[1]Dr. R. Christy Pushpaleela,

Dr. A. Jerline Amutha,

Ms. Mary Ivy Deepa I S,

Dr. K. Viswanathan

Accelerating API Modernization - Automated SOAP to REST Conversion and Container Deployment with Kubernetes in AWS

Journal of Electrical Systems

Abstract: - The purpose of this paper is to analyse the implementation of an Auto boot Framework that automates the SOAP (Simple Object Access Protocol) to REST API conversion process. Furthermore, deploying the resultant microservices in a Kubernetes container.The framework is designed to generate REST (Representational State Transfer) services from existing SOAP implementations as microservices in Spring Boot using pre-defined configurations. Whereas, Microservices provide a solid foundation for the transition and evolution of microservice architectures and its solutions. The XML request can be sourced from any system whereas the resulting artifact file will be automatically deployed in the AWS Kubernetes container.

Keywords: Spring Boot, SOAP, REST AP, MicroService and Kubernetes.

I. INTRODUCTION

The main purpose of this paper is to analyse the automation of SOAP (Simple Object Access Protocol) service to REST (Representational State Transfer) service and container deployment in AWS cloud using Spring Boot and Kubernetes.

Spring Boot is a lightweight open-source framework that simplifies the process of building and deploying applications. It has its own transaction management and services which makesit an efficient tool for the containerized applicationdeployment in an AWS cluster. This paper discusses the features of Spring Boot along with AOP (Aspect- Oriented Programming), MVC framework and AIX. Spring Boot is a bootable framework that does not require XML configuration or an application server to deploy the application. It has an in-built tomcat and Jetty server that helps to reduce development and testing efforts.

Furthermore, the paper highlights thesignificance of microservices as the next natural evolution of SOAP services. Microservicesprovide a solid foundation for the transition and evolution of microservice architectures and its solutions. In addition, they are non-monolithic applications which are independent and changeable services. Each service has its own business logic that makes it an independent decentralized application. The default communication method for microservices is RESTAPI calls though they support SOAP.

1.1 OBJECTIVES

SOAP (Simple Object Access Protocol) is a heavyweight protocol that is widely used in web services. However, our implementation aims to leverage the benefits of REST (Representational State Transfer), an architectural style transfer as well as a lightweight protocol to simplify the web service development. On the other hand, our implementation also emphases on container deployment using Kubernetes. Kubernetes is ideal for non-monolithic and microservice applications which is a lightweight, master-slave architecture. Itcan also manage clusters of containers.

One of the primary reasons for using Kubernetes is to deploy Docker images in multiple machines though Docker is considered to be suitable for single-machine deployments. Eventually, Kubernetes can be used as a microservice platform that simplifies the deployment and management of containers thereby helping in optimizing the usage of resources.

[1]*Associate Professor, Department of Computer Science and Technology, Women's Christian College,Chennai, India. E-mail: christypushpaleela@wcc.edu.in*

[2]*Associate Professor, Department of Computer Science and Technology, Women's Christian College Chennai, IndiaE-mail: jerlineamutha@wcc.edu.in*

[3]*Associate Professor, Department of Computer Science and Technology, Women's Christian College Chennai, India*

E-mail: maryivydeepa@wcc.edu.in

[4]*AWS Lead Solution Architect, Cognizant Technology Solutions, Chennai, India. E-mail: viswanathan_km@yahoo.co.in*

II. LITERATURE REVIEW

The literature review has played a vitalrole in implementing the proposed system. These research papers provided deep insight onthe technologies and frameworks used in the proposed model. The best practices for building microservices, container deployment using Kubernetes and RESTful web servicesare also the key inputs from the literature study. As mentioned by N. Kishore[2], Cloudis an emerging concept which provides services such as Infrastructure, middleware, applications and data security. Every domain has its own flavor of Cloud. Technologies and techniques for Cloud have come a long way and is still evolving. Varun Kumar et al. (2011) highlighted the recent increases in the value of key data from a wide collection of medical data sets which has increased the need for the relevant data mining tools.

Timothy Kinsman et. al [2022] has examined various methodologies that can be employed to create a successful CI/CD pipeline. Artur Cepuc, Robert Botez[2021] used methodologies that encompass a range of tools and address the challenges faced during theimplementation of continuous improvement processes in software development using DevOps.

Aayush Agarwal[2015] described the performance of containers running on hosted services such as Microsoft Azure Kubernetes Services (AKS), Amazon Elastic Container Service for Kubernetes (EKS) and Google Kubernetes Engine.Tingting Chen, Yang Zhang[2021] have done the experimental analysis of GitHub projects. This paper provides valuable insights for experts seeking to deepen their understanding of GitHub Actions[8].

III. METHODOLOGY

3.1 AUTOMATION- SOAP TO RESTCONVERSION SOLUTION

This research work focuses on the framework that auto-generates REST Services for the existing SOAP implementations. This framework generatesmicroservices in Spring Boot using pre-defined configurations. The framework is designed to receive XML requests from any of the source systems followed by converting the XML file to POJO using a wrapper class with the help of the Spring Boot microservice. Spring Boot then provides the workable artifact files for deployment, which are automatically deployed in the application server. Fig 1. Shows the SOAP to REST conversion using Spring Boot. Some of the key features of our framework are:

1. Automatic generation of REST Services for existing SOAP implementations.
2. Microservices are built using Spring Boot and pre-defined configurations.
3. XML requests are converted to POJO usinga wrapper class.
4. Automatic deployment of artifact files in the application server.

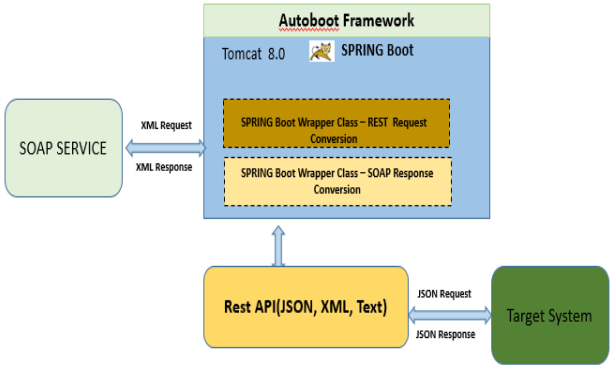

Fig1: SOAP to REST conversion using SpringBoot

The proposed framework aims to simplify the process of generating REST Services for existing SOAP implementations, providing an efficient solution for web service development. This framework would enhance the efficiency as wellas the effectiveness of web service development and deployment process.

3.2. MICRO SERVICE DESIGN PRINCIPLE

Microservices are functionally cohesive, loosely-coupled and independent atomic units of deployment and execution services which interact through a set of well-defined interfaces. Microservices are an approach to the design, architecture and development of an application composed of small and discreet services. The significant characteristics of microservices are,

- Highly cohesive-single responsibility as wellas separate read and write operations
- Loosely coupled to other microservices andservice consumers
- Independently developed via separate CI\CDpipelines
- Data Isolation and Independently deployable
- Independently executable and scalable supporting an on-demand scaling model

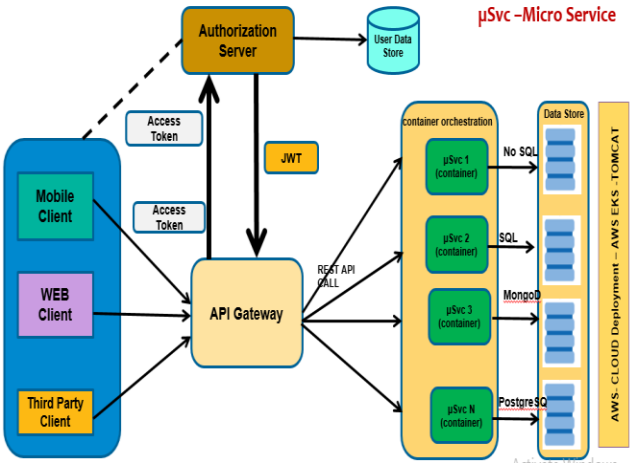

Fig 2. Microservices Architecture

Microservices architecture has become increasingly popular in recent years due to its advantages such as fast delivery, high availability and resilience. Fig 2 explains the microservice architecture. The important features of the microservices include the following,

Easy and Automatic Deployment: Deployment becomes much simpler and faster with microservices. The independent services can be deployed individually, allowing for more flexibility in updating and maintaining the system. Tools such as Jenkins, TeamCity, and Bamboo can automate the deployment process.

Easy Development and Testing: The smaller and more focused nature of microservices makes development and testing more straightforward. Developers can work on individual services, reducing complexity and making it easier to ensure quality.

Fast Delivery: Microservices enable faster deliveryof new features and updates, as changes can bemade and deployed independently without affectingthe entire system.

High Availability and Resilience: Microservices are designed to be fault-tolerant and readily available. If any one of the services goes down then the other services can continue to functionwhich ensures business continuity.

Lightweight and Scalable: Microservices are typically lightweight and can be scaled independently which makes them an ideal choice for applications that need to handle fluctuating loads.

Smaller Development Team: Microservices are utilized in smaller and larger development teams to reduce overhead and increase efficiency.

More Customer Interaction: Microservices make it easier to integrate customer feedback thereby making changes in the applications that lead to an improved overall user experience.

Increased Agility and Reduced Complexity: The modular nature of microservices reduces complexity furthermore increases agility thereby allowing organizations to respond quickly to changing market needs.

IV. KUBERNETES ARCHITECTURE

Kubernetes (K8s) is "When deploy it docker image in multiple machines go for K8s" and single machine go for Docker. In addition it is a container orchestration as well as a open source tool. It is alsocalled as K8s clustering service which is mainly used for non-monolithic application and MS applications. Moreover it is lightweight Master Slave architecture. Kubernetes is a popular open- source platform composed of several key components that includes,

Kube-API server: This component is responsiblefor managing the Kubernetes API which allows users to interact with the cluster and manage its resources.
Kube-Service: This component manages the services that are deployed in the cluster including load balancing and service discovery.
Kube-Scheduler: The Scheduler is responsible for scheduling the Pods onto the nodes in the cluster which is based on the available resources and otherfactors.
Kube-Controller: This component includes several controllers that manage different aspects of the cluster such as Node controller, Replication controller, and Endpoint controller.
Node controller: Furthermore, the Node controller manages the nodes in the cluster by detecting the ineffective nodes and replacing them with new nodes.
Replication controller: This component ensures thata specified number of Pod replicas are running in the cluster at all times.
Endpoint controller: The Endpoint controller manages the endpoints of the services in the cluster which are the network addresses that are used to access the services

Master Node –Activity Implementation for monitoring and managing PODs in the Kubernetes cluster has been established. A systematicapproach guarantees continuous monitoring of POD states and the automatic replacement of failedinstances. The system effectively tracks active,failed, and operational states of PODs. In case ofany POD failure an automatic process on the master node initiates the creation of a new POD while decommissioning the failed one. The system employs a formula to ascertain POD status by calculating the current and target POD states. This ensures the seamless operation of the Kubernetes cluster by minimizing downtime risks associated with POD failures.
Node Affinity -This method ensures that pods arehosted on specific nodes based on the node's labels.
Pod Affinity –This method ensures that related podsare scheduled on the same node by using the same label or label selector

Pod and Node:

A group of containers is called a POD whereas this group of containers run on the same host. This ensures that all containers in the POD are co-located on the same node

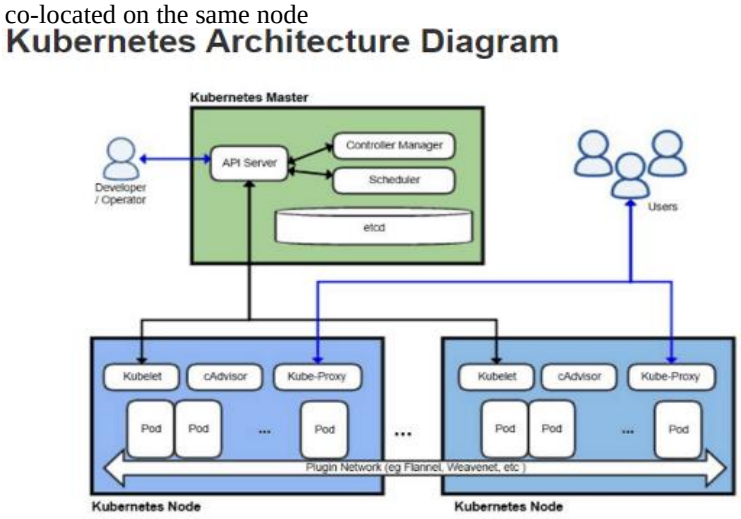

Fig3. Kubernetes Architecture View

V. KUBERNETES CONATINERDEPLOYMENT

.Kubernetes helps organizations build anddeploy applications more efficiently and effectively.

Here are some of the advantages of using Kubernetes.

- **Automatic Rollback:** Kubernetes provides automatic rollback functionality that allows users to

quickly and easily revert to a previousversion of their application if something goes wrong.

- **Auto Scaling:** Kubernetes enables auto scaling of containerized applications, ensuringthat resources are allocated efficiently and automatically scaled up or down based on demand.
- **Horizontal Scaling:** Kubernetes allows for horizontal scaling of applications, enabling organizations to handle increases in traffic andworkload without any downtime.
- **Automated Containerized Environment:** Kubernetes automates the deployment and management of containerized environments,streamlining the development and deploymentprocess.
- **Containerization Benefits:** Containerization is the future of application deployment, providing benefits such as portability, efficiency, and isolation

This approach can help to streamline the development and deployment processes that ensure efficient and scalable services that meet the needs of modern software development. The following is an overview of the approach:

i. Containerize the application: The first step indeploying the application on Kubernetes is to containerize it using a tool like Docker. This involves packaging the application and its dependencies into a container single image.

ii. Define your application as a set of microservices:Breaking down the application into smaller, independent services, each with its own API and functionality.

iii. Define Kubernetes manifests: Kubernetes usesYAML or JSON files called manifests to define the desired state of the application. These manifests define the containers, services, and otherresources needed to run the microservices on Kubernetes.

iv. Deploy your microservices: Use the kubectl command-line tool to deploy the microservices to Kubernetes. Kubernetes will automatically create the necessary resources to run your microservices, including PODs, services, and replica sets.

5. Monitor and manage the microservices: Kubernetes provides a range of tools formonitoring and managing The microservices,including the Kubernetes Dashboard, kubectl commands.

6. Scale your microservices: Kubernetes enables auto scaling and horizontal scaling of microservices, allows the application to handle increases in traffic and workload without any downtime.

7. Update your microservices: Using Kubernetes to roll out updates to the microservices ensures that the application is always up-to-date and running smoothly.

This approach can help to build and deploy microservice applications in a structured and efficient manner, and ensure that the services are scalable and reliable.

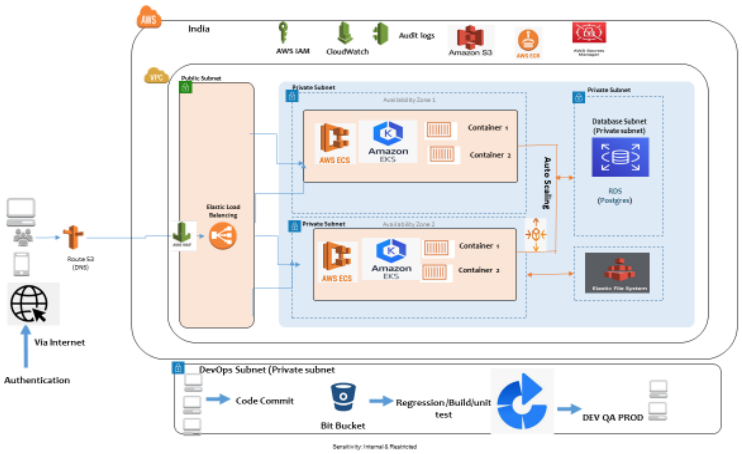

Fig 3. Kubernetes Deployment Process

VI. CONCLUSION

This research analysed the process of converting a SOAP service to a REST API full microservice. The deploymenf of the microservice application in AWS Kubernetes container has also been analysed.

This research work makes a significant contribution to the field of microservices and container deployment. Specifically it provides a detailed method for converting SOAP services to REST APIs and deploying microservices in AWS Kubernetes container. This approach ensures efficient and scalable services that meet the needs of modern software development.

One of the key contributions of our analysis is its focus on AWS cloud and Kubernetesdeployment.

The findings provide practical insights for developers and organizations who are seeking to improve the efficiency and scalability oftheir services. This approach can help organizations save time and resources, whileimproving the overall quality of their services.

REFERENCES

[1] A. Cepuc, -A. Ivanciu and V. Dobrota, "Implementation of a Continuous Integration and Deployment Pipeline in Amazon Web Services Using Jenkins, and K8s," 2020 19th RoEduNet Conference: (RoEduNet), 2020.

[2] R. Dangayach, “AWS Fargate – Procss To Deploy With Containerized Application”.

[3] D. Jones, “Containers vs. VM : https://www.netapp.com/blog/containers-vs-vms/.

[4] J. Galvis, S. and James “What, Why, How: Run Serverless Kubernetes Pods Using Amazon EKS,” IOD, 09-Mar-2020.

[5] Dobrota, "Deploying a Dockerized Application With Kubernetes on Cloud Platform," 2020), 2020, pp. 471-476

[6] Andrew D Multi-agent Cluster Scheduling - Scalability & Flexibility, pp. 16-18, 2012.

[7] Christudas, "Introducing MS", Practical MS Architectural Patterns, pp. 21-34, 2019..

[8] Mosteller, "Understanding the Birthday Problem", Springer Series in Statistics Selected Papers of Mosteller, pp. 349-353..

[9] Hamad, "An Overview of Hadoop Scheduler Algorithms", Modern Applied Science, vol. 12, no.8, pp. 69, 2018.

[10] K. Gopalan, „„Fast rovisioning through scatter- gather,“‘ in Proc. 7th IEEE Int. Conf. Computer. (CLOUD), Anchorage, AK, USA, Jun./Jul. 2014,pp. 376–383.

[11] B. Egger, „„Efficient live migration of virtual machines using shared storage,“‘ in Proc. 9th SIGPLAN/SIGOPS Int. Conf. Virtual Execution Environ., Houston, TX, USA,2013, pp. 41–50.