COPADO

How to Simplify Salesforce as You Scale: A Developer's Guide

15 Recommendations for Adding Users, Apps and Production Orgs

Scaling the Salesforce platform enables development teams to accommodate business growth and meet customer requirements. But without the right process or the right tools, your Salesforce expansion can spark an avalanche of complexity and risk. Let's look at a few benefits of scaling and explore 15 recommendations for a friction-free process. This guide will show you how to scale Salesforce with a DevOps process by:

- Introducing sandboxes to the process.
- Adding team members to handle increased changes.
- Increasing the number of apps and permissions managed on Salesforce.
- Adding production orgs to support multiple business units or geographic regions.

Introducing Sandboxes

After implementing Salesforce, companies often appoint one or two admins to manage the system and customize it directly in production. However, after you roll out Salesforce to the sales and service organization, making changes in production is a recipe for broken functionality and poor customer experiences. That's where sandboxes come in — they give teams a safe environment to make changes and test them without the risk of disrupting live functionality.

Recommendation #1: Leverage user stories to track configuration and code changes.

While sandboxes are a great way to guarantee working functionality, you still need a tool to move changes to production once they are ready. Salesforce change sets can be used for deployments — but they are only designed to handle a few changes at a time. With change sets, it's difficult to track which changes are ready for promotion (especially as your team grows).

To tackle this challenge, Copado introduced user story-based change management in 2013. This approach enables agile companies to write user stories that define what each change is intended to accomplish. When a team is ready to deploy one or more user stories into production, they could create a change set with all the metadata, but this becomes impractical if multiple people are working on similar stories.

How Do You Grow Your Team?

Recommendation #2: Isolate your developers in their own personal environments.

Change sets fall short when teams grow larger and ramp up the number of user stories. Since many business processes involve the same objects and fields, multiple changes often overlap and create conflicts. To prevent your team from changing items in the same environment, Salesforce recommends using isolated sandboxes where each developer has their own environment.

Recommendation #3: Manage multiple changes by using version control as your source of truth.

When a change is deployed to production, there's a chance it could wipe out other changes made by other developers. With a version control system, you can manage changes to a common code base made by multiple developers. By using tools like Git as the source of truth, you can adopt a source-driven development process to merge changes, identify conflicts and prevent overwritten code.

Recommendation #4: Create feature branches in Git that correspond to user stories.

Synchronizing the contents of your Salesforce environment with the Git repository can help you become more aware of potentially conflicting changes. Through the click of a button, you can commit the user story changes to the source code repository. This creates a "feature branch" in the repository that corresponds to the changes listed on the user story.

Expanding the scale of Salesforce development requires you to adopt a process and a set of tools to manage this complexity — and DevOps has been proven over the years to be the easiest and most effective solution.

Recommendation #5: Adopt a process and tools designed for both pro-coders and low-coders.

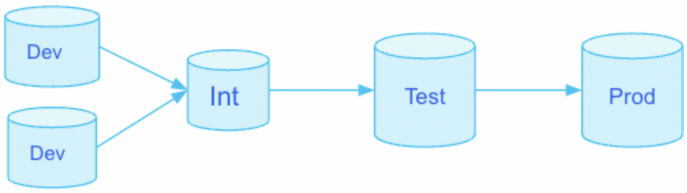
While professional coders may understand what it takes to merge branches in Git, a typical admin does not. According to industry analysis, the vast majority of Salesforce development teams include a blend of low-code and pro-code skill sets.

To support both types of developers in the same process, Copado provides DevOps tools for low-code admins and professional coders alike. Non-technical team members can leverage a point-and-click interface on Salesforce, while developers with traditional coders can use a Command Line Interface (CLI) with the same capabilities. It's a single tool that implements a unified process for both.

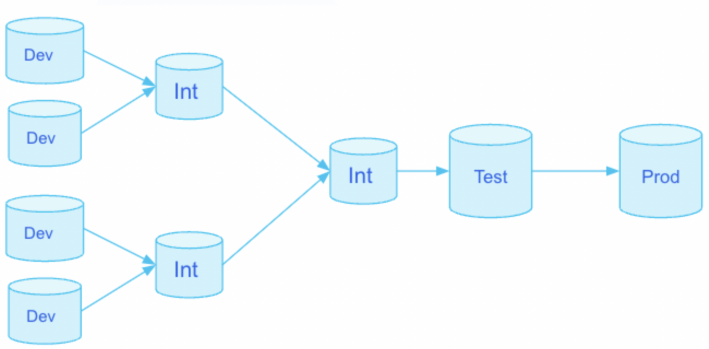
Recommendation #6: Organize your environments into a pipeline that provides integration and testing prior to production.

As your team expands in size, you need to add a new environment for each team member. These can be long-term environments like developer sandboxes or temporary ones like scratch orgs. To centralize development efforts, individual orgs should be organized into a pipeline that feeds into a single production environment (without connecting directly to production).

At a minimum, each developer org should connect to an integration environment where changes can be merged and tested. As a best practice, Copado recommends that teams insert a dedicated test environment between integration and production environments. If your company performs User Acceptance Testing (UAT), it may make sense to dedicate an environment for that as well. Manual testing often takes a long time and you may not want to disturb the environment during live testing.



As you add developers to your team, it may not be practical to require them all to push changes into a single integration environment. To keep things simple, divide developers into multiple scrum teams and provide an integration sandbox for each team. These integration orgs feed into a second level integration environment. Depending on the number of teams and organizational structure, you might even need three levels of integration.



Recommendation #7: Keep user stories small and release often.

Once developers are isolated and the pipeline is created, you're all set to generate changes and push them through the process. As a best practice, developers should focus on one user story at a time and release it when it's ready. Copado calls this a promotion. Each user story should be defined and written with the smallest unit of change that is still releasable — even if that means creating multiple user stories to implement a single feature.

The feature you're building may not be ready to be promoted all the way to production, but the changes released in the pipeline will have minimal impact on the rest of your development efforts. This practice is referred to as Continuous Integration and Continuous Delivery (CI/CD).

Some companies prefer to wait until the end of the day to promote user stories — and often have multiple stories ready to go by then. To make promotion more efficient, Copado enables users to batch multiple stories in a promotion. You can merge changes to form a "Promotion Branch" in the version control system and identify potential conflicts within that selection of stories before they are merged into the integration org.

Later in the pipeline, release managers can leverage bulk promotions to collect user stories from multiple developers and promote them forward into testing environments and production.

Recommendation #8: Minimize differences in pipeline environments with frequent back promotions.

Keeping differences between environments to a bare minimum reduces the chances of a merge conflict across user stories. You may be tempted to minimize differences by refreshing your sandbox. However, refreshes don't account for in-flight changes that were never committed and can lead to lost work.

To minimize differences and reduce risk, Copado enables teams to perform promotions in the reverse direction: Back promotions. After you make changes to your individual stories, back promotions make it easy to update them across other developers' orgs so they can deal with any conflicts before they try to promote their work. This process can even be automated to trigger back promotions overnight — allowing developers to start their day with an environment that's very close to their integration org.

Recommendation #9: Bundle user stories to improve release efficiency.

Keeping user stories small and releasing often is a best practice for CI/CD. However, many companies operate in highly regulated environments that require manual approval processes before releases. With a process like this, user stories tend to pile up in the final environment before production. *So how do you reliably release 100+ user stories to production in one fell swoop?*

To solve this problem, Copado created user story bundles. Once the review board has approved a selection of user stories, you can combine them in a bundle — which creates a single user story for release. Changes are merged in a release branch and can be promoted into production all together.

How Do You Add Multiple Apps?

Now that you've learned how to build a DevOps process to support a growing team, let's tackle the world of Salesforce app development. Most companies start their Salesforce journey with Sales Cloud and Service Cloud. Next, they typically add Commerce Cloud and Marketing Cloud to reinforce their Salesforce capabilities and start installing a few solutions from the AppExchange. Before long, these companies have 10+ apps to manage…including a few custom ones they built themselves.

Both Salesforce clouds and apps created by independent software vendors (ISVs) provide basic capabilities in a protected package while still enabling you to make changes to enhance their functionality. Many apps use the same core objects from Sales Cloud and Service Cloud (like the Account and Contact objects).

Over time, changes accumulate across these objects into an unmanageable mess known inside Salesforce as the "Happy Soup." So how do you scale app development on Salesforce across multiple apps that share the same core objects?

The first challenge with multiple apps is the flexibility Salesforce builds into profiles and permission sets. It is far too easy to update a profile with all the permissions required for each application. Resist this temptation! Salesforce has announced that they are removing permissions from the profiles available on the permission set. But that doesn't mean you should create a single permission set to replace it.

Recommendation #10: Leverage permission set groups to assign user access based around app personas.

Each app is designed for different user personas with their own set of needs. For example, an HR app might have permissions for admins, business partners, executives, managers and individual contributors. For each major feature, create permission sets and assemble them into groups that represent the application roles. Next, assign the permission set group to each user so it's easy to update individual permissions.

Permission set groups are even more critical for ISV applications. Instead of cloning and editing permissions when the new version of an ISV app is released, create a small permission set that gives extra access where needed and a muting permission set to take away user rights assigned to the wrong person. Then, clone the ISV's permission set group and add these two Permission Sets to a new group.

Recommendation #11: Organize your custom apps into packages and use locked packages for common libraries.

Want to know when changes are made to an app you customized? Assemble your apps in a package! For ISV apps, we recommend organizing your enhancements in an unlocked second generation package. This makes it easy to identify the purpose behind the change and remove it if the ISV app is

ever decommissioned in the future. Unlocked packages are also perfect for small homegrown apps for the same reason. Apps have a way of growing like a weed, so pay attention to feature creep.

Large homegrown apps are more difficult to manage — partly because developers often prefer to reuse code from a common library shared by multiple apps. It can be tempting for a developer to make "tweaks" to the libraries without realizing that it could break other apps that use them. To prevent this, locked packages are recommended for shared libraries.

Managing the contents of a locked package is best done in a dedicated pipeline. Access should be restricted to a small group of senior developers with a process in place for impact analysis whenever a change is required. Communication between teams is key.

Changes specific to the application can still be made in unlocked packages that reference the shared classes in the library. Developers who reference the shared package must make sure they have the latest version of the locked package — which places an extra burden on the release process. While Copado provides the ability to push package updates when new versions are available, a simple notification process and quality gate could be less disruptive.

How Do You Add Multiple Production Orgs?

What happens when you expand to multiple production orgs? And why would you want to do that? Multiple production environments add a layer of complexity to everything mentioned above. But don't worry…every practice to this point is fundamental for the success of multi-org. The most common reasons for using multiple production environments include:

- Multiple Business Units
- Multiple Geographic Regions
- Segregated Applications

Large enterprises harness the power of multiple business units with different product lines or distribution channels — which makes it impractical to serve multiple processes from the same production org. Instead, these companies tend to separate each unit into its own Salesforce environments and use analytics tools pulled from multiple environments when needed.

Likewise, when a multinational company does business in diverse regions with unique regulations and languages, it is often easier to split users across production orgs. Finally, some applications (like human resources automation and financial systems) deal with sensitive data. Salesforce provides flexible sharing and security tools, but many companies prefer to segregate sensitive apps into their own environments and limit the number of admins with the "View All Data" capability.

Recommendation #12: Manage common features in a core locked package installed and updated across all production orgs.

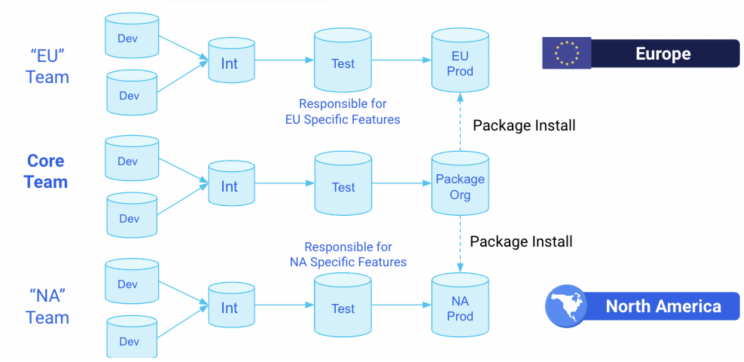
Use cases for multi-region and multi-business unit environments share a common element: They both rely on a core application that is 80-90% the same among all production orgs. However, managing the unique requirements for each individual region or business unit is the key to success.

Not only can multiple applications share common libraries in the same production org — apps can share a common library across multiple product environments. Using a locked package for the common library can protect it from unmanaged changes, and that same package can be installed and upgraded in multiple production orgs.

The core application is managed in a locked package that passes through a dedicated pipeline. Instead of a production environment at the end of this pipeline, there is a packaging org. New package versions are created and tested in this org before being installed in all the other production environments at the appropriate time.

Recommendation #13: Create a pipeline for each region and business unit to manage their unique changes.

Creating a dedicated pipeline for each region or business unit follows the same process as the core application. Since all the changes managed there are unique to a region or business unit, it isn't strictly necessary to create a package in these pipelines. While the core package is installed and upgraded in production, all the other changes may be promoted as usual.



Recommendation #14: Automate as much as possible and limit the number of pipelines managed by individual release managers.

As you add pipelines, you will find that the amount of change is too much for one release manager to handle — even with the help of automation. Pay attention to the release management workload (especially if you keep user stories small and release often).

Recommendation #15: Use a governance org and anonymize test data to reduce the number of users with access to sensitive data.

Which production org will your DevOps tool live in? Copado is installed in a Salesforce production org…but which one? Your developers will typically have admin rights to the org they call home. That gives them the capability to "View All Data" — which may or may not include access to sensitive data.

To reduce risk, create a dedicated Salesforce production environment as the home for developers. Copado calls this a governance org. Developers will receive credentials for the pipelines they work in — but only for the early stages of those pipelines. Release managers will need a login to production, but this should enable you to limit the number of users who need access to a given production environment. If you consistently anonymize data in your sandboxes, they will never be exposed to PII or other sensitive data.

Ready to Scale?

Scaling the Salesforce development process happens over a long period of time as companies grow. The journey looks different for every company. Multinational companies often need to implement multi-production environments from the very beginning, while smaller firms might adopt a single pipeline managed by a handful of team members. By following the recommendations in this guide, you will find it's easy to scale your Salesforce change management to the right level for your organization.

COPADO | #1 DevOps Platform for Salesforce

GET A DEMO