Simplifying Salesforce release management:

a best-practice approach

gearset.com

Gearset

Contents

Executive summary

Managing deployments is generally considered one of the most difficult aspects of the Salesforce platform. Limited tools, complex functionality, and direct impacts on business as usual make it error-prone and labour intensive, particularly for larger organizations.

Gearset is changing this.

Gearset has developed a best-practice release management framework to help businesses gain the most from their Salesforce investment. From individual developers to complex, multi-team projects, this framework provides insights into managing risk, improving deployment success rates, and maximizing cost efficiency. With its combination of an intuitive user interface, powerful functionality, and access anywhere, Gearset supports every stage of this framework and provides a best-of-breed solution for Salesforce release management.

This whitepaper explores what makes Salesforce deployments challenging, how to structure a best-practice release management solution for your business, and how Gearset can enable faster, more reliable deployments to your organizations.

The challenge of Salesforce deployments

With its easy-to-use point-and-click interface, Salesforce has made application development accessible to everyone. Teams all over the world are benefiting from the increased productivity Salesforce brings them. In fact, so many things about Salesforce are simple, quick, and effective that one area stands out as particularly tricky: deployment.

Deployment is the riskiest point in any software development project. It represents the moment of truth for development teams, as new features are pushed live for the first time, and there are many ways in which it can go wrong. Organizations can be extremely intricate, with huge numbers of objects, and dependencies between these objects add to the complexity, with the result that it's all too easy to make an error

during the deployment process. Forgetting to deploy just one object can potentially cause a deployment to fail.

With infrastructure deployments, as in the typical Salesforce deployment scenario, the stakes are even higher, as the development team needs to fix any critical problems that arise before these problems start to interfere with people's jobs. Failed deployments can mean delays, missed deadlines, and late nights for the development team while they attempt to troubleshoot the issues.

How do people approach deployment at the moment?

Never deploy

For some people, deployment isn't a problem, because they work directly in their production environment, which means they never have to deploy. This can be a quick and efficient way of working for some companies. It makes sense if, for example, you don't have sufficient customization of your organization to require a development team, or if you need to keep costs to a minimum, as there's no need for separate development or staging environments.

However, there's always a risk associated with working directly in production, because you're effectively making changes to the live organization that's being used by your business on a daily basis. This means you need to be certain that the changes you're making won't break anything. In addition, there's the possibility that if multiple people are editing the production organization directly, they might end up overwriting each other's changes. There are also limitations in terms of the changes you can make – you can't write Apex code in a production environment, for example.

Environment manager as a “gatekeeper”

At the other end of the spectrum, if there are existing applications and users to consider, an environment manager (or release manager) might make every developer work in their own sandbox environment. This person will then personally manage and review every change before it gets to production in order to ensure it won’t cause any problems when deployed.

As teams grow in size and organizations get more complex, this role of “gatekeeper” can become more than one person can handle, as well as being extremely manual and tedious. And, as with any time-consuming manual task, there’s a chance that things will be missed and mistakes will be made.

Change sets

This tool is available via the Salesforce portal. Two organizations are configured so that they can send and receive changes between each other, and you can look through the objects in the source organization and choose which ones to deploy. Once the change set has been built, it’s deployed and staged in the target org, where you can accept the changes. The two organizations will then have the same specific metadata described by the change set.

The change set workflow is good for quick changes and for smaller organizations, and the graphical user interface (GUI) makes change sets accessible to a wide range of users, which is useful if non-developers need to implement changes. However, it can be hard to scale when you have many developers working on a team or many environments to manage as there’s no support for version or source control, and accidental overwriting of changes made by other developers is common. Destructive changes aren’t supported, and, while it does offer a basic dependency analysis, this can be unreliable, often flagging up false positives and making it cumbersome for larger deployments. The change sets tool doesn’t support any kind of governance control and there’s no way to track who has made changes to the organization, which limits how useful it is for auditing and reporting. But for many people, the biggest drawback is that many metadata object types are not supported in change sets.

Deployment teams often spend many hours running manual post-deployment steps to finish a release. This is cumbersome, error-prone, and entirely avoidable with the right deployment solution.

"We were exclusively using change sets and manual configuration for deployments. Custom settings or profiles could easily take a whole deployment window to manually deploy. If anything went wrong, or there were an excessive amount of changes that need to be done manually, things could get problematic."
Nadia Mayard, Salesforce Program Manager at Sutter Health

Force.com migration tool (Ant)

The Salesforce migration tool is based on Ant and allows finer-grained access to the deployment process. It's useful for automating or controlling the process more accurately, and creating artifacts that can be recorded or reused. Since it lacks a GUI, the Ant tool has a steeper learning curve than change sets, but brings greater power.

The lack of GUI, however, is a barrier to many non-developers, and running deployments requires manual editing of metadata which is error-prone and time-consuming. Dependencies for the deployment package must be individually identified and incorporated, requiring a deep knowledge of Salesforce. Comparison of files downloaded from different environments is a manual process, usually involving a specific diff tool to bridge the gap. Cases of deployment failures with obscure failure messages (which turn out to be down to missing a character when copying between text files) are all too common. As the tool is tied to a specific development environment on a computer, the Force.com tool is not well-suited for remote working or mobile teams, and its complexity makes it inaccessible to the majority of Salesforce users.

"Deployments with Ant take a lot of iteration. Unless you know exactly the patch you want to send, and the associated dependencies, you have to go back and forth and build the deployment piece by piece."
Teodros Negussay, California Department of Industrial Relations

Release management – why is it important?

Release management provides a framework to control when and where changes are promoted from one Salesforce organization to another. This builds on deployment management, which looks at how to move a change from one organization to another. Whether from a developer sandbox to integration testing or from user acceptance testing (UAT) to production, release management is the framework to enable effective organizational control, and should be implemented alongside your deployment management process.

Developing applications on the Salesforce platform is fast and easy. As we outlined in the previous section, there are a number of tools currently used to manage the deployment of these applications, including change sets and the Force.com migration tool. While using deployment management tools alone may be enough in some small organizations, a more integrated approach which employs release management as well is preferable as organizations grow in size and complexity.

To demonstrate, consider working with large-scale enterprise applications or very complex changes to an organization. Performing live edits of an application with a highly customized interface in a production environment no longer makes sense. Not only is it inherently risky, but the limitations of the web tools may also make it simply unfeasible.

Similarly, development involving larger teams necessitates a process to manage integration testing. Changes and fixes from multiple development environments must pass through integration testing before they're promoted to the production organization. This is beyond the scope of simply using change sets, and teams will quickly run into problems if they rely on this tool alone.

Many industries also have a legal obligation to meet certain regulatory requirements. A clear release management framework can help companies meet those requirements in key areas such as user access, data availability, and increased visibility across departments.

How do people approach release management at the moment?

The appropriate level of release management is heavily influenced by the size and complexity of your development projects and the tolerance of your business for risk.

No release management

For simple changes where the risk of interrupting business continuity is low, features are sometimes rapidly promoted from development to production environments, often with little concern for release management. Many businesses take the view that release management is not appropriate or will not deliver any value for these simple changes.

While these kinds of changes can be quickly deployed because they require little effort from the deployment management perspective, a reliable and simple process for release management is still advisable. Even basic records of deployment history and the changes released can provide a useful source of information when tracking progress, especially if you're working within a large project or if a problem arises after deployment.

Basic release management

When working with complex changes or multiple development environments, basic release management is usually a requirement due to the need for source control and integration testing. Reintegrating changes back into the production organization adds complexity to the development process due to the continually moving goalposts of the production environment, and many businesses begin to implement some sort of formal process at this level.

With projects of this size, properly planned release management provides a way to track changes as they're moved between organizations, create a clear audit trail, and reduce conflicts. Without proper planning, however, release management can become more of a hindrance than a benefit, slowing the development process down and creating choke points for project progress.

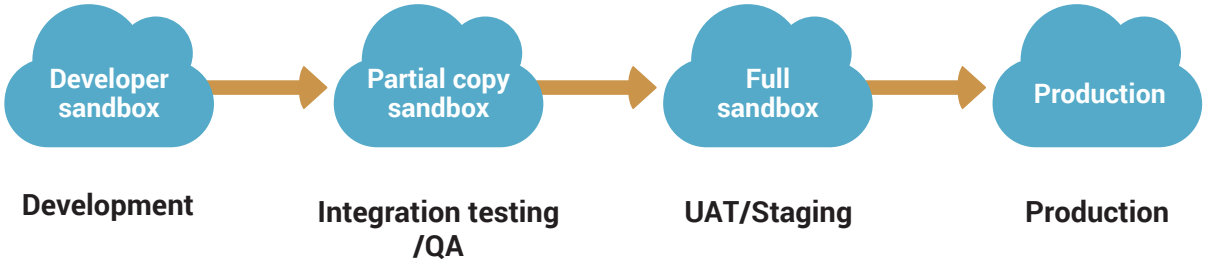
Integrated release management

Projects involving intricate applications that affect a large number of users often require multiple development and testing environments, and a dedicated UAT process. These projects, running over an extended period of time, usually involve several teams working on different development cycles and may require several rounds of integration testing before being deployed to production. Due to the complexity of these projects, with development efforts happening concurrently, release management is a complex but essential task, and almost all businesses employ a structured approach to support large project success.

Creating a best-practice release management model for Salesforce

To help businesses get the most from their Salesforce investment, Gearset has developed a best-practice model for release management in Salesforce. The model provides a high-level overview of organizational structure to best achieve successful deployments which meet users' needs, and also integrates simple deployment management into the process using Gearset. The model can be adapted to suit all needs, from very simple changes which require minimal release management all the way through to complex projects with a lengthy development cycle.

The model is based around information flow between separate Salesforce organizations with different purposes. Salesforce provides several organization types, each with its own set of features, performance, and pricing. This model aims to provide a guideline for achieving a balance between cost-efficiency and functionality.



Org type: Developer sandbox
Used for: Development

Developer sandboxes are isolated organizations which copy metadata (but not production data) into a different environment for coding and testing.

Developer sandboxes should be used for all development work, and are completely removed from the production environment. Each team member should have access to a copy of the metadata from the production environment, and their own independent developer sandbox in which they can make their changes. They may manage this using the Salesforce portal or with local IDEs, such as Eclipse or MavensMate. In larger projects, developers may maintain multiple environments for this purpose, and developer sandboxes should be linked to a source control system to allow easy promotion to the partial copy sandbox.

Org type: Partial copy sandbox
Used for: Integration testing/QA

Partial copy sandboxes copy both metadata and some production data, and have a larger amount of storage space to work with. This makes them well-suited as testing environments using selected production data.

Once development work on a feature is complete, it should be checked in to your source control system and deployed to your partial copy sandbox for integration or QA testing. In large teams or projects, a process of continuous integration, whereby committed changes are automatically built and tested, should be applied between developer sandboxes and partial copy sandboxes for fast and automated functional testing of features. This will allow for rapid iteration and bug fixing on changes and minimize the risk of clashes with other developers' environments. Having passed testing, features should be checked against project goals to ensure they're meeting customer needs before being promoted to UAT.

Org type: Full sandbox
Used for: UAT/Staging

Full sandboxes copy the whole production organization and all data. They're useful for coding and testing changes, and for training.

Before beginning UAT, the full sandbox organization should be cloned back from production. This allows new features to be tested in relation to your entire production environment, minimizing the risks of unexpected errors or unintended effects on other aspects of the organization. Due to the limitations on refreshing sandbox states, this should be carried out as part of a planned testing phase, rather than merely on the fly. Testing should cover whether the feature meets the needs and objectives of the project and users, as well as its stability and integration with the production environment. In the event of any issues, the feature must return to the development and testing process before coming back through to UAT.

Org type: Production
Used for: Production

The production environment is live and has users accessing data.

Before any feature is promoted to production, it should be functional (integration testing), meet the needs of the users (UAT) and not cause any disruption to the production environment (tested with real data). To minimize disruption, deployment to production should be made during scheduled maintenance windows, ideally when no users are on the system. Having been through this rigorous testing, there should be a very low chance of any unexpected errors or deployment failures which could disrupt the business. A detailed report should be maintained for every production deployment to aid project reporting and in case there are any unexpected issues.

Finding the right balance for your business

It may not be appropriate for your business to implement the entire release management model described above. Factors such as team size, budget, and organizational complexity will affect the depth to which the model is applied. To illustrate, let's look at three examples using the different levels of release management outlined in the previous section: none, basic, and integrated.

For smaller projects currently working under the 'No release management' approach, it may be appropriate to skip the integration testing and UAT and simply deploy between a developer sandbox and production. While it may be tempting to work directly in production environments, it's highly recommended that a development environment is used for testing prior to promotion to production. This structure supports rapid iteration and keeps complexity and cost to a minimum, while still significantly reducing the risks associated with working directly in production environments.

Teams working with several developers or under the 'Basic release management' approach will be using some form of source control and integration testing. In these scenarios, a process from development sandbox to partial copy sandbox to production (skipping the UAT phase) may provide enough structure to avoid code duplication and conflicts, while still allowing for some level of testing and code review prior to release.

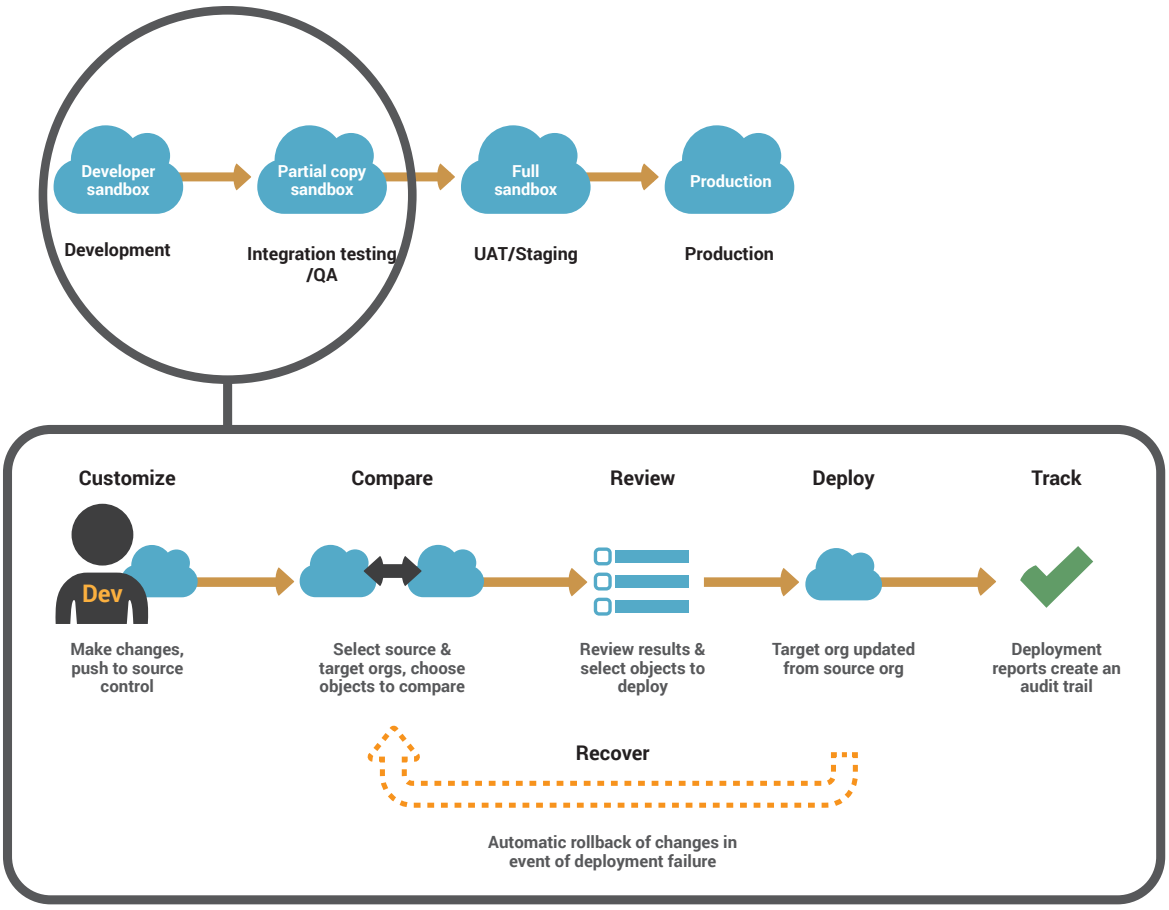
People working in large teams, on complex projects, or on changes which affect a majority of the user base, are likely to already be working under the 'Integrated release management' approach. Projects of this nature require tracking and merging changes from different code branches, user acceptance testing, and other involved processes to ensure a successful release. For these teams, our model provides a best-practice template to guide their current release management approach. The creation of an effective audit trail as a result of our model simplifies project management and reporting while protecting data integrity.

How Gearset improves your release management

Gearset is a comprehensive release management solution which streamlines deployments, reporting, and compliance for your Salesforce environments. Designed to lower the technical barrier to entry, it enhances team collaboration and offers orders-of-magnitude improvements in the time spent on deployments.

"No matter how you operate, if you use Salesforce, Gearset immediately simplifies and expedites how you deploy."
Alex Jones, Project Manager, Xaxis

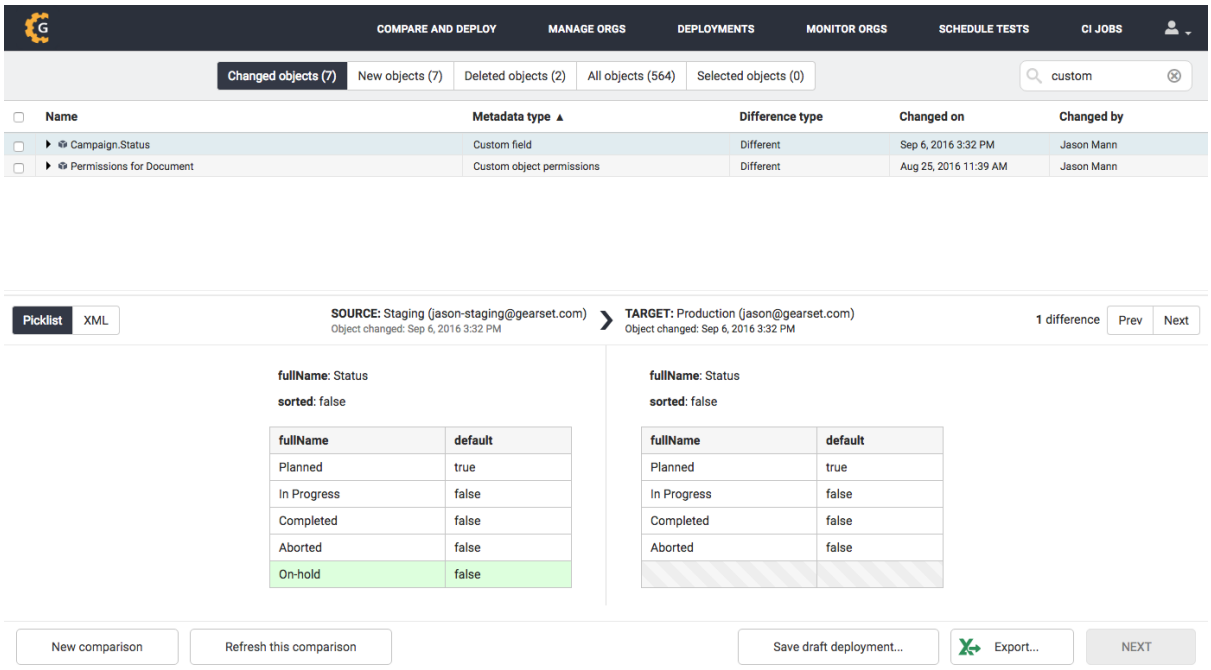
"Being able to quantify the time saving - what used to take 8 hours, now takes less than an hour with Gearset - that's huge, and it really speaks for itself."
Nadia Mayard, Program Manager, Sutter Health



Compare and deploy

Compare any two Salesforce organizations and view line-by-line configuration differences, giving you instant insight into the state of your environments. Automatic XML highlighting helps you find what you want, fast, and dependency analysis suggests the components required for a successful deployment.

Validate your releases and deploy the changes, safe in the knowledge that rollback is available at any time, and a detailed audit trail of activity is maintained in the app. Full support of new, changed, and deleted objects (destructive changes) makes Gearset powerful yet easy to learn, and PDF deployment reports are simple to share or incorporate into your user story tracking.



Schedule and automate

Schedule deployments to coincide with maintenance windows or sprint finishes. Email notifications keep your team in the loop with all releases. Set up continuous integration jobs to automatically keep two orgs in sync.

Test

Automate your unit testing, receive test failure notifications and easily debug errors. Track code coverage change over time and share results with your team with simple email, SMS, Slack and Chatter integration.

Monitor

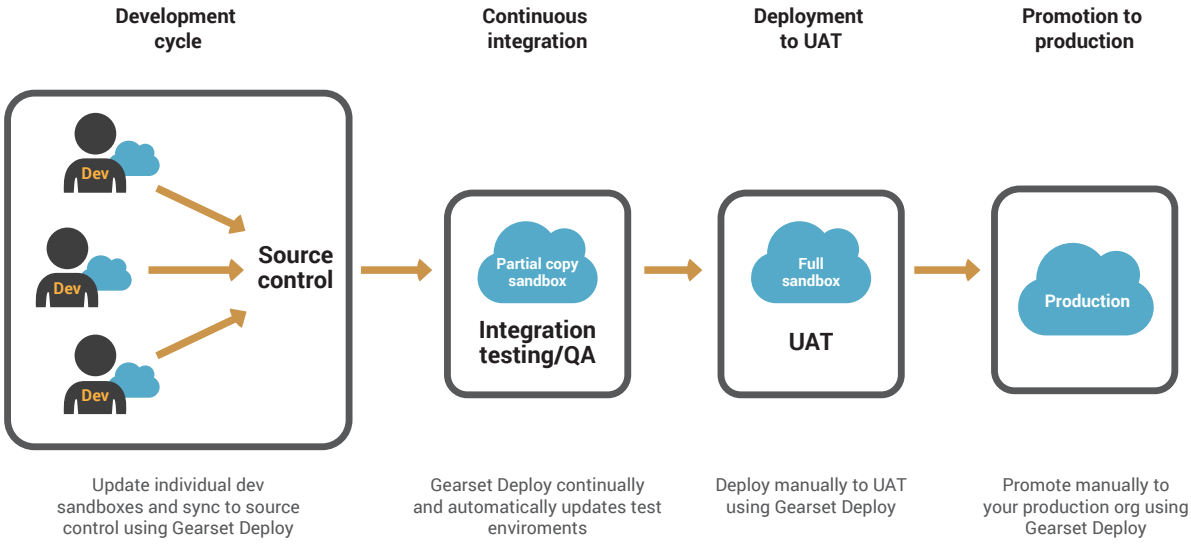
Track configuration changes being made to your orgs to prevent accidental overwrites. Inspect day-by-day audit logs and see exactly what changes were made, when, and by who, making managing multiple work streams a breeze. Roll back any unwanted changes with a couple of clicks.

Share and control

Collaborate with team members to prepare and deploy changes more effectively. Ensure SOX compliance with user roles and permissions, and delegated credential management. Manage team access and licensing self-service from the app.

Source control

Compare and deploy from any GitHub, Gitlab or Bitbucket repository and branch to your Salesforce organizations. Deploy files stored locally on your machine for full compatibility with on-premise version control or IDEs. Automate deployments from an integration branch to your Salesforce environments with continuous integration jobs.



Access anywhere

Access Gearset on almost any device with a web browser, with no packages to install in your orgs. Anywhere you can access Salesforce, you can access Gearset. Gearset is built around an intuitive, user-friendly GUI. Incredibly quick to master, you can get straight to work whether you're a seasoned developer or you've just joined a new project. Everything is managed with just a few clicks and there's no need to use a command line.

Secure by design

Data security is built into every facet of Gearset. Org access is managed via OAuth to protect user credentials. Advanced defence-in-depth techniques protect your metadata, including encryption at rest and in transit. Gearset is hosted in ISO 27001 compliant AWS datacenters that Salesforce and Heroku trust for their compute needs, and is trusted by industry leaders in healthcare, government, financial and education institutions around the world.

Conclusion

This whitepaper has demonstrated the challenges that face businesses around Salesforce deployment management, and the approaches commonly used to manage deployments. It then moved on to consider what release management is, how it builds upon deployment management, the benefits of employing it, and how businesses currently approach it. Based on best practice, Gearset presented a framework for the release management process, and demonstrated how the model can be modified to provide benefits to businesses and teams of any size. Finally, we looked at how Gearset provides a best-of-breed solution to enable effective release management through streamlined deployments, reporting, and compliance for your Salesforce environments.

Find out more at www.gearset.com or get in touch with us at team@gearset.com

gearset.com