International Journal of Management, IT & Engineering
Vol. 15 Issue 11, November 2025,
ISSN: 2249-0558 Impact Factor: 7.119
Journal Homepage: http://www.ijmra.us, Email: editorijmie@gmail.com
Double-Blind Peer Reviewed Refereed Open Access International Journal - Included in the International Serial Directories Indexed & Listed at: Ulrich's Periodicals Directory ©, U.S.A., Open J-Gate as well as in Cabell's Directories of Publishing Opportunities, U.S.A

DevOps Frameworks in the Context of Salesforce Platform Delivery

Pritha Das Mazumdar*

Abstract

Organizations are recognizing the need for more disciplined, scalable and automated approaches to managing change as cloud-based platforms like Salesforce are becoming increasingly central to the enterprise landscape. However, applying DevOps principles to a low-code, metadata driven environment like Salesforce presents its own unique challenges. This paper explores the rationale behind the rise of DevOps, how organizations structure their sandbox to production strategies and adopt DevOps tooling like Azure DevOps or Flosum. The paper explores the rationale behind the rise of DevOps, how to structure the sandbox to production strategies while adopting DevOps tooling like Azure DevOps and Flosum

Keywords:

Salesforce DevOps;
Continuous Integration;
Continuous Deployment;
Azure DevOps;
Flosum.

Author correspondence:

Pritha Das Mazumdar,
Senior Manager – Salesforce Practice
PricewaterhouseCoopers Advisory Services LLC, Atlanta, USA
Email: pritha_dasmazumdar@yahoo.com

1. Introduction: What is DevOps

DevOps is a set of tools and practices that helps organizations build, test and deploy software more reliably and at a faster pace. It enables organizations to evolve and deliver their products more quickly than those with traditional development and release cycle, which can provide a competitive edge. With this, instead of pushing out a release once a fortnight or longer, new features can be delivered to end users daily and bug fixes can be turned around in a matter of hours, leveraging Continuous Integration/Continuous Deployment (CI/CD) pipelines.

It also fosters a cultural collaboration between software developers and enterprise IT operations. Developers want agility i.e. the ability to release new features and bug fixes quickly with little friction to provide value to users as soon as possible, while Operations teams are laser focused on stability ensuring systems and applications running on software are up and running with no failures or downtime. DevOps helps meet both these needs – it provides agility for developers, allowing multiple releases or rollouts per day, week or a month depending on the appetite of the organization and at the same time, automated testing and manual reviews at each stage of the build process assures the operations team that

* Doctorate Program, Linguistics Program Studies, Udayana University Denpasar, Bali-Indonesia (9 pt)

defective builds are failed well before the end up in the live production environment. Finally, there are logs and metrics collected at each stage of the process, providing more visibility to developers and the operations teams alike.

2. Significance of DevOps in the context of Salesforce

Salesforce, as a leading cloud-based Customer Relationship Management (CRM) platform, offers extensive customization and configuration capabilities to cater to diverse business needs. However, the complexity and continuous evolution of Salesforce applications requires a robust approach to manage changes, deployments and integrations efficiently. This is where Salesforce DevOps comes into play, addressing a few key challenges listed below:

- **Metadata Management**: Salesforce metadata is complex and massive, making it difficult to manage a CI/CD setup. The solution lies in carefully tracked changes, deciding which metadata should be managed in Source Version Control and leveraging that as the source of truth for deployments.

- **Deployment Complexity**: Salesforce environments can be complex, with dependencies across various components and metadata types. DevOps practices help manage this complexity through automated deployment pipelines.

- **Quality Assurance**: Ensuring high-quality releases with minimal disruptions is crucial. Static Code Analysis with CI/CD pipeline and automated regression testing help maintain reliability and performance of the application.

- **Collaboration and Transparency**: Salesforce projects often have cross-functional teams staffed and working towards the release. DevOps fosters a culture of transparency and shared responsibility, enhancing collaboration and visibility across the development lifecycle.

3. Demonstration of Continuous Integration and Continuous Deployment with stages

Continuous Integration and Continuous Deployment (CI/CD) sits at the heart of DevOps. This entails integrated processes to automate, build, test and deployment

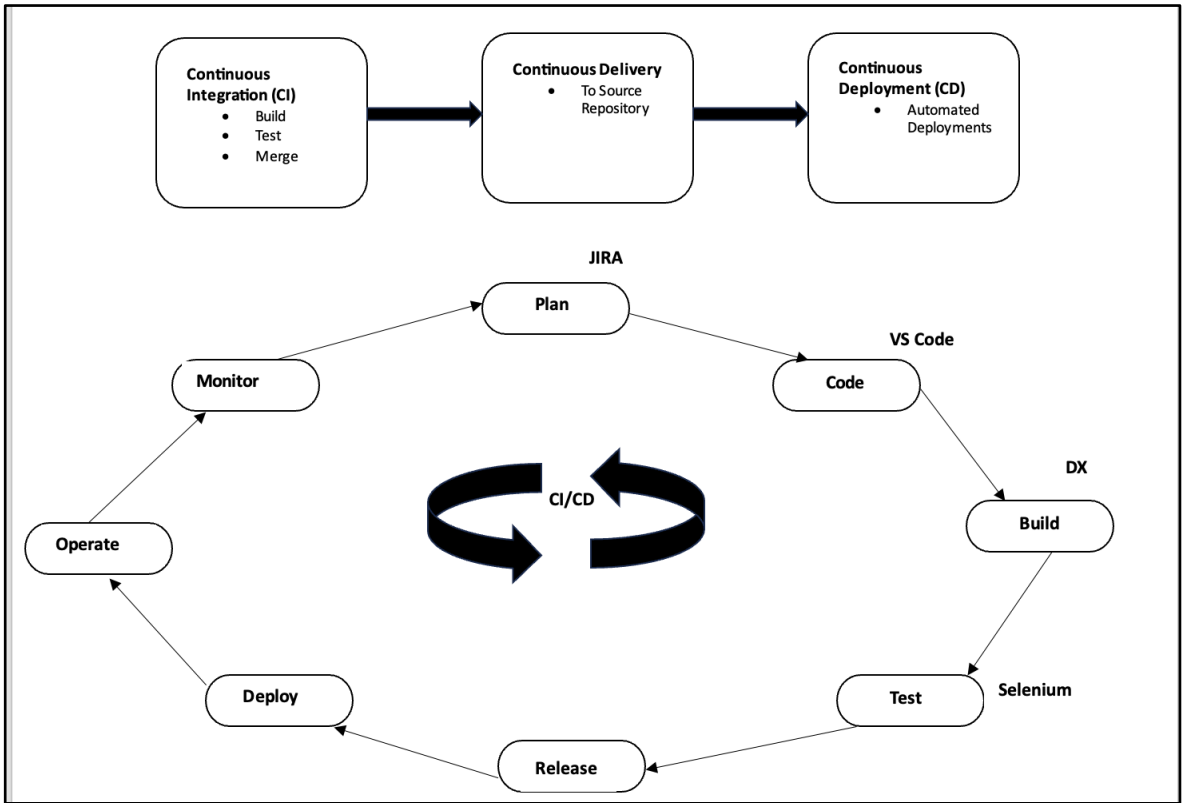

Figure 1. *CI/CD Lifecycle in ALM*

Continuous Integration (CI)

- In the Build phase, a compilation of the application takes place – this is where the build is validated in accordance with the organizational compliance requirements.
- In the Test phase, the code is tested automatically before delivered to production improving collaboration and quality.

Continuous Delivery

- Continuous Delivery takes the applications and delivers them to selected infrastructures.

Continuous Deployment (CD)

- In this phase, the application is deployed to the intended environment(s) / platform(s).

4. Recommended Branching Strategy in Salesforce Projects

A branching strategy drives how the development team uses Version Control like Git branches to manage new features, bugs, hotfixes and releases.

Models commonly seen in Salesforce projects:

- **Feature Branching Model**: Each new feature or user story is developed in its own Feature branch
- **Release Branching Model**: Used for preparing a release, stabilizing code and fixing defects in higher environments(sandboxes)
- **Integrated Branching Model**: Used for having features merged and integrated for functional testing
- **Hotfix Branching Model**: This is branched directly from production and is used predominantly for urgent production fixes to address showstopper defects impacting the application in production

4. Recommended Environment Strategy in Salesforce Projects

An environment strategy drives how different salesforce orgs(sandboxes of different types or scratch orgs or production) are used to support the software development lifecycle (SDLC)

Commonly seen Salesforce environments in a project could look like the following:

- **Developer Sandboxes/Scratch orgs**: Individual environments for building and unit testing used by developers on the team
- **Integrated Sandbox**: Environment where features from developer orgs or scratch orgs are integrated and tested together, used by the testers on the team
- **SIT/UAT Sandbox**: Environments dedicated specifically for system integration testing or business user acceptance testing
- **Staging/Pre-Prod**: Environment for final validation of the code package that mirrors the Production environment
- **Production**: Live customer environment

5. Significance of applying correct Branching and Environment Strategies in Enterprise IT Projects

It is important to align the Salesforce environment to the dedicated right branch. As an example, a developer sandbox should be aligned to a Feature Branch, while a UAT sandbox is best aligned to a Release branch. This strategy is more full proof when there's a mature DevOps tool to automate deployments moving metadata consistently. Below are the reasons why its important to have the strategy built out:

- **Continuous Parallel Development**: Often Salesforce projects have multiple developers/teams working on different features. It is important to ensure that they can maintain synergy without stepping or overwriting each other's changes.

- **Reduced Risk:** Having a combination of Hotfix and Pre-Prod environment allows quick fixes to be deployed without destabilizing or disrupting ongoing development

- **Controlled Releases**: Features or changes move predictably following a standard release path Dev -> Integrated -> SIT/UAT -> Production, aligned to the right protocols

- **Audit & Traceability**: Provides a clear trace of what has been built, tested, approved and deployed

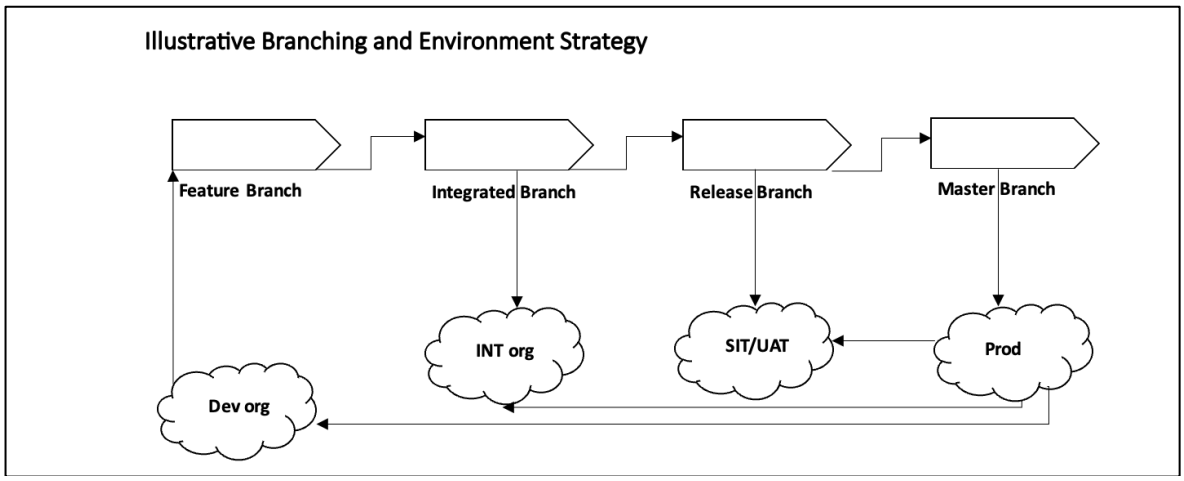

Figure 2. *Branching & Environment Strategy*

6. Comparing Azure DevOps and Flosum across different Implementation Dimensions

This section provides a comparative evaluation of Azure DevOps and Flosum in the context of Salesforce implementation lifecycle management. While both platforms enable version control, CI/CD automation, and release governance, their underlying architectures and integration patterns differ significantly. Azure DevOps, as a broad enterprise DevOps platform, offers flexibility through customizable pipelines and integrations with diverse toolchains, making it suitable for organizations with hybrid technology stacks. In contrast, Flosum delivers a Salesforce-native solution purpose-built for metadata deployments, compliance, and auditability within the Salesforce ecosystem. Below is a practical comparison of their relative capabilities

Capability Dimension	Flosum	Azure DevOps
Native capabilities, readiness for Salesforce	More suitable	Less suitable
Customizations Supported	More suitable	Less suitable
Built in Data Backup Options	More suitable	Less suitable
Initial Cost of Implementations/Timeline	Less suitable	More suitable
License & Maintenance Cost	Less suitable	More suitable
Ease of use	More suitable	Less suitable
Automated code coverage check	More suitable	Less suitable
Ability to integrate with other tools	Less suitable	More suitable

7. Conclusion

To summarize, DevOps is a critical enabler for speed, quality and governance in enterprise delivery. A well defined branching and environment strategy combined with the right DevOps tooling secures these outcomes. Both Flosum and Azure DevOps bring unique strengths – while Flosum provides Salesforce native simplicity, Azure DevOps offers broader enterprise integration and scalability. It is important to keep in mind that success is less about the tool but more about how effectively DevOps strategy and best practices are embedded into the Salesforce delivery lifecycle.

References

[1] Koppanathi, S. R. (2023). *"Salesforce DevOps Strategies"*. European Journal of Advances in Engineering and Technology, 10(3), 75-81. Available: https://www.ejaet.com/PDF/10-3/EJAET-10-3-75-81.pdf

[2] Patel, A. K. (2024). *"Streamlining Development: Best Practices for Salesforce DevOps and Continuous Integration"*. Journal of Mathematical & Computer Applications. ISSN: 2754-6705. Available: https://onlinescientificresearch.com/articles/streamlining-development-best-practices-for-salesforce-devops-and-continuous-integration.pdf

[3] "Salesforce Application Lifecycle Management (for ISVs)." Flosum Blog. https://www.flosum.com/blog/salesforce-application-lifecycle-management-for-isvs.

[4] How to build a Salesforce CI/CD pipeline using Azure DevOps. Gearset Blog. https://gearset.com/blog/how-to-build-a-salesforce-release-pipeline-with-azure-devops/