



Transforming Salesforce Commerce Cloud: DevOps Implementation for Scalable Solutions

Rishitha Kokku

Senior Software Engineer
Optum Services Inc.

Chirag Pethad

Platform Owner Salesforce Service Cloud
PetSmart Inc.

Abstract: In today's fast-paced e-commerce landscape, agility, efficiency, and seamless customer experiences are paramount. Salesforce Commerce Cloud (SFCC) stands out as a leading platform empowering businesses to deliver personalized and scalable online shopping experiences. Integrating DevOps practices into SFCC implementations can further enhance development workflows, streamline deployments, and ensure high-quality, reliable e-commerce solutions.

This white paper delves into the strategic implementation of DevOps for Salesforce Commerce Cloud, outlining the benefits, challenges, and a detailed step-by-step guide to adopting DevOps methodologies. By leveraging automation, continuous integration, and collaboration, organizations can significantly improve their SFCC projects' speed, quality, and scalability.

Keywords: Commerce Cloud, DevOps, CI/CD, Certificates, Pipeline, Account Manager

Introduction

As e-commerce continues to evolve, businesses must adopt robust methodologies to stay competitive. DevOps, a blend of development and operations practices, has emerged as a critical framework for enhancing software delivery and operational efficiency. When applied to Salesforce Commerce Cloud (SFCC), DevOps can transform how e-commerce solutions are developed, deployed, and maintained. Implementing DevOps for a Salesforce e-commerce project can streamline development, enhance collaboration, and accelerate delivery cycles.

This whitepaper provides a comprehensive overview of implementing DevOps for SFCC, offering insights into the benefits, addressing potential challenges, and presenting a structured approach to integration. Whether you're a retailer aiming to enhance your online presence or a development team seeking to optimize your SFCC workflows, this guide serves as a valuable resource.

Overview of DevOps

DevOps is a combination of Development and Operations that aims at simplifying the delivery process and improving the quality of the product. The main goal of DevOps is to automate processes in development, testing, deployment to increase efficiency, quality and reliability of the software releases. DevOps principles are integrated by Continuous Integration, Continuous Delivery and Infrastructure as code. In an SDLC, DevOps implementation is crucial to streamline the development and deployment process and to close any

gaps between software development and IT operations. In Salesforce Commerce Cloud (SFCC), managing the E-Commerce platform is tricky and complicated and it requires a focus-driven DevOps environment to streamline the operations and ensure quality.

Overview of Salesforce Commerce Cloud (SFCC)

Salesforce Commerce Cloud or SFCC is a cloud-based e-commerce platform designed to help businesses create seamless online shopping experiences and also provides tools to manage online storefronts, handle orders, manage product catalogs, and personalize customer experience. It is part of Salesforce's Customer 360 suite that offers features like AI-powered personalization, mobile-first design, multi-channel selling, and integration with other Salesforce products, enabling businesses to deliver a unified and efficient customer journey. Commerce Cloud is suitable for both B2C (Business-to-Consumer) and B2B (Business-to-Business) commerce solutions.

Some of the Key features of Salesforce Commerce Cloud (SFCC) are:

1. **Multi-Channel Support:** SFCC enables businesses to manage and sell products across multiple channels, providing a unified customer experience.
2. **Customizations and Flexibility:** The platform offers tools for customizing storefronts and managing product catalogs, promotions, pricing, and more, enabling businesses to tailor the shopping experience to their brand and customer needs.
3. **Personalization:** With built-in AI powered tools, SFCC enables personalized product recommendations, targeted marketing, and customized user journeys based on customer behavior and preferences.
4. **Scalability:** SFCC can scale to handle high traffic volumes and transactions, making it ideal for both small retailers and large enterprises.
5. **Cloud-Based:** SFCC eliminates the need for on-premise infrastructure, providing automatic updates, high availability, and reduced maintenance efforts.
6. **Integration with Salesforce Ecosystem:** SFCC seamlessly integrates with Salesforce's CRM, Marketing Cloud, Service Cloud, and other Cloud offerings.
7. **Global Support:** It supports multiple languages, currencies, and payment methods, enabling businesses to operate globally and offer localized experiences for customers in different regions.

Benefits of DevOps for Salesforce Commerce Cloud (SFCC)

Implementing DevOps practices within Salesforce Commerce Cloud (SFCC) environments offers a multitude of benefits that enhance both the development and operational aspects of e-commerce solutions. By integrating DevOps, organizations can achieve greater efficiency, reliability, and agility, ultimately delivering superior shopping experiences to their customers. Below are the key benefits of adopting DevOps for SFCC:

1. Faster Time-to-Market

Accelerated Development Cycles

- **Continuous Integration and Continuous Deployment (CI/CD):** Automating the build, testing, and deployment processes significantly reduces the time required to release new features and updates.
- **Frequent Releases:** Enables organizations to deploy updates multiple times a day or week, ensuring that new functionalities reach customers swiftly.

Rapid Feedback Loops

- **Short Iterations:** Quick feedback from automated tests and monitoring tools allows for immediate identification and resolution of issues, speeding up the overall development process.

2. Enhanced Collaboration and Communication

Cross-Functional Teams

- **Unified Goals:** DevOps fosters a culture where development, operations, QA, and other teams work towards common objectives, improving overall project alignment.
- **Shared Responsibilities:** Teams collaborate more effectively, sharing ownership of the entire application lifecycle from development to production.

Improved Transparency

- **Visibility:** Tools and practices associated with DevOps provide greater visibility into processes, enabling all stakeholders to stay informed about project statuses and changes.

3. Increased Deployment Frequency and Reliability

Automated Deployments

- **Consistency:** Automated deployment pipelines ensure that deployments are consistent across different environments, reducing the risk of human error.
- **Scalability:** Easily scale deployments to handle varying traffic loads, especially important during peak shopping periods.

Reliable Releases

- **Rollback Mechanisms:** Automated rollback strategies allow for quick recovery in case of deployment failures, maintaining system stability.
- **Blue-Green Deployments:** Minimize downtime and reduce deployment risks by running two identical production environments.

4. Improved Quality and Reduced Errors

Automated Testing

- **Comprehensive Test Coverage:** Integration of automated unit, integration, and end-to-end tests ensures that code changes are thoroughly vetted before deployment.
- **Early Bug Detection:** Identifying and fixing issues early in the development cycle reduces the cost and effort associated with post-release bug fixes.

Code Quality

- **Code Reviews and Static Analysis:** Incorporating peer reviews and automated code analysis tools enhances code quality and maintainability.

5. Enhanced Security and Compliance (DevSecOps)

Integrated Security Practices

- **Shift-Left Security:** Embedding security checks early in the development process helps identify vulnerabilities before they reach production.
- **Continuous Monitoring:** Real-time monitoring and automated security scans ensure ongoing protection against threats.

Compliance Adherence

- **Automated Audits:** Streamlined processes for compliance checks make it easier to adhere to regulations such as GDPR, PCI DSS, and CCPA.
- **Secure Configurations:** Infrastructure as Code (IaC) ensures that security configurations are consistently applied across all environments.

6. Greater Scalability and Flexibility

Infrastructure as Code (IaC)

- **Automated Provisioning:** Manage and scale infrastructure resources programmatically, allowing for rapid adjustments based on demand.

- **Consistent Environments:** Ensure that development, testing, and production environments are identical, minimizing environment-specific issues.

Modular Architecture

- **Microservices and APIs:** DevOps practices support the development and deployment of microservices and APIs, enabling more flexible and scalable architectures.

7. Cost Efficiency

Resource Optimization

- **Automated Processes:** Reduce manual intervention and associated labor costs through automation of repetitive tasks.
- **Efficient Resource Utilization:** Optimize infrastructure usage, minimizing waste and lowering operational costs.

Reduced Downtime

- **Proactive Monitoring:** Early detection and resolution of issues prevent costly downtimes and maintain revenue streams, especially critical for e-commerce platforms.

8. Enhanced Customer Experience

Faster Feature Delivery

- **Continuous Updates:** Regularly delivering new features and improvements keeps the storefront competitive and responsive to customer needs.
- **Personalization and Optimization:** Quickly implement AI-driven personalization and other enhancements to improve the shopping experience.

Reliability and Performance

- **High Availability:** Ensuring system stability and uptime leads to a seamless shopping experience, fostering customer trust and loyalty.
- **Performance Monitoring:** Continuous monitoring helps maintain optimal performance, reducing page load times and enhancing user satisfaction.

9. Better Risk Management

Incremental Changes

- **Smaller Deployments:** Deploying changes in smaller increments reduces the risk associated with large-scale updates.
- **Continuous Monitoring:** Ongoing oversight allows for the immediate identification and mitigation of potential risks.

Disaster Recovery

- **Automated Backups and Rollbacks:** Ensure that data is consistently backed up and that systems can be quickly restored in case of failures.

10. Continuous Improvement and Innovation

Data-Driven Decisions

- **Analytics and Insights:** Leveraging monitoring tools and user feedback to inform continuous improvements and drive innovation.
- **Iterative Enhancements:** Regularly refining processes and technologies to adapt to changing business needs and technological advancements.

Encouraging Experimentation

- **Safe Testing Environments:** DevOps practices provide safe spaces for experimentation without risking production stability, fostering a culture of innovation.

Step by Step Implementation of SFCC pipelines

Jenkins is a widely used CICD tool that can automate the deployment and testing cycles. SFCC's unique architecture requires a detailed Jenkins process to migrate the changes. Commerce Cloud has cartridges that are being developed that need to be compiled and deployed to the destination environment. Jenkins does the compilation, building and uploading these packages into the SFCC environments. Here is a detailed procedural guide to build the Jenkins jobs for E-Commerce deployments.

1. Pre-requisites

- Jenkins installation.
- Node.js installed and configured under "Manage Jenkins".
- Github and Jenkins pipeline plugins installed.
- SFCC account to access the environments.
- Pipeline plugin to create the pipelines for deployments.
- CLI plugin to execute the commands.

2. Certificate Creation

A key step in deploying the code to Dev, CICD and Staging environments is to create certificates for authentication. These Certificates will be used to access the environments via the CICD tools.

- **Generate public and private keys:** Below command can be used to create the keys. The private key stays on your local machine while the public key is uploaded to the Salesforce Commerce Cloud Account.

```
ssh-keygen -t rsa -b 2048 -m PEM -f sfcc_deployment_key
```

- **Upload the public key to Salesforce Account Manager:** This step is required to allow Salesforce to authenticate the deployments. Login to Account Manager --> API Clients --> Create a new API client for each environment --> Upload the public key. Assign the appropriate permissions to the client for deployments, activations and data imports.
- **Setup OAuth2 Authentication:** The sfcc-ci tool with OAuth2 authentication will be used based on the Client ID and Client secret.

```
npm install -g sfcc-ci
```

- Configure the sfcc-ci tool with the Client ID, Client secret and Private key.
- Run the below command to login

```
sfcc-ci client:auth --client-id <YOUR_CLIENT_ID> \
                  --client-secret <YOUR_CLIENT_SECRET> \
                  --key <PATH_TO_PRIVATE_KEY>
```

3. Pipeline Creation

This pipeline will be used to build, compile and test for the SFCC platform. This pipeline uses Jenkinsfile located in the root directory of the repository. Below are the stages that will be executed as part of the pipeline.

- **Pull SCM from Version Control:** The Jenkins pipeline will pull the latest source code from the repository that contains the storefront code for SFCC.

- **Build Code:** Node.js tools will be required to build the code pulled from the above step. This step ensures the sanity of the pipeline.
- **Deploy to SFCC Environment:** SFCC CLI commands will be used to automate the deployment process. This will be the next stage in the pipeline. This stage authenticates into the SFCC environment using the client ID and Secret created on the WebDav, deploys the code into the sandbox instance and activates the version mentioned.

```

pipeline {
  agent any
  environment {
    SFCC_HOST = 'sfcc-hostname' |
    SFCC_USER = 'sfcc-username'
    SFCC_PASSWORD = 'sfcc-password'
  }
  stages {
    stage('Code SCM') {
      steps {
        git branch: 'main', url: 'git@github.com:your-repo.git'
      }
    }
    stage('Install Dependencies') {
      steps {
        sh 'npm install'
      }
    }
    stage('Build') {
      steps {
        sh 'npm run build' // Custom build script
      }
    }
    stage('Deploy to SFCC') {
      steps {
        script {
          sh 'curl --user $SFCC_USER:$SFCC_PASSWORD \
            --upload-file ./cartridges path-to-your-SFCC-webdav'
        }
      }
    }
  }
}

```

- **Jenkins Build execution:** This pipeline can be replicated to create multiple jobs to other environments and this is considered as a best practice to deploy. Another approach to deploying to multiple environments can be achieved if the above stage is replicated for deployments.

Challenges of Implementation

Implementing DevOps for Salesforce Commerce Cloud (SFCC) can bring significant benefits in terms of automation, efficiency, and collaboration, but it also comes with its own set of challenges. These challenges stem from the unique nature of Salesforce Commerce Cloud as a platform, its architecture, and the integration of DevOps tools and practices. Here are the key challenges to be aware of:

1. Limited Native DevOps Support

SFCC does not inherently provide strong built-in support for DevOps practices. It doesn't come with native CI/CD tools or easy infrastructure management options which creates a challenge in setting up automated deployment pipelines and version control systems.

2. Complex Development Lifecycle

SFCC uses Digital and Storefront Reference Architecture (SFRA), which can be complex to manage. Teams often need to work across multiple instances such as staging, production, and development sandboxes, which can make synchronization difficult and time consuming.

3. Customizations and Integrations

SFCC is highly customizable, allowing for tailored solutions. These customizations can complicate DevOps implementations. Integrating third-party systems, plugins, or custom code into automated pipelines requires additional configuration which becomes difficult to maintain.

4. Continuous Integration (CI) and Continuous Deployment (CD) Gaps

There are often gaps in ensuring seamless deployment workflows, particularly with data models, custom configurations, and sandbox management in SFCC. This requires more manual intervention compared to other platforms.

5. Sandbox and Environment Management

SFCC environments (such as staging and production) can have different configurations, making it challenging to ensure consistency between them. Each instance may also require manual setup, making the process prone to errors.

6. Data Management in DevOps

Synchronizing data (catalogs, pricing, products) and ensuring data consistency across all environments (development, staging, and production) without affecting live data can be difficult. SFCC typically has different configurations and data sets across environments, which could cause issues when promoting changes.

7. Testing and Quality Assurance (QA)

Automating tests for SFCC is challenging, especially for custom storefronts and third-party integrations. Testing is crucial but automating end-to-end and regression tests for complex and custom e-commerce workflows can be resource-intensive.

8. Security and Compliance Integration (DevSecOps)

Ensuring security in a DevOps pipeline (often called DevSecOps) requires integrating security checks early in the development process. SFCC is subject to specific e-commerce regulations like PCI DSS for payment data security, and managing compliance in a fast-moving DevOps pipeline can be difficult.

9. Knowledge and Skill Gaps

DevOps for SFCC requires a combination of knowledge in Salesforce technologies, development best practices, and DevOps tools. Many teams may lack this blend of skills and expertise, leading to slower adoption and misconfigurations.

10. Deployment Rollbacks and Downtime

Handling deployment failures and rollbacks in SFCC without causing downtime can be challenging due to the nature of the platform. With limited support for blue-green or canary deployments natively, rollback processes may need custom implementations.

11. Monitoring and Logging

Implementing robust monitoring and logging solutions for SFCC can be more difficult compared to traditional infrastructure. SFCC does not natively offer deep observability tools, requiring integration with third-party tools for monitoring application performance, transaction errors and server or platform health.

12. Continuous Feedback Loops

Fast feedback is essential to continuously improve development practices. Gathering user feedback, performance metrics, and error reports quickly from production can be difficult if not well integrated into the DevOps pipeline.

Best Practices

Due to its architecture and complexity involved with SFCC, developing a pipeline involves several best practices to ensure reliability.

1. Encrypt Sensitive Information

Information like Credentials and Client IDs and Secrets must be stored in the Jenkins Credential plugin rather than having them hardcoded in the pipeline or Jenkinsfile. Make use of the environment variables in Jenkins to create an entry of the Credentials and refer the variables in your pipelines for better security.

2. Separate pipelines for environments

It's advisable to separate the deployments to each environment instead of executing all at once. This helps keep the environments isolated and independent and have more control on the pipelines. Below is the sample code to add parameters to the job for execution. When this is included in the Jenkinsfile, you will have an option to execute the job to a specific environment. This will speed up the deployment process and ensure the right version is deployed to the required environments.

```
parameters {  
    choice (  
        name: 'Environment',  
        choices: ['sandbox','staging','production'],  
        description: 'Deployment Environment'  
    )  
}
```

3. Make use of artifacts

Plan to use the artifactory to store the successful builds to deploy across the environments. Successful builds are tested and validated and are reliable versions. 'ArchiveArtifacts' command can be used to store the artifacts from the builds.

4. Enable Roll-back option

Include Roll-back stage as optional in your pipeline and execute it when necessary. Option to revert the code comes handy in case of system failures. Revert to the previous version that has been tested and validated successfully to restore the application.

5. Monitor the deployment process

Adopt third party tools that are designed specially to monitor the deployment patterns. Review the patterns to identify areas of improvement and achieve 100% automation. Add post-build actions to notify the appropriate teams of the build results.

Conclusion

Salesforce Commerce Cloud is a robust, scalable, and customizable platform that helps businesses create engaging e-commerce experiences while leveraging the power of Salesforce's broader ecosystem. By embracing DevOps principles and leveraging the right tools and practices, your Salesforce Commerce Cloud implementation can achieve greater efficiency, reliability, and success, ultimately delivering exceptional shopping experiences to your customers. Implementing DevOps for Salesforce Commerce Cloud can be transformative, but it requires addressing several key challenges such as limited native DevOps support, managing customizations, automating testing, and ensuring security and compliance. Overcoming these challenges involves leveraging third-party tools, automating manual processes, and investing in the right skills and infrastructure.

Carefully planning your DevOps strategy and using best practices enables organizations to enjoy the benefits of faster deployments, improved collaboration, and better-quality software releases for their SFCC environments. By following this step-by-step guide from this whitepaper, you can establish a robust DevOps pipeline that enhances your development workflows, improves product quality, and accelerates the delivery of your e-commerce solutions.

References

- [1] King, M., & Murphy, M. (2021). *Salesforce B2C Solution Architect's Handbook: Leverage Salesforce to create scalable and cohesive business-to-consumer experiences*. Packt Publishing. ISBN: 9781800567227
- [2] DevOps, a new approach to cloud development & testing, 2020, <https://papers.ssrn.com/abstract=4004330>
- [3] Tatineni, Sumanth. "Applying DevOps Practices for Quality and Reliability Improvement in Cloud-Based Systems." *Technix international journal for engineering research (TIJER)* 10.11 (2023): 374-380.
- [4] Online / Digital Storefronts - <https://www.salesforce.com/commerce/online-store-platform/>
- [5] Salesforce Commerce Cloud - <https://www.rafter.one/what-is-salesforce-commerce-cloud/>
- [6] SFCC Developer Center - <https://developer.salesforce.com/developer-centers/commerce-cloud>
- [7] Salesforce for Commerce - <https://www.salesforce.com/eu/solutions/by-role/commerce/>