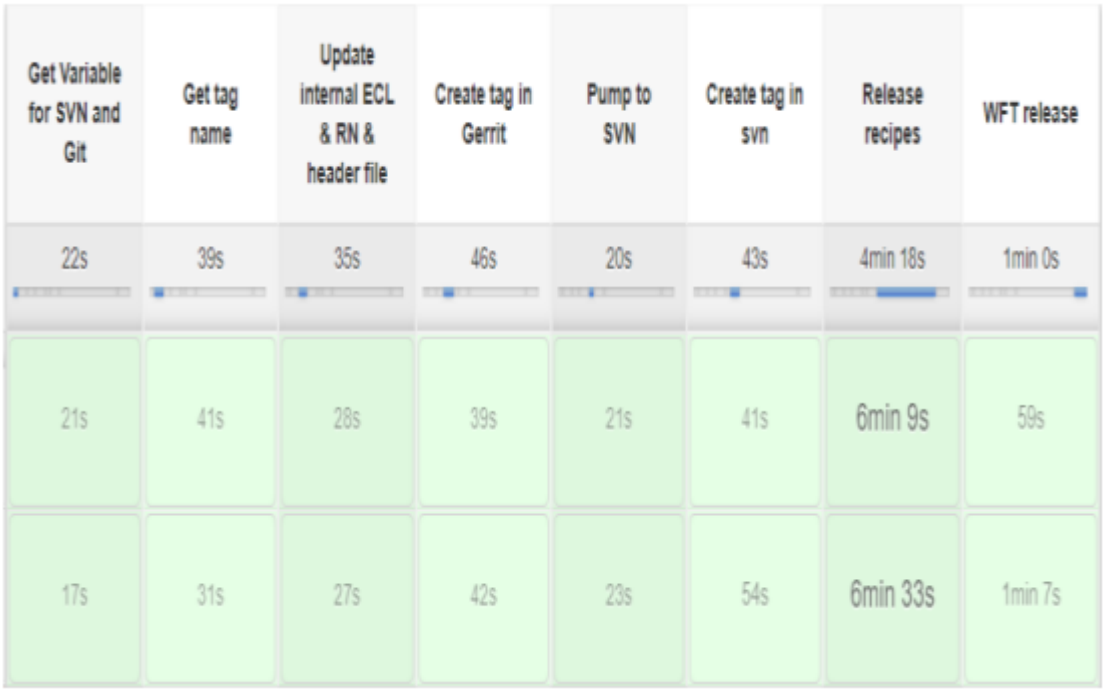

Fig 3.5: Jenkins Job

Coop

Coop is developed for the participation of two or more substances or other specialists to deliver a combined impact more noteworthy than the whole of their isolated impacts. During the

software continuous integration process, huge amounts of data will be generated, but those data spread across too many places to synergistic this, Developers are in a process of creating an end-to-end view when it comes to the connections between CI systems and to show how the content propagates from the commit to system tests of a build that contains that commit, to have a Continuous Integration data visualization purpose.

Tracking job

Repo configuration is done with Project name and Gerrit server name. Internally it connects to the Gerrit server and track the recent commits updated by the developer to the respective repo by system component provided during the repo configuration. Python scripts is implemented to post repo configuration to coop by json template.

Chapter 3

SYSTEM REQUIREMENT ANALYSIS

3.1 SYSTEM REQUIREMENTS

3.1.1 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list especially in case of operating systems.

An HCL list tested compatibility and sometimes incompatible hardware devices for a particular application or operating system. The following sub sections discuss the various aspects of hardware requirements.

PROCESSOR	Intel core, i5
RAM	16 GB
HARD DISK	237GB

3.1.2 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and prerequisites that needs to be installed on a computer to provide optimal functioning of an application. These requirements are generally not included in the software installation package and need to be installed separately before the software is installed.

OPERATING SYSTEM	Windows 10,64 bit,linux
TECHNOLOGIES	GIT,GERRIT
LANGUAGES	PYTHON

Chapter 4

Reflections & Outcomes

The internship at Nokia Networks was an excellent experience. During internship period, there was so much of learning in all technical and non – technical aspects. This reflected hugely, helping to become a better human. The technical knowledge, non – technical knowledge and the learning outcomes acquired during the internship period is described in detail in this chapter.

4.1 Technical Knowledge

As an engineer, it is very vital to view every aspect or a feature in a technical manner to make the lives of people easy and comfortable by implementing the visually imagined idea in real time and real world. It is the technical view and approach given to brain – storming ideas which transformed this idea into reality. It improves the logical and analytical skills, and the problem – solving skills which is utmost important for working in R&D department. Nokia Networks being a product – based company provided ample of opportunities to explore and improve the technicalities during the internship work. The technical knowledge acquired during the internship is described below.

4.1.1 Usage of Tools

The tools which were used explicitly during the internship were Gerrit, Jenkins, Zuul and RF specific testing tools. As discussed in Chapter 2 and 3, Gerrit is a code review tool, where code changes are reviewed. Technically, code changed using git bash commands are merged after it going through Gerrit review process. Zuul is an API gateway application. It is interdependent on Gerrit. Jenkins is an automated CICD tool, which continuously integrates and deploys the code and also checks for build failures of different software packages. Jenkins is mainly used for automation of build process of various software packages. The artifactory management manages the source files stored. The popular artifactory management tool is JFrog artifactory. The source codes in addition with binary files are stored in artifactory, known as artifacts. It also stores the meta data, which has data and details about the stored files. Docker containerization was learnt. The RFspecific tools, WFT usage was learnt, which is the heart of the RF departmental work. The testing tool Knife was explored.

4.1.2 Programming Languages

Any idea is implementable using executable code. In Nokia Networks, the programming languages used are mainly python scripting and groovy scripting. The codes written in git is python scripts and that in Jenkins is Groovy, which is similar to Java scripts. The codes are written and modified according to requirement of feature builds. Coding helps in improving logical and analytical thinking. Thus, coding skills were improved significantly during the internship, which in turn helped in acquiring core technical skill regarding the work done.

4.1.3 Other Technical Skills

Linux operating system is widely used in servers for its stability and security purpose. The CLI provides secured operations than Graphical User Interface (GUI). Thus, linux commands were learnt and executed in Ubuntu system. Learning linux commands also helped in faster learning of git bash commands. Further, shell scripting (also known as bash scripting) was learnt. Shell scripting is used when a group of commands are to be run repeatedly. Instead of running all single commands multiple times, when written as a shell file and run, the commands in the shell file is executed. Also, RF team specific tools such as testing tools, repository access, artifactory access and creation (Jfrog artifactory) were learnt. In addition, git bash commands, svn commands were explored and learnt

4.2 Soft Skills Knowledge

Soft skills play an important role in describing the work done in proper understandable and formal way. It helps in having good and cordial relation with the co – workers and create an amicable and comfortable work environment. Soft skills are important to convey the information in effective way. The soft skills acquired during the internship at Nokia Networks is described in this section.

4.2.1 Agile Way of Working

In corporate world, it is important to keep the work in progress as well as to deliver products without any failure in its functioning. This also requires to check if there are any build failures or errors happening in work. Agile way of working records the progress of work done by employees on every day basis, where it is required to update the work done everyday to concerned person. This also helps in having knowledge of co – workers work and if similar kind of work is assigned, will become easy to complete. The working culture followed in Agile way of working was learnt. Agile way of working is implemented as sprints – based works. Agile working way includes planning phase, where planning for the sprints is performed, exit meetings, where the tasks completed in the sprint are explained, retrospective meetings are held, where the overall feedback of the sprint is taken from the team members. It discusses both what went well, what did not go well and what can be improved. It helps to constantly improve the efficiency and maintain a healthy work – life balance.

4.2.2 Communication Skills

Communication skills is one of the most important skills which is very much required to have information exchange between people. It also builds an amicable, inter – personal and cordial relationship with co – workers. When proper formal communication is being made, it becomes easier to make the other person understand the information wanted to be conveyed. It also aided to improve the hold on language, the fluency and vocabulary of the language. It also helped in connecting to people and interacting with them easily in better way. Attending meetings and knowledge transfer sessions helped in explaining the concepts effectively and quickly. It helped to understand and react to the situation in more matured way. It helps in sharing creative ideas and solving problems. It also ensures transparency between people who are communicating. It also helps to be more interactive during meetings and knowledge transfer sessions. It also helps in building trust with the co – workers and maintain good relationship with them.

4.2.3 Other Skills

Other soft skills acquired during internship at Nokia include team building capability which helps to learn to work in a group and achieve a common goal. The cultural team events organized by Nokia Networks also helped in showcasing cultural talents and develop talent and skills. The way of organizing an event was learnt, as an opportunity was given to aid in organizing event. Overall development in personality was observed, which helped in improving the skill on working as a team with one common goal. It helped in acquiring professional approach of working. It also helped to improve time management and critical thinking. It also helped in adapting to different situations. It helped in writing skills as documentation was learnt

4.3 Learning Outcomes

Internship at Nokia Networks was a great experience. It provided an insight about the industrial and corporate world, giving a jump to view work professionally. It helped me acquire both technical and non – technical knowledge and provided an exposure to explore new things. Technically, programming languages scripting, usage of different tools, using different systems were learnt. Learning and adapting to agile way of working, attending meetings, team building

skills, were learnt. Also, by attending and participating in meetings, was able to know and understand other peoples' work. Soft skills such as communication skills, story – telling were improved. Also, participated in various events organized by Nokia and inter – personal skills were improved. During internship period, an overall progress in technical, non – technical and cultural behavior was seen.

Conclusion & Future Work

source code management (SCM) is an integral part of software development for all organizations. Continuous Integration, deployment and Continuous delivery are of utmost importance because tasks like builds and testing needs to execute regularly on a daily basis in continuity.

Jenkins, Gerrit, Git, MOBXterm and other tools help in the SVN(subversion) to Git migration process and making the task easier. Jenkins tools and plugins, are used to carry out automations and static code analysis, reduce the burden of manual execution and repeated procedures.

The use of migration is to get the latest commit and logs, developer can switch to any feature branch by keeping the present work in staging area, developer can work offline, only while pushing commits to repository developer need an internet connection. In any case data in the central server is lost can be backup by any of the developer's local machine.

REFERENCES

[1] https://www.nokia.com/ <For referring reports>

[2] Nokia Annual CSR Report, 2019-20

https://www.nokia.com/sites/default/files/202104/CSR_Annual%20Report_26Mar2021_7Apr.pdf <For referring reports>

[3] Nokia in 2021, https://www.nokia.com/system/files/2022-03/nokia-ar21-en.pdf

[4] Nokia – Single RAN

https://www.nokia.com/networks/mobile-networks/airscale-radio-access/single-ran-advanced

[5] Airscale Cloud RAN

https://www.nokia.com/networks/mobile-networks/airscale-radio-access/cloud-ran/

[6] Vinayak Pujari, Rajendra Patil and Kamija Tambe, "Research Paper on Future of 5G

Wireless System", Emerging Advancement and Challenges in Science, Technology and

Management, ISSN 2231-2137, Apr. 2021.

[7] Deepender Manoj, U Shrivastava and J K Verma, "A Study on 5G Technology and Its

application in Telecommunications", International Conference on Computational

Performance Evaluation, pp. 365-371, 2021.

[8] N H Phuoc Dai, L Ruiz and R Zoltan, "5G revolution:

Challenges and opportunities",
International Symposium on Computational Intelligence and Informatics, pp. 211-216, 2021.

[9] Sadiq Iqbal, Jehad M Hamamreh, "A Comprehensive Tutorial on How to Practically
Build and Deploy 5G Networks Using Open-Source Software and General-Purpose, Off-theShelf Hardware", RS Open Journal on Innovative Communication Technologies, vol. 2,
2021.

[10] Meer Zafarullah Noohani, Kaleem Ullah Magsi, "A Review of 5G Technology:
Architecture, Security and wide Applications", International Research Journal of Engineering
and Technology, vol. 7, Issue. 5, May 2020, ISSN – 2395-0072.

[11] Amin Salih Mohammed, Suzan Tahsein Husein, M Sivaram, V Porkodi, "4G and 5G
Communication Networks Future Analysis", International Journal of Engineering Research
and Technology, vol. 12, no. 3, 2019.

[12] Woosik Lee, Eun Suk Suh, et.al, "Comparative Analysis of 5G Mobile Communication
Network Architectures", Journals, vol. 10, Issue 7, 2020.

[13] Naing Kyaw, "Study on Basic Principles and Signalling Functions of eNodeB in LTE
Network", Journal of Engineering Education and Applied Science, vol. 1, Issue 1, Feb. 2020.

[14] Sai Keerti Boddepalli, “Performance Evaluation of v – eNodeB using Virtualized Radio
Resource Management”, University of Nebraska – Lincoln, 2018.

[15] Mohammad Asif Habibi, Meysam Nasimi, et. al, “A Comprehensive Survey of RAN
Architectures Toward 5G Mobile Communication System”, IEEE Access, vol. 7, 2019.

[16] Md. Farhad Hossain, Ayman Uddin Mahin, et. al, “Recent Research in Cloud Radio
Access Network (C-RAN) for 5G Cellular Systems – A Survey”, Journal of Network and
Computer Applications, vol. 139, pp. 31-48, 2019.

[17] Yekta Turk, Engin Zeydan, Cemal Alp Akbulut, “On Performance Analysis of Single
Frequency Network with CRAN”, IEEE Access, vol. 7, 2019.

[18] Supatsara Wattanakriengkrai, Bodin Chinthanet, et.al, “GitHub Repositories with Links
to Academic Papers: Public access, traceability, and evolution”, Journal of Systems and
Software, vol. 183, Jan. 2022.

[19] https://git-scm.com/doc

[20] Gerrit – Code Review,
https://gerrit-review.googlesource.com/Documentation/config-gerrit.html

[21] Suresh, Anitha G S, Shruthi L, “Gerrit Migration”, International Research Journal of
Engineering and Technology, vol. 8, Issue. 5, May 2021, ISSN –

2395-0072.

[22] Ioannis K Moutsatsos, Imtiaz Hossain, et. al, “Jenkins – CI, an Open – Source
Continuous Integration System, as a Scientific Data and Image – Processing Platform”,
Original Research, 2016, DOI: 10.1177/1087057116679993.

[23] Jenkins, https://www.jenkins.io/

[24] Paul Stothard, “An Introduction to Linux for Bioinformatics”, University of Alberta, Jan.
2016.

[24] Kirti Kaushik, Jyoti Yadav, Kriti Bhatia, “Shell Script & Advance Features of Shell
Pr5gramming”, International Journal of Computer Science and Mobile Computing, vol. 4,
Issue 4, Apr. 2015.

[26] Shivangi Shandilya, Surekha Sangwan, Ritu Yadav, “Shell Scripting and Shell
Programming in Unix”, International Journal of Innovative Research in Technology, vol. 1,
Issue 11, 2014.

[27] Zuul, https://zuul-ci.org/
[28] JFrog Artifactory, https://jfrog.com/artifactory/

[29] Groovy Language Documentation, Version 5.0.0
https://docs.groovy-lang.org/next/html/documentation

[30]NokiaAnnualCSRReport,2019-20,
https://www.nokia.com/sites/default/files/

2021-04/CSR_Annual%20Report_26Mar2021_7Apr.pdf
[31] Nokia in 2021, https://www.nokia.com/system/files/2022-03/nokia-ar21-en.pdf
[22] https://git-scm.com/doc

[33]Gerrit–CodeReviewhttps://gerrit-review.googlesource.com/Documentation/config-gerrit.html
[34] Zuul, https://zuul-ci.org/
[35] Jenkins, https://www.jenkins.io/