DevOps

A Project-II Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE & ENGINEERING

BY

Samriddhi Kakkar

EN18CS301225

Under the Guidance of

Mr. Sachin Solanki

Department of Computer Science & Engineering

Faculty of Engineering

MEDI-CAPS UNIVERSITY, INDORE- 453331

May 2022

Report Approval

The project work **"DevOps"** is hereby approved as a creditable study of an engineering/computer application subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approve any statement made, opinion expressed, or conclusion drawn therein; but approve the "Project Report" only for the purpose for which it has been submitted.

Internal Examiner

Name: Mr. Sachin Solanki

Designation: Assistant Professor

Affiliation

External Examiner

Name:

Designation:

Affiliation

Declaration

I/We hereby declare that the project entitled **"DevOps"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in 'Computer Science' completed under the supervision of **Mr. Sachin Solanki, Assistant Professor, Computer Science and Engineering,** Faculty of Engineering, Medi-Caps University Indore is an authentic work.

Further, I/we declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Samriddhi Kakkar

(EN18CS301225)

Signature and name of the student(s) with date

Certificate

I/We, **Mr. Sachin Solanki** certify that the project entitled **"DevOps"** submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Samriddhi Kakkar** is the record carried out by her/them under my/our guidance and that the work has not formed the basis of award of any other degree elsewhere.

Mr. Sachin Solanki	Mr. Jadeja Rajpal
Computer Science & Engineering	DevOps Engineer
Medi-Caps University, Indore	Unthinkable Solutions LLP

Dr. Pramod S. Nair

Head of the Department

Computer Science & Engineering

Medi-Caps University, Indore

Offer Letter of the Project work-II/Internship

unthinkable

Dated: August 19, 2021

Subject: Appointment-cum-Offer Letter

Dear Samriddhi Kakkar,

We are pleased to offer you the position of **Intern** in our Company.

Your engagement shall be subject to the terms expressed herein and the on-job training letter, and uniform terms and conditions of the Company, which are also accessible from Company's office server.

Subsequent to your completion of Internship Period you will be designated as Junior Associate – IT. Acceptance of this letter is by expression acceptance of the service agreement of Minimum contractual period. Minimum Contractual Term" means the Internship Period i.e. (initial minimum 6(Six) Months which will be extended till the time you have not appeared for your final semester exams) and 1 (One) year thereafter. This period shall exclude any leaves availed by you apart from the stipulated earned leaves as per company's policy.

The date for commencement of your joining is January 2022. Your reporting officer is **Ms. Himani Gautama (Manager - Talent Management)**.

Upon joining, you will receive the detailed confirmation letter and you will be required to signed copy of the enclosed Joining Report along with documents requested.

Stipend and Salary Structure:

- Stipend Structure- With effect from your joining date, you will be entitled for the monthly stipend of INR 20,000 during internship.
- Salary Structure- Post completion of internship, below will be CTC breakup **for one year.**

	Post completion of internship	
	0-06 Months	**7-12 Months**
Particulars	**Amount (INR)**	**Amount (INR)**
Basic Salary	15100	15200
HRA	7550	7600
Travelling Allowance	1600	1600
Special Allowance	6517	11582
Gratuity	726	731
Health Insurance	277	277
Total Salary	**31770**	**36990**
Performance Bonus*	3530	4110
Monthly CTC	**35300**	**41100**
Expected In Hand Salary (pre-TDS)	34272	40067
Yearly CTC	**458400**	
Yearly Incentive**	**41600 (After 01 Year in permanent employment)**	
Total Annual CTC	**500000**	

Performance Bonus*& Yearly Incentive:** As per prevailing policy of company

Wishing you a long and fruitful stay,

For Unthinkable Solutions LLP

_____

Authorized Signatory

Completion certificate/Letter

unthinkable

Date: May 05, 2022

To Whomsoever It May Concern

This is to certify that Samriddhi Kakkar D/o Sumit Kakkar, R/o C-3A/146B, Janakpuri New Delhi BTECH student of Medi-Caps University, is doing internship with Unthinkable Solutions LLP from **14th Feb 2022** under DevOps project.

Till now, her overall performance is satisfactory. We found that she is sincere, hardworking, and result oriented.

For Unthinkable Solutions LLP

UNTHINKABLE SOLUTIONS LLP

Authorized Signatory

Registered Office :
Applane Solutions LLP
6th Floor, Metropolis Hissar Mall,
Opposite Vidyut Sadan,
Delhi Road, Hissar-125005

Corporate Office :
9th Floor, Tower B-1,
DLF Silokhera SEZ,
Sector-30, Gurugram,
Haryana-122001, India

Tel.: +91-124-6817000
info@unthinkable.co
www.unthinkable.co
LPIN : AAO-4243

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal,** who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Dilip K Patnaik,** Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. I also thank **Prof. (Dr.) D K Panda,** Pro Vice Chancellor, **Dr. Suresh Jain,** Dean Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. I would also like to thank my Head of the Department **Dr. Pramod S. Nair** for his continuous encouragement for betterment of the project.

I express my heartfelt gratitude to my **External Guide, Mr. Jadeja Rajpal**, Junior Associate IT, Unthinkable Solutions LLP as well as my Internal Guide, **Mr. Sachin Solanki,** Assistant Professor, Department of Computer Science, MU, without whose continuous help and support, this project would ever have reached the completion.

I would also like to thank my team at Unthinkable Mr. Anjan Kumar Jha, Mr. Ayush Sethi, and Mr. Deepanshu Jain who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report. Without their support this report would not have been possible.

Samriddhi Kakkar
(EN18CS301225)
B.Tech. IV Year
Department of Computer Science & Engineering
Faculty of Engineering
Medi-Caps University, Indore

Abstract

The problem statement of this project was:

Microservice infrastructure setup for Demo, Dev, QA, Staging and Production environment with monitoring enabled and logging tools setup.

The requirements of this project project are as mentioned below:

1. Client wanted to migrate an application running on monolithic architecture on an ec2 machine to cloud with microservices architecture with high availability, scalability.
2. Setup secure AWS architecture for the EKS environment. setup roles, user for bastion host.
3. Infrastructure setup on AWS using EC2, IAM - Roles, Groups, Permissions, Policies, MFA, S3, ECS, ECR, CloudFront, CloudWatch, Load Balancer.
4. Setup & configuring jenkins for CI/CD using dockerfile.
5. Implementation of logging tool with fluentd and sent logs to elasticsearch.
6. Implementation of continuous monitoring with prometheus/grafana.

Table of Contents

List of Figures

List of Abbreviations

S. No	Abbreviation	Full Form
1.	AWS	Amazon Web Services
2.	EC2	Elastic Compute Cloud
3.	VPC	Virtual Private Cloud
4.	EKS	Elastic Container Service for Kubernetes
5.	S3	Amazon Simple Storage Service
6.	K8s	Kubernetes

Chapter-1

Introduction

1.1 Introduction

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. A software development life cycle (SDLC) model is a conceptual framework describing all activities in a software development project from planning to maintenance. This process is associated with several models, each including a variety of tasks and activities.

In traditional models like in the waterfall model there is a problem of a one-way stream of work. Due to which if there is any mistake the whole process repeats and there is no interaction with customers. Now, this is solved in agile by splitting the whole development plan into several iterations for a better level of production efficiency. The agile model also includes customer interaction with the company to rectify the mistakes. But there is another problem faced in Agile too.

Here, the problem arises when the development team continuously changes the code for better performance and sends the code to the operations team for testing. But there may be a delay in the operations team feedback in situations like if the developers sent code for review at night but due to the unavailability of the operations team, there will be a delay in the project feedback.

So, DevOps is the solution to this problem. DevOps is a practice or a methodology in which the development team and operations team work together by including automation at the initial stages. So they can work on rapidly changing systems, fix bugs, and help to deliver a good quality of software in time.

In our project, we've provided the client an environment from scratch. Firstly, we have setup the architecture for development environment including the EKS setup of backend api's, frontend on cloudfront, monitoring tools with prometheus/grafana, logging tools with fluentd, setup script for rotation of logs, securing their credential with secret manager and we are managing all deployment from CI/CD. We built a secure, scalable, and available infrastructure for them. Upon successful setup of the development environment, we have set up demo env, QA env, UAT env, prod env, and we are still optimizing the infrastructure and making it more secure.

1.2 OBJECTIVES

The major goals of this project are:

1. To provide a highly scalable and available application.
2. To provide a secure cloud based setup.
3. Automation of environment, alerts, and monitoring.

1.3 ABOUT THE ORGANIZATION

Unthinkable is a custom software development company specializing in designing, developing and scaling, beautiful products with high-speed efficiency. We can turn your ideas into innovative software applications faster and drive rapid business growth. Our proactive, professional, and cost-effective software development service delivery has included us amongst the best in the business.

They believe in taking a radically different approach to software development. By solving problems at the technology level, we eliminate the misuse of scarce engineering talent to solve the same problem repeatedly across the solution, ultimately freeing them to deliver more value to the customers.

Unthinkable houses a team of 200+ developers catering to all the facets of software development. Be it Startups or Enterprises, SMEs or Governments, Product Companies or Digital Agencies, we cater to all of their diverse technology requirements. They build software solutions abiding by the traditional methodologies and develop products using decoupled software components to reduce redundant codes.

Their domain knowledge, technology expertise, and the use of proven methodologies have yielded high-quality solutions that add value to your business. This has also allowed them to serve over 100 customers with a faster time to market. Their experience puts us in a position where they can take an unconventional approach to software development. This idea also led to the birth of the "Unthinkable".

Chapter-2

System Requirement Analysis

2. Implementation Technology

2.1 AWS:

Amazon Web Services (AWS) is a cloud service from Amazon, which provides services in the form of building blocks, these building blocks can be used to create and deploy any type of application in the cloud.

These services or building blocks are designed to work with each other, and result in applications that are sophisticated and highly scalable.

Fig 1. Services provided by AWS

2.1.1 EC2 Instance: Amazon Elastic Compute Cloud, EC2 is a web service from Amazon that provides resizable compute services in the cloud. They are re-sizable because you can quickly scale up or scale down the number of server instances you are using if your computing requirements change.

Fig 2. EC2 Logo

An instance is a virtual server for running applications on Amazon's EC2. It can also be understood as a tiny part of a larger computer, a tiny part which has its own Hard drive, network connection, OS etc. But it is actually all virtual. You can have multiple "tiny" computers on a single physical machine, and all these tiny machines are called Instances.

2.1.2 **Virtual Private Network (VPC):** The backbone for the security of this architecture is VPC (Virtual Private Cloud). VPC is basically a private cloud in the AWS environment that helps you to use all the services by AWS in your defined private space. You have control over the virtual network and you can also restrict the incoming traffic using security groups.

Overall, VPC helps you to secure your environment and give you a complete authority of incoming traffic. There are two types of VPCs, Default VPC that is by default created by Amazon and Non-Default VPC that is created by you to suffice your security needs.

Fig 3. VPC Logo

2.1.3 **Subnets:** Subnets are like breaking a large network into sub-networks. Maintaining a smaller network is easy as compared to maintaining a large network.

Take the example of an organization. There are different teams such as Finance, Support, Operations, Technology, HR, Sales & Marketing. The data accessible to the Technology team cannot be given to the Sales & Marketing team, the data for the HR team cannot be given to the Operations team and vice-versa. Here, you

create sub-networks such that accessing and maintaining the network becomes easier.

2.1.4 **Amazon Elastic Container Service for Kubernetes (EKS):** Amazon Elastic Container Service for Kubernetes (Amazon EKS) is a managed service that makes it easy for users to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane. Since Amazon EKS is a managed service it handles tasks such as provisioning, upgrades, and patching.

Definition aside, let's take a look at the benefits of Amazon EKS:

- ➔ Amazon EKS runs the Kubernetes management infrastructure across multiple AWS Availability Zones, thereby freeing users from maintaining Kubernetes control planes.
- ➔ Infrastructure running on Amazon EKS is secure by default by setting up a secure and encrypted communication channel between worker nodes & Kubernetes endpoint.
- ➔ AWS actively works with the Kubernetes community and makes contributions to the Kubernetes code base.
- ➔ Applications managed by Amazon EKS are fully compatible with applications managed by any standard Kubernetes environment.

Fig 4. EKS Logo

2.1.5 **S3 Bucket:** Amazon Simple Storage Service (S3) is a storage for the internet. It is designed for large-capacity, low-cost storage provision across multiple geographical regions. Amazon S3 provides developers and IT teams with Secure, Durable and Highly Scalable object storage.

S3 is Secure because AWS provides:

- Encryption to the data that you store. It can happen in two ways:
  - ➔ Client Side Encryption
  - ➔ Server Side Encryption
- Multiple copies are maintained to enable regeneration of data in case of data corruption

- Versioning, wherein each edit is archived for a potential retrieval.

S3 is Durable because:

- It regularly verifies the integrity of data stored using checksums e.g. if S3 detects there is any corruption in data, it is immediately repaired with the help of replicated data.
- Even while storing or retrieving data, it checks incoming network traffic for any corrupted data packets.
- S3 is Highly Scalable, since it automatically scales your storage according to your requirement and you only pay for the storage you use.

Fig 5. S3 Logo

2.1.6 Load Balancer: Load Balancing refers to efficiently distributing incoming network traffic across a group of backend servers in a manner that increases speed and performance.

The Elastic Load Balancer is similar to all the other load balancers. It basically serves the purpose of a single point of contact for the client. To make it simple, the client sends a request to the load balancer and it then distributes them to the targets like EC2, lambda function, etc in different Availability Zones.

There are three types of Elastic Load Balancers available. You can use the one best suited for your work:

➔ Classic Load Balancer: This is the most basic form of the load balancer, that was used initially for classic EC2 instances. It is operated on connection level as well as request level. The main drawback of this type of load balancer is, it does not support features like host-based routing or path-based routing. Once the load balancer is configured, it balances load across the servers regardless of what is present on the server which might end up reducing the efficiency and performance in certain situations.

➔ Network Load Balancer: Network Load Balancer works at layer 4 of the OSI Model(connection level). It is mainly used for load balancing TCP traffic. This Load Balancer can handle millions of traffic and is best suited for maintaining low latencies. It's capable of handling uncertain traffic while maintaining a static IP address.

➔ Application Load Balancer: This type of Load Balancer is used when there is HTTP and HTTPS traffic routing. I'm sure you guys are aware of the OSI Model. This load balancer works at the Application layer of the OSI Model. It provides advanced routing features such as host-based and path-based routing. It also works well with containers and microservices. I

Fig 6. Load Balancer Logo

2.1.7 CloudFront: Amazon CloudFront speeds up distribution of your static and dynamic web content, such as .html, .css, .php, image, and media files. When users request your content, CloudFront delivers it through a worldwide network of edge locations that provide low latency and high performance. It provides high security with the 'Content Privacy' feature. It facilitates GEO targeting service for content delivery to specific end-users and uses HTTP or HTTPS protocols for quick delivery of content. It is a popular AWS service as it is less expensive, and charges for the data transfer only.

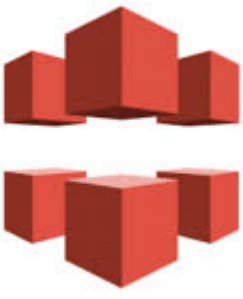

Fig 7. CloudFront Logo

2.2 Docker:

Before understanding docker, one key term that needs to be understood is containers.

➔ Containers: Containerization is a type of Virtualization which brings virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system.

Following are the reasons to use containers:

- Containers have no guest OS and use the host's operating system. So, they share relevant libraries & resources as and when needed.
- Processing and execution of applications are very fast since application specific binaries and libraries of containers run on the host kernel.
- Booting up a container takes only a fraction of a second, and also containers are lightweight and faster than Virtual Machines.

Docker is a platform which packages an application and all its dependencies together in the form of containers. This containerization aspect ensures that the application works in any environment.

As you can see in the diagram, each and every application runs on separate containers and has its own set of dependencies & libraries. This makes sure that each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

So a developer can build a container having different applications installed on it and give it to the QA team. Then the QA team would only need to run the container to replicate the developer's environment.

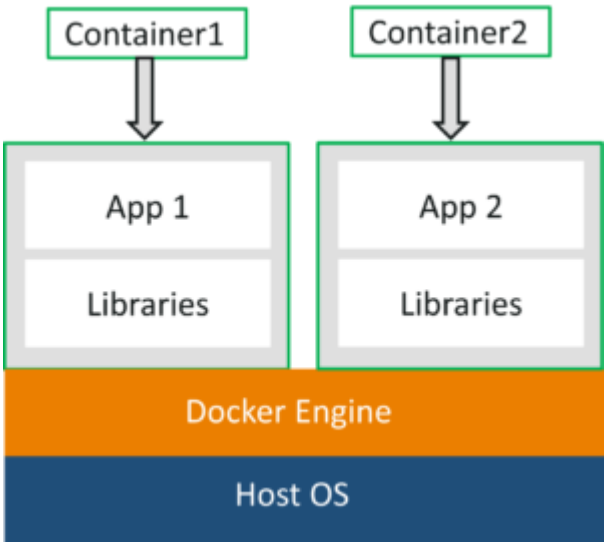

Fig 8. Docker Container Structure

2.3 Kubernetes:

Containers are a good way to bundle and run your applications. In a production environment, you need to manage the containers that run the applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by a system?

That's how Kubernetes comes to the rescue! K8s provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more.

K8s provides you with:

1. Service discovery and load balancing
2. Storage orchestration
3. Automated rollouts and rollbacks
4. Automatic bin packing
5. Self-healing
6. Secret and configuration management

Kubernetes architecture can be understood with the help of following diagram:

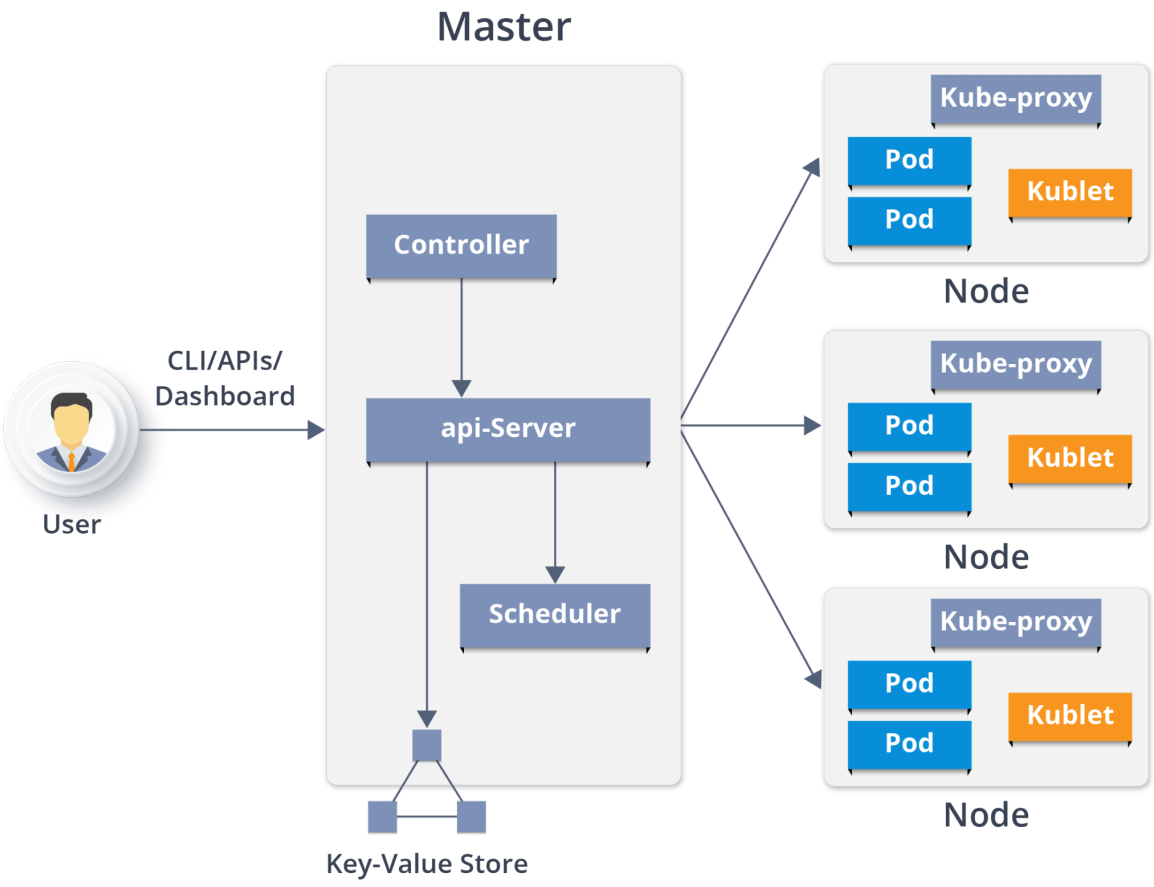

Fig 9. Kubernetes Architecture

2.3.1 Master Node

It is the entry point for all administrative tasks which are responsible for managing the Kubernetes cluster. There can be more than one master node in the cluster to check for fault tolerance. More than one master node puts the system in a High Availability mode, in which one of them will be the main node where we perform all the tasks. For managing the cluster state, it uses etcd in which all the master nodes connect to it.

The components of a master node are explained below:

- ➔ API Server:
  - Performs all the administrative tasks through the API server within the master node.
  - In this REST commands are sent to the API server which validates and processes the requests.
  - After requesting, the resulting state of the cluster is stored in the distributed key-value store.
- ➔ Scheduler:
  - The scheduler schedules the tasks to slave nodes. It stores the resource usage information for each slave node.
  - It schedules the work in the form of Pods and Services.
  - Before scheduling the task, the scheduler also takes into account the quality of the service requirements, data locality, affinity, anti-affinity, etc.
  -
- ➔ Controller Manager:
  - Also known as controllers.
  - It is a daemon which regulates the Kubernetes cluster which manages the different non-terminating control loops.
  - It also performs lifecycle functions such as namespace creation and lifecycle, event garbage collection, terminated-pod garbage collection, cascading-deletion garbage collection, node garbage collection, etc.
  - Basically, a controller watches the desired state of the objects it manages and watches their current state through the API server. If the current state of the objects it manages does not meet the desired state, then the control loop takes corrective steps to make sure that the current state is the same as the desired state.

2.3.2 ETCD

- ETCD is a distributed key-value store which stores the cluster state.
- It can be part of the Kubernetes Master, or, it can be configured externally.
- etcd is written in the Go programming language. In Kubernetes, besides storing the cluster state (based on the Raft Consensus Algorithm) it is also used to store configuration details such as subnets, ConfigMaps, Secrets, etc.

- A raft is a consensus algorithm designed as an alternative to Paxos. The Consensus problem involves multiple servers agreeing on values; a common problem that arises in the context of replicated state machines. Raft defines three different roles (Leader, Follower, and Candidate) and achieves consensus via an elected leader.

2.3.3 Worker Node

It is a physical server or you can say a VM which runs the applications using Pods (a pod scheduling unit) which is controlled by the master node. On a physical server (worker/slave node), pods are scheduled. For accessing the applications from the external world, we connect to nodes.

The components of a worker node are explained below:

➔ Container Runtime:
- To run and manage a container's lifecycle, we need a container runtime on the worker node.
- Sometimes, Docker is also referred to as a container runtime, but to be precise, Docker is a platform which uses containers as a container runtime.

➔ Kubelet: It is an agent which communicates with the Master node and executes on nodes or the worker nodes. It gets the Pod specifications through the API server and executes the containers associated with the Pod and ensures that the containers described in those Pod are running and healthy.

➔ Kube-proxy:
- Kube-proxy runs on each node to deal with individual host subnetting and ensure that the services are available to external parties.
- It serves as a network proxy and a load balancer for a service on a single worker node and manages the network routing for TCP and UDP packets.
- It is the network proxy which runs on each worker node and listens to the API server for each Service endpoint creation/deletion.
- For each Service endpoint, kube-proxy sets up the routes so that it can reach it.

➔ Pods: A pod is one or more containers that logically go together. Pods run on nodes. Pods run together as a logical unit. So they have the same shared content. They all share the same IP address but can reach other Pods via localhost, as well as shared storage. Pods don't need to all run on the same machine as containers can span more than one machine. One node can run multiple pods.

2.4 Jenkins:

Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

With Jenkins, organizations can accelerate the software development process through automation. Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis, and much more.

Jenkins achieves Continuous Integration with the help of plugins. Plugins allow the integration of Various DevOps stages. If you want to integrate a particular tool, you need to install the plugins for that tool. For example Git, Maven 2 project, Amazon EC2, HTML publisher etc.

The image below depicts that Jenkins is integrating various DevOps stages:

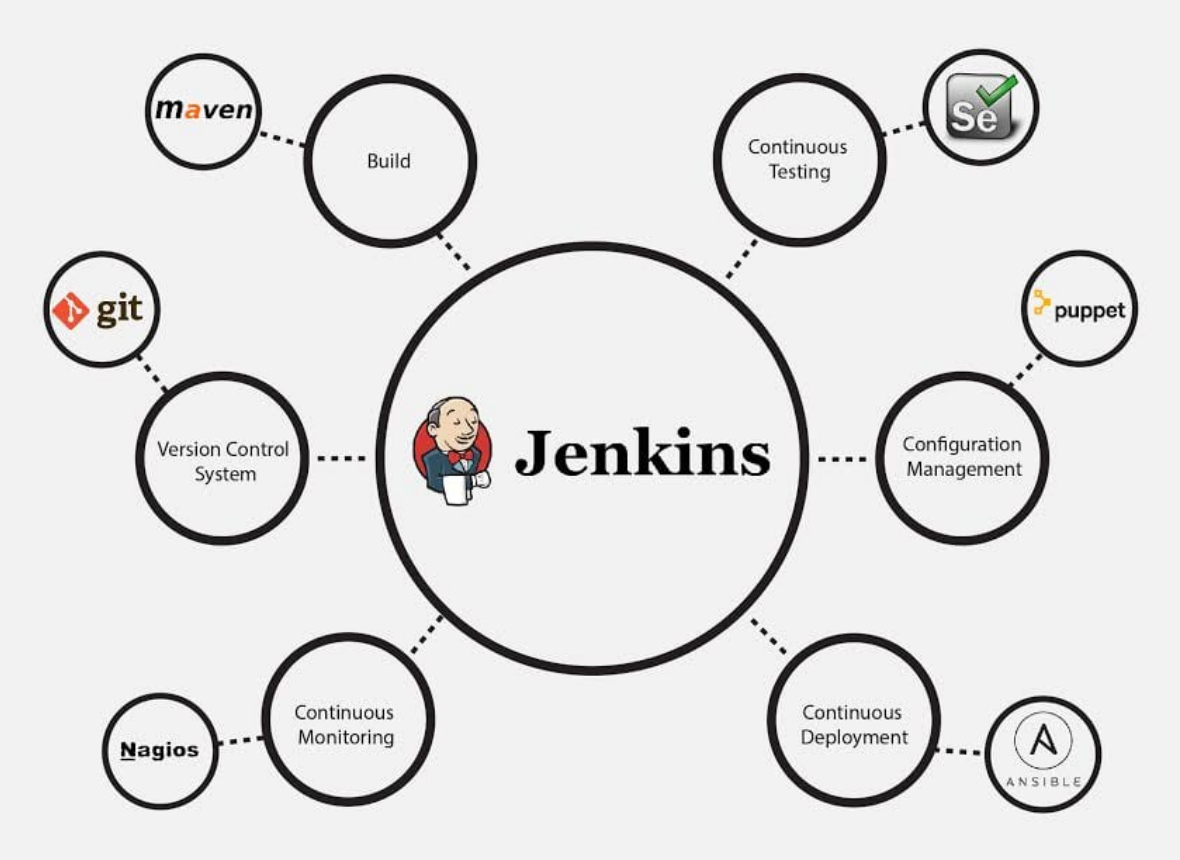

Fig 10. Jenkins working

2.5 Prometheus/Grafana

Prometheus is an open-source systems monitoring and alerting toolkit originally built at SoundCloud. Since its inception in 2012, many companies and organizations have adopted Prometheus, and the project has a very active developer and user community. Prometheus collects and stores its metrics as time series data, i.e. metrics information is stored with the timestamp at which it was recorded, alongside optional key-value pairs called labels.

Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength.

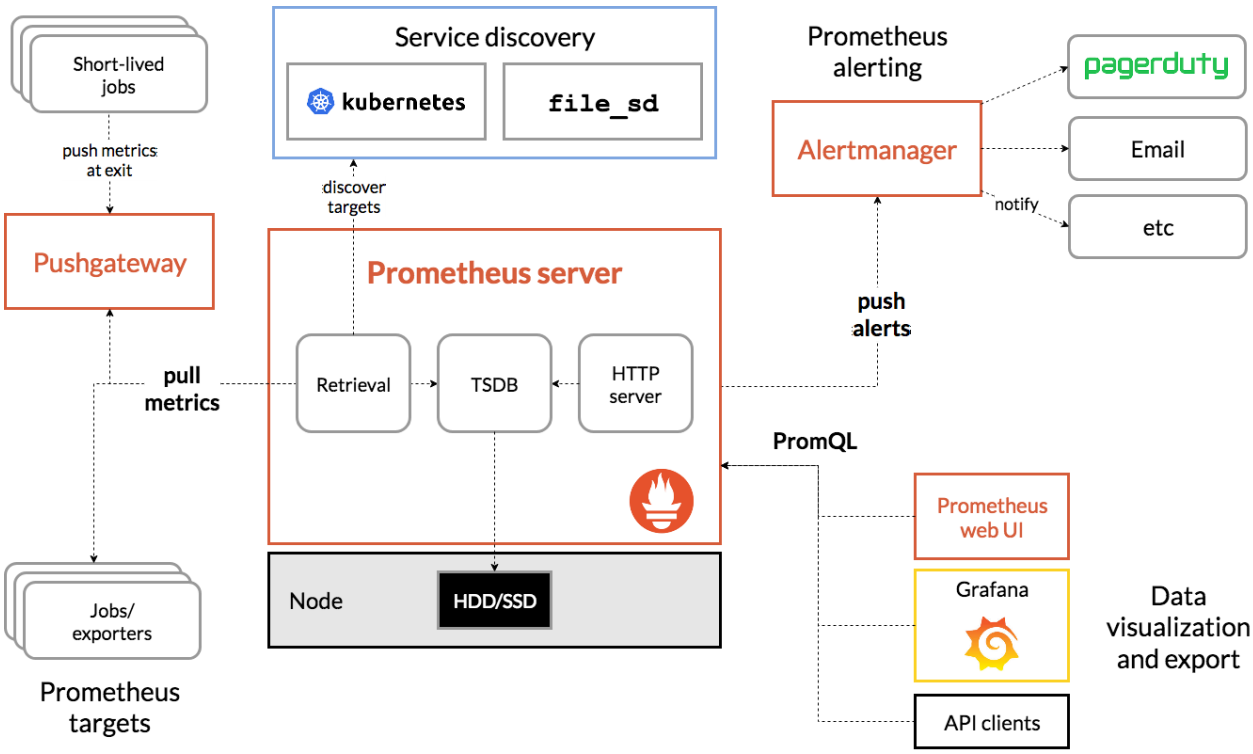

Fig 11. Prometheus and Grafana architecture

Grafana is an open source solution for running data analytics, pulling up metrics that make sense of the massive amount of data & to monitor our apps with the help of cool customizable dashboards. Grafana connects with every possible data source, commonly referred to as databases such as Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL etc.

Grafana being an open source solution also enables us to write plugins from scratch for integration with several different data sources. The tool helps us study, analyze & monitor data over a period of time, technically called time series analytics.

It helps us track the user behavior, application behavior, frequency of errors popping up in production or a pre-prod environment, type of errors popping up & the contextual scenarios by providing relative data.

Chapter-3

METHODOLOGY

3.1 Architecture Diagram of Project

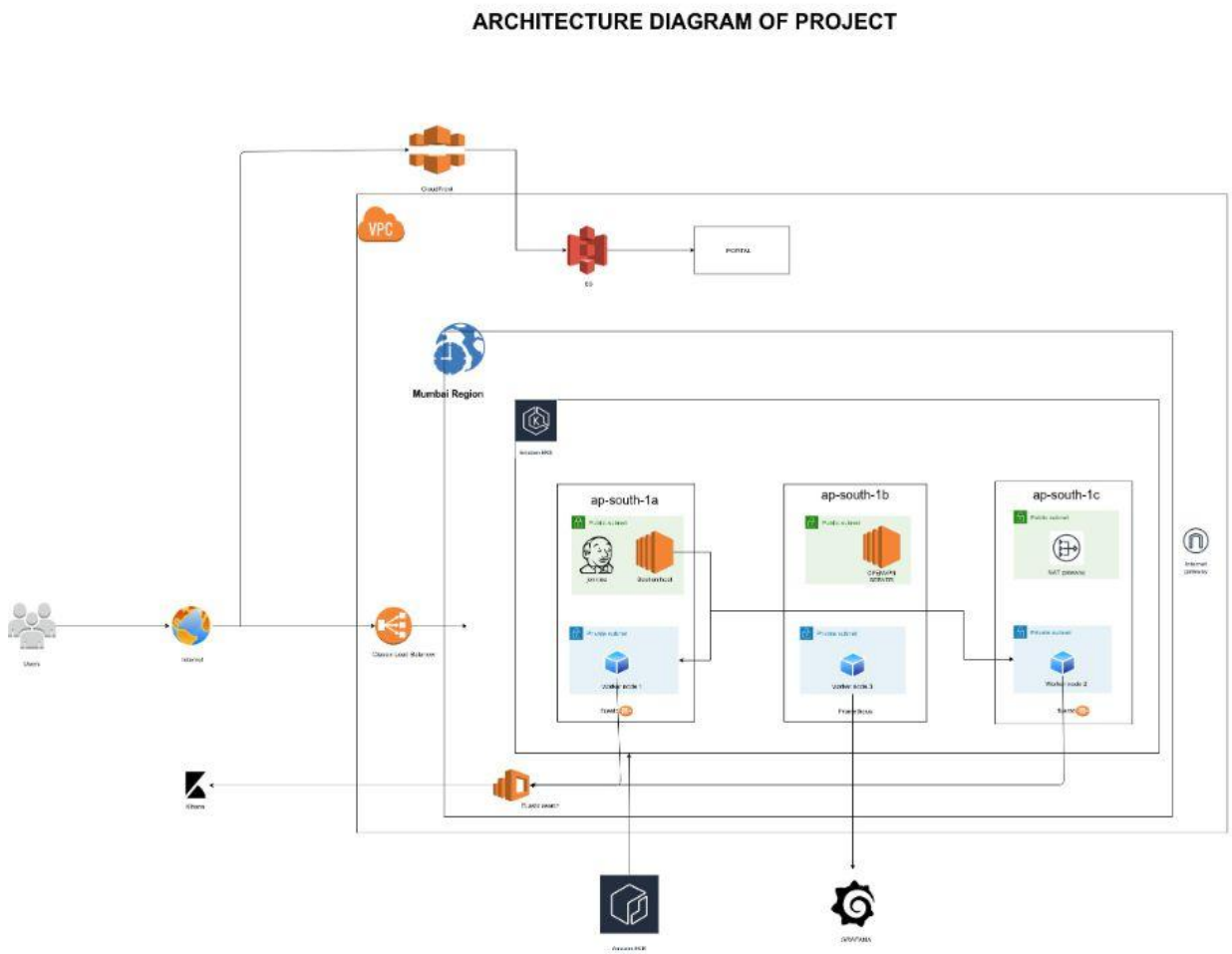

Fig 12. Architecture Diagram of Project

3.2 Architecture Diagram of Jenkins

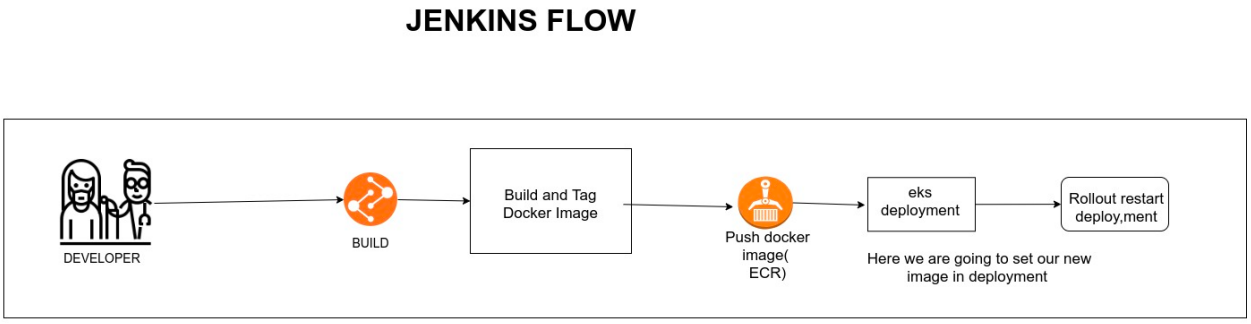

Fig 13. Architecture Diagram of Jenkins

3.3 Architecture Diagram of API Gateway

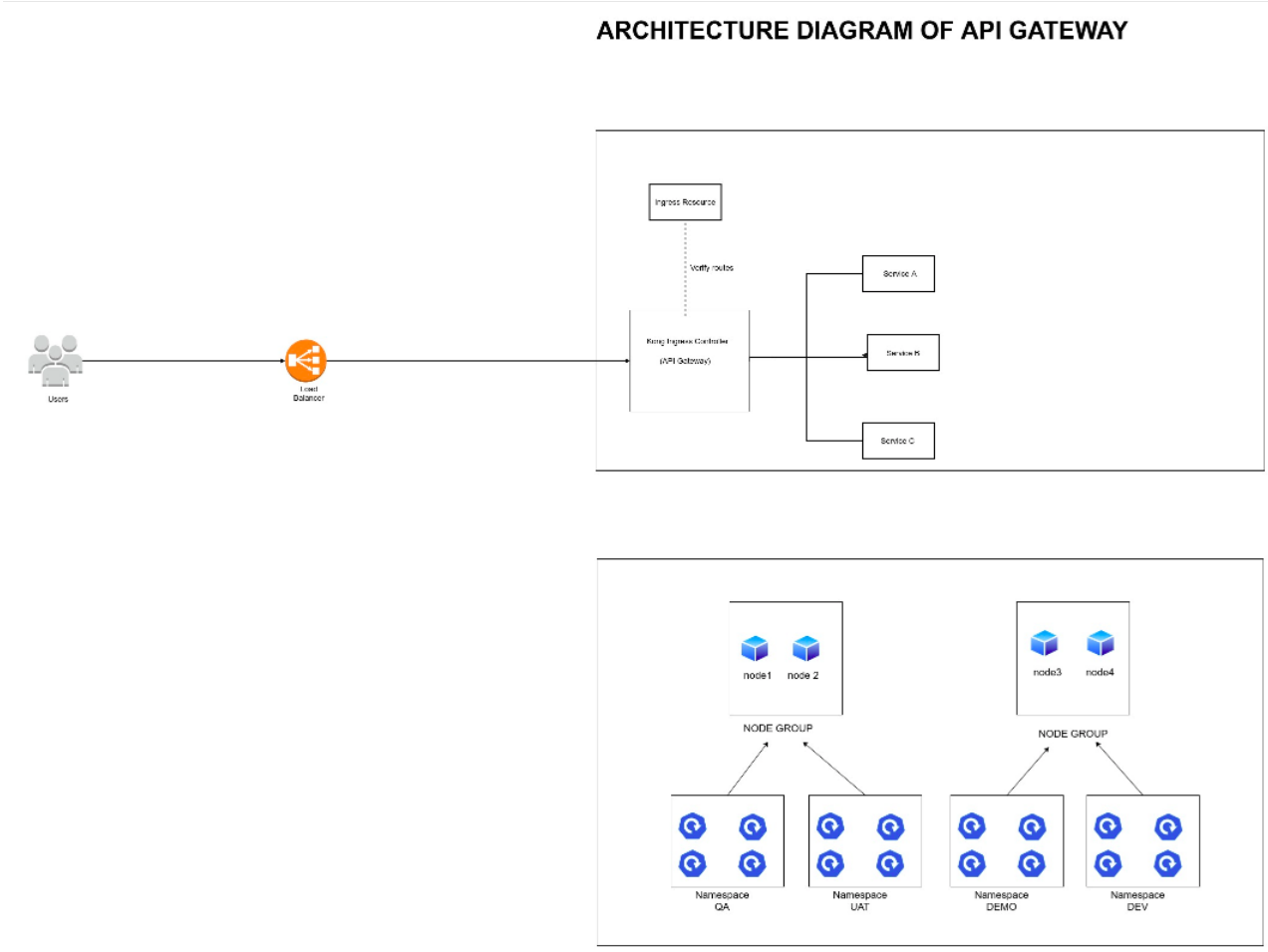

Fig 14. Architecture Diagram of APi Gateway

Chapter-4

Limitations

1. AWS does have general cloud computing issues when you move to a cloud such as a downtime, limited control, and backup protection. However, these flaws can be overcome after some time. This makes them a temporary issue.
2. As security is one of the main features so AWS limits some of its features which cannot be changed at all are-
    - ➔ EC-2 classic- Maximum of 500 per instance and each Security Group can have a maximum of 100 permissions.
    - ➔ EC2-VPC- Up to 100 security groups per VPC.
3. If you need more immediate or intensive assistance, you'll have to opt for paid support packages among 3 which are-
    - ➔ Developer: $29/month.
    - ➔ Business: Greater of $100
    - ➔ Enterprise: Greater of $15,000

Chapter-5

Conclusion

The report detailed the requirements for the implementation of DevOps Project. The migration from a monolithic application to a cloud based application was successfully provided to the client. We've been providing continuous monitoring and upgradations as per requirement to the client, since the initiation of the project. This project required in-depth knowledge and analysis of client requirements as well as AWS infrastructure.