CASE STUDY

Autorabit Ensures Continuous Integration for Schneider Electric

CASE SUMMARY

This Case Study explains how AutoRABIT ensured Continuous Integration of Salesforce applications for its client, Schenider Electric, in less than a fortnight, while setting up GIT from scratch, reducing the execution time and expediting the production releases for the client.

COMPANY OVERVIEW

Schneider Electric is a French multinational corporation, which develops connected technologies and solutions that ensure safe, reliable, efficient and sustainable ways of managing energy and processes. It specializes in electricity distribution, automation management, and produces installation components for energy management. The Group has a strong commitment to sustainable development and invests in R&D to innovate and differentiate itself from its competition.

The company has its headquarters in Rueil-Malmaison, France, and is also based at the World Trade Center of Grenoble. Schneider Electric share is traded on the Paris Stock Exchange – Euronext Paris. It is also a part of the Euro Stoxx 50, Euronext 100 and the S&P Global 100 index.

Schneider Electric has its development and business teams spread across the globe. There are multiple releases (major, minor and hotfix) happening concurrently, with multiple global project teams contributing to the releases.

The leadership team in Schneider Electric felt the need for a streamlined release management solution driven by automation, which addresses the day-to-day challenges they faced related to Salesforce environment management, some of which are listed in the 'The Challenges' section below.

THE CHALLENGE

SCHNEIDER FACED THE FOLLOWING CHALLENGES:

Multiple Development Tracks:

There were several project teams, viz., Sales, Service, PRM, Marketing, CCC, etc., working with deliverables for minor and major releases. Integrating all these changes from multiple development teams (100+ members), without stepping on each other's shoes, was a challenge, as metadata, such as profiles and permission sets, were shared for various development tracks.

Multiple Releases:

There are minor releases (config. only), major releases (quarterly once) and hot fix for any business-critical issues that happen simultaneously. Ensuring constant sync between the development and release environments across the releases was a challenge.

Multiple Environments:

There were quite a few environments such as SIT, UAT, Performance, Training, Innovation, etc. Maintenance of these environments was a challenge, especially with many pre-refresh and post-refresh activities that had to be done manually after every production release.

Complex Salesforce Orgs and Heavy Deployments:

Schneider's Salesforce environments are large and complex, with large custom objects, profiles and permission sets, and typical deployments contain a few thousands of members.

Governance and Audit for the Deployments:

Several types of releases happened concurrently and changes were directly made in release environments such as SIT and UAT, which made it difficult to track the changes from a governance and audit perspective.

Environments Out-of-Sync and Challenges with Refresh:

Since changes happened directly on the Salesforce environments during releases and some of the BR's may not go-live to production – being rejected in UAT phase – the developer sandboxes, release environments and production Org were out-of-sync. Frequent refresh caused developers to lose their work-in-progress as well as induced lot of manual steps for the development teams.

Lack of Version Control:

Since multiple development tracks happened concurrently, changes to the shared metadata members, such as Custom Objects and Profiles, happened resulting in overwriting of code. It was not possible to maintain versions of changes, track them and rollback in the Salesforce environment.

Manual Steps in Migration:

There are many pre-migration and post-migration steps involved in deployments, in addition to large deployment footprint.

Exclusion Lists in Deployment:

Schneider Electric Development had a Security Guidelines and Architecture Team, whose recommendations required certain metadata to be excluded in packaging and deployments. Examples include login IP ranges, SSO configurations, etc. There was no way to skip parts of metadata in the existing deployment process, without heavy manual intervention.

Data Migration:

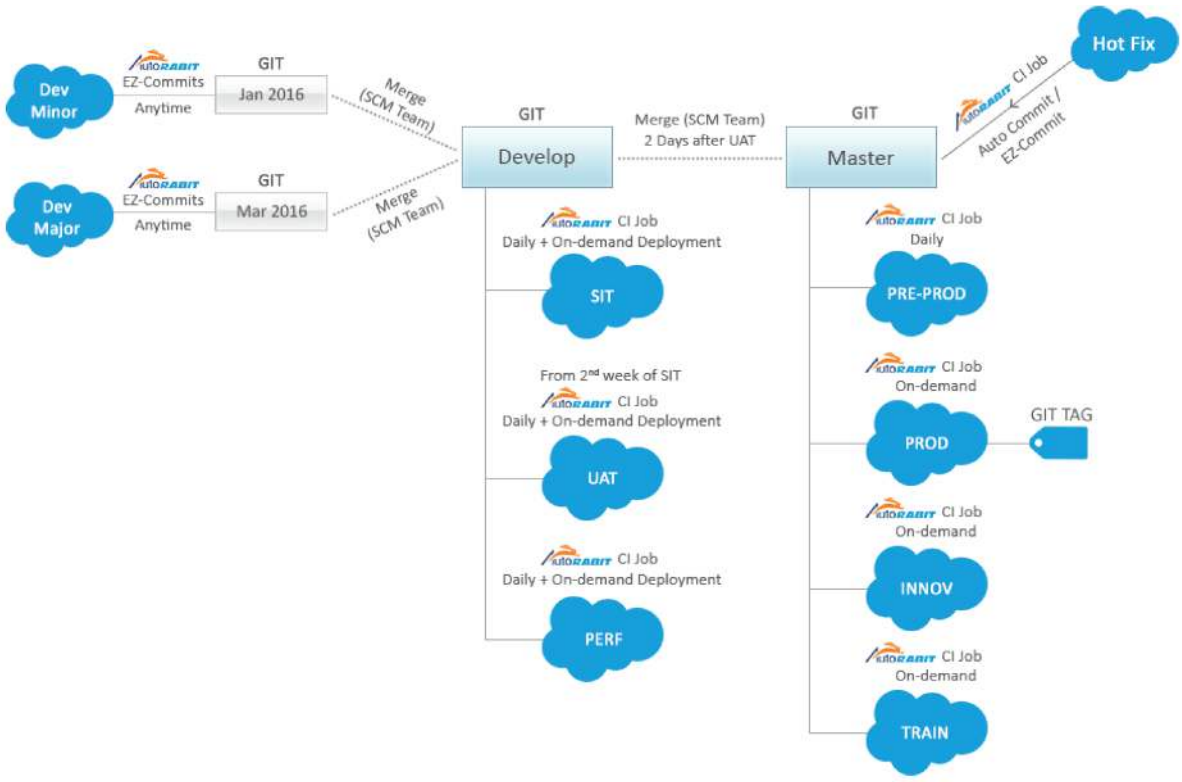
Schneider Electric had a business model of complex data. The inclusion of a wide range of Salesforce ecosystem-managed packages added more complexity to data migration, with several relationships and circular data references.

AUTORABIT'S SOLUTION

AutoRABIT team visited the client's offices in Paris and Bangalore, India, and understood their challenges in-depth, with a release automation discovery session.

AutoRABIT's implementation was driven by a robust Continuous Integration/Continuous Delivery maturity model (as shown in appendix below) that determines the client's current state vs the ideal state, and the maturity model mapping exercise helped the client's leadership team to discover their current state, prioritize their goals and set up a phase-wise implementation for their release management, with Version Control and Continuous Integration being identified as the primary success factors.

AutoRABIT team set up GIT repository for the client's team, outlined a Continuous Integration driven release automation flow from development to production (as shown in the picture below), created the branching model in GIT, provided a customized easy-check-in module – where the development teams can push their changes, along with dependencies into Version Control, without much effort – and also conducted onsite training on the best practices in Version Control and Continuous Integration.



AutoRABIT team also worked with the client's leadership teams on deployment strategies related to hotfix, performance, etc., and taught them the best practices.

Based on the above release automation discovery model, AutoRABIT team worked with the client's leadership team in creating Continuous Integration jobs, whose changes are automatically and effortlessly deployed from various branches into Salesforce environments.

The entire set up, implementation, training and rollout was completed in a span of two weeks and the client's team was able to use AutoRABIT for end-to-end release automation in a very short time.

AutoRABIT team understood the data migration requirements and designed Data Loader jobs to set up data for various client environments that are fully automated.

RESULTS/BENEFITS

The client reaped the following benefits through AutoRABIT's Continuous Integration suite

- The client's team, comprising 100 members, started using AutoRABIT tool seamlessly for their Continuous Integration of Salesforce applications in less than 15 days
- AutoRABIT's Selective Apex Testing reduced the client's execution time and expedited their production releases
- Seamless Data Migration with parent-child relationship saved 50% time of the client in environment set up
- AutoRABIT enabled seamless Org synchronization of the client across geographies
- AutoRABIT set up GIT from scratch and trained the team on best practices and ensured release process automation driven by Version Control in a short time of two weeks.
- AutoRABIT extended automated support to the client's unsupported metadata through customizations.
- AutoRABIT was able to bring governance and auditability for their deployments into various release environments

APPENDIX

AutoRABIT's Maturity Model

Level	Build Process	Environments and Deployment	Testing	Reporting
Level 4	• All build issues addressed through automation	• Continuous deployment to production • Feature sandboxes • Automated provisioning	• 100% code coverage • All testing automated • Code tested against large number of potential configurations	• Real-time visibility
Level 3	• Triggered builds • Code, meta-data, documentation, requirements subject to version control	• Separate Dev, Test, and Prod environments • Single button deployment	• Very high test coverage • Test interaction between code, meta-data, and data • Users create automated test scripts	• Report trend analysis
Level 2	• Build on commit • Dependencies managed	• Standardized deployment process • Full Sandbox • Selective Promote, Change sets • Some automated deployment	• Automated daily functional tests	• On-demand reporting
Level 1	• Source control utilized • Automated build process	• Documented deployment process • Defined branching strategy	• Large amounts of testing done at build	• Reports visible to entire team
Level 0	• Manual or semi-manual build process	• Changes made directly to Production	• Manual or no testing	• Status visible only to individual developer