International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

Unlocking Agile Software Deployment: The Power of Canary Deployments

Mansi Puthran[1], Flavia Gonsalves[2]

[1]Student, Institute of Computer Science, Mumbai Educational Trust - MET ICS, Mumbai, India

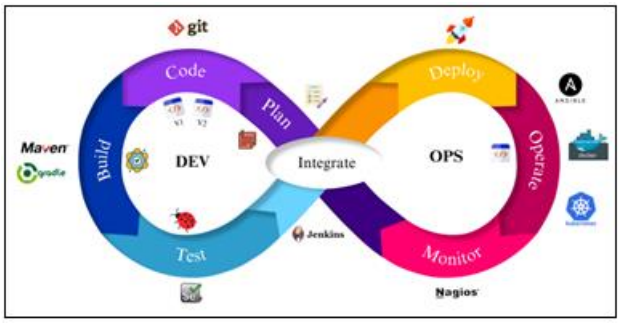
[2]Professor, Institute of Computer Science, Mumbai Educational Trust - MET ICS, Mumbai, India

Abstract: ***Canary deployments offer a strategic approach to software deployment, enabling organizations to gradually roll out updates and monitor their impact. This research paper provides a comprehensive analysis of canary deployments, covering their working mechanisms, adoption trends, and implications in modern software development. Through insights into frequency, adoption rates, tool analysis, efficiency evaluations, and case studies, organizations can gain valuable guidance for implementing canary deployments effectively and confidently navigating software updates.***

Keywords: Canary Deployments, Rolling deployment, Blue and Green, Feature Toggle, DevOps

1. Introduction

There used to be two separate departments during the inception of software development; the development team; in charge of making plans, designs & structures aimed at creating software while operations team will take care of testing and implementation. The operations team would inform the developers of errors and required modifications that caused most times the slowing down of the whole process. While the operations team was still testing the previous work, this waiting period would be a time when the development team would either do nothing or begin new projects; such could prolong project implementation period by weeks or even months. Now what if these departments merged? How could the "Wall of Confusion" dividing them be taken down? That's where DevOps comes in. DevOps is a move from a separated approach to a more consolidated one, a continuous flow indicated by the infinity sign showing that it is a cycle that never stops through development, feedback, and improvement. Helping businesses adjust faster hence achieving rapid deployments that are more reliable, DevOps brings together those responsible for creating a software program and those responsible for running their system operations. In spite of possible communication problems, DevOps promotes more effective software development through better teamwork. DevOps comprises a number of stages that are in turn backed by different tools and practices. We will thus delve into these stages in order to fathom how DevOps brings about clearer and more efficient workflows.



2. Phases of DevOPS

DevOps is made up of multiple steps that are backed up by different systems and behaviors. We shall delve into these stages in order to discover how DevOps enables smoother and better workflows.

A) Planning Phase:
This is where everything begins. The development team creates a plan outlining the objectives of the application and the features to be delivered to the customer. It's crucial to set clear goals at this stage, as it lays the foundation for the entire project.

B) Coding Phase:
Once the plan is set, the coding begins. Developers write the code, often collaborating on the same project. To manage different versions of the code and avoid conflicts, they use version control tools like Git. This approach allows multiple team members to work on the same codebase while keeping track of changes. If necessary, they merge different versions of the code to create a cohesive final product.

C) Build Phase:
After the coding is done, it's time to build the software. This involves transforming the source code into an executable form. Tools like Maven and Gradle help automate this process, ensuring that the code is ready for testing and deployment.

D) Testing Phase:
With the software built, it's now put through a series of tests to check for bugs or errors. Automation plays a significant role here, with tools like Selenium performing repetitive tests quickly and efficiently. This phase ensures the software is stable and meets quality standards before it's deployed in a seamless cycle.

These phases form the core of the DevOps lifecycle, allowing teams to work collaboratively, deliver software more quickly, and maintain a consistent quality throughout the process.

E) Deployment Phase:
If the code passes all the tests, it's ready for deployment. The operations team takes over, deploying the software to the production environment. This phase can be automated with tools like Ansible, Docker, and Kubernetes, which simplify the deployment process and ensure consistency across environments.

F) Monitoring and Integration Phase:
Once deployed, the product is continuously monitored for performance and issues. Tools like Nagios help automate this monitoring, providing real-time insights into the software's behavior. Any feedback from this phase is sent back to the planning phase, creating a continuous loop of improvement.

Jenkins plays a key role in this phase, automating the process of continuous integration and ensuring the code is built, tested, and deployed in a seamless cycle.

These phases form the core of the DevOps lifecycle, allowing teams to work collaboratively, deliver software more quickly, and maintain a consistent quality throughout the process.

3. Other Deployment Techniques

In DevOps, deployment is all about keeping the process smooth and reliable. It's part of an ongoing cycle where automation plays a big role, ensuring that software gets updated and released without a hitch.

There are various deployment techniques, and this research paper focuses on canary deployment. But before diving into the specifics of canary deployment, let's take a look at some other common deployment techniques that are practiced in the industry. This way, we can understand how canary deployment stands out and why it's an important part of the DevOps toolkit.

a) Big Bang Deployment

One of the earliest methods for deploying changes to production is the Big Bang Deployment. It's a bit like ripping off a band-aid—all at once, in one go. With Big Bang, all the changes are pushed simultaneously, which usually involves a brief period of downtime. This downtime occurs because you have to shut down the old system before you can switch on the new one.

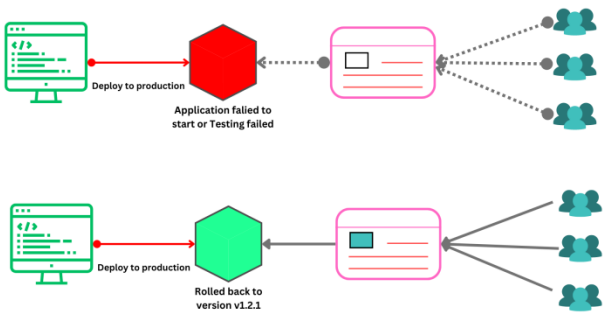


Figure 2: Big Bang Deployment Workflow

While the downtime is typically short, it can cause significant problems if things don't go as planned. That's why preparation and thorough testing are critical with this approach. If there's an issue, the usual solution is to roll back to the previous version. But here's the catch: rollback isn't always straightforward. It can still disrupt users, and there's always the risk of data loss or other complications. This is why it's essential to have a solid rollback plan in place.

Despite its risks, sometimes Big Bang Deployment is the only option—like when you're upgrading a complex database and you can't make incremental changes. It requires careful planning, but it's an approach that's been around for a long time and is still used in specific scenarios.

b) Rolling Deployment

Rolling deployment is a bit like updating your system one step at a time. Instead of pushing changes all at once, you roll them out gradually, updating one server at a time. For example, if you have 10 servers, you start with the first one, update it, and bring it back online. If everything's okay, you move on to the next server, and so on, until you've updated them all.

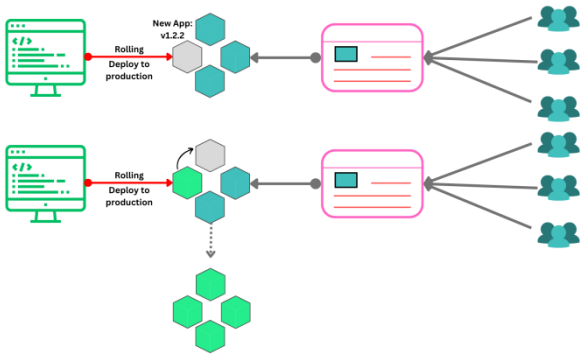


Figure 3: Rolling Deployment Workflow

This approach has a few advantages. It usually avoids downtime because while one server is being updated, the others are still working. It also lets you catch problems early since you're updating incrementally.

However, rolling deployment can be slower and doesn't completely eliminate risk. If an issue slips through, it could still spread as you update more servers. Additionally, you can't control who gets the new version first—all users gradually see it as servers are updated. Despite these drawbacks, rolling deployment is popular because it balances risk with user impact. It allows you to make updates in a careful, methodical way, reducing the chance of major disruptions.

c) Blue-Green Deployment

Blue-green deployment involves maintaining two identical production environments—one active and one idle, often called "blue" and "green." At any given time, one serves users while the other is used for testing.

Here's how it works: you deploy new changes to the idle environment (green), while the active environment (blue) continues to run the current version. This way, users experience no downtime during the transition. Once the new

version is tested and ready, you switch the load balancer to direct traffic from blue to green. If any issues arise, you can quickly switch back, providing a simple rollback option.

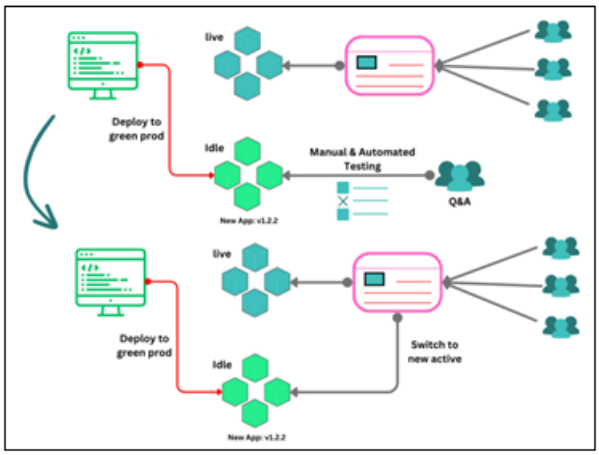


Figure 4: Blue-Green Deployment Workflow

The downside of this approach is that it requires more resources, as you essentially maintain two identical systems. It can also be complex to ensure data synchronization between the two environments. Despite these challenges, blue-green deployment remains popular because it offers smooth transitions and easy rollback options without disrupting users.

4. Canary Deployment

Now that we've covered the other deployment techniques, let's talk about canary deployments, which is the focus of this research paper. Unlike traditional methods, canary deployments offer a gradual and controlled way to roll out software updates. This approach reduces risk and provides useful feedback during the deployment process. Let's explore how canary deployments work and why they've become a favorite for teams looking to innovate without sacrificing stability.

Canary deployment is named after the old mining practice of using canaries to detect toxic gases. Miners used canaries because these birds were more sensitive to dangerous fumes. If the canary showed signs of distress, it was a warning signal to get out quickly. In software deployment, canary deployments work similarly, offering an early warning system for potential issues before a full roll out.

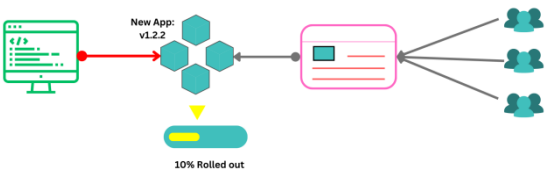


Figure 5: Canary Deployment Workflow

Here's the idea. Instead of releasing a new software version to all servers or users at once, we start with a small subset, our canaries. It could be a small percentage of servers or a specific group of users, chosen based on various criteria.

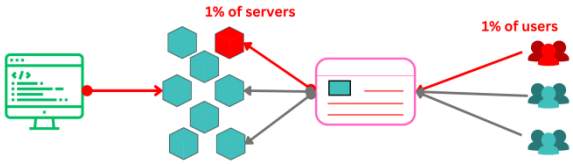


Figure 6: Small subset of users and servers

For instance, we might begin by deploying to a single server, a small cluster, or even a specific geographic region. This lets us see how the new version behaves in a real-world setting, but on a smaller scale.

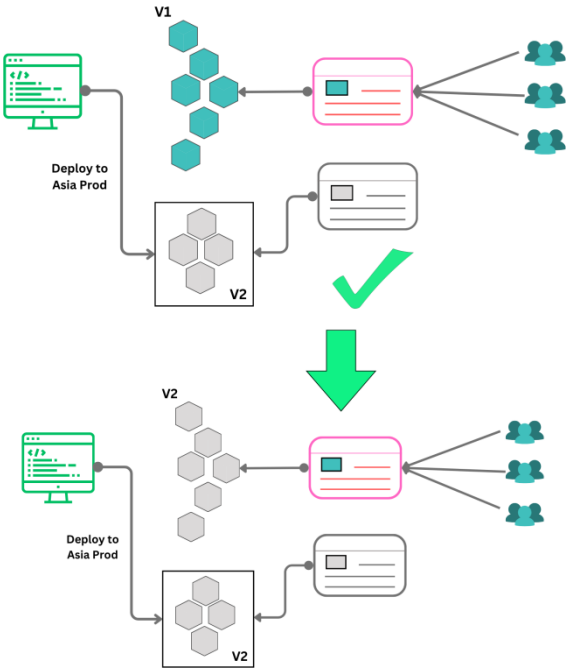
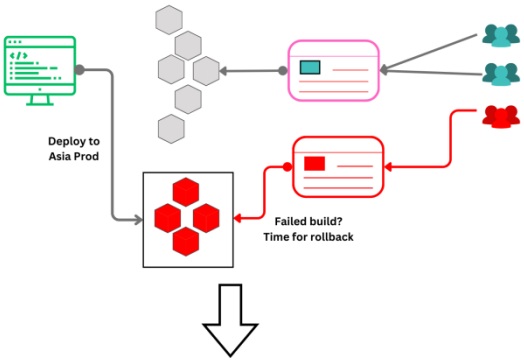


Figure 7: Successful Scenario

If things go well and the new version behaves as expected, we can start extending the deployment to more servers or users. But if there's a problem, we have a safety net. We can pause the deployment, fix the issues, and avoid impacting the majority of users. This incremental approach provides both safety and control.



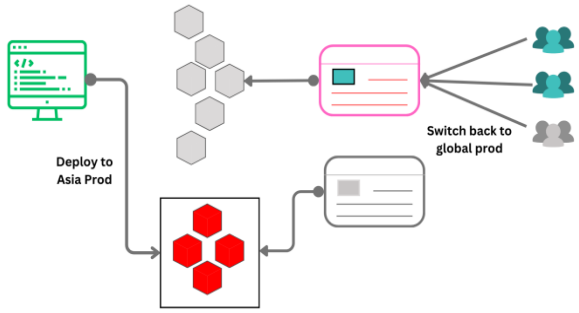


Figure 8: Failed Scenario

Canary deployment also allows for targeted roll out, which is particularly useful for directing deployments to specific groups, like a certain region or device type.

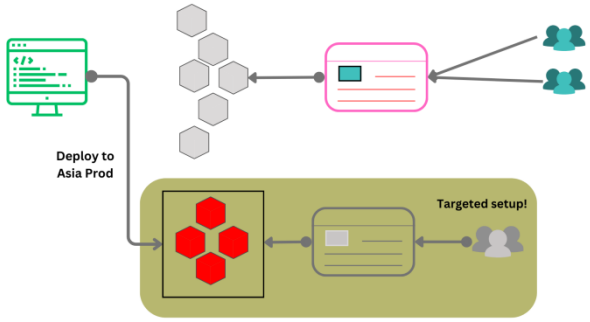


Figure 9: Targeted roll out

However, it has its own set of challenges. You need careful monitoring and automated testing to keep track of the canaries. The infrastructure has to be ready to ramp up or halt the deployment if needed. Implementing canary deployment can get complex, especially if there are database changes or compatibility issues with existing systems.

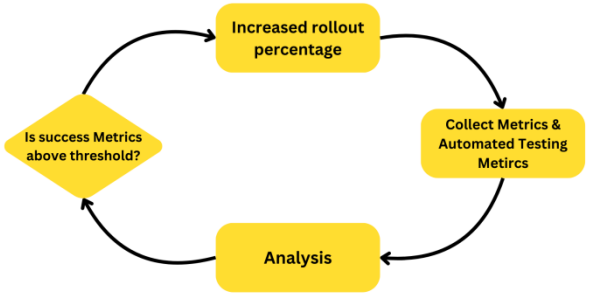


Figure 10: Canary Deployment Challenges

It's worth noting that canary deployment is rarely a standalone strategy. It's often combined with rolling deployment or other approaches, bringing together the best aspects of multiple deployment strategies. Despite the complexity, many organizations find canary deployment valuable for reducing risk and improving the reliability of software updates.

A. How to pick servers and users for roll out?

When doing canary deployments, picking up which servers and users get the new updates first is very important. It affects how smoothly the deployment goes and how quickly you can spot and fix any problems. So let's talk about some of the approaches we can use to make this decision.

1) **Geographical Location:** One option is to base your roll out on geographical location. This method works well when a feature is intended for a particular region. For example, when WhatsApp launched UPI payments, they only released it to users in India, where the system is used, rather than to their global user base. This type of geographical segmentation is a form of canary deployment because it lets you test changes in a specific context.
2) **User Cohorts:** Another approach is to target user cohorts. This involves rolling out features to a defined group based on specific criteria. For example, you might choose to release a new feature to users aged 25 to 30 who work in the tech industry. This targeted deployment allows for focused testing and feedback from a specific segment of users.
3) **Random Selection:** Random selection is a simpler method. Instead of focusing on a particular group or location, you randomly select a percentage of users or servers to receive the new update. This can be done by directing a random 5% or 10% of traffic to a particular server, without knowing which users will be affected. This randomness helps avoid bias and can be an effective way to gather diverse feedback. Beta users are another type of cohort often used for early testing. Companies like Google Play ask users if they want to sign up for beta programs, giving them access to early releases of apps. When a new version is ready, it is rolled out to these beta users, allowing them to test and provide feedback before a wider release. This is similar to canary deployment, but it's more about A/B testing—testing one version against another to compare results.
4) **Sticky + Random Selection:** Sticky plus random selection combines the benefits of random selection with consistency. You select a few people at random, and then consistently send new updates to those same people. This approach allows you to track how changes impact a steady group over time.
5) **Internal Employees:** Finally, internal employee rollouts are used by companies like Facebook. They have two versions of their app: the "yellow" version, used by internal employees, and the "blue" version, which is released to the public. When a new feature is developed, it's first released to internal employees for testing and feedback. Once they're satisfied, it's rolled out to external users. This method ensures that any major issues are caught and fixed internally before the public rollout.

The selection criteria for a rollout depend on specific use cases and goals. There's no strict rule that works for everyone, so we have to choose the approach that best suits our needs.

B. How can we combine Canary Deployment with out deployment techniques for better efficiency?

1) Canary Deployment with Rolling update-

Combining rolling updates with canary deployment is like having the best of both worlds in the software update game. Let's break it down with a simple example. Imagine you're a big online retailer, and you've just

developed a shiny new feature for your website. Now, you want to roll it out to your users, but you don't want to risk disrupting everyone's shopping experience if something goes wrong.

Here's where the magic of rolling updates and canary deployment comes in. Instead of pushing the new feature to all your servers at once, you start with just a few. These servers act as your canaries, testing the waters before diving in completely. You monitor how the new feature performs on these servers, checking for any bugs or performance issues.

Meanwhile, the rest of your servers continue serving the old version of your website, keeping things running smoothly for the majority of your users. As you gain confidence in the new feature's stability, you gradually roll it out to more servers.

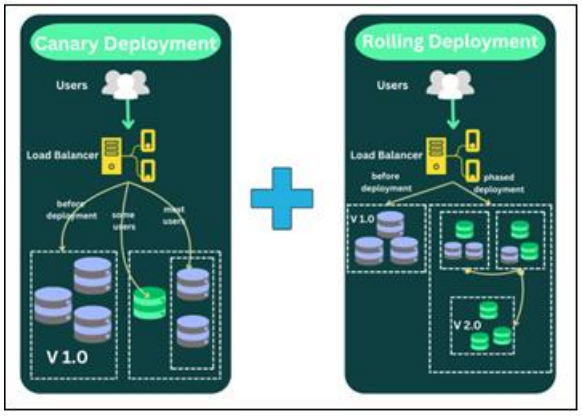


Figure 11: Canary with Rolling update

By combining rolling updates with canary deployment, companies can ensure a smooth and efficient rollout of new features while minimizing the risk of disruptions. It ensures that your users always have a great experience, no matter what updates you're making behind the scenes.

2) Canary Deployment with Feature Toggle-

Before talking about how these two will work together let us first understand what is Feature toggle. Feature toggle stands a bit apart from the other strategies we discussed. It's not about deploying a new version of the entire application, but rather about managing specific new features within the application. With Feature Toggle, we introduce a toggle or a switch in the code for new features. This allows us to turn the feature on or off for certain users or circumstances. Think of it as a gate that we can open or close. It controls who gets to see the new feature.

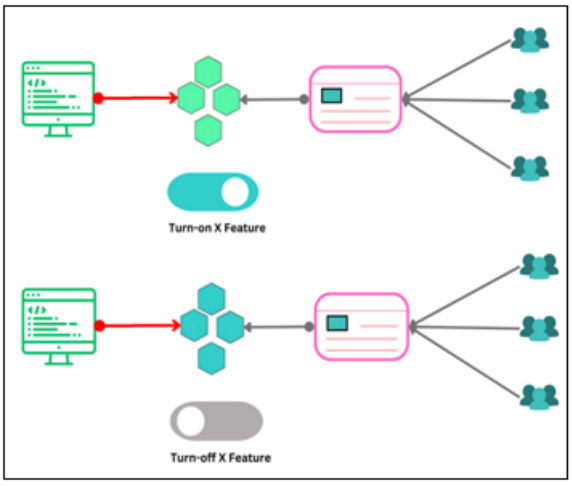


Figure 12: Canary with Feature Toggle

Now we can combine feature toggle and canary deployment in such a way- We can turn on the Feature Toggle for just the canary users, letting them test out the new feature while the rest of the user base carries on with the current version. Feature Toggle offers excellent control over new features and a lot of targeted user testing. It's great for A/B testing or gradually rolling out a feature to see how it performs.

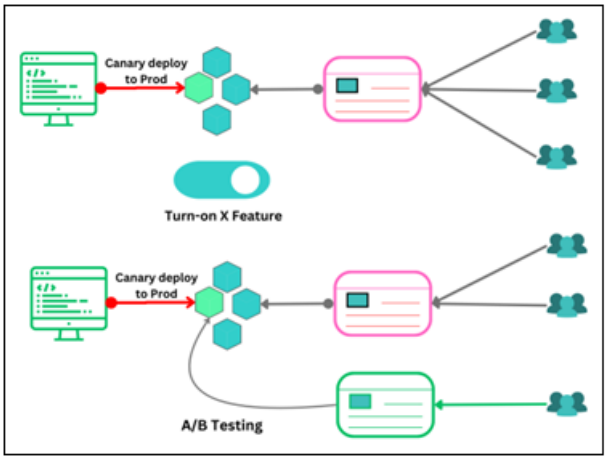


Figure 13: Canary with Feature Toggle (A/B Testing)

However, Feature Toggle has its downsides. If not managed properly, toggles can add complexity to the code base and make testing more difficult. All obsolete toggles need to be cleaned up to prevent toggle debt, which can make the system increasingly hard to maintain.

5. Survey Analysis

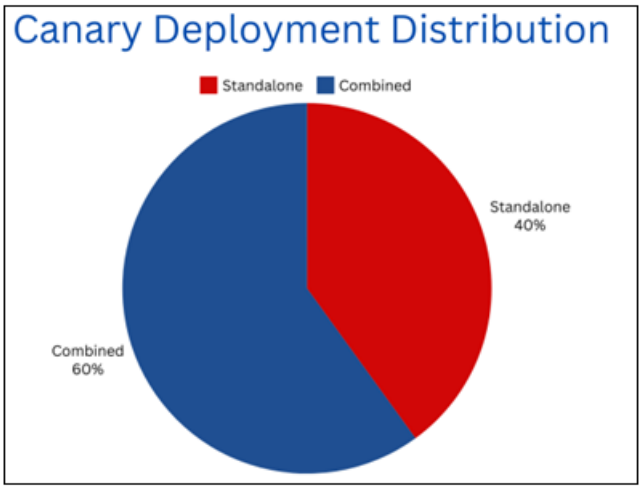


Figure 14: Pie Chart of Canary Deployment Distribution

From the above statistics it was found that 60% of the organizations/companies combine canary deployment with other deployment techniques and 40% of companies use it as standalone.

By combining canary deployment with other techniques, organization can increase the strengths of each approach to achieve more robustness and flexibility in deployment.

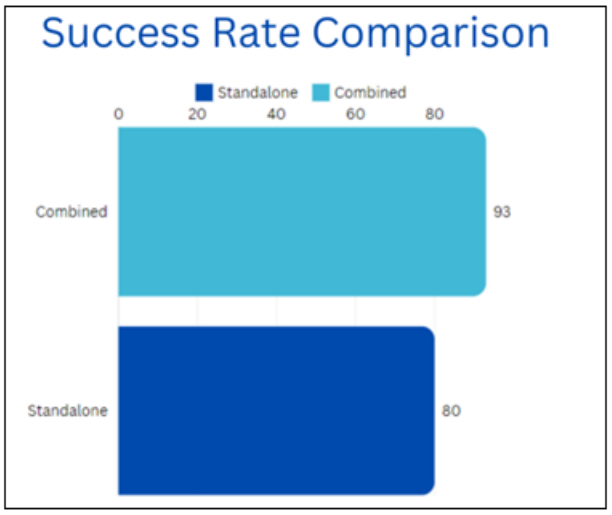


Figure 15: Bar Graph of Success Rate Comparison

From the above statistics it was found that the success rate when organizations use combined techniques and standalone canary technique differs. Success rate for combined (canary deployment with other techniques) is 93% and for standalone (canary deployment) is 80%. This may be because combined techniques offer additional layers of redundancy and error mitigation, leading to higher success rates compared to standalone canary deployments.

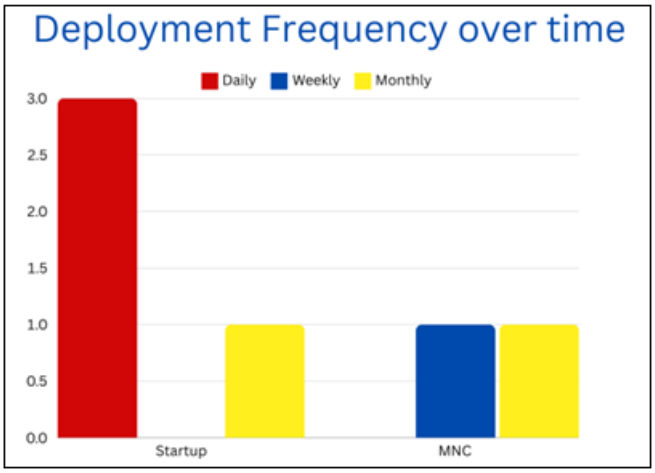


Figure 16: Bar Graph of Deployment Frequency over time

From the above statistics it was found that startups tend to deploy updates more frequently , while MNCs typically opt for less frequent deployments such as weekly or monthly. It may be because startups prioritize agility and rapid iteration and deploy updates frequently to stay competitive and responsive to market demands. In contrast to this, MNCs prioritize stability and risk mitigation, opting for less frequent deployments to minimize disruptions.

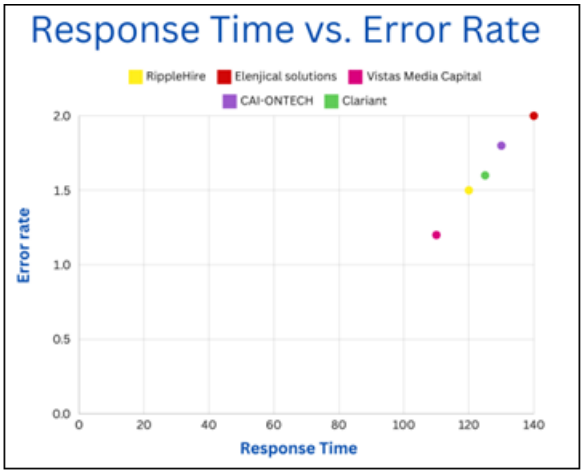


Figure 17: Scatter Plot for Response Time vs Error Rate

The above scatter plot illustrates relationship between response time and error rate for each company that took part in this survey. It's evident that response time have an inverse correlation with error rate. Companies with lower response times generally exhibit lower error rates, indicating a more efficient and stable system.

6. Findings

1) Most companies prefer blending canary deployment with other techniques, showing a growing tendency towards mixed deployment strategies.
2) When combined with other methods, canary deployment shows a higher success rate compared to its standalone counterpart, indicating better effectiveness through integration.
3) Looking at response time versus error rate, we can see a interesting relationship highlighting the need to streamline response times to minimize deployment errors.

4) Deployment frequency varies across different companies. Startups tend to deploy changes daily, while larger corporations opt for weekly or monthly cycles. This reflects diverse deployment habits influenced by company size and culture.

7. Conclusion

In conclusion, this study shows how canaries are used in different ways in the software industry. By looking at different deploying techniques, how canary deployment from others, how well this deployment work- what are the success rate, response time and error rate, and how often they are used that is the frequency of deploying the code or applications.

We've learned a lot about how companies handle this process. The findings highlights the importance of tailored deployment strategies, combining canary deployment with other techniques for effectiveness. As technology keeps evolving, its super important to know how to use it right and make sure everything works smoothly when we release new software.

References

[1] Netflix Tech Blog. (2018). Canary Analysis Service: Empowering continuous delivery at Netflix scale. Netflix Tech Blog.
[2] Kubernetes Documentation. (n.d.). Canary deployment. Retrieved May 14, 2024, from: https://kubernetes.io/docs/concepts/cluster-administration/manage-deployment/
[3] SoundCloud Engineering. (2016). Deployment: From Canary to 500-Node Swarm. SoundCloud Engineering Blog. Retrieved from https://developers.soundcloud.com/blog/deployment-from-canary-to-500-node-swarm
[4] Google Cloud Blog. (2018). How Google does canary releases. Google Cloud Blog. Retrieved from https://cloud.google.com/blog/products/management-tools/how google-does-canary-releases
[5] DevOps Institute. (n.d.). Retrieved May 14, 2024, from https://devopsinstitute.com/
[6] Kim, G., Humble, J., Debois, P., & Willis, J. (2016). The DevOps handbook: How to create world-class agility, reliability, & security in technology organizations. IT Revolution Press.