e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
Volume:03/Issue:06/June-2021 Impact Factor- 5.354 www.irjmets.com

AUTOMATE APPLICATION DEPLOYMENTS BY MODERNIZING CI/CD PIPELINES

Harsh Vardhan Kataria[*1], **Chandra Shekhar Panwar**[*2], **Abhilash Patel**[*3], **Dr. Preeti Narooka**[*4]

[*1,2,3]B. Tech Student, Computer Science Department, Geetanjali Institute Of Technical Studies Udaipur, India.

[*4]Assistant Professor, Computer Science Department, Geetanjali Institute Of Technical Studies Udaipur, India.

ABSTRACT

Imagine a world where product owners, Development, Quality Assurance, Operations, and Information Security Teams work together, not only to help each other, but also to ensure that the overall organization succeeds in terms of profitability, security and reduced costs. They allow the quick flow of planned work into production by working together toward a single objective (e.g. performing tens, hundreds, or even thousands of codes deploy per day), while achieving world-class stability, reliability, availability, and security. In our world, Development and IT Operations are antagonists; testing of applications and InfoSec activities happen only at the end of a project, too late to correct If any issues are discovered, and practically any vital task involves too much backbreaking work and too many handoffs, keeping us waiting all of the time. Not only does this contribute to extremely long lead times to get anything done, but the quality of work, especially production deployments, is also problematic and chaotic, resulting in negative impacts to customers and business. As a consequence, we fall well short of our objectives, and the whole business is unhappy with IT's performance, leading to budget cuts and irritated, disgruntled staff who feel helpless to influence the process and its consequences. Our main goal is to create a platform for developers, which not only compiles & runs the app with minimal configurations but also abstracts the deployments part with loose architecture & negligible learning curve, and developers will be able to get all the benefits of multi cloud platforms without any hassles.

Keywords: Continuous Integration, Continuous Deployment, Multi-Cloud Deployments, Kubernetes, Tekton Pipelines, Docker.

I. INTRODUCTION

The purpose of this project is to build a platform for developers which only requires minimal configurations to run the app with a negligible learning curve of backend deployment. All the benefits of the cloud, scaling, availability, disaster-recovery, security-constraints, and huge reduction in costs etc., will be handled by the platform along with great isolation among different infrastructure levels with features including Automatic Builds, Resource Limits, CI/CD Pipelines, and Automatic Scaling as per requirements.

The main objectives are: -

i. To setup Automatic Builds for different programming languages.
ii. Resource Limits for projects deployed.
iii. Automatic Scaling based on Thresholds as per requirements.
iv. Continuous Integration & Continuous Deployment Pipelines.
v. Per Second Billing based on resource consumption.

II. BACKGROUND OVERVIEW

A. Existing System- On analyzing the current methodologies & deployment plans followed by different IT Firms, we scrutinized that more than 90% of these firms deploy new updates/patches only on the weekends and that too without any senior developer on-board in order to assist the deployment. Secondly, almost every other feature that is being deployed is tightly coupled with the whole technology stack.

B. Drawbacks of Existing System- At the production level, one thing that always haunts a developer is whether the code that he/she has written will be successfully deployed or not. Secondly, even if that works out in production, every time there’s an added dependency or update/feature one has to set up different versions of OS, SDK's, Application Environments, and managing firewalls in order to test applications on different platforms which is a huge overhead. Proposed System- On analyzing these drawbacks, We found the need of a platform for developers, where they have to provide only the source and general instructions to run the app (minimal configurations), and they will be able to get all the benefits of cloud along with scalability, reliability, usability, performance effectiveness and huge cut in costs of deployment and most important great isolation among different features along with loosely coupled architecture of the whole technology stack.

III. METHODOLOGY

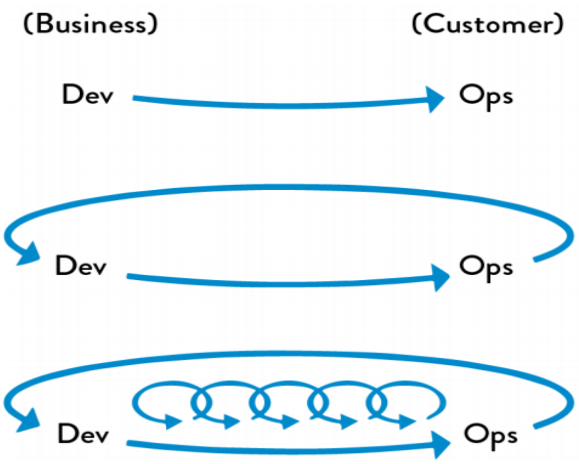

Figure 1: DevOps Methodology

Devlops is a software development (Dev) and operations (Ops) technique (Ops). The goal is to make it easier for teams to communicate so that they can create, test, and distribute software more rapidly and efficiently. The key factor that differentiates DevOps Methodology from others is the deployment process & automation, while other methodologies emphasizes mostly on software development process, DevOps actively focusses on the release process of a software in secure & reliable manner with very less manual intervention of testing & approving things manually. The main concept of DevOps is to manage end-to-end Engineering Process by collaborating Development & Operation teams work together. It is necessary for both the teams to fully understand the software release & software/hardware requirements & its implications to the deployment.

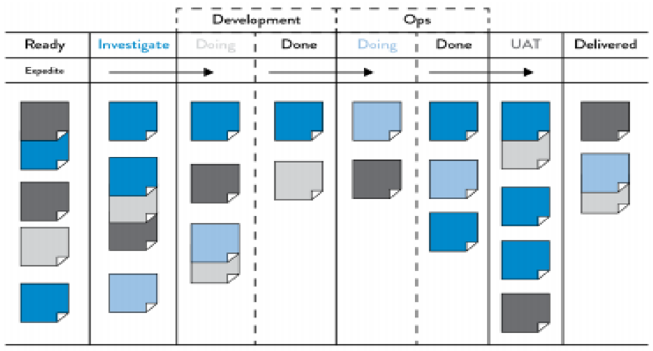

Figure 2: DevOps Work Plan

IV. SCOPE

Since DevOps practices & methodologies came into existence, many fortune companies have seen world-class performance increases and thus achieving revenue increases, faster deployments and less chaos in deployment times. DevOps has enhanced product development while simultaneously lowering the risk of failure. The pipeline process has been controlled by DevOps in such a way that high-quality software can be generated with very less effort. During the development, testing, and deployment phases of an application, one must enable constant feedback of the release process in order to reduce the security concerns that arises after it hits the production servers. Our project not only enables developers to achieve World Class Reliability, Continuous Integrations & Delivery, and Huge Cost Reductions but also enables them to deploy their applications at much higher scale over multi-cloud and managed services.

Future Scope of Our Project: -

A. AI/ML Pipelines: - The software development landscape has been altered by DevOps. However, artificial intelligence and machine learning may help us to automate our application in a more regulated manner. Applying AI and ML to the pipelines can help us run builds and automation in a much better with closer insights a control and provide a cost-effective approach to automate an entire working pipeline.

Platform as a Service (PaaS): - Platform as a service (PaaS) is a growing field with a lot of applications. Gone are the days when developers had to worry about putting together a full application architecture. PaaS enables teams to expand development capabilities without adding staff, potentially lowering engineering expenses. The developers have no control over the underlying cloud infrastructure, such as the network, servers, operating systems, or storage, but they do have control over the apps that are deployed and maybe the application hosting environment's configuration settings. following Diagram better illustrates the differences: -

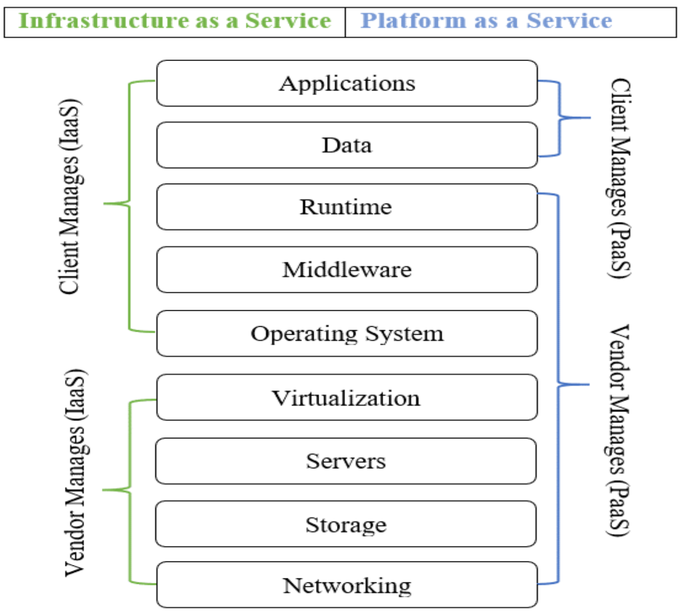

Figure 3: Difference between IaaS and PaaS

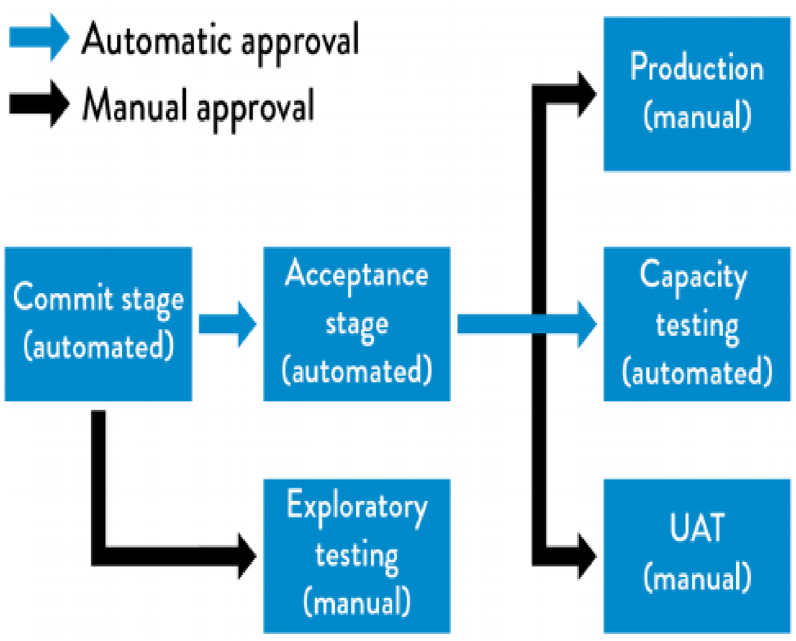

Figure 4: Activity Diagram

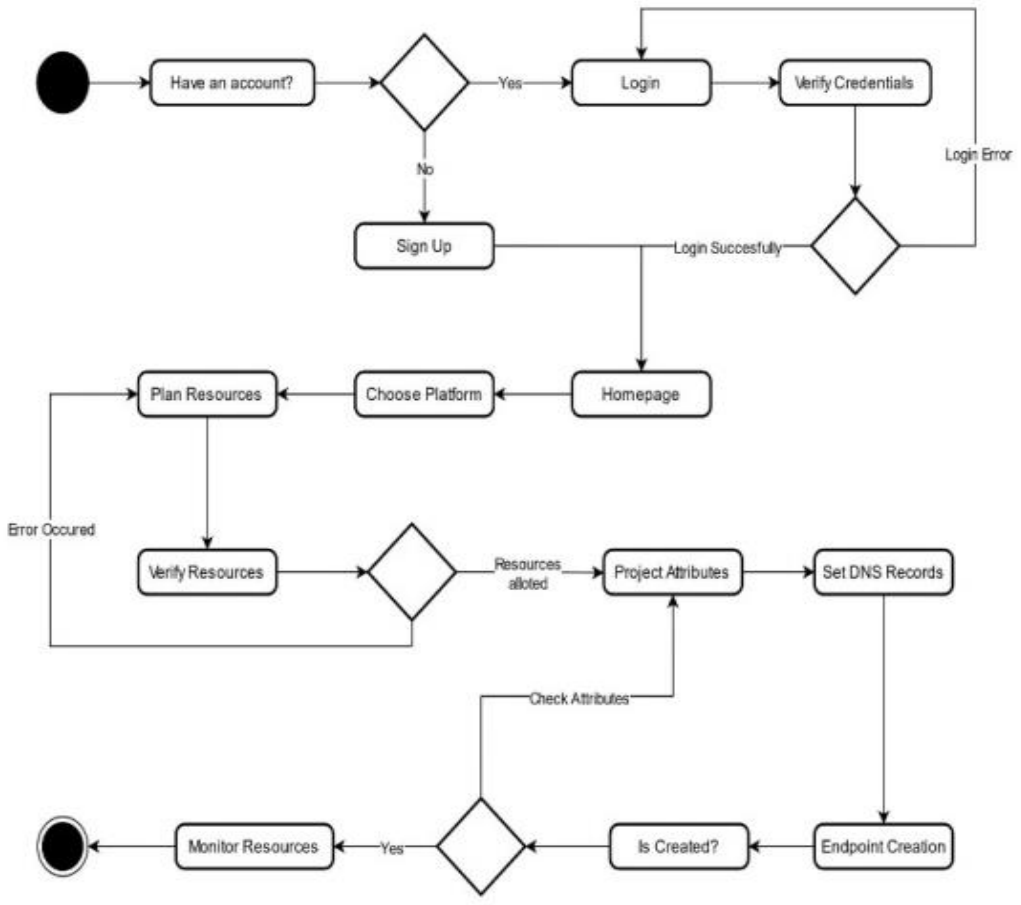

Figure 5: CI/CD using Tekton Pipelines

B. Containers Overriding Configuration Management: - The technology world is being shaken by container orchestration platforms. Container orchestration techniques have grown to the point where they might eventually replace multiple tools like Ansible, Chef, and Puppet. Kubernetes is the most well-known and commonly used container orchestration technology today, but there are many more on the way. Container orchestration solutions, when properly designed, may simplify infrastructure provisioning and many of the complexity that come with it.

V. WORKING & RELEVANCE

A. Distributed Workloads: - In modern architectures, large number of services are delivered over the network via APIs. These are distributed by distribution systems that run on several servers and configurations in various places, all of which communicate with one another to coordinate their operations.

B. Load Balancers for Zero Downtime: - Gone are the days we used to simply deploy an application in one VM and used to point a DNS to it. Load balancing and horizontal scalability is achieved by making systems distributed like these.

C. Restful API's: - We are using these exposed APIs on a daily basis, all around the globe, these systems should be highly reliable and available, i.e. They can't fail, and there should be no downtime. Because these services are used all over the world, they must be scalable without requiring a complete overhaul of the present infrastructure. For your containerized application, Kubernetes provides appropriate services to do all of this.

D. CI/CD Using Tekton Pipelines: - Continuous Integration (CI) allows us to continuously integrate code into a single shared and easy to access repository. Continuous Delivery (CD) helps us to push code from the repository to production on a regular basis. CI/CD creates a fast and effective process of getting a product to market much before the competition it also helps us in releasing new features and bug fixes.

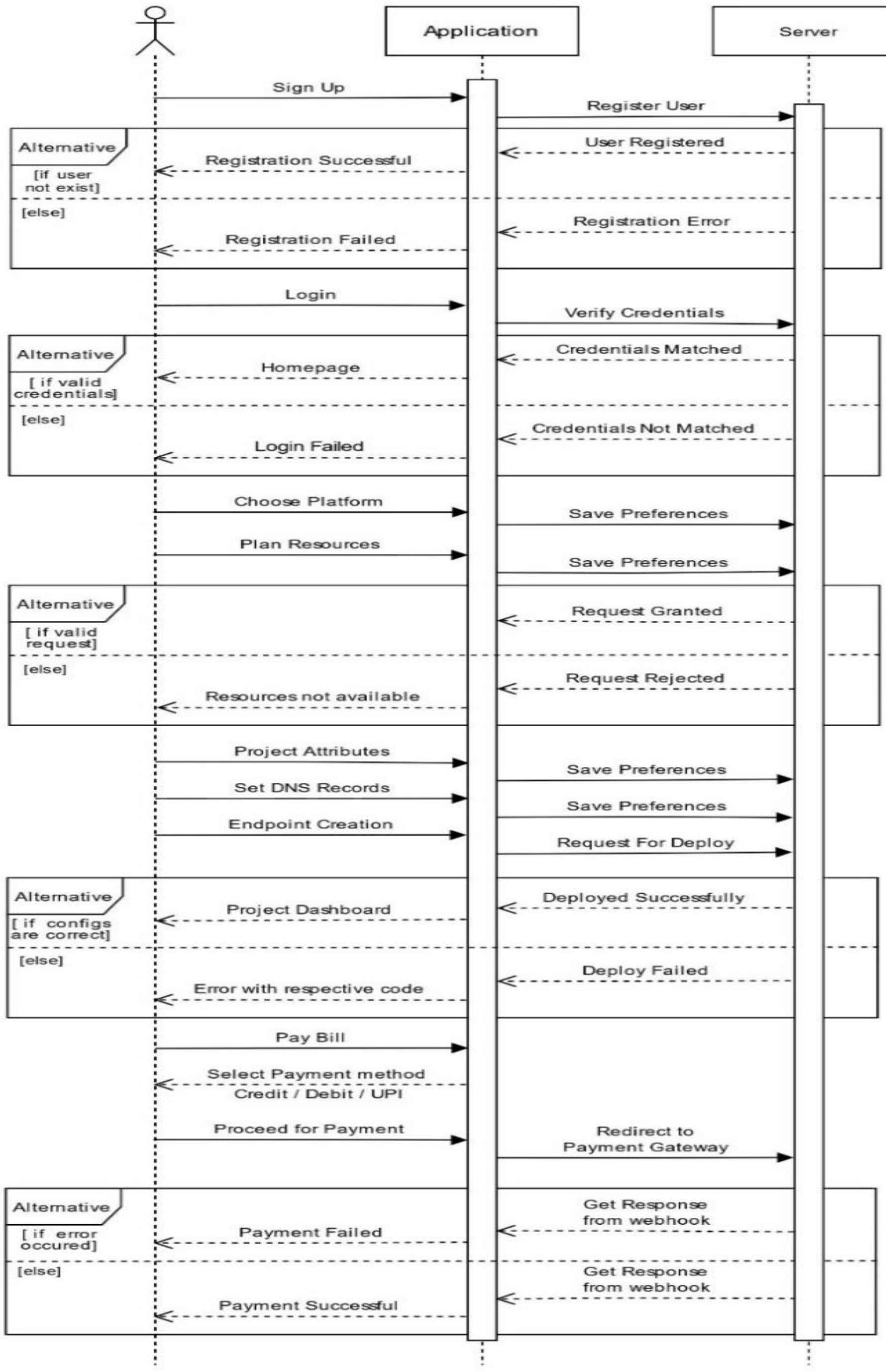

Figure 6: Sequence Diagram

E. Sequence Diagram: - Below Diagram illustrates the flow of a user when they interact with our platform.

VI. CONCLUSION

As per the analysis, many IT Firms have shown that releases & deployments should not be high-risk, tightly coupled and does not requires hundreds or even thousands of engineers to deploy at production. Instead, it should be done is such a manner that should be in everyone's routine & part of daily workflow. By doing this, we can minimize the impact of failed deployments & chaos at production level & reduce the lead times further from months to minutes to even seconds. Our project enables this by minimizing the time & efforts required to build & compile the application at different organizational environments (Test, Production) and supports different deployment strategies (Blue-Green, Canary, A/B Testing, Shadow, Ramped) by enabling automatic builds for different programming languages & creating Continuous Integration & Continuous Deployment Pipelines in order to minimize the release times. On integrating with Kubernetes & multi-cloud platforms, the application could be scaled indefinitely based upon requirements & user traffic, this not only supports the isolation b/w different applications but also enables per second billing approach to monetize the resources based on consumption per second.

VII. REFERENCES

[1] Gene Kim, Patrick Debois, Professor John Willis, Jez Humble, John Allspaw (2017), “The DevOps Handbook” in Paperback.

[2] Citrix Application Delivery and Security, Retrieved From https://www.citrix.com/en-in/solutions/app-delivery-and-security/what-is-containerization.html

[3] Alexander Kainz (7 Jul 2020), Microservices vs. Monoliths: An Operational Comparison – The New Stack [Blog Post], Retrieved From: https://thenewstack.io/microservices-vs-monoliths-an-operational-comparison/

[4] What is Kubernetes [Documentation] Retrieved From:

[5] https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[6] What is CI/CD? [Blog Post] Retrieved From: https://www.redhat.com/en/topics/devops/what-is-ci-cd.