Annexure-IX (a)

Software Development Intern

Nyalazone Solutions Private Limited

A training report

Submitted in partial fulfillment of the requirements for the award of a degree of

Bachelor of Technology
Computer Sciences and Engineering

Data Science

Submitted to

LOVELY PROFESSIONAL UNIVERSITY

PHAGWARA, PUNJAB

From 12/12/2022 to 20/05/2023

SUBMITTED BY

Name of the Student:
Kanigolla Naga Venkata Bala Likhith

Registration Number: 11903700

Student Signature:

K.N.V. Bala Likhith

SUBMITTED TO

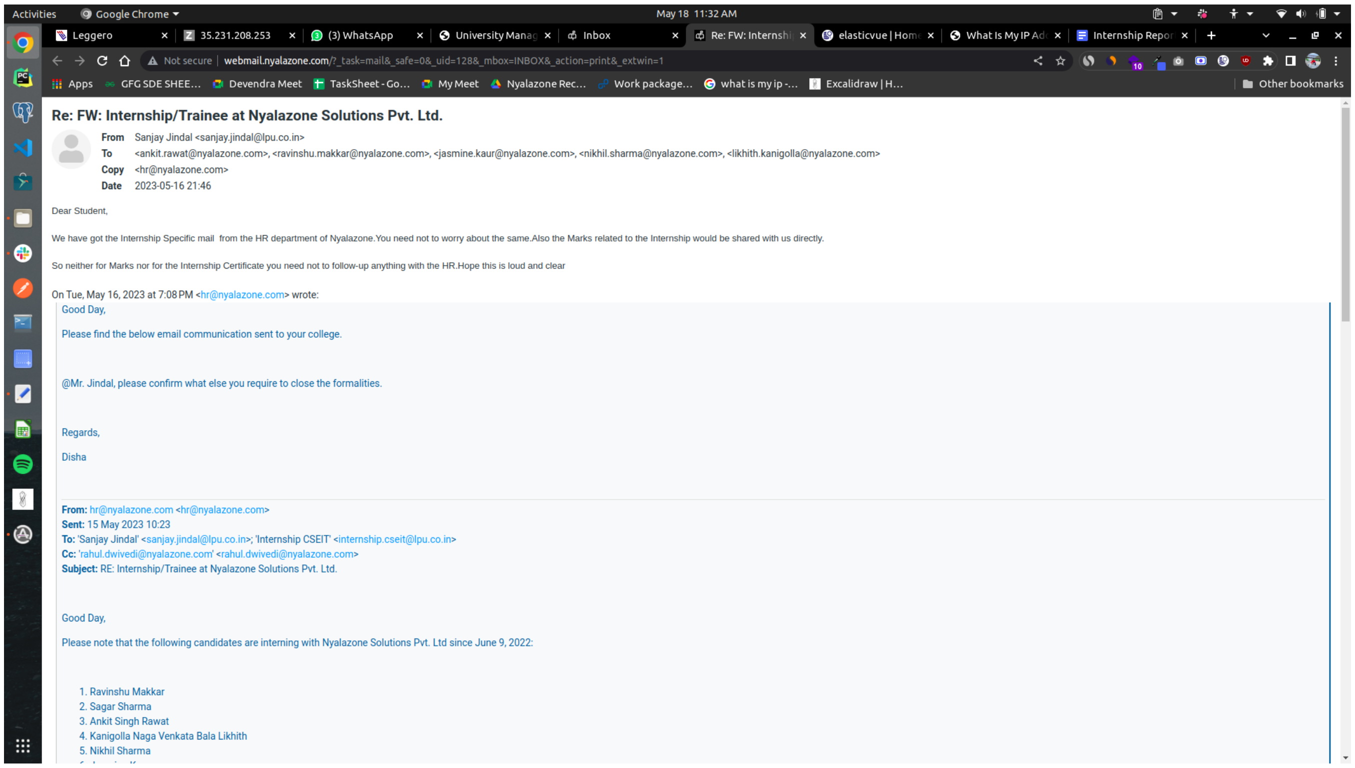
Name of the Supervisor:
Dr. Anshu Sharma

UID: 28431

Signature of the Supervisor:

TABLE OF CONTENTS

Internship Certificate



Re: FW: Internship/Trainee at Nyalazone Solutions Pvt. Ltd.

From Sanjay Jindal <sanjay.jindal@lpu.co.in>
To <ankit.rawat@nyalazone.com>, <ravinshu.makkar@nyalazone.com>, <jasmine.kaur@nyalazone.com>, <nikhil.sharma@nyalazone.com>, <likhith.kanigolla@nyalazone.com>
Copy <hr@nyalazone.com>
Date 2023-05-16 21:46

Dear Student,

We have got the Internship Specific mail from the HR department of Nyalazone.You need not to worry about the same.Also the Marks related to the Internship would be shared with us directly.

So neither for Marks nor for the Internship Certificate you need not to follow-up anything with the HR.Hope this is loud and clear

On Tue, May 16, 2023 at 7:08 PM <hr@nyalazone.com> wrote:

Good Day,

Please find the below email communication sent to your college.

@Mr. Jindal, please confirm what else you require to close the formalities.

Regards,

Disha

From: hr@nyalazone.com <hr@nyalazone.com>
Sent: 15 May 2023 10:23
To: 'Sanjay Jindal' <sanjay.jindal@lpu.co.in>; 'Internship CSEIT' <internship.cseit@lpu.co.in>
Cc: 'rahul.dwivedi@nyalazone.com' <rahul.dwivedi@nyalazone.com>
Subject: RE: Internship/Trainee at Nyalazone Solutions Pvt. Ltd.

Good Day,

Please note that the following candidates are interning with Nyalazone Solutions Pvt. Ltd since June 9, 2022:

1. Ravinshu Makkar
2. Sagar Sharma
3. Ankit Singh Rawat
4. Kanigolla Naga Venkata Bala Likhith
5. Nikhil Sharma

To whomsoever it may concern

I, Kanigolla Naga Venkata Bala Likhith, 11903700, hereby declare that the work done by me as "Software Development Intern" from December 2022 to May 2023, under the supervision of Milan Garg, Data Engineer , Nyalazone Solutions Private Limited, and Dr. Anshu Sharma, Lovely Professional University, Phagwara, Punjab, is a record of original work for the partial fulfillment of the requirements for the award of the degree, Bachelor of Technology in the field of Computer Science and Engineering with Data Science specialization.

K.N.V. Bala Likhith

Kanigolla Naga Venkata Bala Likhith (11903700)

Dated:20th May 2023

Annexure-IX (c): Declaration by the supervisors

To whomsoever, it may concern

This is to certify that Kanigolla Naga Venkata Bala Likhith, 11903700 from Lovely Professional University, Phagwara, Punjab, has worked as an Intern in Nyalazone Solutions on "Software Development" under my supervision from December 2022 to May 2023. It is further stated that the work carried out by the student is a record of original work to the best of my knowledge for the partial fulfillment of the requirements for the award of the degree, Bachelor of Technology in Computer Science and Engineering.

Name of the External Supervisor	Name of the Internal Supervisor
Milan Garg	Dr. Anshu Sharma
Designation of the External Supervisor	Designation of the Internal Supervisor
Data Engineer	
Signature of the External Supervisor	Signature of the Internal Supervisor
Dated:	Dated:

Annexure-IX (c): Acknowledgement

I would like to acknowledge and thank Nyalazone Solutions and Lovely Professional University for this internship opportunity. Working at Nyalazone Solutions is a great chance for learning and professional development. I am also grateful for having a chance to meet so many wonderful people and professionals who helped me through this internship period.

I express my deepest thanks to Milan Garg and Arnab Sharma for taking part in useful decisions & giving necessary advice, mentorship and guidance, and arranging all facilities to make life easier and learn new things every single day. I choose this moment to acknowledge his contribution gratefully.

I would like to acknowledge the sense of respect I have for my mentor and teammate Devendra Singh Negi for being the best mentor one could ask for and patiently explaining everything whenever I'm stuck.

I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.

Lastly, I give my sincere thanks to all those who have helped me during this internship period. I extend my deep sense of gratitude to all my family, faculty, colleagues, and friends who have directly or indirectly encouraged and helped me to complete my project successfully.

K.N.V. Bala Likhith

Kanigolla Naga Venkata Bala Likhith (11903700)

Dated: 20th May 2023

Analytics: Analytics shows you how your content is performing. It also gives you additional insights on the status of your content including troubleshooting information for failure rates on any specific flow.

API: Application Programming Interface is an access point and a set of routines, protocols, and tools to specify how various software components interact.

Cloud Deployment: Cloud deployment is one of the ways to deploy content and make it available to your users. In this scenario, content is published to Servers.

Deployments: Deployments are various ways to host and deliver content to your end-users.

Flow: Flows are a sequence of step-by-step instructions that users are led through in your application to complete a task.

Task List: The Task List is a widget that gives you the ability to assign a list of topics users must complete as part of a training program.

Versioning: Versioning is the ability to save and view an older version.

Job: Jenkins Freestyle Project is a repeatable build job, script, or pipeline that contains steps and post-build actions. It is an improved job or task that can span multiple operations. It allows you to configure build triggers and offers project-based security for your Jenkins project. It also offers plugins to help you build steps and post-build actions.

Pipeline: Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins.

DBM: A database model is a type of data model that determines the logical structure of a database. It fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational model, which uses a table-based format.

CLI: A command-line interface (CLI) is a text-based user interface (UI) used to run programs, manage computer files and interact with the computer. Command-line interfaces are also called command-line user interfaces, console user interfaces and character user interfaces. CLIs accept as input commands that are entered by keyboard; the commands invoked at the command prompt are then run by the computer.

GIT: Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Agile Server: The Agile Application Server is the center of the Agile system, the base for the PLM platform, where all common services and business logic reside for the entire solution. The Agile Application Server runs on industry-leading J2EE application servers. As the System Configuration Overview figure illustrates, all client servers and users connect to the Application Server either directly or indirectly. The application server connects to the components in a persistence layer where product content is stored.

GCP: Google Cloud Platform, offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search, Gmail, Google Drive, and YouTube.

AWS: Amazon Web Services, Inc. is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals, companies, and governments, on a metered pay-as-you-go basis. These cloud computing web services provide distributed computing processing capacity and software tools via AWS server farms.

SSH: The Secure Shell Protocol is a cryptographic network protocol for operating network services securely over an unsecured network. Its most notable applications are remote login and command-line execution. SSH applications are based on a client–server architecture, connecting an SSH client instance with an SSH server.

SCP: SCP (secure copy) is a command-line utility that allows you to securely copy files and directories between two locations.
With scp, you can copy a file or directory:

- From your local system to a remote system.
- From a remote system to your local system.
- Between two remote systems from your local system.

Dashboard: In business computer information systems, a dashboard is a type of graphical user interface which often provides at-a-glance views of key performance indicators relevant to a particular objective or business process.

Zabbix: Zabbix is an open-source software tool to monitor IT infrastructure such as networks, servers, virtual machines, and cloud services. Zabbix collects and displays basic metrics
Slack Bot: A bot is a type of Slack App designed to interact with users via conversation.A bot is the same as a regular app: it can access the same range of APIs and do all of the magical things that a Slack App can do.

Kafka Broker: There are one or more servers available in Apache Kafka cluster, basically, these servers (each) are what we call a broker.

Kafka Topics: Basically, Kafka maintains feeds of messages in categories. And, messages are stored as well as published in a category/feed name that is what we call a topic. In addition, all Kafka messages are generally organized into Kafka topics.

Kafka Partitions: In each broker in Kafka, there are some number of partitions. These Kafka partitions in Kafka can be both a leader or a replica of a topic. So, on defining a Leader, it is responsible for all writes and reads to a topic whereas if somehow the leader fails, replica takes over as the new leader.

Kafka Producers: In simple words, the processes which publish messages to Kafka are what we call Producers. In addition, it publishes data on the topics of their choice.

Kafka Consumers: The processes that subscribe to topics and processes as well as read the feed of published messages, is what we call Consumers.

Offset in Kafka: The position of the consumer in the log and which is retained on a per-consumer basis is what we call Offset. Moreover, we can say it is the only metadata retained on a per-consumer basis

Kafka Consumer Group: Basically, a consumer abstraction offered by Kafka which generalizes both traditional messaging models of queuing and also publish-subscribe is what we call the consumer group. However, with a consumer group name, Consumers can label themselves.

Introduction about the company:

Nyalazone Solutions is a company focused on AI, Analytics, and Data Management Solutions for enabling a robust New Age Enterprise Data Architecture. Our offerings address the need of today's elastic Enterprise boundary that is continuously readjusted to include nontraditional data generation sources like Social Media, Contents, Weblogs, sensors, machine-generated and Geospatial data, and the dynamic changes of OLTP systems like CRM and ERP, etc.

We leverage Machine Learning, Neural Networks, and Natural Language Processing to build solutions that are based on Artificial Intelligence Models. The solutions based on our platform are successfully used by various private-sector and public-sector organizations to fulfill their analytical needs in critical areas of business operations. We understand the need for data-driven decision-making for distinct operational and competitive market advantage.

Businesses today grapple with the "Datafication" of the world that is extensive and unstructured. We provide solutions that will help you to establish correlations in this world of Big Data so that you can derive value and formalize a sustained cycle of value generation using your enterprise data.

We provide our customers with comprehensive solutions for Data-driven Decision making, based on a platform-centric approach with horizontal and vertical industry solutions. Our offerings span the spectrum of the 'Data Milieu', all the way from Data Engineering to Applied Data Science. Our Robotic Process Automation (RPA) solutions help customers in increasing operational efficiency, reduce costs, and respond to the dynamic business environment with minimal risks of human errors.

The mission of the Company:

Our mission is to establish a low-cost data management platform for an enterprise that is agile and responsive to business needs. While this is achievable by using a combination of technology, our philosophy 'Elegant is Simple' ensures that our offerings are easy to maintain and have a low cost of ownership.

Quick Facts about the Company:

Website: http://www.nyalazone.com

Type: Private Company

Phone No: +91 (0) 120 7195316

Email: info@nyalazone.com

Industry: IT Services and IT Consulting

Key People: Saurabh Kumar(Co-Founder and Chief Executive Officer)

Company size: 50-100 employees

Address: G002, Nyalazone Solutions , GM IT Park, Sector 142, 201301 Noida, India

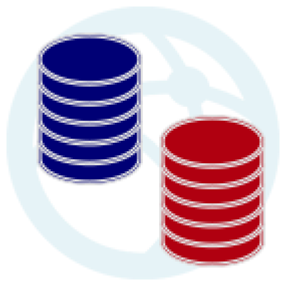
Founded: 2014

Specialties: Big Data, Hadoop, Data Analytics, and Data Warehouse

Company Services:

- **Data Migration**

Our service offering leverages the DDS module of our platform and runs pre-migration data cleansing and transformation needs in real-time. The module runs its proprietary in-memory database for storing commonly used data structures required for transformation and cleansing routines. The in-memory processing capability of the module ensures real-time responses to transformation needs. Leggero Dynamic Data-source is also used in data migration and Integration exercises.



Why choose Nyalazone for your complex data migration handling?

Our data migration services leverage the capabilities of the platform to handle complex migrations, the platform's ability to handle unstructured data opens up avenues for source data extraction that other tools conveniently ignore.

The ability to configure migration logic, persist it and subsequently use it for iterative migration exercises helps drastically reduce time and human errors.

Our ML based data cleansing algorithms help in creating a virtual migration source that is contextual to the target migration system without the need for cumbersome human interventions.

How about using printable text reports of a legacy ERP system as a data source and converting it to strictly normalized SAP ISU input? Leggero DDS has achieved the task and de-risked an SAP implementation

- **Agile BI**

Our Business Intelligence consulting and implementation services challenge the traditional approach to data discovery, insights, and analysis. We leverage our Platforms that have been architected, keeping in mind the ease of use and flexibility while ensuring non-functional requirements like scalability and response time. It provides multidimensional data analysis capability in real-time for large to very large datasets. The Business Intelligence module has a proven track record of ingesting data from multiple sources and formats, transforming and presenting insights in a fraction of the time and costs usually required.

Our expertise in architecting Data Lakes and Business Intelligence solutions is proven and unmatched in the realization of short implementation time and measurable ROI. Our experts help you in data discovery and work with your business teams to define high-impact business measures that enable efficient business operations. Our teams have implemented BI solutions tracking hundreds of KPIs that rely on multiple transactional systems. Our agile implementation approach ensures short sprints for quick wins in an ever-changing business environment.

- **Digital Enterprise Consulting**

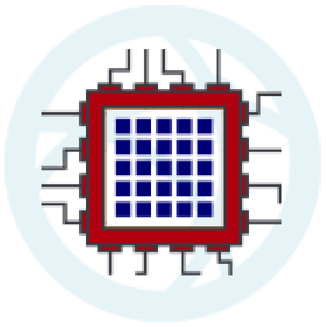
The core foundation of Digital Transformation is in having the ability to make relevant and contextual data available for real-time decision-making. The ground reality is that the data is not available in the form or at the time you want it.
We leverage our expertise to consult customers in identifying the least effort and most effective digital transformation initiative to create a successful transformation journey. Our expertise in various Predictive and Prescriptive Models coupled with RPA has helped us create high-impact transformation initiatives.

Why should I request a consultation with Nyalazone today?

- We will help you identify your quick wins

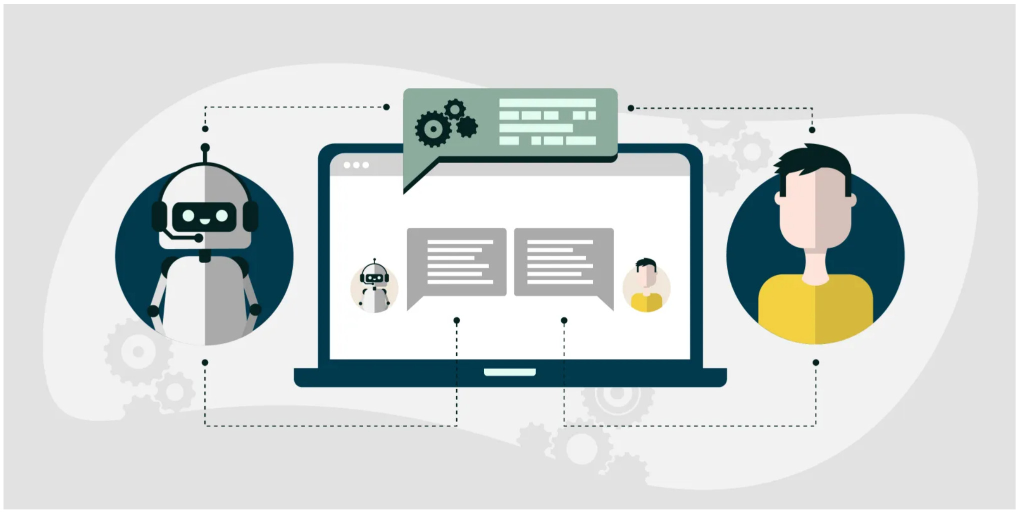
- We will help you in identifying point solutions that work with your existing technology stack.
- Our expertise in Bots and RPA will help you implement high-impact automation solutions.
- Are you embarking on a Digital Transformation Path in your enterprise? Talk to us. We will be happy to have a no-cost session with your team for a roadmap workshop.

- Machine Learning-based Robotic Process Automation (RPA)



Smart machine learning algorithms coupled with Natural Language Processing and workflow enable us to build highly efficient and accurate 'bots' that automate critical business processes, which otherwise are reliant on human intervention. Time-consuming repetitive tasks that are prone to human errors are best left to the 'bot army'.
Whether it is an assistant for your sales team, a bot that presents the most suitable products/services for your customer, a bot assisting your customer retention team to help them identify potential churn or a bot that effectively predicts a 'time-bomb' of an issue that can lead to a large scale social unrest because of service delivery issues, our bots are available 24/7 to help you succeed.

Natural Language Processing



Our smart machine learning algorithms coupled with Natural Language Processing capabilities allow us to extract information about people, places, events, and much more. Mentioned in text documents, articles, or emails which in turn tackle various problems such as providing sentiment analysis for customers to identify opinions and to help them understand what customers think about their products and services or parse intent from customer conversations happening in a call center to flag various issues beforehand. Beyond determining simple polarity, our models help you better understand what is behind an expressed opinion, and predict a favorable or unfavorable outcome, which can be extremely relevant in understanding and driving behavioral decisions.

Why choose complex AI/ML-based RPA or Model needs?
- Relevant to your business
- Services leveraging a comprehensive platform that supports concept to deploy lifecycle, supported by strong workflow and integration capability.
- Proven and referenceable expertise
- Flexible commercial engagement models
- World-class performance

Platforms

- **Leggero**

Leggero

The Leggero Data Management & Analytics Platform allows you to store, access, homogenize and analyze data from various sources and in different formats. The platform is built to scale for multiple terabytes of data and handle a continuous in-flow of data. The system provides features to collate and correlate data from various sources and analyze it for various facts based on multiple dimensions. The platform has abilities to handle unstructured data like Text Files and XML Files and convert them to datasets. These datasets can further be used for simple reporting or be used to run various models for complex analysis. The platform can be deployed on dedicated hardware environments or on the cloud depending on the implementation needs and preferences. The analyzed information can be presented in the form of reports, charts and graphs that can be configured for various roles and privileges to access reports and charts/graphs.

The platform is a comprehensive data lifecycle management solution, it provides the necessary tools and frameworks that enables complex data engineering, data storage, metadata management, meta-semantics, advanced analytics and visualization.

The platform supports Big Data over a distributed computing environment for large scale data processing and analysis. The platform has to be deployed in a clustered environment if large volumes of data is processed, it can be installed and run on a non distributed environment in case of lesser volume of data processing and management.

System Architecture and Functionality

The Platform has been architected keeping in mind the ease of use and flexibility while ensuring nonfunctional requirements like scalability and response time. The information analyzed can be presented in the form of Reports and Dashboards that comprise charts and graphs. The data can be simply published as a text dataset in case a different system requires to consume the data. The data can be presented in single-series or multi-series charts for example: Pie Charts, Area Charts, or Bar and Column charts. The system has the ability to analyze and publish data for times, series or trends and other data series analysis.

Why choose Nyalazone's Leggero Data Management & Analytics Platform?

- Highly scalable
- Advanced visualization
- You are not restricted to a design time schema of the warehouse
- Co-existent with traditional warehouse and marts
- Significantly Low TCO

- **DDS**

Dynamic
Data
Source

Nyalazone’s Dynamic Data Source platform allows the configuration of various transformation rules that are persisted and subsequently used for iterative data cleansing and transformational needs. The module runs its own proprietary, in- Memory Database (Data Objects) for storing commonly used data structures that are required for transformation and cleansing routines. The in-memory processing capability of the module ensures near real-time responses on transformation needs. DDS facilitates data transformation exercises with various features that allow for very complex rules to be applied to an attribute or an instance level of a source entity.

The unique concept of ‘Computed Columns’ allows for multiple source columns to be used as inputs for transformation exercises to derive an output column.

Data cleansing and consolidation are facilitated with Fuzzy Logic Matches and Pattern Matches. The platform has the ability to convert unstructured data to structured data for migration needs. Multiple sources can be joined, correlated or used for mappings to perform an attribute or instance-level transformation.

Why choose Nyalazone's Leggero Data Management & Analytics Platform?

Real-Time Lightning Fast Data Transformation Handle Multiple Data Sources:

- JSON, XML
- Complex Multiline Text
- Excel, word & PDF
- RDBMS
- Web Scraping
- In Memory Data Objects, Fuzzy Logic & Pattern Match, Join Across Different Source, High Volume Data Handling, Handle Data Transformation

Tasks can be done like:
- Mapped Transformation
- Dynamic &Amp; Function Based Lookups
- Self-Computed Columns
- Grouped Functions
- Filter Functions
- Vertical Aggregation, Horizontal Extraction, Persistent Transformation Rules

- **Intelli ML**

The IntelliML module allows for building and deploying Bots. It provides support for strong workflow and integration capabilities which allow the deployment of production ready bots. Deploying bots that help in automating tasks that are mission-critical and cannot be left to human errors, or tasks that rely on individual judgment that induces errors because of the emotional state of an individual or tasks that are repetitive and time consuming that takes thousands of man-hours and yet do not yield results. These are just some of the cases fit for RPA using Bots deployed on our platform.

Some of our 'Busy Bots':

Churn Management Predictor
Churn Management Predictor helps to predict if a customer is potentially going to drop out. The model will predict the likelihood of customers dropping out. The model can be run periodically on the entire customer dataset to produce a list of potential dropouts. This list can be used for various remedial actions to assist with customer retention. The Conversion Model (described above) can be used to retain a customer by predicting suitable product alternatives.

Conversion Predictor
Conversion Predictor helps by suggesting the best product or service alternative to a customer based on Demographic Segmentation, behavioral profile, usage profile, other financial data available and financial needs analysis.

Effective Call Time
This Bot identifies the best time to call a potential customer based on identified parameters, it works with the campaign manager to create an outbound call list that enhances the chance of having effective communication with the potential customer that will lead to a sale or efficient service call.

New Sales Predictor
New Sale Prediction Model allows for the prediction of "best fit" products or services for a particular customer based on their fitment in a specific demographic segmentation. The customer segmentation is based on an unsupervised learning model. It leverages various behavioral attribute-relevant predictions that then drive the sales process.

Application to Customer Support
This Bot helps in tracking and predicting a successful lead-to-cash-cycle. Based on various customer interactions, requests and response time/frequencies or artifact exchanges, the bot will help in the continuous monitoring of leads to an effective sale.

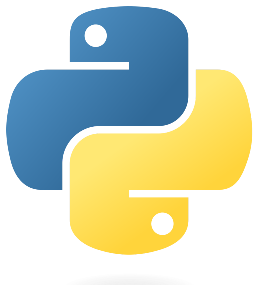
Email to Service Request / Tickets Generator
This model uses Natural Language Processing (NLP) and Machine Learning Algorithms to categorize emails that can be used to create leads, service requests or tickets etc. The model relies on historical data and emails to identify key issues raised in an email and further classify

the email to a particular service request category and sub-category. This model helps in improving operational efficiency by reducing human interventions.

Critical Issue Identifier
This model is based on NLP and ML algorithms and helps in identifying critical issues highlighted in emails. This helps in identifying potential “time bombs” and addresses such issues without letting them blow up, thus avoiding the significant cost and reputational damages.

Why choose Nyalazone for your machine learning algorithms and AI?
Nyalazone is an end-to-end provider for ML-enabled Models and Bots, we take complete ownership right from the data source to the bot-driven action. Our matured platform tools and methodology reduces the time spent on data engineering tasks, which allows us to focus better on Business Process adaptation for the model development and iteratively improving the models as well. Our typical bot deployment cycle is 90 days which helps our customers realize quick wins and thus get better support for the transformation initiatives.

1. Python:



Python is an interpreted, object-oriented, high-level, dynamically semantic programming language. It is particularly desirable for Rapid Application Development as well as for usage as a scripting or glue language to tie existing components together due to its high-level built-in data structures, dynamic typing, and dynamic binding. Python's straightforward syntax prioritizes readability and makes it simple to learn, which lowers the cost of programme maintenance. Python's support for modules and packages promotes the modularity and reuse of code in programmes. For all popular platforms, the Python interpreter and the comprehensive standard library are freely distributable and available in source or binary form.

Python frequently causes programmers to fall in love due to the enhanced productivity it offers. The edit-test-debug cycle is extraordinarily quick because there is no compilation step. Python programmes are simple to debug since a segmentation failure is never caused by a bug or incorrect input. Instead, the interpreter raises an exception when it finds a mistake. The interpreter prints a stack trace if the application doesn't catch the exception. Setting breakpoints, evaluating arbitrary expressions, inspecting local and global variables, stepping through the code one line at a time, and other features are all possible with a source level debugger. Python's ability to perform introspection is demonstrated by the debugger, which is developed in Python.

Python's ability to perform introspection is demonstrated by the debugger, which is developed in Python.

History:

In February 1991, Van Rossum published the code (labeled version 0.9.0) to alt.sources. Already present at this stage in development were classes with inheritance, exception handling, functions, and the core datatypes of list, dict, str and so on. Also in this initial release was a module system borrowed from Modula-3; Van Rossum describes the module as "one of Python's major programming units' ' Python's exception model also resembles Modula-3's, with the addition of an else clause. In 1994 comp.lang.python, the primary discussion forum for Python, was formed, marking a milestone in the growth of Python's user base

2. Docker:

Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers.The service has both free and premium tiers. The software that hosts the containers is called Docker Engine.It was first started in 2013 and is developed by Docker, Inc.

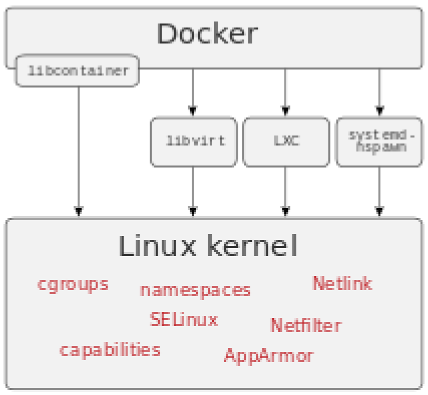
Background

Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels.Because all of the containers share the services of a single operating system kernel, they use fewer resources than virtual machines.

Operation:

Docker can package an application and its dependencies in a virtual container that can run on any Linux, Windows, or macOS computer. This enables the application to run in a variety of locations, such as on-premises, in public (see decentralized computing, distributed computing, and cloud computing) or private cloud.When running on Linux, Docker uses the resource isolation features of the Linux kernel (such as cgroups and kernel namespaces) and a union-capable file system (such as OverlayFS) to allow containers to run within a single Linux instance, avoiding the overhead of starting and maintaining virtual machines.Docker on macOS uses a Linux virtual machine to run the containers.

Because Docker containers are lightweight, a single server or virtual machine can run several containers simultaneously.A 2018 analysis found that a typical Docker use case involves running eight containers per host, and that a quarter of analyzed organizations run 18 or more per host.It can also be installed on a single board computer like the Raspberry Pi.



The Linux kernel's support for namespaces mostly isolates an application's view of the operating environment, including process trees, network, user IDs and mounted file systems, while the kernel's cgroups provide resource limiting for memory and CPU.Since version 0.9, Docker includes its own component (called "libcontainer") to use virtualization facilities provided directly by the Linux kernel, in addition to using abstracted virtualization interfaces via libvirt, LXC and systemd-nspawn.

Docker implements a high-level API to provide lightweight containers that run processes in isolation.Docker containers are standard processes, so it is possible to use kernel features to monitor their execution—including for example the use of tools like strace to observe and intercede with system calls.

Components
The Docker software as a service offering consists of three components:

● Software: The Docker daemon, called dockerd, is a persistent process that manages Docker containers and handles container objects. The daemon listens for requests sent via the Docker Engine API.The Docker client program, called docker, provides a command-line interface (CLI), that allows users to interact with Docker daemons.
● Objects: Docker objects are various entities used to assemble an application in Docker. The main classes of Docker objects are images, containers, and services.
● A Docker container is a standardized, encapsulated environment that runs applications.A container is managed using the Docker API or CLI.
● A Docker image is a read-only template used to build containers. Images are used to store and ship applications.
● A Docker service allows containers to be scaled across multiple Docker daemons. The result is known as a swarm, a set of cooperating daemons that communicate through the Docker API.
● Registries: A Docker registry is a repository for Docker images. Docker clients connect to registries to download ("pull") images for use or upload ("push") images that they have built.

Registries can be public or private. The main public registry is Docker Hub. Docker Hub is the default registry where Docker looks for images.Docker registries also allow the creation of notifications based on events.

Tools:

- Docker Compose is a tool for defining and running multi-container Docker applications.It uses YAML files to configure the application's services and performs the creation and start-up process of all the containers with a single command. The docker-compose CLI utility allows users to run commands on multiple containers at once, for example, building images, scaling containers, running containers that were stopped, and more.Commands related to image manipulation, or user-interactive options, are not relevant in Docker Compose because they address one container.The docker-compose.yml file is used to define an application's services and includes various configuration options. For example, the build option defines configuration options such as the Dockerfile path, the command option allows one to override default Docker commands, and more. The first public beta version of Docker Compose (version 0.0.1) was released on December 21, 2013.The first production-ready version (1.0) was made available on October 16, 2014.

- Docker Swarm provides native clustering functionality for Docker containers, which turns a group of Docker engines into a single virtual Docker engine. In Docker 1.12 and higher, Swarm mode is integrated with Docker Engine.The docker swarm CLI utility allows users to run Swarm containers, create discovery tokens, list nodes in the cluster, and more.The docker node CLI utility allows users to run various commands to manage nodes in a swarm, for example, listing the nodes in a swarm, updating nodes, and removing nodes from the swarm.Docker manages swarms using the Raft consensus algorithm. According to Raft, for an update to be performed, the majority of Swarm nodes need to agree on the update.

Docker Volume facilitates the independent persistence of data, allowing data to remain even after the container is deleted or re-created.

History

Docker Inc. was founded by Kamel Founadi, Solomon Hykes, and Sebastien Pahl during the Y Combinator Summer 2010 startup incubator group and launched in 2011.The startup was also one of the 12 startups in Founder's Den first cohort.Hykes started the Docker project in France as an internal project within dotCloud, a platform-as-a-service company.

Docker debuted to the public in Santa Clara at PyCon in 2013.It was released as open-source in March 2013. At the time, it used LXC as its default execution environment. One year later, with the release of version 0.9, Docker replaced LXC with its own component, libcontainer, which was written in the Go programming language.

In 2017, Docker created the Moby project for open research and development.

3. Search Engine:

We use an open source leading search engine. Coming to what is Open Source

Open source software is software with source code that anyone can inspect, modify, and enhance.

"Source code" is the part of software that most computer users don't ever see; it's the code computer programmers can manipulate to change how a piece of software—a "program" or "application"—works. Programmers who have access to a computer program's source code can improve that program by adding features to it or fixing parts that don't always work correctly.

Some software has source code that only the person, team, or organization who created it—and maintains exclusive control over it—can modify. People call this kind of software "proprietary" or "closed source" software.

Only the original authors of proprietary software can legally copy, inspect, and alter that software. And in order to use proprietary software, computer users must agree (usually by signing a license displayed the first time they run this software) that they will not do anything with the software that the software's authors have not expressly permitted. Microsoft Office and Adobe Photoshop are examples of proprietary software.

Open source software is different. Its authors make its source code available to others who would like to view that code, copy it, learn from it, alter it, or share it. LibreOffice and the GNU Image Manipulation Program are examples of open source software.

As they do with proprietary software, users must accept the terms of a license when they use open source software—but the legal terms of open source licenses differ dramatically from those of proprietary licenses.

A search engine is a software program that helps people find the information they are looking for online using search queries containing keywords or phrases.

Search engines are able to return results quickly even with millions of records by indexing every data record they find.

4. Zabbix

Zabbix is an open-source software tool to monitor IT infrastructure such as networks, servers, virtual machines, and cloud services. Zabbix collects and displays basic metrics.
Zabbix is designed primarily as an IT infrastructure monitoring tool. New features are generally released every six months to major versions and every 1.5 years to LTS versions.

Released under the terms of GNU General Public License version 2, Zabbix is free software that does not require a license to use any of its features. Even though Zabbix is open-source software, it is a closed development software product, developed by Zabbix LLC based in Riga, Latvia.

Early in its history, Zabbix was described as simple to set up compared to other monitoring solutions.However, later it was considered by some to need a significant amount of manual configuration. As an open-source product however Zabbix focusses on the usage of existing tools and functionality as well as proprietary solutions to achieve a scalable monitoring solution.

5. Fluentd

Fluentd is a widely used, cross-platform, open-source data collection software project that was initially developed by Treasure Data. This powerful tool is primarily written in the Ruby programming language and is designed to handle big data sets, specifically semi-structured or unstructured data. It is capable of analyzing event logs, application logs, and clickstreams, making it a valuable tool for a wide range of applications. According to Suonsyrjä and Mikkonen, the "core idea of Fluentd is to be the unifying layer between different types of log inputs and outputs".

One of the most significant advantages of Fluentd is its versatility. It can be used on Linux, macOS, and Windows, making it a useful tool for a wide range of platforms. Fluentd was developed by Sadayuki Furuhashi as a project of the Mountain View-based firm Treasure Data. Its source code was released as open-source software in October 2011, and the company announced $5 million of funding in 2013. Treasure Data was then sold to Arm Ltd. in 2018, further expanding the reach and potential of Fluentd.

Fluentd has gained widespread recognition and adoption in the industry since its initial release. In 2013, Amazon Web Services recommended Fluentd as one of the data collection tools, along with Apache Flume or Scribe. Google Cloud Platform's BigQuery also recommends Fluentd as the default real-time data-ingestion tool and uses Google's customized version of Fluentd, called google-fluentd, as a default logging agent

Fluent Bit is another project that is being developed under the Fluentd project umbrella. It is a log processor and log forwarder and is currently a CNCF sub-project. Unlike Fluentd, Fluent Bit is written only in C and has no dependencies, making it an excellent choice for embedded Linux and container environments

While Fluentd is a powerful tool, it does consume a significant amount of memory resources since it is
written in both C and Ruby and built as a Ruby gem. However, with the development of Fluent Bit, memory usage has decreased, making it an even more attractive option for users who prefer lower memory consumption

In conclusion, Fluentd is a highly versatile, open-source data collection software project that has gained widespread adoption in the industry. It is capable of analyzing a wide range of data sets and is available on multiple platforms, making it an ideal tool for various applications. Additionally, the development of Fluent Bit has further expanded Fluentd's reach, making it even more attractive for users who prefer lower memory consumption.

6. Grafana

Grafana is an open-source analytics and monitoring platform that provides a wide range of data visualization options. It allows users to query, visualize, alert on, and understand metrics no matter where they are stored. Grafana is widely used in various domains such as DevOps, IoT, and Business Intelligence.

The first version of Grafana was released in 2014 by Torkel Ödegaard. Grafana has become popular because of its ease of use, extensibility, and the ability to visualize complex data in a way that is easy to understand. Grafana has a large and active community that contributes to the development of plugins and dashboards.

Grafana supports a wide range of data sources, including popular time-series databases such as Prometheus, InfluxDB, Graphite, Elasticsearch, and more. It also supports SQL databases such as MySQL, Postgres, and Microsoft SQL Server. Grafana supports cloud platforms such as Amazon Web Services, Google Cloud Platform, and Microsoft Azure.

Grafana provides a wide range of data visualization options such as charts, graphs, heatmaps, gauges, and tables. It also provides advanced features such as annotations, alerts, and dashboards. Grafana dashboards can be customized and shared with others, allowing teams to collaborate and share data insights.

One of the key features of Grafana is its ability to create and manage alerts. Alerts can be created based on various conditions such as thresholds, patterns, and anomalies. Grafana can send alerts to various channels such as email, Slack, PagerDuty, and more. Grafana also provides an API to create, manage, and query alerts.

Grafana has a plugin system that allows users to extend its capabilities. There are hundreds of plugins available that add new data sources, visualizations, and integrations. Grafana plugins are easy to install and use, and they can be shared with the community.

Grafana has become popular in the DevOps community because it provides a unified platform for monitoring and visualization. It allows DevOps teams to monitor infrastructure, applications, and services in a single place. Grafana supports popular monitoring tools such as Prometheus, Graphite, and InfluxDB, making it easy to integrate with existing infrastructure.

Grafana is also widely used in the IoT domain. It provides a platform to collect, analyze, and visualize data from sensors and devices. Grafana supports popular IoT protocols such as MQTT and Modbus. Grafana allows users to create custom dashboards for IoT data, making it easy to understand and analyze sensor data.

Grafana has become popular in the Business Intelligence domain because of its ability to visualize complex data in a simple and easy-to-understand way. Grafana supports SQL databases, making it easy to create dashboards and reports for business data. Grafana provides features such as drill-down, filtering, and exporting to PDF and CSV, making it easy to analyze and share data insights.

In conclusion, Grafana is a powerful and flexible platform for monitoring, analyzing, and visualizing data. It supports a wide range of data sources and provides a wide range of visualization options. Grafana is widely used in the DevOps, IoT, and Business Intelligence domains. Its ease of use, extensibility, and active community make it an attractive choice for organizations looking to gain insights from their data.

7. Kafka

Apache Kafka is an open-source distributed streaming platform that is designed to handle large-scale, high-throughput, real-time data feeds. It was originally developed by LinkedIn and later donated to the Apache Software Foundation in 2011, and has since become one of the most popular distributed streaming platforms in use today.

At its core, Kafka is a distributed messaging system that allows for the exchange of data between applications or microservices in a fault-tolerant and scalable manner. Data is published to a Kafka topic, which is a logical stream of data that can be partitioned across a cluster of Kafka brokers. Consumers can then subscribe to these topics to consume the data in real-time or at their own pace, depending on their needs.

One of the key benefits of Kafka is its ability to handle large amounts of data at scale. Kafka is designed to handle millions of messages per second, making it an ideal solution for use cases such as real-time analytics, log aggregation, and event streaming. It is also highly scalable, with the ability to add more brokers to a cluster as needed to handle increasing workloads.

Kafka is also fault-tolerant, which means that it can continue to function even if one or more brokers in a cluster fail. This is achieved through the use of replication, where data is replicated across multiple brokers to ensure that there is always a backup in case of failure. Additionally, Kafka is designed to handle a variety of data formats, including binary, JSON, and Avro.

Kafka's architecture is composed of four main components: producers, consumers, topics, and brokers. Producers are responsible for publishing data to a Kafka topic, while consumers subscribe to these topics to consume the data. Topics are logical streams of data that can be partitioned across multiple Kafka brokers, and brokers are the nodes that make up the Kafka cluster.

Kafka also provides a number of additional features that make it a powerful tool for real-time data processing. These include support for stream processing through the Kafka Streams API, which allows developers to build real-time processing applications using a simple Java API. Kafka also integrates with a number of popular data processing frameworks such as Apache Spark, Apache Flink, and Apache Storm.

One of the main use cases for Kafka is log aggregation. Many organizations generate a large amount of log data, which can be difficult to manage and analyze in real-time. Kafka can be used to collect log data from multiple sources and stream it to a central location for processing and analysis. This makes it easier to identify issues and monitor the health of distributed systems.

Another use case for Kafka is real-time analytics. By streaming data to Kafka topics in real-time, organizations can build real-time analytics pipelines that allow them to make data-driven decisions quickly. Kafka also provides support for complex event processing, which allows for the detection of patterns and trends in real-time data.

Kafka has become an essential tool for many organizations that rely on real-time data processing. Its ability to handle large amounts of data at scale, while providing fault-tolerance and support for a variety of data formats, makes it an ideal solution for a wide range of use cases. Additionally, Kafka's rich ecosystem of tools and integrations make it easy to build complex data processing pipelines that can handle even the most demanding workloads.

8. Apache Flink

Apache Flink is an open-source distributed data processing platform that allows for real-time data streaming and batch processing. It was developed by the Apache Software Foundation, and its first version was released in 2014. Flink is designed to be fault-tolerant, scalable, and efficient, making it ideal for large-scale data processing applications. In this report, we will discuss the architecture, features, and use cases of Apache Flink.

Architecture:

Apache Flink has a unique architecture that makes it stand out from other data processing platforms. It follows a dataflow-based architecture that allows for parallel data processing. The dataflow graph is a directed acyclic graph (DAG) that is built by the Flink engine. Each node in the DAG represents an operator that performs a specific function on the data. The data is processed in-memory, which makes Flink very fast.

The Flink cluster consists of two types of nodes: Job Manager and Task Manager. The Job Manager is responsible for scheduling and coordinating tasks, while the Task Manager is responsible for executing the tasks. The data is partitioned and distributed across the Task Managers for parallel processing. In case of a Task Manager failure, Flink automatically redistributes the data to other Task Managers and resumes processing without any loss of data.

Features:

Apache Flink has several features that make it a powerful data processing platform. One of the key features is its support for real-time data processing. Flink can handle streaming data with low latency and high throughput. It can also handle out-of-order data and perform windowing operations on the data stream.

Flink also supports batch processing, which makes it a versatile platform for different types of data processing applications. It can handle batch processing of large datasets with high performance and fault-tolerance. Flink's support for batch processing allows for hybrid processing, where both real-time and batch processing can be combined in a single application.

Flink provides a rich set of APIs for data processing, including Java, Scala, and Python. These APIs allow for easy integration with other data processing platforms and tools. Flink also supports a wide range of connectors for data sources and sinks, including Kafka, Hadoop, and Elasticsearch.

Another key feature of Flink is its support for stateful stream processing. Flink can maintain state across stream processing operations, which allows for complex processing of data streams. Flink's support for stateful processing makes it a powerful platform for applications such as fraud detection, anomaly detection, and predictive analytics.

Use Cases:

Apache Flink has several use cases in different industries. In the financial industry, Flink is used for real-time fraud detection and transaction monitoring. Flink can process large volumes of financial data in real-time, which allows for quick detection and prevention of fraudulent activities.

In the e-commerce industry, Flink is used for real-time product recommendations and customer personalization. Flink can process user behavior data in real-time and provide personalized recommendations to users.

In the telecommunications industry, Flink is used for network monitoring and optimization. Flink can process real-time network data and detect anomalies and potential issues before they become major problems.

Introduction

The company operates on a task-based structure, where each individual is assigned a specific task with a deadline for submission. The assigned tasks are reviewed daily by supervisors and mentors to provide guidance and support, with problem statements discussed during daily meetings scheduled at specific times.

My daily meeting is scheduled at 11:30 a.m. every day, where I discuss the previous day's progress with senior employees who review and verify the work completed. During the meeting, we discuss various topics, including the task requirements, available resources, and the expected outcomes. We also develop a workflow to ensure that the task is completed efficiently and effectively, without any errors or loopholes.

Once the plan is finalized, I begin working on the assigned task. In my internship, I have worked on various tasks, and each one has provided me with valuable learning outcomes.

During my internship at the company, my primary task was to conduct research on implementing new software projects as instructed by my mentors. This research process included several stages, which were crucial for the successful implementation of the software.

The first stage was the compatibility analysis, where I had to ensure that the proposed software was compatible with the existing systems and platforms used by the company. This was essential to prevent any potential conflicts that could arise during implementation and to ensure the smooth functioning of the software.

After ensuring the compatibility, the next stage was to identify the use cases of the proposed software. This involved studying the different scenarios in which the software would be used and how it could benefit the company. This information was then used to define the project requirements and objectives, which would serve as a roadmap for the implementation process.

In the third stage, I had to research and evaluate different implementation approaches for the project. This involved analyzing various strategies for developing and deploying the software and selecting the most efficient and effective one that would meet the project's requirements.

Once the best implementation approach was identified, I had to learn the tech stack required to implement the software. This involved researching and familiarizing myself with the programming languages, frameworks, and tools needed to develop and deploy the software. This was a critical stage that required me to have a good understanding of the software development process and experience with programming languages and technologies.

After learning the required tech stack, I had to implement and test the software in a local environment to ensure that it worked as expected. This involved writing code, building the software, and running tests to verify its functionality and performance. Any issues or bugs that were discovered during this stage were addressed and resolved before moving on to the next stage.

Finally, the software was implemented on the company's server for actual usage. This involved deploying the software on the production environment and conducting additional tests to ensure its compatibility and reliability. Any issues or bugs that were discovered during this stage were addressed and resolved to ensure the software's smooth functioning.

Throughout the research process, I learned various skills and gained valuable experience in software development, including compatibility analysis, use case analysis, implementation approaches, learning tech stacks, and implementing and testing software. I also learned the importance of proper planning and execution in software development and how it can affect the success of a project.

EFG Stack Implementation:

The EFG (Elastic, Fluentd, Grafana) stack is a popular open-source monitoring solution that is used to collect, store, and visualize data. It is composed of three main components:

Elasticsearch is a distributed, scalable, and highly available NoSQL database that is designed for storing and querying large amounts of data.
Fluentd is a data collector that can be used to collect data from a variety of sources, including servers, applications, and networks.
Grafana is a powerful visualization tool that can be used to create interactive dashboards and charts from data stored in Elasticsearch.
The EFG stack is a powerful and flexible monitoring solution that can be used to monitor a wide variety of systems and applications. It is easy to set up and use, and it is scalable to meet the needs of even the most demanding applications.

Elasticsearch:
Elasticsearch is a distributed, scalable, and highly available NoSQL database that is designed for storing and querying large amounts of data. It is a popular choice for storing log data, metrics data, and other types of time-series data.

Elasticsearch is a key-value store, which means that data is stored in documents that are indexed by a unique key. This makes it easy to search for and retrieve data. Elasticsearch also supports a variety of querying methods, including full-text search, faceted search, and geospatial search.

Elasticsearch is a scalable solution that can be easily expanded to meet the needs of growing applications. It is also highly available, with built-in features for replication and failover.

Fluentd:
Fluentd is a data collector that can be used to collect data from a variety of sources, including servers, applications, and networks. It is a powerful tool that can be used to collect data in a variety of formats, including JSON, XML, and CSV.

Fluentd is a flexible tool that can be customized to meet the needs of different applications. It supports a variety of input and output plugins, and it can be configured to collect data from a variety of sources.
Fluentd is a popular choice for collecting log data. It can be used to collect log data from a variety of sources, including servers, applications, and networks. Fluentd can also be used to collect metrics data from applications and systems.

Grafana:
Grafana is a powerful visualization tool that can be used to create interactive dashboards and charts from data stored in Elasticsearch. It is a user-friendly tool that can be used to create dashboards that are tailored to the needs of different users.

Grafana supports a variety of data sources, including Elasticsearch, Prometheus, and InfluxDB. It also supports a variety of visualization types, including line charts, bar charts, and pie charts.

Grafana is a popular choice for visualizing log data. It can be used to create dashboards that show the status of applications, the performance of systems, and the health of networks.

EFG Stack Benefits:
The EFG stack is a powerful and flexible monitoring solution that offers a number of benefits, including:

- Scalability: The EFG stack is scalable to meet the needs of even the most demanding applications.
- High availability: The EFG stack is highly available, with built-in features for replication and failover.
- Flexibility: The EFG stack is flexible and can be customized to meet the needs of different applications.
- Ease of use: The EFG stack is easy to set up and use, even for beginners.
- Cost-effectiveness: The EFG stack is a cost-effective solution that can be used to monitor a wide variety of systems and applications.

EFG Stack Use Cases:
The EFG stack can be used to monitor a wide variety of systems and applications, including:

- Web applications: The EFG stack can be used to monitor the performance and health of web applications.
- Databases: The EFG stack can be used to monitor the performance and health of databases.
- Servers: The EFG stack can be used to monitor the performance and health of servers.
- Networks: The EFG stack can be used to monitor the performance and health of networks.
- IoT devices: The EFG stack can be used to monitor the performance and health of IoT devices.

Difficulties Faced:

During the implementation of the EFG stack, it was found that the easiest part was setting up Elastic and Grafana. These two components did not require much configuration, and the setup process was relatively straightforward. However, when it came to Fluentd, there were some challenges that needed to be addressed.

One of the main difficulties in implementing Fluentd was processing the different file formats. While some log formats, such as Apache and Nginx logs, were easy to process, there were customized logs that required custom formatting. This meant that a lot of research had to be conducted to figure out how to process these logs properly. Additionally, there were issues with determining the appropriate flush intervals for different logs. This required careful consideration to ensure that the logs were processed efficiently without overloading the system.

Once the Fluentd implementation was working correctly, the files were processed and sent to Elastic search indexes. However, even after the logs were successfully sent to Elastic search indexes, there were still challenges that needed to be addressed. One of the most difficult parts of the implementation was processing the application logs.

Application logs can be highly customized and can vary greatly from one application to another. As a result, it was challenging to determine how to process these logs correctly. The team had to carefully analyze the logs and determine the best way to extract the relevant information. This required a deep understanding of the application and how it worked.

Another challenge with processing application logs was ensuring that they were processed efficiently. Because application logs can be highly detailed, they can also be quite large. This meant that the processing time for these logs could be significant, which could impact the overall system performance. To address this issue, the team had to carefully balance the processing requirements with the available system resources.

In addition to the technical challenges associated with implementing the EFG stack, there were also organizational challenges that needed to be addressed. One of the biggest challenges was ensuring that all team members were on the same page and had a clear understanding of the implementation process. This required frequent communication and collaboration between team members.

Another challenge was managing the workload associated with the implementation. The implementation process required a significant amount of research and testing, which could be time-consuming. To address this issue, the team had to carefully prioritize tasks and ensure that everyone had a clear understanding of their responsibilities.

Despite the challenges associated with implementing the EFG stack, the project was ultimately successful. By carefully analyzing the log formats, determining the appropriate processing intervals, and optimizing the system resources, the team was able to successfully implement the EFG stack and provide valuable insights into the system performance. Additionally, the project helped to build stronger communication and collaboration within the team, which will be valuable for future projects.

Zabbix Implementations:

Zabbix is an open-source monitoring software that helps organizations to monitor and track the performance of their IT infrastructure. The software can monitor a wide range of components in the IT infrastructure, such as servers, network devices, databases, applications, and more. One of the essential components of the Zabbix monitoring system is the Zabbix agent, which is responsible for collecting data from the monitored host and sending it to the Zabbix server for processing and analysis.

The Zabbix agent is a small software that is installed on the host that needs to be monitored. The agent is responsible for collecting data from various sources on the host and sending it to the Zabbix server. The agent is lightweight and designed to consume minimal system resources, making it suitable for use in various types of environments.

Setting up the Zabbix agent involves several steps, including installing the agent software, configuring the agent, and verifying that the agent is communicating with the Zabbix server correctly. The following sections provide more details on each of these steps.

- **Installing the Zabbix Agent**

The first step in setting up the Zabbix agent is to install the agent software on the host that needs to be monitored. The Zabbix agent is available for various operating systems, including Windows, Linux, Unix, and macOS. The installation process for the agent may vary depending on the operating system being used. However, the general process involves downloading the agent software from the Zabbix website, extracting the files, and running the installation script.

- **Configuring the Zabbix Agent**

After installing the Zabbix agent, the next step is to configure the agent to communicate with the Zabbix server. The configuration process involves editing the agent configuration file, which is typically located in the /etc/zabbix/ directory on Linux and Unix systems or in the installation directory on Windows systems.

The configuration file contains several parameters that need to be set up, such as the hostname or IP address of the Zabbix server, the port number, and the communication protocol to be used.

Other parameters that can be set up in the configuration file include the interval at which the agent sends data to the server, the log file path, and the monitoring items to be enabled.

- **Verifying the Zabbix Agent Communication**

After configuring the Zabbix agent, the next step is to verify that the agent is communicating correctly with the Zabbix server. This can be done by checking the Zabbix server dashboard or by using the Zabbix agent status command on the monitored host.

The Zabbix server dashboard provides real-time data on the performance of the monitored hosts, including metrics such as CPU usage, memory usage, network traffic, and more. By checking the dashboard, administrators can ensure that the Zabbix agent is collecting data from the monitored host and sending it to the server.

Another way to verify the Zabbix agent communication is by using the Zabbix agent status command on the monitored host. This command provides information on the status of the Zabbix agent and its communication with the server. Administrators can use this command to check the agent's configuration, the version of the agent software, and the last communication time with the server.

- **Troubleshooting Zabbix Agent Issues**

Despite its ease of use, setting up the Zabbix agent can sometimes be challenging, especially when dealing with complex IT infrastructures. Some of the common issues that may arise during the Zabbix agent setup include configuration errors, network connectivity issues, and firewall problems.

To troubleshoot these issues, administrators can use various tools such as network monitoring tools, firewall logs, and Zabbix agent logs. The Zabbix agent logs provide information on the communication between the agent and the server, and any errors that may have occurred during the communication process. By analyzing these logs, administrators can identify the root cause of the issue and take the necessary steps to resolve it.

Kafka Architecture for Data Replications:

In this task, research plays a crucial role in identifying the best implementations, workarounds, local implementations, pros and cons, drawbacks, and use cases. The process of researching involves collecting and analyzing information from various sources, such as online articles, technical documentation, and case studies. The research phase helps in understanding the available options and selecting the best one based on the requirements and limitations of the project. It also enables us to identify potential problems and work on solutions before implementation, ensuring a smoother process overall. Additionally, researching the use cases helps in understanding how the solution has worked in similar situations, providing insights into how it can be tailored to the specific project's needs.

Researching also helps in identifying the drawbacks and limitations of the solution, enabling the team to work on workarounds and solutions beforehand, minimizing the risks of failure during implementation. Local implementations help in testing the solution in a controlled environment, enabling the team to work on fine-tuning the solution before moving on to the production environment. The research phase is crucial in identifying the technical requirements and limitations of the solution, helping the team to learn and understand the tech stack required for implementation. In summary, research is a crucial phase in any project, especially when it comes to implementing new software. It helps in identifying the best options, minimizing the risks of failure, and ensuring a smoother implementation process overall.

First Implementation:

Kafka is a distributed streaming platform that is used to build real-time data pipelines and streaming applications. Kafka provides a messaging system that is capable of handling high volumes of data in real-time, making it an ideal choice for data streaming applications. One of the key features of Kafka is its ability to handle data from a variety of sources, including databases. The Debezium connector is a tool that enables data to be extracted from databases and streamed into Kafka. This makes it possible to integrate databases with Kafka and build real-time data pipelines.

The first step in implementing the Kafka architecture is to set up the basic infrastructure. This involves installing Kafka and the necessary components, such as the Debezium connector and the Kafka connector. Once the infrastructure is set up, the next step is to connect the database to Kafka using the Debezium connector. This involves configuring the connector to extract data from the database and stream it into Kafka.

The Debezium connector is designed to work with a variety of databases, including MySQL, PostgreSQL, and MongoDB. It monitors the database transaction log and streams changes to the data into Kafka. This makes it possible to capture changes to the data in real-time and process it as it is generated. The Debezium connector is a powerful tool that enables real-time data integration between databases and Kafka.

Once the data is in Kafka, it can be processed and analyzed using a variety of tools. One common use case for Kafka is to stream data into Elasticsearch. Elasticsearch is a distributed search and analytics engine that is capable of handling large volumes of data in real-time. By streaming data from Kafka into Elasticsearch, it is possible to build real-time search and analytics applications.

The Kafka connector is a tool that enables data to be streamed from Kafka into Elasticsearch. It is designed to handle large volumes of data and can scale to handle very high volumes of traffic. The Kafka connector can be configured to stream data into Elasticsearch in real-time, making it possible to build real-time search and analytics applications.

In summary, the Kafka architecture provides a powerful platform for building real-time data pipelines and streaming applications. By using the Debezium connector to extract data from databases and stream it into Kafka, it is possible to integrate databases with Kafka and build real-time data pipelines. By using the Kafka connector to stream data into Elasticsearch, it is possible to build real-time search and analytics applications. The Kafka architecture is a powerful tool that enables organizations to build real-time data processing and analytics applications.

Second Implementation:

The second step in the implementation process involved setting up Kafka connectors to send data from Kafka to Elasticsearch. The objective was to create a pipeline that would allow for real-time data processing, analysis, and visualization in Kibana. To achieve this, the Kafka Connect Elasticsearch Sink Connector was used.

Kafka Connect is a component of Apache Kafka that allows for the integration of Kafka with external systems. Connectors are plugins that are responsible for moving data between Kafka and other systems. There are two types of connectors: source connectors and sink connectors. Source connectors move data from an external system into Kafka, while sink connectors move data from Kafka to an external system.

The Kafka Connect Elasticsearch Sink Connector is a sink connector that allows data to be moved from Kafka to Elasticsearch. It provides a reliable and fault-tolerant way to index Kafka data into Elasticsearch, ensuring that data is delivered to Elasticsearch even in the event of a network or Elasticsearch outage.

To set up the connector, the first step was to install and configure the Kafka Connect Elasticsearch Sink Connector. The connector can be installed either by downloading it from the Confluent Hub or by building it from source. Once the connector was installed, the configuration file had to be edited to specify the Elasticsearch cluster and index where the data was to be stored.

The configuration file for the Elasticsearch Sink Connector consists of several key properties, including the name of the connector, the Kafka topic to consume from, the Elasticsearch index to write to, and the connection details for the Elasticsearch cluster. Additionally, the connector can be configured to handle errors and failures, specify the number of tasks to run in parallel, and set other properties related to performance and scalability.

Once the configuration was set, the next step was to start the connector. This was done using the Kafka Connect REST API, which provides a web-based interface for managing connectors. The connector was started by sending a POST request to the Kafka Connect REST API, specifying the name of the connector and the configuration file.

With the Kafka Connect Elasticsearch Sink Connector up and running, data could be streamed from Kafka to Elasticsearch in real-time. The connector constantly monitors the Kafka topic for new messages, and as soon as a message is received, it is transformed and indexed into Elasticsearch. This allowed for real-time data processing and analysis, as well as real-time visualization in Kibana.

In conclusion, the second step in the implementation process involved setting up Kafka connectors to send data from Kafka to Elasticsearch. The Kafka Connect Elasticsearch Sink Connector was used to provide a reliable and fault-tolerant way to index Kafka data into Elasticsearch, allowing for real-time data processing, analysis, and visualization in Kibana. With this step complete, the Kafka architecture was fully set up and ready for use in the company's data pipeline.

Drawbacks and issues:
While working on the KSQLDB faced some issues in implementing the company's use cases with the KSQLDB so upon research I Found a new one which is Apache Flink

KSQLDB and Apache Flink are both powerful stream processing frameworks that allow users to process real-time data streams. However, each framework has its own set of advantages and disadvantages.

One of the main drawbacks of KSQLDB is its limited functionality compared to Apache Flink. KSQLDB is designed to be a simplified SQL-like language for stream processing, and therefore it is not as flexible as Flink in terms of complex data processing. Flink provides more advanced data processing capabilities, such as support for machine learning algorithms, graph processing, and complex event processing.

Another disadvantage of KSQLDB is its limited scalability. While KSQLDB can handle small to medium-sized workloads, it may struggle with large-scale data processing tasks. Apache Flink, on the other hand, was specifically designed to handle large-scale data processing, making it a better choice for organizations dealing with high volumes of real-time data.

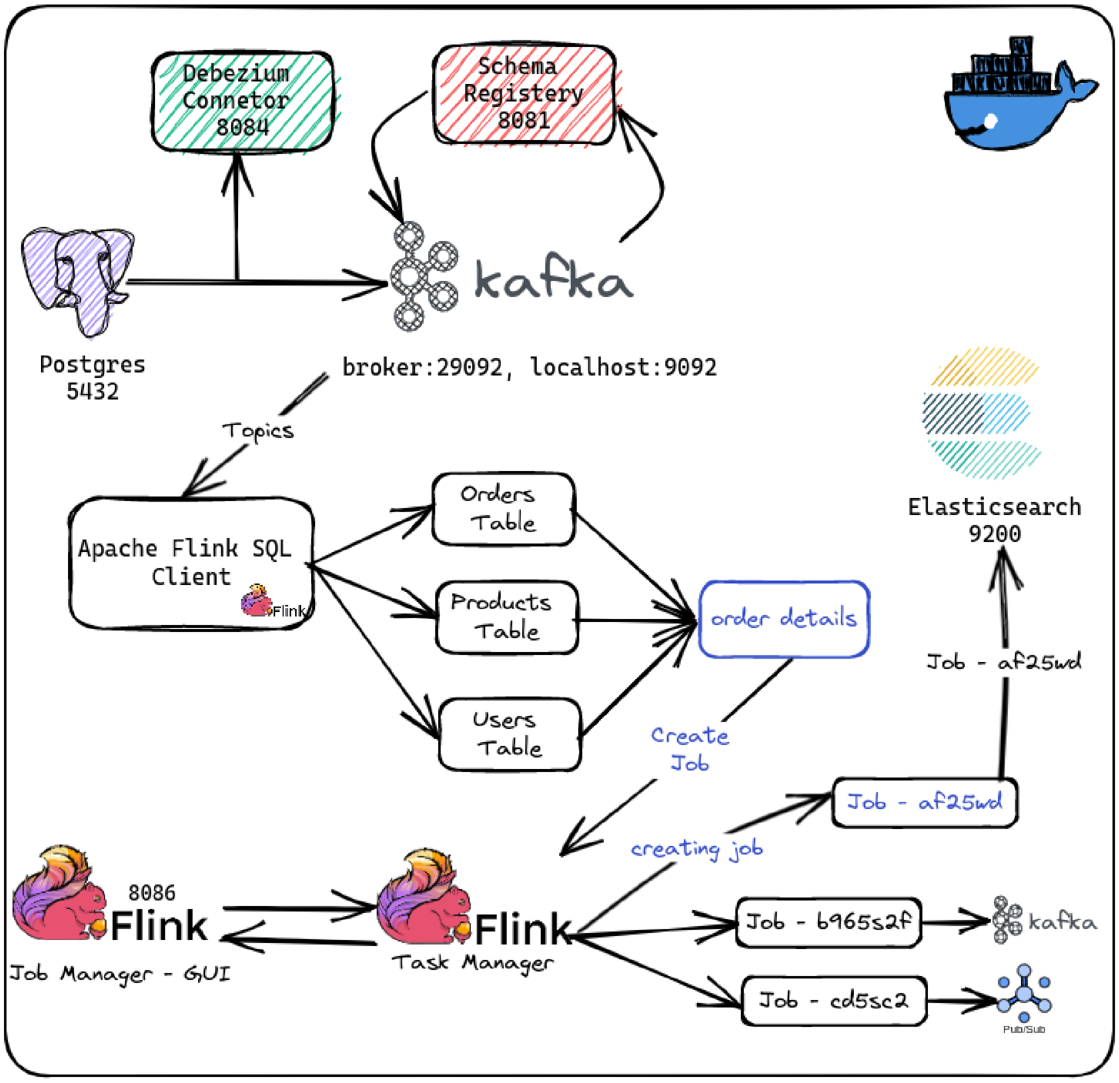
In addition, KSQLDB is tightly integrated with Kafka, which means that users who do not use Kafka may find it challenging to incorporate KSQLDB into their existing data processing pipelines. Apache Flink, on the other hand, can be integrated with a wide range of data sources and sink connectors, making it more versatile in terms of data integration.

Another drawback of KSQLDB is its relative immaturity compared to Apache Flink. While KSQLDB has been in development for several years, it is still a relatively new framework that may lack some of the features and stability of more established stream processing frameworks like Flink.

Finally, KSQLDB's reliance on SQL-like syntax can make it less accessible for users who are not familiar with SQL. While SQL is a widely-used language in the data processing industry, some users may find the syntax and structure of KSQLDB queries challenging to understand and write.

In summary, while KSQLDB has its advantages in terms of simplicity and tight integration with Kafka, it may not be the best choice for organizations that require advanced data processing capabilities, large-scale data processing, or integration with non-Kafka data sources. Apache Flink, on the other hand, provides more advanced data processing capabilities and greater flexibility, making it a better choice for organizations that require these features.

The Final Implementation:



Apache Flink is a distributed computing system that provides fast, scalable, and fault-tolerant processing of large data sets. In this data pipeline, Flink is used to process data received from Kafka, perform operations on it, and then send the processed data to ElasticSearch.

The first step in the pipeline is to use the Debezium connector to connect to the database and send data changes to Kafka topics. Debezium is a tool that provides change data capture (CDC) for databases, allowing you to capture and stream changes to your data in real-time. This ensures that any changes made to the database are immediately available for processing.

Once the data is in Kafka, Apache Flink processes it using a variety of operators. These operators include filtering, mapping, aggregating, and joining data streams. Flink allows for complex operations to be performed on the data with high throughput and low latency.

One of the key benefits of Apache Flink is its ability to handle streaming data. Traditional batch processing systems are limited by the fact that they process data in batches, meaning that data is processed only when a batch is completed. Streaming processing, on the other hand, allows for real-time processing of data as it is received.

Apache Flink's support for streaming data means that it can process data as it's received, which is crucial for applications that require real-time insights. For example, in an e-commerce application, it's important to be able to process data in real-time so that you can react to changes in customer behavior and adjust your strategy accordingly.

In addition to its support for streaming data, Apache Flink also provides fault-tolerance and scalability. Flink uses a distributed architecture, which means that it can handle large volumes of data and scale horizontally as needed. If one node in the cluster fails, Flink automatically redistributes the data and continues processing without any interruption.

Apache Flink also provides support for high-level APIs, which makes it easy to write complex data processing workflows without having to write low-level code. This makes it possible for developers to focus on the business logic of their applications rather than the intricacies of distributed computing.

After the data has been processed in Apache Flink, it is sent to ElasticSearch for indexing and storage. ElasticSearch is a distributed, full-text search engine that provides powerful search capabilities and fast data retrieval.

One of the benefits of using ElasticSearch is that it provides real-time search results. This means that as new data is added to the index, it's immediately available for search queries. This is important for applications that require real-time search results, such as e-commerce sites that need to display search results to customers in real-time.

In addition to its search capabilities, ElasticSearch also provides support for analytics and data visualization. This means that you can use ElasticSearch to perform complex data analysis and visualize the results using tools like Kibana.

Overall, the implementation of Apache Flink in this data pipeline provides fast, scalable, and fault-tolerant processing of large data sets. Its support for streaming data, fault-tolerance, and scalability make it an ideal choice for applications that require real-time processing and insights. Additionally, its support for high-level APIs and ease of use make it a good choice for developers who want to focus on the business logic of their applications. When combined with tools like Debezium and ElasticSearch, Apache Flink provides a powerful data processing and analytics platform that can help organizations gain valuable insights from their data.

Once the Apache Flink pipeline was developed, it was tested locally to ensure its correctness and performance. This was done by setting up a local Flink cluster using Docker and running the pipeline on it. After confirming that the pipeline was working as expected, it was deployed to the server for production use.

The deployment process involved setting up a Flink cluster on the server, configuring the necessary resources such as memory and CPU cores, and deploying the Flink job to the cluster. The Flink cluster was set up using the Flink distribution package, which was downloaded from the Apache Flink website.

One of the challenges faced during the deployment process was ensuring that the resources allocated to the Flink cluster were sufficient to handle the workload. This required a thorough understanding of the Flink job's resource requirements, which was gained through experimentation and benchmarking.

Once the Flink job was deployed to the server, it was monitored using Flink's built-in metrics and monitoring tools. These tools provided real-time insight into the job's performance and resource usage, allowing for early detection and mitigation of any issues that arose.

Overall, the implementation of the Apache Flink pipeline proved to be a success, as it provided a robust and scalable solution for processing and analyzing database data. It enabled the company to gain valuable insights into their data, which was used to improve their products and services.

However, it should be noted that the implementation of the pipeline was not without its challenges. The complexity of the Apache Flink framework meant that it required a high level of expertise to set up and configure, which may not be feasible for smaller organizations with limited technical resources. Additionally, the use of Flink may not be necessary for less complex data processing tasks, and simpler frameworks such as Kafka or Spark may suffice.

In conclusion, the implementation of the Apache Flink pipeline proved to be a valuable addition to the company's data processing infrastructure. Its ability to handle large volumes of data in real-time and perform complex analytical operations made it an ideal solution for the company's use case. However, the complexity of the framework and its resource requirements may make it unsuitable for some organizations.

Other Tasks:
During the internship, various tasks were performed which include writing Python scripts, Bash scripts, building Jenkins pipelines, Docker images, and issue resolutions in them, and log

processing. These tasks were performed in order to enhance the productivity of the team and to provide solutions to various problems.

Python scripting was done in order to automate the repetitive tasks and to reduce the manual efforts required for the same. The scripts were written to process the data, perform data analysis, and to automate the data extraction process. The scripts were also used to generate reports and to provide insights into the data.

Bash scripting was done to automate the tasks on the Linux operating system. Bash scripts were written to automate the deployment process, to create backups of the data, to install and configure the required software, and to perform various other tasks. The scripts were also used to schedule the tasks and to provide notifications for the same.

Jenkins pipelines were built in order to automate the continuous integration and continuous delivery process. The pipelines were built to automatically build, test, and deploy the software. The pipelines were also used to generate reports and to provide insights into the quality of the software.

Docker images were built to simplify the deployment process and to provide a consistent environment for the software. Docker images were built for the different services and applications required by the team. The Docker images were also used to provide a reproducible environment for testing and development.

Issues were resolved in the Python scripts, Bash scripts, Jenkins pipelines, and Docker images. The issues were identified and resolved in order to enhance the functionality and to provide solutions to the problems faced by the team. The issues were also resolved to improve the performance and to reduce the errors in the software.

Log processing was done to extract insights from the logs generated by the various services and applications used by the team. The logs were processed to identify the issues, to track the performance, and to provide insights into the usage of the services and applications. The logs were also processed to identify the patterns and trends in the data.

Overall, these tasks were performed in order to enhance the productivity and efficiency of the team. The tasks were performed with the aim of providing solutions to the problems faced by the team and to improve the quality of the software. These tasks helped in developing the skills and knowledge of the intern and contributed to the growth of the team.

Chapter - V Learning Outcomes

During the internship, I had the opportunity to work on various technologies, including the Elastic, Fluentd, Grafana (EFG) Stack and Kafka Architectures. These technologies are widely used in the industry and are critical for processing, analyzing and visualizing data. Through this experience, I gained several valuable learning outcomes.

One of the significant learning outcomes was gaining knowledge about the EFG stack. The EFG stack is widely used for monitoring and visualization of data in real-time. During the implementation, I learned how to use Elastic Search to store and search data, Fluentd to collect and forward logs, and Grafana to visualize and monitor the data. The implementation helped me understand the various components of the stack, how they work together, and their significance in the data processing pipeline. I also learned how to create custom dashboards and visualizations in Grafana, which was an important skill in data analysis.

EFG Stack Outcomes:

- Learned about the use cases and benefits of the EFG stack for log processing and analysis.
- Gained knowledge on the architecture and components of the EFG stack, including ElasticSearch, Fluentd, and Grafana.
- Learned how to configure and deploy the EFG stack to process logs and visualize data in Grafana.
- Learned about the importance of data formatting and how to customize Fluentd configuration for specific log formats.
- Discovered the potential drawbacks of the EFG stack, such as high resource usage and limited data processing capabilities.

Another important learning outcome was the implementation of the Kafka architecture. During the implementation, I learned how to use Kafka as a distributed messaging system to process and store data. I also learned how to use Kafka Connect to import data from different sources and how to use Kafka Streams to process data in real-time. Additionally, I learned how to use the Debezium connector to capture database changes and stream them to Kafka. Through this implementation, I gained a deeper understanding of how to use Kafka to process and store data and its significance in building distributed systems.

Kafka Architecture Learnings:

- Learned about the architecture and components of Kafka, including brokers, topics, and partitions.
- Gained knowledge on the use cases and benefits of Kafka, such as real-time data processing and message streaming.
- Learned how to set up a Kafka cluster and deploy Kafka connectors to integrate with other systems.
- Discovered the importance of data serialization and how to customize Kafka producers and consumers for specific data formats.
- Compared and contrasted Kafka with other data processing frameworks, such as Apache Flink.

Apart from these, I also learned several valuable skills in other tasks performed during the internship. I gained proficiency in Python scripting, which helped me automate several tasks and build efficient scripts. I also learned how to write Bash scripts to automate tasks in the Linux environment. The experience of building Jenkins pipelines helped me understand the importance of continuous integration and continuous deployment (CI/CD) in software development. The experience of building Docker images helped me understand containerization and its advantages in software development. Additionally, I gained valuable experience in issue resolution, which helped me improve my problem-solving skills.

Other Tasks Outcomes:

- Learned how to write Python and Bash scripts to automate common tasks and streamline workflows.
- Gained knowledge on building Jenkins pipelines for continuous integration and deployment.
- Learned how to create Docker images and deploy containerized applications.
- Developed skills in issue resolution and troubleshooting, particularly with log processing and system configurations.

Overall, the internship provided me with valuable experience in critical technologies that are widely used in the industry. Through the implementation of the EFG stack and Kafka architecture, I gained a deeper understanding of data processing and visualization. The experience of working on different tasks helped me develop proficiency in several important skills like scripting, CI/CD, containerization, and problem-solving. These learning outcomes will be valuable to me as I move forward in my career in software development and data engineering.

During my internship, I had the opportunity to work on various tasks that involved implementing new technologies, building scripts and pipelines, and resolving issues in existing systems. Through this experience, I have gained valuable knowledge and skills in different areas of software development and have become more confident in my abilities as a developer. In this report, I will discuss my overall learning outcomes from these tasks and outline future opportunities for me to continue building on this experience.

One of the key learning outcomes from my internship was gaining proficiency in implementing and working with different technologies. Specifically, I gained experience in implementing the EFG (Elasticsearch, Fluentd, Grafana) stack, which involved setting up data processing pipelines for large volumes of data. I learned how to use Elasticsearch as a distributed search and analytics engine, Fluentd as a log collector and processor, and Grafana as a visualization tool. This experience gave me a better understanding of how these technologies work together and how they can be used to build scalable data processing pipelines.

Another important learning outcome was gaining experience in building scripts and pipelines. I worked on building Python scripts for data processing, Bash scripts for automating tasks, and Jenkins pipelines for continuous integration and deployment. Through these tasks, I learned how to write efficient and reusable code, how to automate repetitive tasks, and how to build and deploy software in a more efficient manner.

In addition to building new systems, I also gained experience in resolving issues in existing systems. I worked on resolving issues in Docker containers, Kubernetes clusters, and other systems. This experience gave me a better understanding of how to diagnose and fix issues in complex software systems, as well as how to work collaboratively with other developers to resolve issues.

Overall, my internship provided me with a wealth of experience and knowledge in various areas of software development. I feel more confident in my abilities as a developer and have a better understanding of how different technologies and systems work together.

Looking to the future, there are several opportunities for me to continue building on this experience. One area where I would like to gain more experience is in building more complex software systems, such as distributed systems or microservices. I would also like to continue gaining experience in cloud-based technologies, such as AWS or Azure, as these are becoming increasingly important in modern software development. Additionally, I would like to explore opportunities to work with different programming languages and frameworks, as this will help me to develop a more versatile skillset.

In terms of professional development, I plan to continue building my knowledge and skills by attending conferences, networking with other developers, and contributing to open-source projects. These activities will not only help me to stay up-to-date with the latest trends and technologies in software development, but also give me opportunities to connect with other developers and learn from their experiences.

In conclusion, my internship provided me with a wealth of experience and knowledge in different areas of software development. Through my work with different technologies, building scripts and pipelines, and resolving issues in existing systems, I gained valuable insights into the complexities and challenges of modern software development. Looking to the future, I am excited about the opportunities to continue building on this experience and developing my skills as a software developer.

References

1. Elastic: https://www.elastic.co/
2. Fluentd: https://www.fluentd.org/
3. Grafana: https://grafana.com/
4. Kafka: https://kafka.apache.org/
5. Debezium: https://debezium.io/
6. KSQLDB: https://ksqldb.io/
7. Apache Flink: https://flink.apache.org/
8. Docker: https://www.docker.com/
9. Jenkins: https://www.jenkins.io/
10. Python: https://www.python.org/
11. Bash: https://www.gnu.org/software/bash/
12. PostgreSQL: https://www.postgresql.org/
13. MySQL: https://www.mysql.com/
14. Oracle: https://www.oracle.com/
15. Redis: https://redis.io/
16. MongoDB: https://www.mongodb.com/
17. Amazon Web Services (AWS): https://aws.amazon.com/
18. Microsoft Azure: https://azure.microsoft.com/
19. Google Cloud Platform (GCP): https://cloud.google.com/
20. Logstash: https://www.elastic.co/logstash
21. Nyalazone: http://nyalazone.com/