Getting started with Salesforce DevOps Center and Elements.cloud

Elements
.cloud

About this Guide

It is intended to explain the core DevOps principles so that organizations can maximize the benefits of migrating from DevOps Center. It has been written with Elements.cloud team's experience of the pilot, working with customers on the pilot, and building a partner extension alongside the DevOps Center team.

> *DevOps is actually a vast, and potentially confusing subject. While Salesforce DevOps Center helps Trailblazers replace change sets rather easily, there is more to using this tool for effective app development. DevOps Center Implementation Guide from Elements.cloud takes readers on a gentle tour of DevOps and the release management principles used by Salesforce DevOps Center. Then you get the step-by-step guide on getting the job done. This guide should be part of every Trailblazer's startup kit for using the Salesforce DevOps Center."*
>
> **Vernon Keenan, Senior Industry Analyst, SalesforceDevOps.ne**t

This Guide is organized into three sections.

- An overview of DevOps best practice principles
- How DevOps Center works and the Elements integration
- Implementing DevOps Center

Introduction to DevOps Center

Moving metadata between Sandbox and Production Orgs has never been easy, particularly for the declarative developer – i.e. the Admin. Change Sets are a painful experience and replacing them has been long overdue.

Fortunately, that wait is over. DevOps Center is the free replacement for Change Sets and it goes into public Beta in June with GA in Fall 22. So we should expect some announcements at Dreamforce22 in September about GA and the roadmap.

> *DevOps Center is all about change and release management and introducing DevOps best practices to our entire community, regardless of where you fall on the low-code to pro-code spectrum*
>
> **Karen Fidelak – Salesforce DevOps Center Product Manager**

The goal of DevOps Center is to allow low code / no code declarative developers (Admins) to drive deployments in the same rigorous manner that pro-code developers (Developers) do, but with an easily understood point and click interface. There will then be just one single source of configuration and code that is managed in GitHub which will improve team collaboration and compatibility across all functions; admins, developers, release managers, QA, and business stakeholders.

The vision is that DevOps Center will support a unified deployment approach across Salesforce platforms – Lightning, Heroku, Mulesoft, Commerce, Marketing, and 3rd party ecosystem. The first release supports the Lightning platform and it has been architected to enable partners to build extension packages.

Launch blog [DevOps Center is in open beta!](#) written by Karen Fidelak.

Core DevOps principles

DevOps is not just for technology projects

The development, launch, and subsequent changes to any product or service all go through the same lifecycle; requirements definition, development of a set of planned changes, the implementation and testing of those changes, and then the release. Let's call this the "implementation lifecycle" which is a generic term.

The level of rigor of the implementation lifecycle process is driven by the complexity and the risk of what is being launched. As Salesforce 360 is becoming a reality in organization after organization, the Salesforce core platform is no longer an isolated tactical application. It is a strategic platform in the IT landscape with integration to other Salesforce clouds and external systems. Making uncontrolled changes can have a material impact as it introduces risks; business risk, technical risk, and regulatory/reputational risk. This means Salesforce needs a formal process for implementation if you are going to accelerate the speed of innovation.

Implementation lifecycle

There are different terms that you will hear to describe the Salesforce implementation lifecycle. These terms include Development Operations (DevOps), Application Lifecycle Management (ALM), Software Development Life Cycle (SDLC), and Continuous Integration / Continuous Delivery (CI/CD).

DevOps is the term that is gathering momentum in the ecosystem to describe the implementation lifecycle and the applications that support it. However, DevOps is often depicted as the entire implementation cycle as you can see in the images below. You will recognize it as it is in Salesforce DevOps presentations, articles, and training materials.

A huge impact on the success of the project is the upfront analysis and design. This happens in the "Plan" phase. In the Plan phase in the image below are the following activities: capturing requirements, process mapping, release planning, architecture design, user design, technical design, creation of user stories/work items, and the risk assessment in terms of business, technical and regulatory risk of the user stories and therefore the release. Whew!! Each of these activities individually could be as large as the "Test" or "Release" activities that have their own segment of the donut.

Modern Application Lifecycle Management (ALM)
The fast, efficient, and trusted path to build on Salesforce
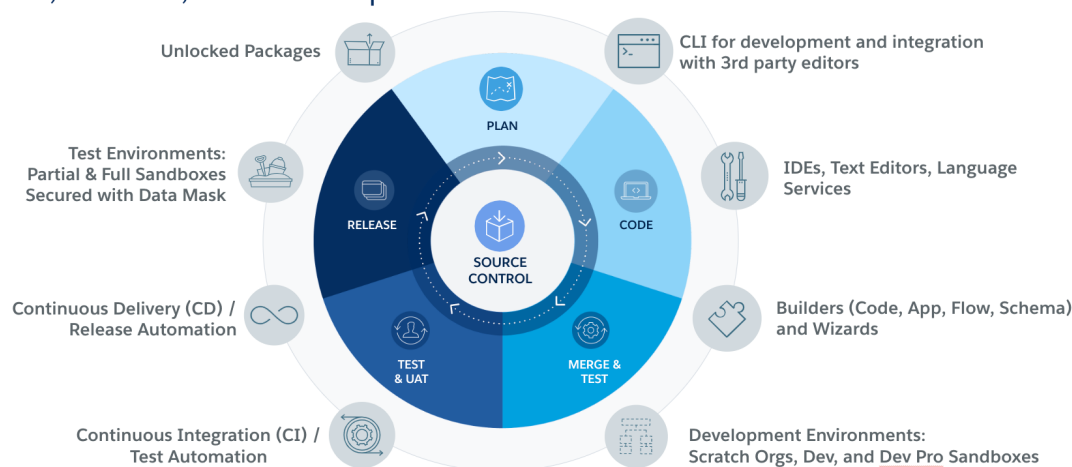
As Abraham Lincoln, a famous President and a low-code expert said "*Give me six hours to chop down a tree and I will spend the first four sharpening the ax.*" These are wise words applied to any project, but they're even more relevant when the misguided application of agile and the compelling speed of low code means that developers rush into the build phase without sufficient analysis and planning. The resulting rework and tech debt eliminate any benefits of speed.

So the core of any DevOps implementation is putting in place a set of documented and agreed ***implementation*** processes that cover the entire lifecycle, not just the development activities. You need to get all the stakeholders - platform owners, business analysts, designers, architects, admins, developers, release managers, and consultants - literally, on the same page. BTW these are roles or skill sets, not individuals. We know that in many - or most - organizations these roles are covered by just a few people.

As the implementation processes will vary very little by industry or size of the organization, later on in the guide we discuss each step in more detail in the implementation lifecycle that is shown in the image below.
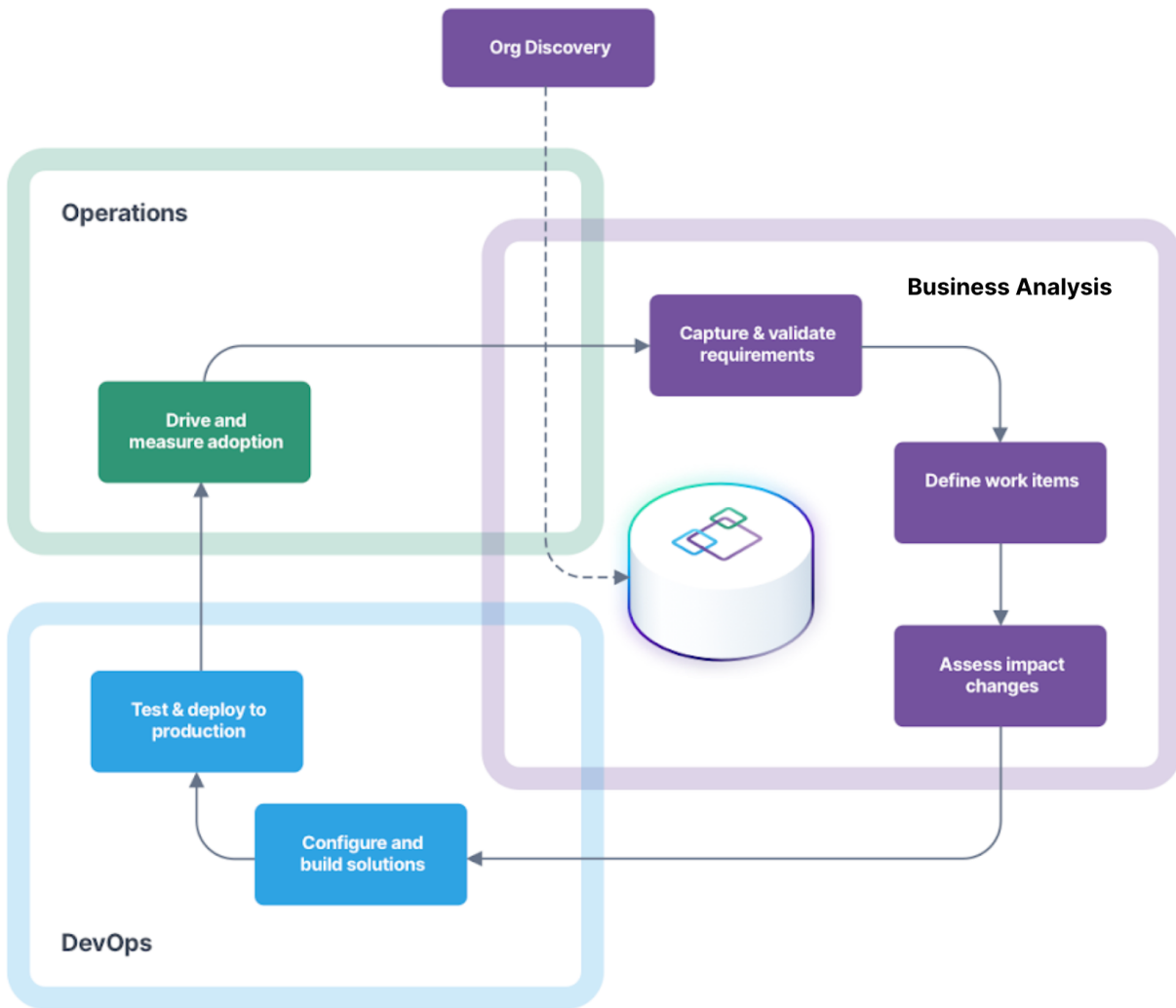
Low Code No Code vs Pro-code

Low Code No Code (LCNC) is declarative development. It is drag and drop. It is quicker than coding. It is getting more powerful with automation like Flow which means there is less need to code. There are of course business requirements that can only be satisfied by writing code. The important thing is that LCNC is coding; declarative development. So it needs to follow the same disciplines as coding. It is just easier and more intuitive. And the great news is that Salesforce is considered by industry analysts to be the #1 LCNC platform.

In addition, there are LCNC platforms that sit on top of Salesforce or extend what can be done by declarative developers. Examples are form-building apps like GetFeedback and FormAssembly, or workflows like Nintex and Workato. Also, there are integration platforms

that could be considered LCNC. These include Zapier and Celonis Make (originally Integromat).

The LCNC developers have traditionally been called Salesforce Admins but it is better to consider them as declarative developers to make the point that they are developing applications.

Pro-code developers are simply solving more technically challenging requirements that are beyond the drag-and-drop capabilities of the Salesforce core platform. They are writing code and are familiar with using Command Line Interface (CLI), source control systems, and scripting to push their changes through the pipeline.

DevOps Center gets both LCNC and Pro-code developers working in the same way.

Environments, pipelines & branching

A pipeline is a series of orgs (environments) that metadata changes go through from development to production. So you can have more than one pipeline. You could have 3 different pipelines; high-risk changes, low-risk changes, and hot-fix.

Each developer (LCNC and Pro code) will have their own dev org - their branch. So, at some point in the pipeline, the changes from the different developer orgs need to be merged. In the image below you can see the 2 different dev environments have been defined along with the sandboxes and production.

DevOps Center Environments

DevOps Center allows you to define the pipelines and it manages the movement of metadata through the pipeline. In the open beta and initial GA release. There is a single path rather than parallel path options. What that means is DevOps Center can support multiple developers feeding into the pipeline. But then, after that, it can only be a single flow through to Production.

Source Control

One of the core principles is that all metadata changes are saved to a source control system that is external to Salesforce. This means that all metadata for each org in the pipeline will be stored in the source control system. And when a metadata item is pushed or promoted from one org to the next in the pipeline, the source control system needs to be updated.

Don't worry if this seems overwhelming. DevOp Center manages the source control system behind the scenes and hides the complexity. In the open beta and initial GA release of DevOps Center, GitHub is the supported source control system. You only need the free GitHub account. Other systems such as BitBucket are in the product roadmap. There is a Trailhead badge on Git and GitHub Basics if you are interested in what a source control system does.

CI/CD

CI/CD stands for Continuous Integration/Continuous Delivery. The concept is that work can be broken down into smaller chunks and using automation you can move to a daily release, or even more frequently. To get a change from the developers into production, they will be moved through the different sandboxes in the pipeline. So part of the tooling is detecting changes. Automated tests need to be run and if successful then the change is moved to the next stage in the pipeline. So these automation jobs can get quite complex.

Below is an image and an excerpt from Gearset's [Launchpad training site.](#) where you can learn more about CI/CD.

> *"Across all software stacks, CI/CD automation is just a script that gets triggered on a periodic basis or when a change to your repository is detected. But automation jobs can get very complex. When a change is detected, code has to be fetched, compilers run, a variety of automated tests need to execute in different runtime environments, and so on. In the Salesforce world, it's a little more straightforward because a Salesforce org is both your compiler and the runtime that your code executes under. For this reason, CI/CD within the Salesforce context is primarily focused on automating testing and deployments."*
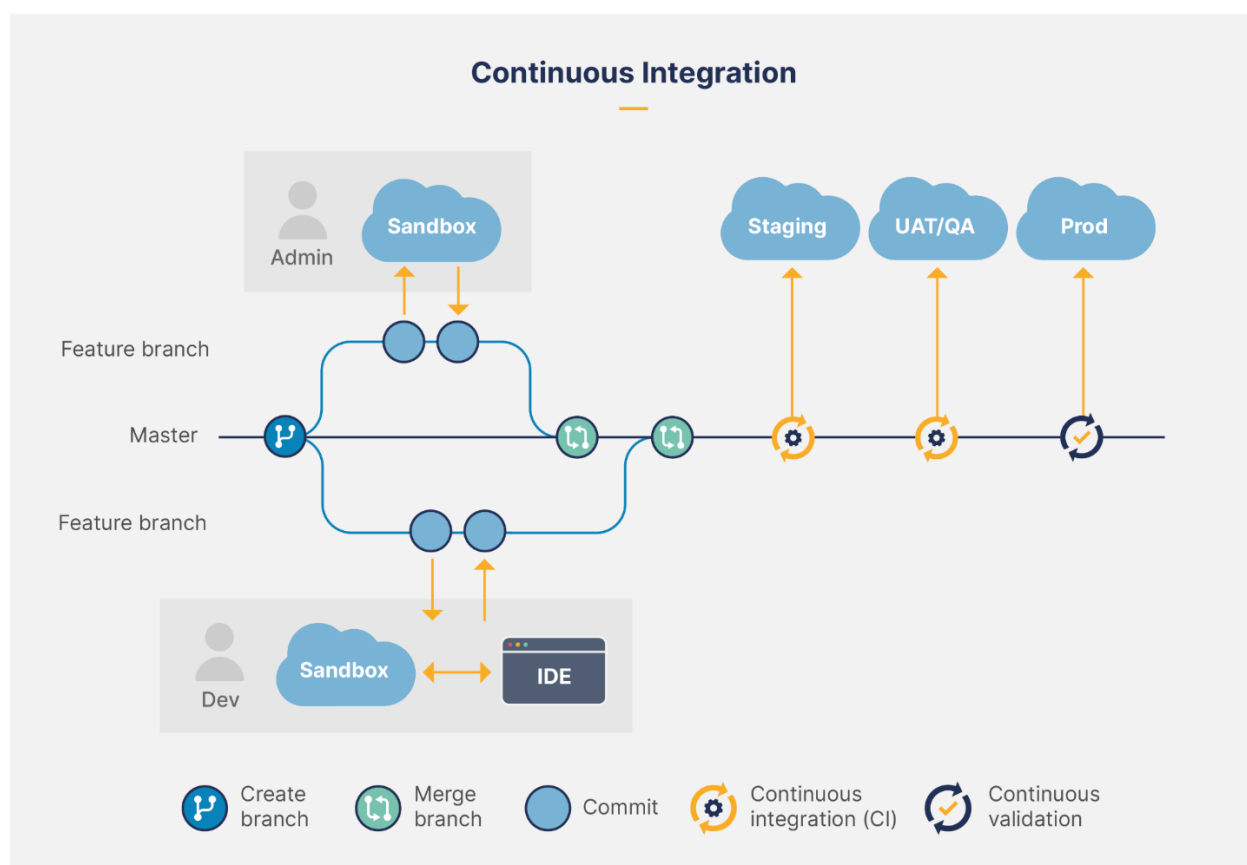

Image from Gearset Launchpad

While DevOps Center doesn't have built-in automation for CI/CD yet, it will happily work alongside it. As you can see from Gearset's maturity curve below, DevOps Center will get you to Level 2 and once you start to work and refine your DevOps processes you can get to Level 3. CI/CD is Level 4 and 5 but you will need to invest in more sophisticated tooling than DevOps Center.

| Level 1: Beginner	Level 2: Novice	Level 3: Practitioner	Level 4: Expert	Level 5: Luminary
Tools None / Change Sets	**Tools** Ant / DX CLI, Git, Change Sets	**Tools** DevOps tool with change monitoring, Ant / DX CLI, Git	**Tools** CI-based DevOps tool, Git, DX CLI (optional)	**Tools** CI-based DevOps and backup tool, Git, DX CLI (optional)
Process Development in sandboxes or directly in production; infrequent releases.	**Process** Development in shared sandboxes; a subset of metadata in source control.	**Process** Development in individual sandboxes; changes in production cloned back to Git.	**Process** CI workflow with a majority of metadata in Git; automated unit testing; single-click releases.	**Process** Continuous delivery to integration with automated metadata and data backups.
Benefits You're ready to start on your DevOps Journey.	**Benefits** You use Git as backup to help familiarize your team with source control.	**Benefits** You combine features during development rather than a stressful release window.	**Benefits** You integrate changes via Git with regular and reliable releases to production.	**Benefits** You've added robust data restore to your high-cadence release process.

Gearset Maturity curve

Shift Left

What exactly is Shift Left? In simple terms, it is the process of early analysis or validation which means we shift the identification of a problem to the left i.e. earlier in the plan. There are numerous studies showing the financial benefits of Shift Left but it is pretty self-intuitive. It is 100x cheaper to fix an issue in analysis than in production. It is also 100x less stressful.

In the diagram below the blue area is where the effort is expended in Shift Left. The purple area is the effort in a traditional model. What is important is dollar signs are the cost to fix any problem at any phase. As you can see it gets more expensive the further down the lifecycle that you discover an issue.
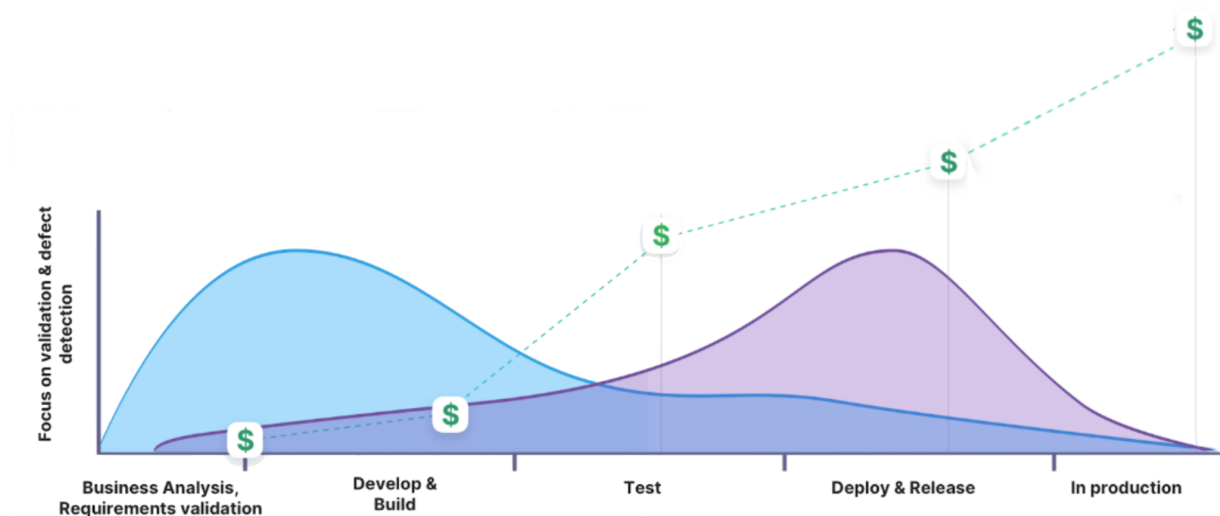
Shift Left

Unfortunately, DevOps talks about Shift Left as *earlier testing* as they are considering only the Develop & Build and Test phases. But if we look further left..... Are the requirements really understood before the app starts being developed? How powerful would it be to know the impact of changes to Org metadata before it breaks the Org? What is the cost of downtime created by a patch that was rushed into production? Without accurate analysis, you may be delivering the wrong features with the risk of breaking the org which impacts adoption and erodes trust, even if you deploy to Production faster and often.

Understanding and validating the business requirements and the impact of the change from a technical, business, and regulatory basis is the cornerstone of Shift Left. This work is called business analysis (BA) and Salesforce has just announced the Salesforce BA Certification. Here is the exam guide which sets out the scope of BA. You should use the implementation of DevOps Center to put in place a solid BA process and documentation standards.

Metadata dictionaries and documentation

Salesforce configurations can get complex quite quickly. Poor documentation of metadata changes can lead to runaway technical debt which compromises agility. If the purpose and use of a metadata item is not understood, then there is a risk that changing it will break the org. So often a new, but similar, metadata item is created. And so it goes on. Multiple fields, flows, picklist values, record types etc.

Salesforce sessions on tech debt and cleaning up your org recommend that you have a metadata dictionary to track changes. You need a metadata dictionary for each org - Production and Sandboxes - where metadata changes can be documented. The good news is that the metadata dictionaries can be built and maintained automatically and quite a lot of automated documentation can be generated. The nightly sync can build a change log and alert you to changes. The information about metadata items can be generated. The field population can be calculated. The impact assessment and dependencies can be analyzed and documented in a visual tree. This is available to all teams working in different sandboxes to make better design and build decisions, avoiding conflicts well before any metadata is committed. This just leaves manual documentation such as process maps, architecture diagrams, specs, and images or screenshots to be added to or linked to the metadata items.

You should also consider building metadata dictionaries for other apps (Salesforce non-core, other clouds on-prem and custom) across your IT estate. Then you can link any metadata item in them to your Salesforce orgs and to any manual documentation.

Managed Packages and data as metadata

There are several development approaches. Clearly, there is updating the metadata in an org, and installing Managed Packages from the AppExchange. Salesforce reported that 89% of customers use partner apps.

But the development in an org can be a combination of these approaches. You can develop the changes in an org the same way that an ISV develops an app in a Managed Package. Think like a Product Manager and structure changes as an app. This could be as an unlocked package. Whether it is a Managed Package or Unmanaged Package the changes will be visible in the org and will be synced to the metadata dictionary.

Some managed packages store configuration as data in objects rather than metadata. Examples are Vlocity, now a core part of Salesforce Industries, nCino the banking platform ISV, and CPQ (Configure Price Quote). These config records are stored in the metadata dictionary so that you can link and document them, but DevOps Center cannot currently manage them.

Salesforce Clouds

Salesforce is not a single product. At the heart of it is the core platform. In the image below the core platform covers Sales, Service, Platform, Safety, Sustainability and Industries. Arguable, also Partners as they. create managed packages that extend the core platform So there are links to Slack, Marketing, Commerce, Analytics and Integration that are outside the core platform. And then there are links to the other enterprise applications that your organization uses for finance, product management, fulfillment etc.

DevOps Center currently supports changes to core lightning platform, but not configuration changes that are stored as data records which is how some of the Industry solutions are built. The DevOps Center vision is to support the development of more of the Customer360 clouds.
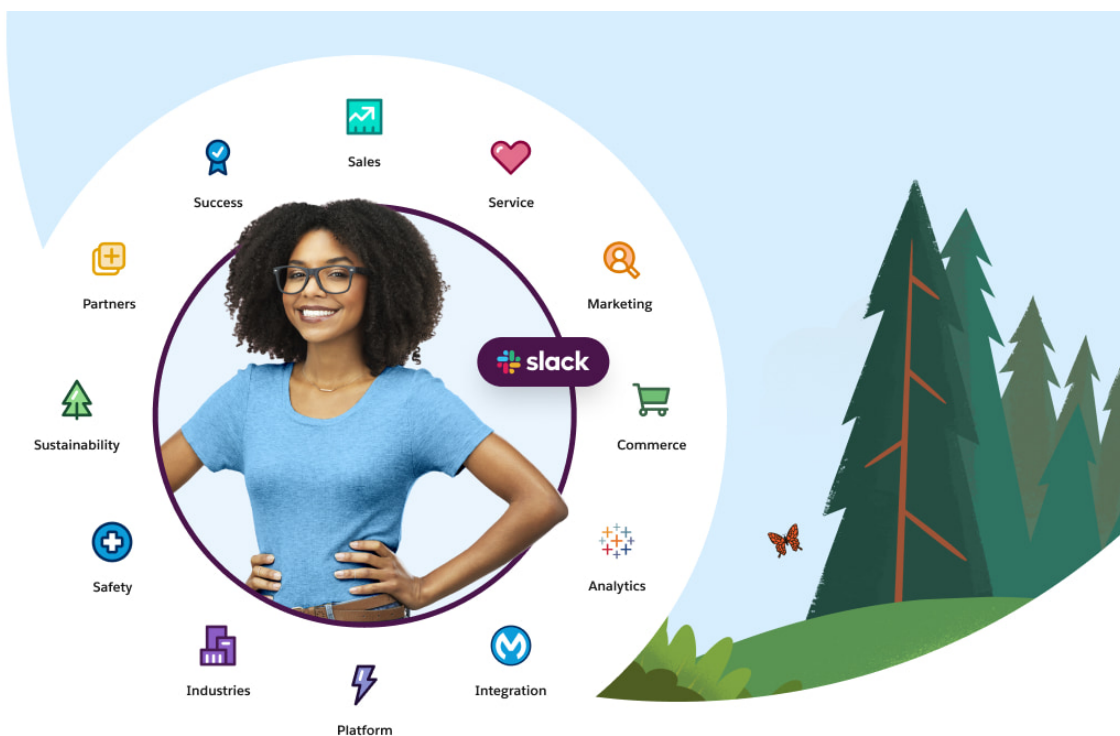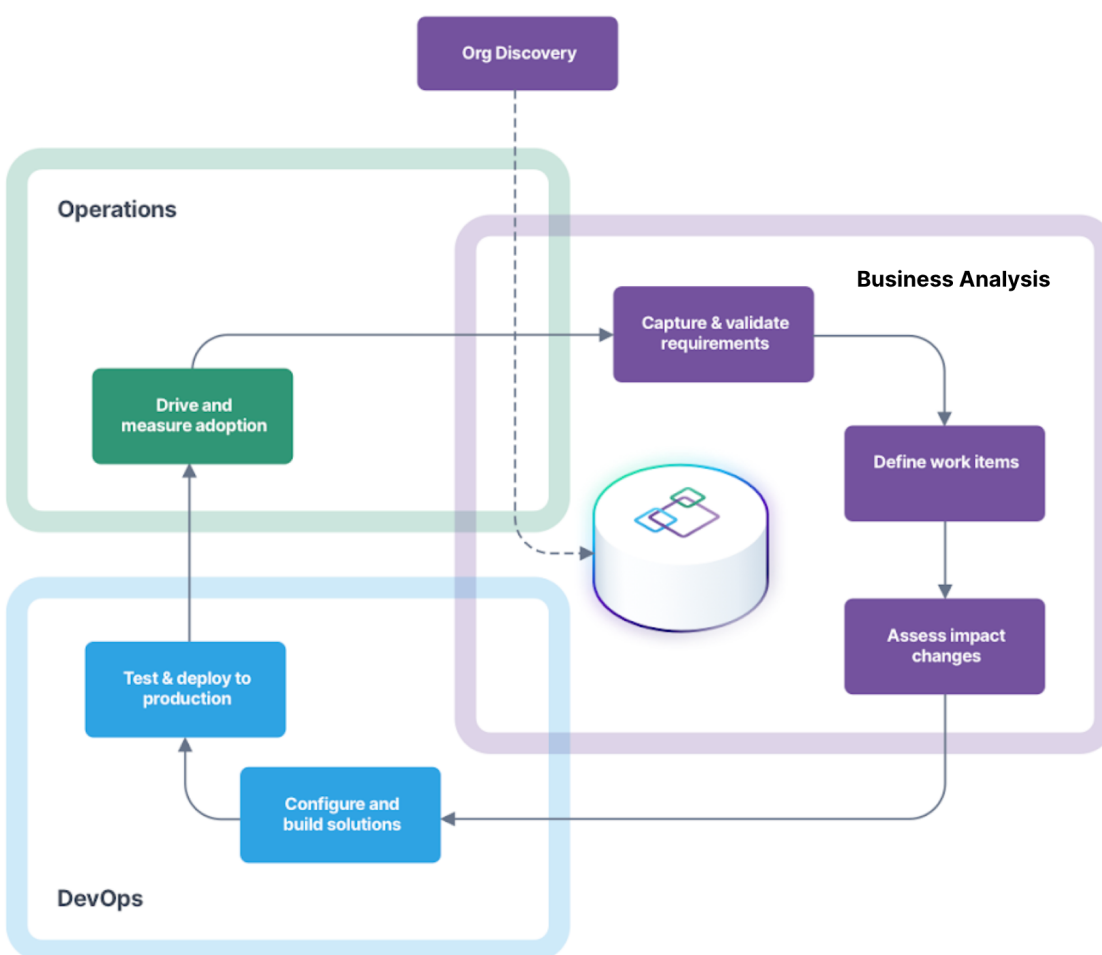
Implementation lifecycle in detail

Every change goes through an implementation lifecycle. Implementing DevOps Center is a change process, so it will go through the implementation change cycle. How meta is that!

You should use the implementation of DevOps Center as the catalyst to document and get a shared agreement on your implementation lifecycle processes. Even if they are currently well documented, DevOps Center will open up opportunities to streamline them. Getting more rigorous business analysis in place will dramatically improve your Salesforce ROI and user adoption. Simply replacing change sets with DevOps Center is a huge missed opportunity.

Your terminology for each step may be different but every change (should) go through each of the steps in the diagram below. Each step is described in more detail below.

Org discovery

Initial discovery

You walk into an organization and there is little or no overall understanding of the configuration of Salesforce. You have no idea of the scale or complexity of the org. Where the org is hitting limits. Where the technical debt is highest. What changes will have the highest risk? You are literally blind. It is scary. It is uncomfortable. It knocks your confidence. Elements.cloud can automatically analyze the org, build metadata dictionaries, add documentation, and create the impact and dependency analysis visualizations.

Documenting evidence

As you navigate around your org you will spot things; metadata that can be deleted or optimized; notes about what you've discovered who, how, and importantly, why things were configured. You can also tag or document ideas for improvement. This is a working, living repository of knowledge about the metadata - not just a snapshot report. Step away from GoogleDocs or GoogleSheets. There is a better way. You can document what you've found inside the org in the Elements.cloud metadata dictionary.

Capture and validate requirements

Capturing requirements

End-users identify changes they want. They may be well defined and thought through. They may be vague wish list items and they need help to identify the real issue and challenge the changes that have been requested. Also, the business has major initiatives, such as CPQ, that they want to implement to improve productivity. All of these are captured as business requirements with as much supporting information as possible. BTW "We need Einstein" is not really a business requirement. These requirements are captured inside Elements.cloud along with supporting information.

Validating requirements with process mapping

The best way to understand and refine the requirements is a live process mapping workshop with the end-users. It will drive out the true requirements and identify new ones. It will enable a far better architected solution to be designed. The process maps should also be used for UAT and user training, so they will have a life beyond this phase of the project. In fact, as Salesforce implementations are iterative, the process maps are a

valuable ongoing asset. Here is the [process mapping Trailhead badge](#) and a [live process mapping session with Ryan Reynolds](#), actor, and owner of Aviation Gin. Elements.cloud is the process mapping application that supports the UPN (Universal Process Notation) standard for Salesforce.

Architecting solution

The Salesforce "Well Architected" program encourages that any solution is architected and documented using the Salesforce Diagrams standard. Capturing the architect's design decisions is an important step to establish the detailed changes that need to be made to the system. Rushing straight to solutioning can have a profound longer-term impact on system performance, technical debt, and maintainability. Architecture diagrams can be drawn in Elements.cloud with all the power of connecting to the other business analysis content.

Define Work Items

Creating User Stories

An User Story is the definition of the changes to the metadata that needs to be driven through the pipeline of orgs, from a business perspective. A User Story is created in Elements.cloud during business analysis work. The User story can be linked to all impacted metadata items with the documentation created through the business analysis phase to provide context. Conflicts can be identified early if the same metadata item is linked to multiple user stories in the same or different releases.

Understanding the impact/risk of changes

During business analysis, the risk of a User Story can be estimated. This will allow the correct level of development and testing resources to be allocated, and which pipeline to allocate the work to. The risks to be considered are technical (which metadata items in which systems are impacted and the knock-on impact), business (how big a change is it to the operation processes), and regulatory (will it break any compliance regulations). The technical risk assessment requires an understanding of the complexity and dependencies across Salesforce core and external systems.

Generating Work Items

The Elements.cloud User Story will generate a Work Item in DevOps Center. A Work Item has a list of metadata items that are going to be driven from development into production. It may also sync with an issue/ticket in Jira if that is used to track development work. A User Story may be linked to several Work Items to track bug fixes to the original Work Item.

Defining Release

A Release is a logical collection of changes. So a Release in Elements.cloud has one or many User Stories. With Change Sets, the entire Change Set was effectively a Release. This is because it had to have all the metadata items in the release. But with DevOps Center, you can split changes into a number of Work Items and a Release can have many User Stories and hence one or many Work Items associated with it. A Release could be many Work Items for a new app with hundreds of metadata items. Or for an upgrade or a hotfix, a Release could only be just one User Story and therefore one Work Item, with one or two metadata items. An Elements.cloud Release is a collection of User Stories that are connected to DevOps Work Items. The metadata items on the Work Items are kept in sync with the User Stories. At a Release level, inside Elements.cloud, you can see potential conflicts before they hit development or test.

Configure and build systems

Defining a pipeline

DevOps Center makes the process of driving the Work Items through the pipeline significantly easier than change sets. The first task is to define a pipeline which is a series of orgs and branches from development through to production. There can be a number of pipelines. For high-risk changes the pipeline will have a number of sandboxes between development and production - for example Dev, UAT, Staging, Training, Production. For low risk changes it might only be Dev - UAT - Production. For hot fixes it is Dev and Production. DevOps Center makes it so easy to deploy changes that there is less excuse to develop straight into Production.

Developing (LCNC and Pro-code)

The changes may be made using a declarative language or coding. The definition of the work will be the Work Item or a Jira ticket, but with the related Elements.cloud User Story information shown alongside. This extra information will reduce ambiguity and potential rework. With DevOps Center each developer will have their own development org, even declarative developers. Behind the scenes, DevOps Center will create a new branch in GitHub for each Work Item when the Work Item is allocated to a dev org.

Documenting changes

The changes that are made to the metadata items need to be documented so that future impact assessment is faster. Documentation reduces declarative technical debt. If the purpose and use of a metadata item are understood then it can be changed. If not, there is a risk that a change could have an unforeseen impact, so a new, but similar, metadata item is created. The documentation can also be added or linked to each metadata item in

the Elements.cloud metadata dictionary. Inevitably there will be changes to the way the changes are implemented compared to how the solution was originally designed. These changes need to be reflected back into the Elements.cloud User Story so that the proposed testing strategy is correct.

Test and deploy systems

Testing

The Elements.cloud User Story defines the business need and the scenarios that need to be tested. Declarative development needs to be tested. For example, a Flow won't need to be tested to make sure that it will run, in the same way, you need to test code. But you still need to check if the logic is correct and the results are as expected. Testing will be conducted in any sandbox in the pipeline. If a test fails, it needs to be reported back and new Work Items created against the same User Story. This is because once a Work Item has been promoted from Dev, it cannot be updated or back-promoted.

Deploying release through the orgs in the pipeline

A Work Item is *promoted* from one org to the next in the pipeline. All the metadata items attached to the Work Item are moved and DevOps Center manages metadata inside GitHub, the source control system. If there are conflicts or errors that prevent the promotion thrown up by GitHub or DevOps Center they are reported inside DevOps Center. These will need to be resolved before the Work Item can be promoted. The status, org and metadata on the Work Items are kept in sync with the Elements.cloud User Story.

Drive and measure adoption

Help and training

Adoption will be easier if you have built the app that the users need, not what they thought they may want. But end-users will need training or instruction on how to use the new features. This is best achieved by providing point-of-need training embedded or attached to page layouts. Documentation in the Elements.cloud metadata dictionary can be surfaced inside Salesforce as pop-up help.

Gathering feedback

When you are dealing with an existing Org, your change cycle arguably starts (because it is a cycle that has no start) with feedback from end-users. These are best captured "in the moment" when the end-user has the issue. Elements.cloud enhances the standard

help icon by adding the ability to post feedback at object, record type, or field level. Feedback is then triaged and converted into business requirements. Ideally, you do this from the page in Salesforce where the issue or idea occurs. That way the feedback is already linked to the object or field it relates to.

Measuring adoption

If the org has event monitoring switched-on you have access to rich usage data. This requires a separate Salesforce license. The historical usage data is displayed alongside a metadata item in the Elements.cloud metadata dictionary. You can then understand whether the new features you rolled out are being accessed by the people you expected, in the volumes you expected. If not, what changes to communication, training, or features are required? And where do you need to focus efforts to elicit feedback and engage your end-users?

Migrating to DevOps Center

Challenges of Change Sets

This is going to make life for Admins exponentially easier. Those who battle change sets every day know that they are painful, frustrating, and time-consuming.

When you create a change set, you add each metadata item one by one. A new change set needs to be created to push changes between orgs; Dev to UAT, and another identical one to go from UAT to Prod. There is no option to copy a change set with all the metadata items. A change set can get big with hundreds of metadata items because it is a release. With DevOps Center you can break the release into a number of smaller, more manageable work items.

How DevOps Center works

As an Admin who is developing straight into production or using Change Sets to move metadata changes between sandboxes and production, there will be a change to how you work. This is because the aim is to drive modern best practices like source control into the deployment approach. Here are some useful videos

- [Overview and demo of DevOps Center in action including Elements extensions](#)
- [Detailed demo of DevOps Center](#)
- [How to use DevOps Center alongside pro-code developers](#)

There are 3 principles that you need to understand which were discussed earlier at a generic level. Here they are discussed wrt to DevOps Center.

DevOps Center Pipelines

This is the sequence of orgs that a change is pushed through from development to production. You may have several pipelines defined. A pipeline for high risk changes could be Dev (scratch or sandbox) – Test (sandbox) – Staging (sandbox) – Prod (production). Another might be for Hot Fixes which is just Dev – Prod. At Elements.cloud we have 4 different ones – High, Medium, Low and Hot Fix. This enables us to allocate the right level of development and testing resource to any release based on its technical, business and regulatory risk. The lower the risk of the changes, the faster and cheaper it is to get into production.

In DevOps Center you define the Pipelines and link each stage to a different org.
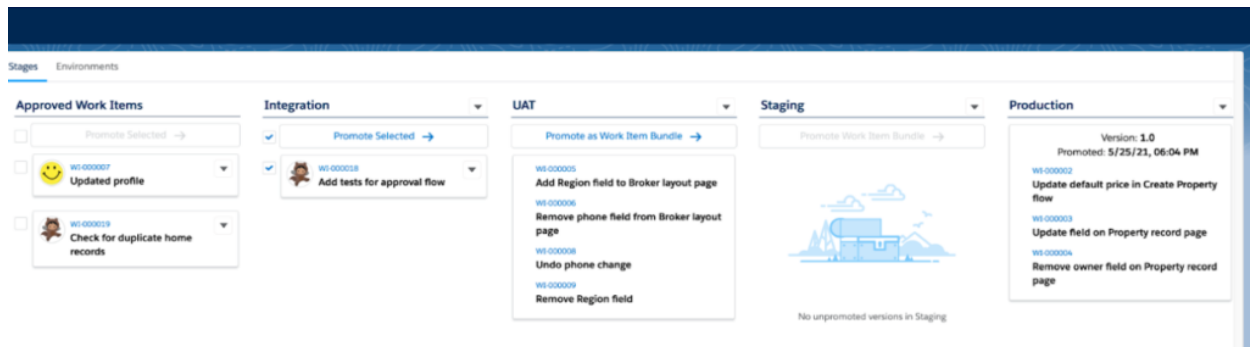
DevOps Center Pipeline

DevOps Center Work Items and Bundles

When you are planning the work you define a Release that is made up of multiple User Stories. These are developed during the business analysis phase when you are understanding the business users' requirements. The more rigorous the analysis, the better the User Story. More time spent here will reduce downstream rework and increase adoption because you are "Building the right thing". This is the work before using DevOps Center.

In DevOps Center the Work Item is the same as an Elements.cloud User Story. The Work Item is the collection of metadata that is being pushed through the pipeline. So a Work Item is the same as a Change Set, or part of it. A Change Set can sometimes be hundreds of metadata items because it is the Release. But with DevOps Center you can break the Release into a number of smaller, more manageable Work Items grouped together as a Bundle. In DevOps Center you group Work Items into a Bundle at a certain point in the Pipeline – i.e. Staging. So when you promote a Bundle it takes all the Work Items and promotes them at the same time. BTW a User Story could be related to multiple Work Items – the original Work Item and a new Work Item with changes to fix issues found in testing.
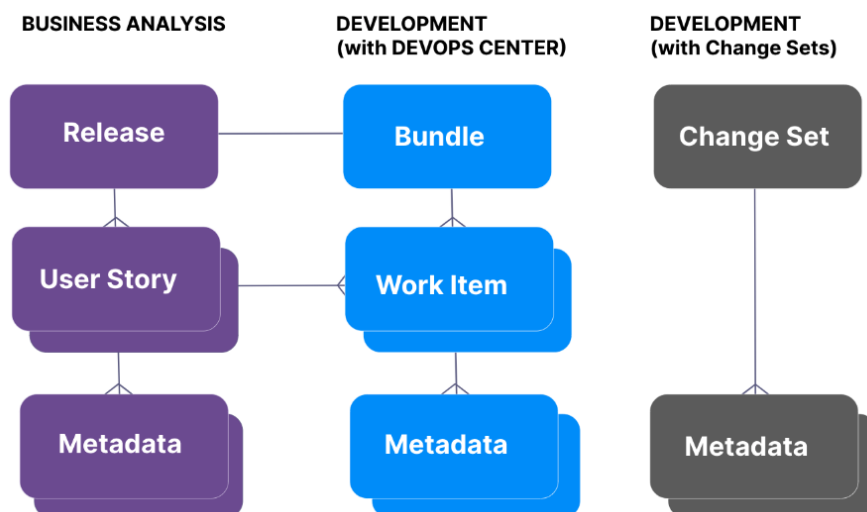
Comparing data: Business Analysis and, before and after DevOps Center

Currently, you need to create a new Change Set and add all the metadata every time you push changes from org to org. You cannot simply copy it. Building Change Sets one metadata at a time is a massive time suck and source of constant frustration. And so many times the Change Set fails and you have to keep cloning it.

We got great feedback from the early pilot users of Salesforce DevOps Center who were also Elements.cloud customers. They saw immediate benefits, but also it helped shape the roadmap.

> *With Change Sets you are ecstatic if it goes through the first time. With DevOps Center you are disappointed if it doesn't.*

> **John Eichsteadt, Platform Owner, Marcus & Millichap**

In DevOps Center when the Work Item is created and linked to a development org it creates a GitHub branch behind the scenes. You can then pull a list of all the metadata changes from that org and select them from the list to add to the Work Item. You can also selectively add other metadata items to the Work Item. You do this once, as it is the Work Item that flows through the pipeline.

DevOps Center showing a Work Item

GitHub Source Control

All the management of GitHub branches and moving metadata between branches and Salesforce orgs is managed by DevOps Center behind the scenes. GitHub is the central source control system but Admins do not need to understand GitHub and the CLI. Developers who are using CLI commands to manage GitHub don't need to make any changes now for the GitHub integration. They just need to be sure they're committing to a branch that matches the Work Item name. Planned for GA is a CLI command for promote, so they can use that command for deploys.

Using GitHub is the biggest change for the Admin but it is important because it provides a consistent way of managing changes to Salesforce and enables better governance. It is forcing the adoption of development best practices, but with all the complexity hidden away.

The good news is that you can use the GitHub free tier.

Considerations and limitations

DevOps Center is a deployment tool. It is free and it has been built as the replacement for Change Sets. So yes. IT. IS. AMAZING.

But you need to understand the scope of DevOps Center. Think of it as an elegant way to drive changes through a deployment pipeline. It is not a sophisticated commercial DevOps tool with features like backup and rollback, testing, ticketing integrations, and conflict checking. Some of these will be in the roadmap and some features will be through partner extensions.

However, the architecture is in place to allow partner-developed extensions. And the DevOps Center team has a clear view of the major roadmap items based on feedback from the pilot and closed beta users. Here are the current considerations.

- Initially it will only work with GitHub, so if your developers are using another source control system such as BitBucket, then either they will need to change to GitHub or wait until a later release of DevOps Center. Using two different source control systems is a recipe for disaster. Support for BitBucket was identified as a major roadmap item.
- The DevOps Center approach currently supports org-based development, rather than package-based development (DX), but for the vast majority of the ecosystem org-based development is the current approach.
- There is currently no integration with Jira – which is the most popular ticketing system – to keep Jira User Stories with DevOps Center Work Items in sync.
- There is no visibility of the same metadata item across different Work Items in the Release in the same Pipeline or across different Pipelines. This makes managing Work Items for the Release Manager challenging, particularly if Pipelines have the same stage set as Bundled.
- The lack of visibility of metadata item conflicts across Work Items makes resolving Github errors difficult, such as the most common rebase error. This is the error that is thrown up when a Work Item cannot be promoted because it needs changes from another Work Item to already be in place. BTW These problems occur when using Change Sets, but are not flagged because a Change Set simply overwrites the metadata, and you only find out when the org breaks. So using DevOps Center, GitHub, and source control puts in place more robust metadata management practices.

Partner Extensions

The DevOps Center team has architected it so that partners can develop extension packages. Their expectation is that partners provide additional capabilities and integrations. The DevOps Center app is installed as a Managed Package. At the time of writing it has 17 custom objects, 859 Apex classes, 15 Apex triggers and 80 Lightning Component Bundles. So it can be enhanced through an extension Managed Package. As the screens are not Lightning Pages there is no ability to drop in Lightning Web Components and that makes any UI-based integration difficult.

The first partner extension is Elements.cloud which was launched at Salesforce TrailblazerDX in March '22 in the Salesforce DevOps Center session. The extension package is providing the support that DevOps Center needs to address some of the current limitations and extend its capabilities.

If you don't have Elements already, it is a paid product. For existing Elements.cloud customers, the extension is a part of the core application, so is free.

The extension adds to the core business analysis, metadata dictionaries and impact and dependent analysis. The additional capabilities are:

- the creation of Work Items linked to a User Story from Elements or Jira
- synchronization of Jira User Stories with Elements User Stories and DevOps Center Work Items
- the aggregated view of metadata across Work Items for conflict resolution and management of releases
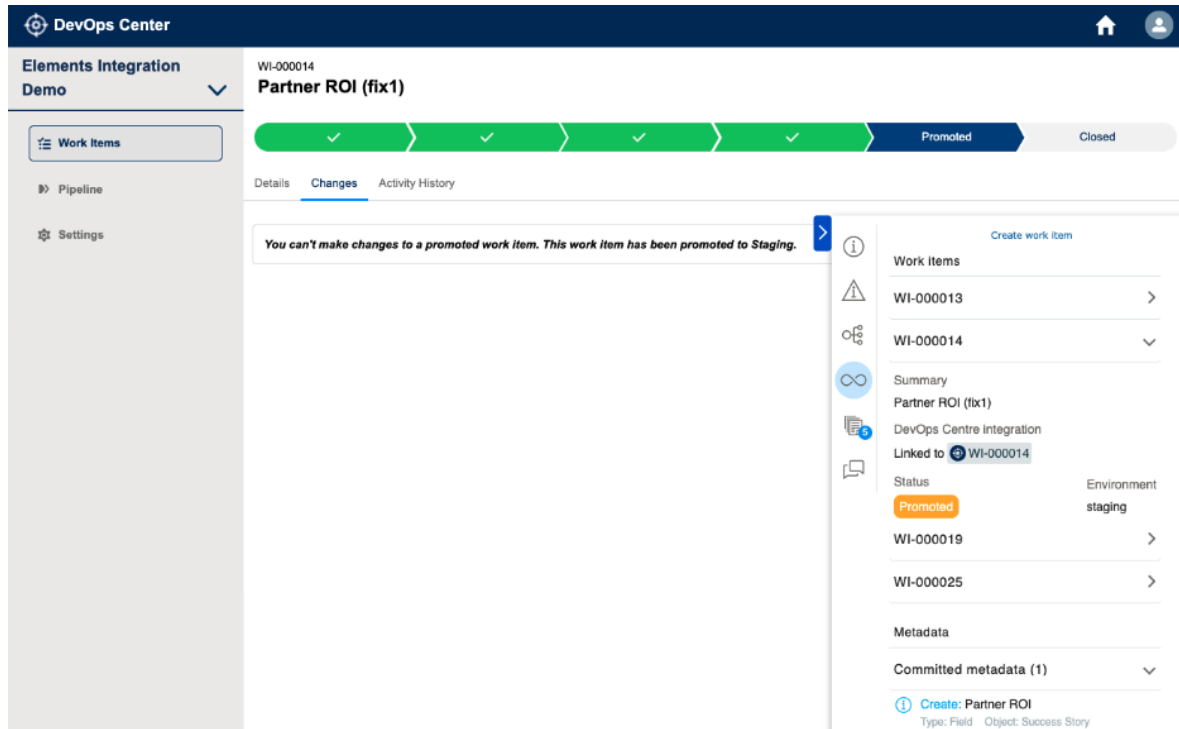
DevOps Center with Elements.cloud right panel
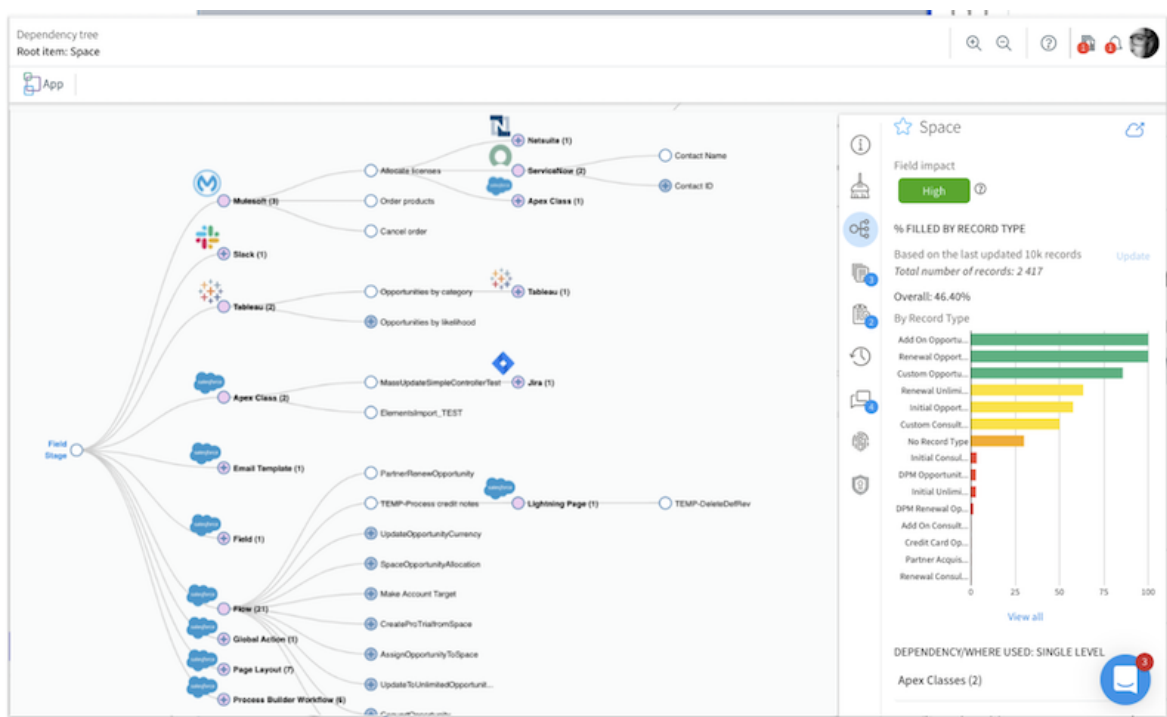
Elements.cloud metadata dependency tree and impact analysis

Tech stack and architecture

The development lifecycle needs to be supported by technology to improve productivity, scalability, and accountability. Using GoogleDocs or a patchwork of free utilities or custom objects is no longer sufficient. Salesforce development now requires a sophisticated set of functionality that supports each phase of the implementation lifecycle. It requires a single source of all change documentation and metadata so that stakeholders can collaborate. Anything less slows down delivery and adds risk.

Elements.cloud, DevOps Center and GitHub together support the entire lifecycle. The functionality required to support the lifecycle is in the table below and the architecture diagram below that.

You can see from the table below that Elements.cloud supports the earlier phases of business analysis. It also supports the integration with DevOps Center.

Feature	Phase	Salesforce Platform	Salesforce DevOps	Elements . cloud	Jira	Github
Feedback	BA			x		
Requirements	BA			x		
Process maps	BA			x		
Architecture/ ERD diagrams	BA			x		
User Stories / Work Items	BA / DevOps		x	x	x	
Releases	BA / DevOps			x	x	

Feature	Phase	Salesforce Platform	Salesforce DevOps	Elements . cloud	Jira	Github
Metadata dictionary and dependencies	BA / DevOps			x		
Development	DevOps	x	x		x	x
Testing	DevOps					
Deployment	DevOps		x			x
Help	Operations	x		x		
Adoption		x		x		

In the pilot phase we talked about a cut-down Elements license for DevOps Center. But talking to clients it became clear that removing core functionality such as metadata dictionaries and impact and dependency analysis did not make sense. They are a critical part of the DevOps process and are needed by DevOps Center for conflict identification and helping make sense of the GitHub errors.

Instead, we are offering unlimited licensing which includes the DevOps Center extension. The license cost is based on the number of end users in the production org. You can work with our team on a free proof of concept so you can build a business case. Elements.cloud/pricing

You do not need to implement Element.cloud to use DevOps Center. You can use DevOps Center on its own. Clearly, you won't have the business analysis capabilities, org dependency trees, conflict checking and integrations. Without these features, you are flying blind and are likely to make changes that will break the org.

You do not need to implement Jira. But Jira is used in a large number of organizations to track development work which is why it is shown in the table and the architecture diagram.

You must implement GitHub with DevOps Center.

The diagram below shows the architecture, integration, and flow of actions between these apps. The purple box in Jira and Salesforce is the right panel showing Elements information using the Chrome Extension
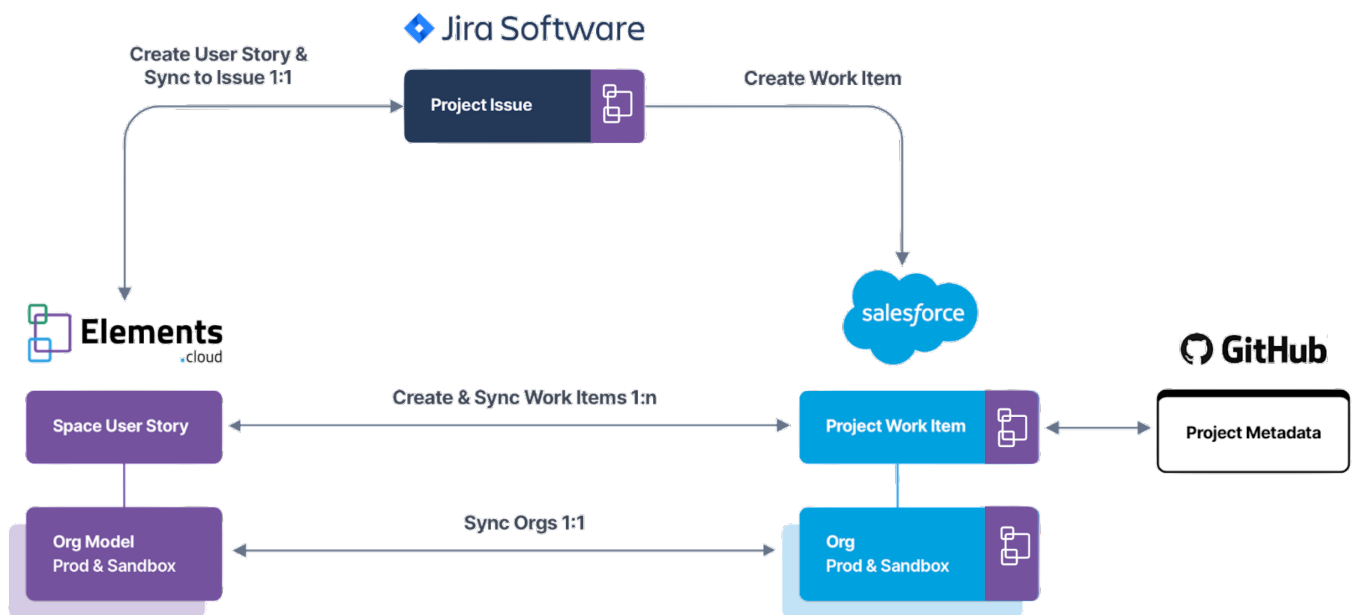
Technical architecture

DevOps implementation

Overview

The implementation below assumes that you are also using Elements.cloud and Jira. If you are not then ignore step 5 and replace it with 5a

1. Streamline implementation lifecycle processes
2. Set up GitHub.com source control accounts
3. Align Pro-code developer GitHub processes
4. Install DevOps Center Managed Package into Prod
5. Create Elements.cloud account and install core Managed Package, DevOps Extension Managed Package and Chrome Extension
6. Follow new process/approach for the next release
7. Post-release review and fine-tune process

1. Streamline implementation lifecycle processes and assign roles

As we said earlier, this is the opportunity to reengineer your implementation processes - from idea to adoption. This is not a hugely time-consuming activity. Process mapping in live workshops can be really effective at gaining consensus AND documenting the agreed processes very quickly. There is a process mapping Trailhead and this short video is a live workshop with Ryan Reynolds (actor and owner of Aviation Gin) where we map out the process for making gin.

You need to think about the earlier business analysis work, and how you risk-assess changes using the metadata dictionary and dependency analysis. That is where you will get the most value. Make sure that you *build the right thing*. Then you can use DevOps Center to *build the thing right.*

Below is a typical top level process diagram for your implementation cycle. Any activity box can drill down to the next level of detail. Any activity box can have attachments for more guidance. And you have version control so you can iterate. That's the power of UPN.
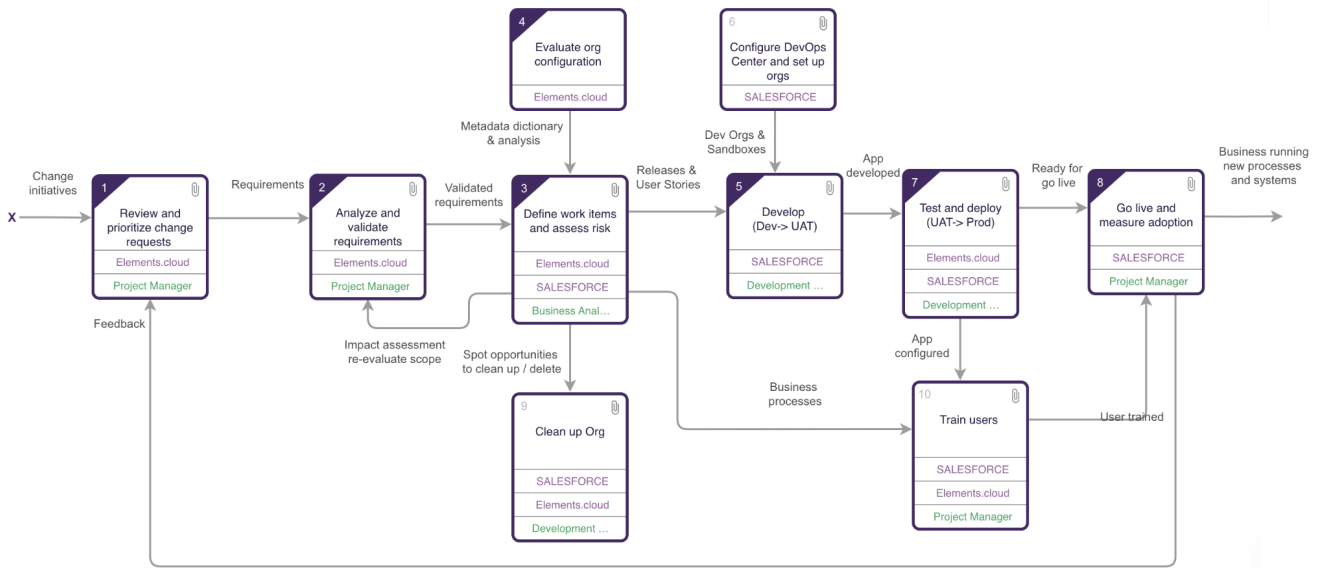
Level 1 process diagram

2. Set up GitHub source control

One of the key principles behind DevOps Center is using a source control system. The supported system is GitHub, so you need a corporate repository and create accounts for each developer. There is no external document for this, so it is explained in full here.

If your organization is new to GitHub

One person needs to be the GitHub owner. First, they need to set up their own free GitHub account. They can skip the personalizations. https://github.com/signup They need to use an email and a unique GitHub username.

Then they create a GitHub organization. Go to *Settings* in the right user account menu. Then select *Organizations* in the left menu.
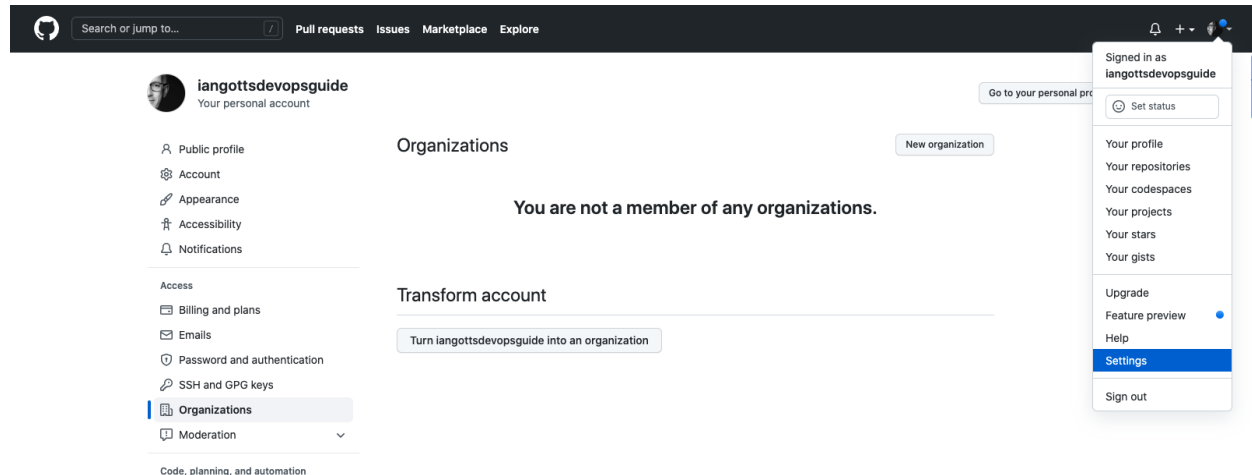
Create organization

Click the New *organization* button and select the free organizations. Then complete the form to set up a business organization.

Create organization

Each of the LCNC developers needs to create a GitHub account. They can skip the personalizations. https://github.com/signup They need to use an email and a unique GitHub username. You then invite the GitHub users to the organization, via their GitHub username, full name or email.

Start collaborating
Welcome to DevOpsGuideOrg

Add organization members

Organization members will be able to view repositories, organize into teams, review code, and tag other members using @mentions.

Learn more about permissions for organizations →

Search by username, full name or email address

> **iangotts**
> iangotts

Complete setup

Invite users to organization

Existing GitHub users

If you have Pro-code (developer) your organization will already have a GitHub organization set up and they will already have GitHub accounts.

Every LCNC developer needs to set up their own free GitHub account. They can skip the personalizations. https://github.com/signup They need to use an email and a unique GitHub username. Then they need to be invited to the GitHub organization.

If your developers are using a different source control system e.g. BitBucket, then they will need to move to GitHub or you shouldn't be considering DevOp Center yet. One of the key benefits of DevOps Center is that LCNC and Pro-code developers are all working off the same core source.

3. Align Pro-code developer GitHub processes

Your Pro-code developers will already have pipelines, branches, and scripts to manage metadata in and out of GitHub. These may need to be tweaked to align them with the new processes that were developed in Step 1. If they are not using GitHub, then they either need to move to GitHub or you should not be implementing DevOps Center.

4. Install DevOps Center Managed Package

DevOps Center is a Managed Package app and needs to be installed into an org so that it can be accessed by developers and release managers. DevOps Center will only be able to be installed in production orgs with Professional, Enterprise, Unlimited or Developer Editions. You will not be able to install it into a sandbox org. There is a new page available in Setup called DevOps Center from where the application can be enabled and installed.

Search for "devops" in the Quick Find box to install the managed package. Then follow the detailed installation instructions in this Salesforce Help topic.

# 5. Setup Elements.cloud

Elements.cloud is not required to make DevOps Center work, but there are huge benefits. Elements is an AWS app with two Managed Packages (core and DevOps Extension) and a Chrome Extension. You can run a free 2-week trial. At the end of the trial period you can run a Proof of Concept and work with us to build a business case, or purchase.

Creating Elements account and Space

Sign up at https://Elements.cloud. Create a user account where the user name is your email address. The signup wizard will ask you to create a Space which is your organization's name.

Connecting orgs

Then connect the Production org. Elements will then start building a metadata dictionary (org model) and analyzing the org impact and dependencies.
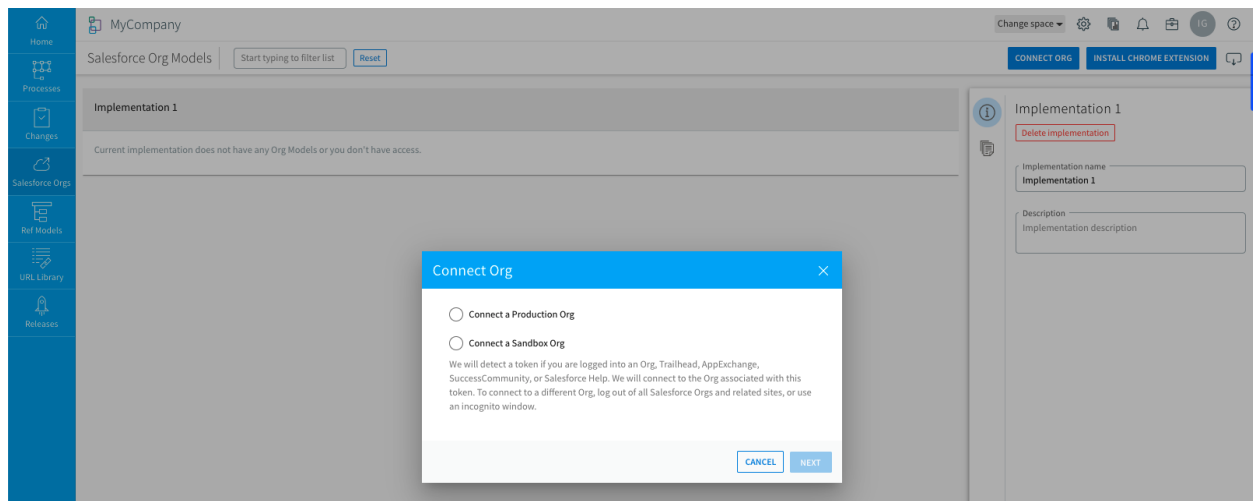

Create user account and Space

You also need to connect all the sandboxes that are in your pipeline to the Elements Space so that Elements can build metadata dictionaries for them.

Installing Managed Packages and Chrome Extension

The Elements core Managed Package and the Elements DevOps Extension Managed Package needs to be installed in the org where DevOps Center is installed. This enables the integration with DevOps Center.

To install the Elements core Managed Package, use the link on the Elements.cloud website under the LOGIN / REGISTER menu item. Also the Elements core Managed Package needs to be installed in any other org that is in the pipeline if you want field population analysis. There is a different Managed Package for Production vs Sandbox.
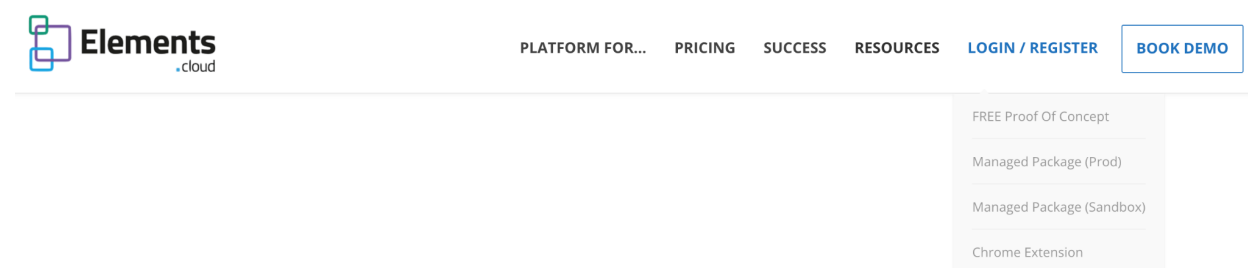
Install managed package

The Elements DevOps Extension Managed Package can be installed by contacting our Support team at success@element.cloud

The Chrome Extension is installed by every Elements user who needs to access the right panel in Setup or DevOps Center or give in-app feedback. The Chrome Extension can be rolled out to users as a standard extension. Here is the help topic. Or you can download it from the link on the Elements website under the LOGIN / REGISTER menu item.
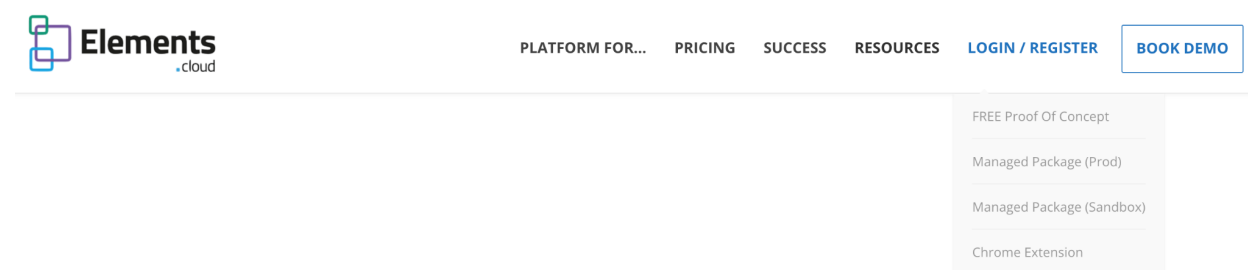
Install Chrome extension

Invite users and give access to orgs

Users can be Admin, Editor, or Viewers in the Space. You can then give them access to each of the metadata dictionaries if they require it. People will have different roles - business analyst, admin, developer, release manager, tester, trainer.

In smaller organizations, one or two people will fulfill all the roles. In larger organizations, people will probably have only one or two roles. When you invite users to Elements you need to decide on their access rights based on their roles. Users are invited in the Users menu item in Space Settings. Here is the help topic.

Jira integration

The Elements Jira integration keeps Jira projects / Elements Releases, and Jira tickets / Elements User Stories / DevOps Center Work Items in sync. Instructions to set up the Jira integration are in the help topic.

5a. Put in place Users Story and Release Management

If you are not going to implement Elements.cloud, then you need to put something in place to document the business analysis, understand your org configuration, manage your user stories, group them into releases and keep them in sync with the Work Items in DevOps Center.

6. Follow new process/approach for the next release

Great. You are ready to prove this on the next changes you are making. You can run your existing change processes in parallel, so you don't need to wait for all the current projects to finish. However, don't pick anything that is too big or visible for the first set of changes; CPQ or a set of board dashboards. Pick something that is small enough to be manageable, but large enough to prove the processes.

7. Post-release review and fine-tune process

After the first week or two of using the process, schedule a review workshop with all those involved in the process. Use the process maps you created to drive the meeting and make changes or add sticky notes to the draft version as you identify improvements. If you are using Elements then you have sticky notes, commenting, and version control.

Resources

Salesforce DevOps resources

[Customer DevOps Center Trailblazer Group](#)

[Partner DevOps Center Trailblazer Group](#)

[Release Notes](#)

[Installation and Configuration](#)

[User's Guide](#)

Links in this document

[DevOps Center installation](#)

[DevOps Center demo video](#)

[Trailhead GitHub Basics](#)

[GitHub signup](#)

[Trailhead Process Mapping](#)

[Salesforce BA Certification guide](#)

[Salesforce Architect site](#)

[DevOps Launchpad (Gearset)](#)

[Elements trial set up](#)

[Elements user management](#)

[Elements Jira integration](#)

Feedback

If you have feedback, find issues or feel this Guide needs more information in any area, then please let us know success@elements.cloud