Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2020, 7(3):337-342**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

DevOps for Database Administration: Best Practices and Case Studies

Balakrishna Boddu

Sr. Database Administrator
balakrishnasvkbs@gmail.com

Abstract: DevOps practices are increasingly being adopted in database administration to enhance efficiency, collaboration, and automation. The key best practices for implementing DevOps in database administration, include continuous integration, continuous delivery, infrastructure as code, and monitoring and alerting. Additionally, it presents real-world case studies highlighting successful DevOps implementations in database environments. By examining these best practices and case studies, database administrators can gain valuable insights into how to leverage DevOps to improve their workflows, reduce downtime, and deliver higher-quality database services.

Keywords: DevOps, Agile, SDLC, GitHub, Jenkins, ec2, rds, Iteration, Optimization, automation.

1. Introduction

The main purpose of DevOps is to deploy continuous software development processes such as continuous delivery and continuous deployment and microservices to support an agile software development lifecycle. Other trends in this context are that software is increasingly delivered through the internet, either server-side (SAS Service) or as a channel to deliver directly to the customer, and the increasingly pervasive mobile platforms and technologies on which this software runs. Database administration plays a critical role in supporting modern applications. However, traditional database management practices can often be time-consuming, error-prone, and hinder agility. DevOps provides a framework for automating and streamlining database tasks, fostering collaboration between database administrators and application developers, and ensuring that database infrastructure aligns with business needs. Organizations are constantly striving to improve their application delivery speed, reliability, and quality. DevOps, a cultural and methodological approach that promotes collaboration between development and operations teams, has emerged as a powerful tool for achieving these goals. While DevOps has traditionally been associated with application development, its principles and practices can also be applied to database administration.

We Will Address the following research questions from a Database Administration perspective in this Paper.

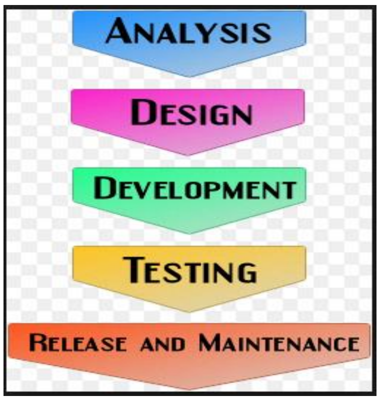
1. Why do We require DevOps for Database Administration?
2. What are the core areas of DevOps required for Database administration?
3. what are the benefits and challenges of using DevOps?

The rise of DevOps has transformed software development and deployment, emphasizing collaboration, automation, and continuous delivery. As databases play a critical role in modern applications, it is essential to integrate database administration (DBA) into DevOps practices. This paper explores the benefits of DevOps for DBA, including improved efficiency, reliability, and scalability. We will delve into key best practices for implementing DevOps in DBA, such as version control, automation, continuous integration/continuous delivery (CI/CD), and monitoring. Additionally, we will present case studies demonstrating the successful application of DevOps principles in real-world database environments.

2. Related Research

There are multiple areas of research required when it comes to Database administration with DevOps. Most of the organization are using agile methodology for their Project scope and improvements for the Software Development lifecycle.

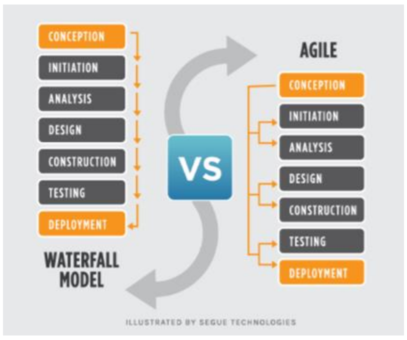
Different Stages of SDLC



Various Methods to Achieve SDLC:

Water Fall Method: Even though we have numerous requirements each requirement has to pass through its SDLC one by one. There might be delays due to system crashes, and differences in environment. Quality cannot be assured until it is implemented and tested in production.

Agile Method: Each requirement is divided into multiple frames and executed. Quality cannot be assured.



It's a combination of tools that increases the ability of an organization to deliver a project in a faster way with quality assurance by making continuous Development/Integration/Testing. If something happens in any software life cycle phase we can easily identify/restore as we need. We have several tools that can be accomplished in each SDLC phase.

Central Version Control System:

The repository will be available in a central location and users have to connect via the network and make changes there.

Distributed Version Control System:

Users can import/have their local database from the Central repo make their changes locally and commit the changes again to the central repo.

Ex: GIT

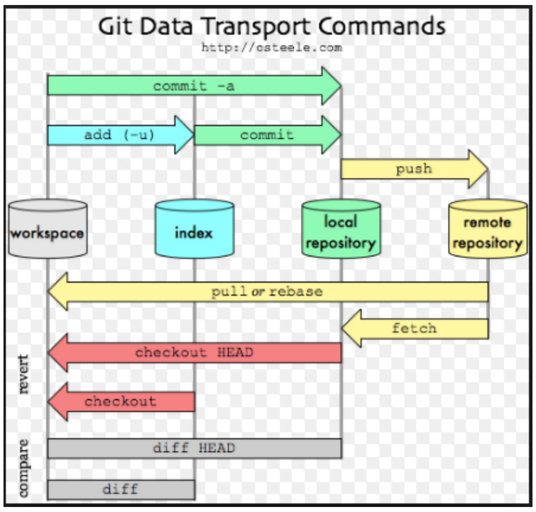
Git & Git Hub:

Git is a version control system also called a distributed version control system should be installed in the client. Which will track the history of all the files in a repository.

Every individual user has his local repo apart from the central repo. Whenever a developer commits any change to a file/folder, Git creates a new version for that file. If the user/developer has any issues with the latest version of the code then the previous/required version can be easily restored.

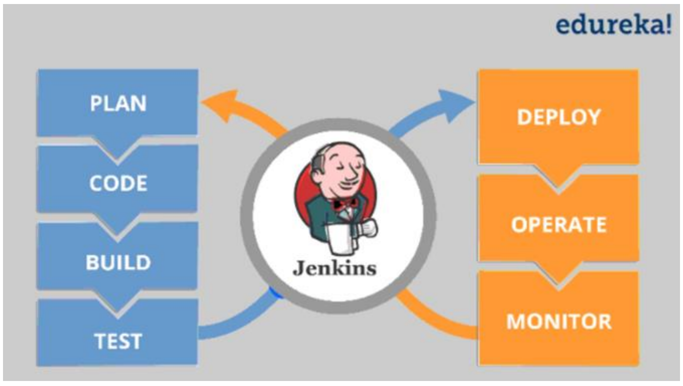
Git Hub is a repository system that is available at https://github.com and users should be able to sign up with their official email/Gmail.

How Git Works:



Jenkins:

Is a Java Application and it is used for continuous integration and continuous delivery. Jenkins by default won't come with any features it can be just a skeleton. Any feature will be available as plugins, so only required plugins can be installed.

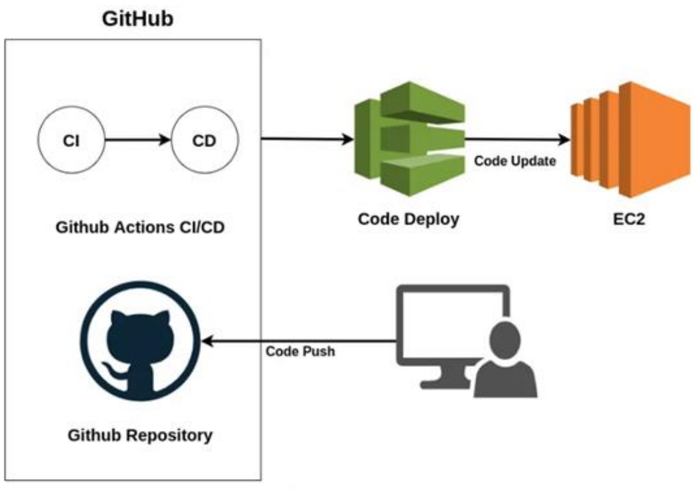


Continuous integration and continuous delivery:

In general Database Administrators and Developers Write the code in their machines and push the changes into shared repositories like Git/Bit Bucket/SVN and at the end of the day if the build starts and encounters any bug it would be difficult to find out at what point of code the issue occurred which may lead to delay in delivery.

In the case of Jenkins, it would trigger the build as soon as the developer checked in any change into the repository and if any bug is introduced users will be notified with an exact report at what point of code the issue started.

Example Process how Database administrator can create EC2 instance with CICD Process:



Overview of CI/CD using CodeDeploy and GitHub Actions

When the User pushes the code that request will go to the GitHub Repository will grab the existing code and Connect to AWSCli The code will Deploy to create an EC2 Machine. So, with this process, we can create N Number of Server with our Manual Work and It will save lots of hours for an organization. The same process will be applicable to create and maintain database changes with our Human intervention.

3. Research Framework

Many executives are excited about DevOps, but they often don't fully understand what it is or how it works. This can lead to unrealistic expectations and disappointment.

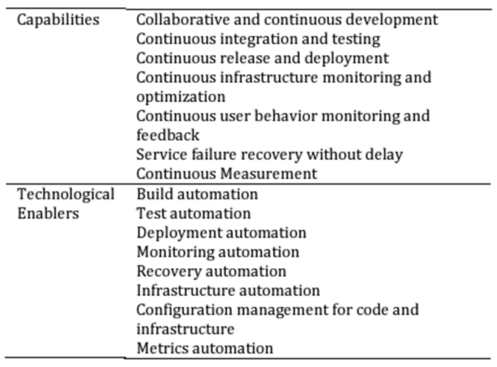
To make DevOps successful, companies need to use a platform that can track and measure performance. This data helps them see if DevOps is working and make improvements.

The real value of DevOps isn't just using a trendy term; it's about having a common way of talking about and improving software quality and delivery

DevOps Roles in Critical Database Administration World:

- Responsible for all kinds of UNIX\Windows administration tasks.
- Virtualize the servers using Cloud (eg: Azure\aws) for test and dev, Prod environments.
- Write Ansible playbooks/roles to manage the configuration in different environments.
- Deploy Docker Engines in Virtualized Platforms for containerization of multiple applications.
- Setting up monitoring tools like Nagios and Amazon Cloud watch to monitor major metrics like Network packets, CPU utilization, and Load Balancer Latency.
- writing Ansible scripts and heavy Shell, Perl, Python, and JSON scripting.
- Deployed and configured GIT repositories with branching, forks, tagging, merge requests, and notifications.
- Coordinate/assist developers with establishing and applying appropriate branching, Labelling/naming conventions using GIT source control.
- Automate the backup/project builds jobs using Jenkins.
- Maintain Servers\Database availability as per SLA. Produce monthly reports for critical servers.
- Assist different teams in all phases of builds. Ensure the documentation when there is a change in any environment.

We Will Achieve (diagram) these many features when We integrate databases into DevOps with the help of Administration.



Capabilities	Collaborative and continuous development Continuous integration and testing Continuous release and deployment Continuous infrastructure monitoring and optimization Continuous user behavior monitoring and feedback Service failure recovery without delay Continuous Measurement
Technological Enablers	Build automation Test automation Deployment automation Monitoring automation Recovery automation Infrastructure automation Configuration management for code and infrastructure Metrics automation

Diagram: Core DevOps Services for Databases

4. Findings

To be agile and keep customers happy, DevOps teams need four important skills. One of these skills is observability. This means every part of the DevOps process should be tracked and measured. Without this data, it's impossible to understand and fix problems.

• **Iteration:** Code fixes and improvements must be rapidly identified, triaged, and developed using data correlated throughout the toolchain to provide deeper insights.

• **Collaboration:** Rapid delivery requires that DevOps teams are on the same page, use the same data, and take action based on the same measurements.

• **Optimization:** Use statistical methods to analyze quantitative data (e.g., performance metrics, cost savings).

Using data to improve DevOps and observability can directly benefit a business. It can make things more efficient, help developers work faster, release apps faster, save money, make customers happier, and increase revenue.

Companies that use a full observability platform can expect to:

• See problems with new apps right away, not hours or days later.

• Fix problems much faster, both after they happen and before they even start.

• Work more efficiently because data is collected and analyzed automatically. This lets developers and operations teams focus on what the business needs, not on building tools.

To make sure DevOps teams are working towards the business goals, they need to release new things often and measure how that affects the business.

5.Devops in Best Practice

When implementing DevOps, several challenges slowed down the process. These challenges made it harder to achieve the goals of DevOps. One big challenge was having employees with the right skills. This meant hiring new people with the right skills or teaching current employee's new things.

Not having enough skilled employees can slow down DevOps because the needed skills might not be available when needed. The skills needed include knowing how to write software, understanding infrastructure, and using the right tools.

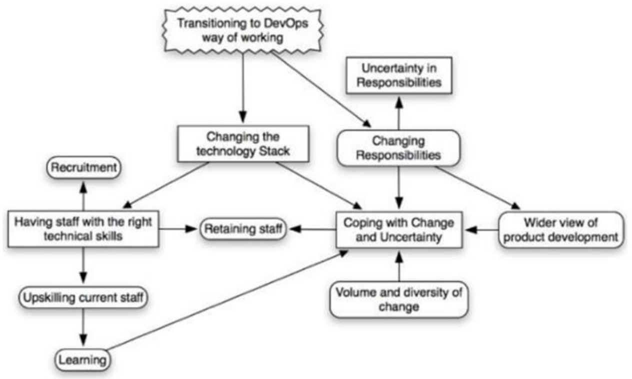

Transitioning to DevOps way of working
Uncertainty in Responsibilities
Changing the technology Stack
Changing Responsibilities
Recruitment
Having staff with the right technical skills
Retaining staff
Coping with Change and Uncertainty
Wider view of product development
Upskilling current staff
Volume and diversity of change
Learning

Diagram: Challenges related to DevOps Adoption

Additional best practices consistent with the DevOps framework:

1. Make small, frequent changes to the database code. This makes it easier to undo changes and find problems early in the development process.
2. Monitor and manage dependencies after each change. Use a microservices approach for database development.
3. Use a fast feedback loop, like you do with application code. However, important feedback might be hidden among all the logs and alerts.
4. Track all changes to the database code. Test often and focus on metrics that are important to the business and user experience.
5. Set up a testing environment that matches real-world use cases. Use a production environment to test how the database works.
6. Automate as much as possible. Find tasks that are repeated and predictable and write scripts to update the database code when new builds are created.
7.

6. Conclusion

Using DevOps practices in database management can greatly improve efficiency, reliability, and quality. By implementing methods like continuous integration, continuous delivery, infrastructure as code, and strong monitoring and alerting, organizations can simplify their processes, minimize downtime, and adapt more quickly to changes. The examples shared show the real benefits of these practices, such as quicker deployments, better teamwork, and more robust databases. As databases keep evolving, adopting DevOps will be crucial for staying competitive and providing effective database services.

References

[1]. **"Database DevOps with Liquibase"** by Brian Lane

[2]. **"DevOps for Dummies"** by Emily Freeman and Jez Humble

[3]. **"High-Performance MySQL"** by Baron Schwartz, Peter Z (translated title: High-Performance MySQL by Baron Schwartz, Peter Z)

[4]. **Liquibase Blog:** https://www.liquibase.com/blog

[5]. **Redgate Blog:** https://www.red-gate.com/solutions/overview

[6]. **Daffodil Software Blog:** https://www.daffodilsw.com/devops-services/

[7]. **DevOps, DBAs, and DBaaS -** This book discusses how DBAs manage data platforms and change requests in a DevOps environment, supporting continuous integration, delivery, testing, and deployment(https://link.springer.com/book/10.1007/978-1-4842-2208-9)

[8]. **DevOps Best Practices for Database -** This article outlines nine best practices to improve software quality and speed up release cycles. (https://www.atatus.com/blog/devops-best-practices-for-database/)

[9]. **A Case Study of a Successful DevOps Implementation -** This case study highlights the challenges, strategies, and benefits of implementing DevOps in an organization. (https://identicalcloud.com/blog/a-case-study-of-a-successful-devops-implementation/)

[10]. **A Guide to Database DevOps -** This guide provides best practices for managing databases using DevOps protocols. (https://www.liquibase.com/resources/guides/database-devops)

[11]. **DevOps for Databases -** This article walks through an example of how DevOps can simplify database management. (https://stackify.com/devops-for-databases/)