	Bansilal Ramnath Agarwal Charitable Trust's Vishwakarma Institute of Information Technology **Department of** **Artificial Intelligence and Data Science**

Name: Netal Prakash Daga		
Class: TY	Division: A	Roll No: 371016
Semester: V	Academic Year:2022-2023	
Subject Name & Code: Cloud Computing and Analytics		
Title of Assignment: Study and explore – case study with devops and site reliability engineering.		
Date of Performance: 15/11/2022	Date of Submission: 06/12/2022	

Aim: Study and explore – case study with devops and site reliability engineering.

Problem Statement: Study and explore – case study with devops and site reliability engineering.

Background Information:

1.] Site Reliability Engineering (SRE) :

Site Reliability Engineering or SRE is an exceptional, software-first approach to Information Technology operations supported by a group of analogous practices. The main focus of SRE is system reliability, which is considered the most basic feature of any product. Once the system is reliable enough then SRE starts to adding new features or producing new products. It also puts efforts for close attention on tracking results, making appreciable performance improvements, and automating operational tasks.

2.] DevOps :

The term DevOps is Development and Operations which was invented in 2009 by Patrick Debois. Nothing but a set of practices in between development and operations in the entire service life cycle. Its principles are the same as those of SRE:

- Application in engineering practices to operations tasks, evaluating results, and belief in automation instead of manual work. But its focus is much larger.
- It appeared in the first place as a culture and mindset that didn't specify how particularly to implement its ideas.
- It's often seen as an inference of main SRE methods so that they can be used by a larger range of organizations.

Similarities and Differences Between DevOps and SREs :

The primary difference between DevOps and SRE is DevOps is all about What needs to be done, SRE talks about how this can be done. Representing similarities and differences between the two:

1.] Reduce Organizational Silos:

Huge enterprises usually have a complex organizational structure, with a lot of teams working in silos. Each team is drawing the product in a different order, not communicating with the rest of the company and as a result, failed to see the big picture as a whole. This can lead to frustration, a setback in deployment, and high costs due to delays.

- DevOps' work is to diminish the silos and to clear that there are not any other teams who are not aligned with any other companies.

- SREs don’t think about how many silos are in the company, but more about how to get everyone into the discussion. This is done by using the same techniques and tools across the company, which helps in sharing ownership across everyone.

2.] Accept Failure as Normal:
DevOps hugs this by consenting to failure as something that is bound to happen, and which can help the teams to learn and grow. SREs do not want any errors or failures, even if it’s something that we can learn from it.

3.] Implement Gradual Change:
Companies want to work rapidly than they did before. They want recurrent releases, continually updating the product, and keeps team members active about new and relevant technology. DevOps are for changes but in a gradual and handled way. Both DevOps and SREs want to move quickly, and research shows that SREs makes lower the cost of failure in comparison to DevOps.

4.] Leverage Tooling and Automation:
As we know before, the primary point for both DevOps and SREs is automation. Tooling and Automation encourage including as much automation and tools as feasible, till they provide value to developers and operations by removing tasks manually.

5.] Measure Everything:
The primary difference here is that SREs roll around the concept of operations as a software problem, which led them to define prescriptive ways for measuring accessibility, toil, uptime, outages. It also ensures that everyone in the company agrees on how to measure dependability and what to do when availability falls out of specification.

Case Study:

The Catalyst

In 2013, the now-infamous data breach at Target exposed its technology organization to criticism and intense scrutiny. Target prioritized rebuilding its engineering culture; to champion excellence and value talent, attempting to evolve into a world-class, high-performing technology organization.

The resulting evaluation revealed challenges that are common across many IT organizations:

- Lack of a dynamic engineering culture (70% outsourced)
- Complex organizational structure (siloed engineering teams far removed from customers)
- Complex IT systems (massive technical debt)

TARGET

Enter DevOps

The DevOps journey at Target started with a small, grassroots team, supported by a couple of key change agents in leadership. Inspired by the Phoenix Project, the team in 2011 was focused on “automating everything” and implementing continuous integration for software development.

By 2014, when Target shifted focus to rebuilding its engineering culture, this team had already proven the value to be gained from DevOps, and were important contributors to the wider DevOps movement. They had recently held the first internal DevOps Days, aiming to build a community connecting the expanding grassroots efforts across internal silos to share and learn from each other.

A new CIO in 2015 brought additional leadership focus to improving cooperation between development and operational resources and striving for a more agile software development process. In order to scale from the grassroots initiatives into an engineering organization transformed by DevOps principles, Target invested heavily in training and workshops. In 2015, Target created "**The Dojo**". The Target Dojo coaches teams through 30-day challenges where they transition from traditional delivery to using DevOps and Agile principles. Teams enter the Dojo with processing times of three-to-six months to provision environments, and they leave capable of doing it in a few hours.

To provide flexibility in service delivery, Target created teams built around infrastructure components, with each team providing API access for other teams to consume components and stitch together environments. The teams have full-stack ownership for the API toolset, with operational resources playing a key part in the development process.

Outcomes: Target was successful in rebuilding its engineering culture, attracting engineers, and increasing innovation. They successfully shortened release cycles, improved application quality, and increased employee morale. Target considers its DevOps movement to intersect between the three values required for the seamless delivery of product:

- Collaboration
- Empathy
- Experiential Learning

There are now three Dojos (the immersive learning environment) across the world, 37 coaches, and 265 completed challenges (up from 14 in 2014). DevOps has transformed the entire culture at Target.

Recommendations:

- Align incentives across development and operational organizations
- Change the way you see and treat failure – fear of failure will stifle innovation
- Prioritize continuous learning – experimenting, testing, failing, and ultimately succeeding
- Shift to a "blameless culture"
- Focus on results and capture metrics, e.g. faster delivery, better quality, overall lower costs
- Build trust with peers across the organization through communication and transparency

GitHub Repo Link:

https://github.com/Netal1702/Cloud-Computing-Analysis

Conclusion:

As DevOps processes grow, there is new increasing demand for professionals with expertise in Key practices and tools. DevOps is not open up new opportunities for Operations personnel but also provides them with a logical career succession. There is a need of Site Reliability Engineering as a particular execution of DevOps.