



How To **Evaluate A** **DevOps** Solution for Salesforce

Introduction

📄 Evaluating which DevOps solution is right for your organization is an important process that can affect productivity, data security, production costs, team morale and more. And because there are many solutions on the market to choose from, comparing and contrasting them all can become quite overwhelming.

👍 In this article, we'll point out some important areas to consider and questions to ask when evaluating which DevOps tools is right for you, in hopes to provide a solid foundation for your search.

Security



All vendors will tell you their solution is secure, however it is up to you to determine if this is true. When conducting your search, we suggest looking at security first - taking a close look at each solution's architecture and completing any security screens needed before diving deeper into the **features and functionality**. You will find this helps you very quickly narrow your search and

prevents you from wasting time evaluating tools that in the end do not meet your needs. Furthermore, installing an insecure solution even just for a trial essentially exposes all of your data. Some important **questions to consider when evaluating a solution's security** are:

Is the Solution really “native” to Salesforce?

Solutions that are built specifically for the force.com platform are able to handle its unique requirements easily (for example the processing of lightning components and static resources) and generally have the most robust

security because data never leaves the platform. However, this is also a tricky area, because some solutions out there claim to be native, but in fact are built on third party platforms that just happen to be owned by Salesforce.



Will the Vendor have access to your data?

It is a common misconception that DevOps tools only move code and meta data and therefore won't have access to your data. In actuality, **all solutions migrate data in one form or another** - and if they are built on AWS or Heroku, this becomes a a huge security risk.

Does the tool require backdoor access to your production org? Do you need to whitelist IP addresses in order for the tool to perform?

While many DevOps tools claim to be "secure," their architecture is built in such a way that they require backdoor access to the production org to function. Whitelisting an address is essentially like giving someone the key through your firewall and into all your internal business functions. **In fact, it is Salesforce's security best practices documentation actually suggests that you shutdown all the IP addresses and marketing access to any other vendor.**

Compliance & Attestations

Are there any industry regulations that need to be met? If so can the tool provide them?

Healthcare institutions, government agencies, financial organizations, or any kind of company that stores any kind of personal customer information need to pay special attention to their specific rules and requirements. Common certifications needed in healthcare for example include PCI and HIPAA. These types of institutions will not be able to utilize a dev ops tool that runs on AWS or Heroku because of their requirements.

What are your data residency requirements?

Many organizations, and even some countries have very strict data residency requirements. In Canada and Singapore for example, it is illegal for data to leave the country. And because many DevOps tools are hosted on AWS or Heroku who utilize offshore resources, this presents an inherent conflict.

Who will have access to your data?

The personnel accessing customer data at Salesforce are all located in the United States and properly vetted and undergo extensive background checks. However, many DevOps tools utilize AWS or Heroku, which do not maintain same standards and are often offshore. If any part of the tool is outside of Salesforce, know there is a risk it may end up in wrong hands.

Technology



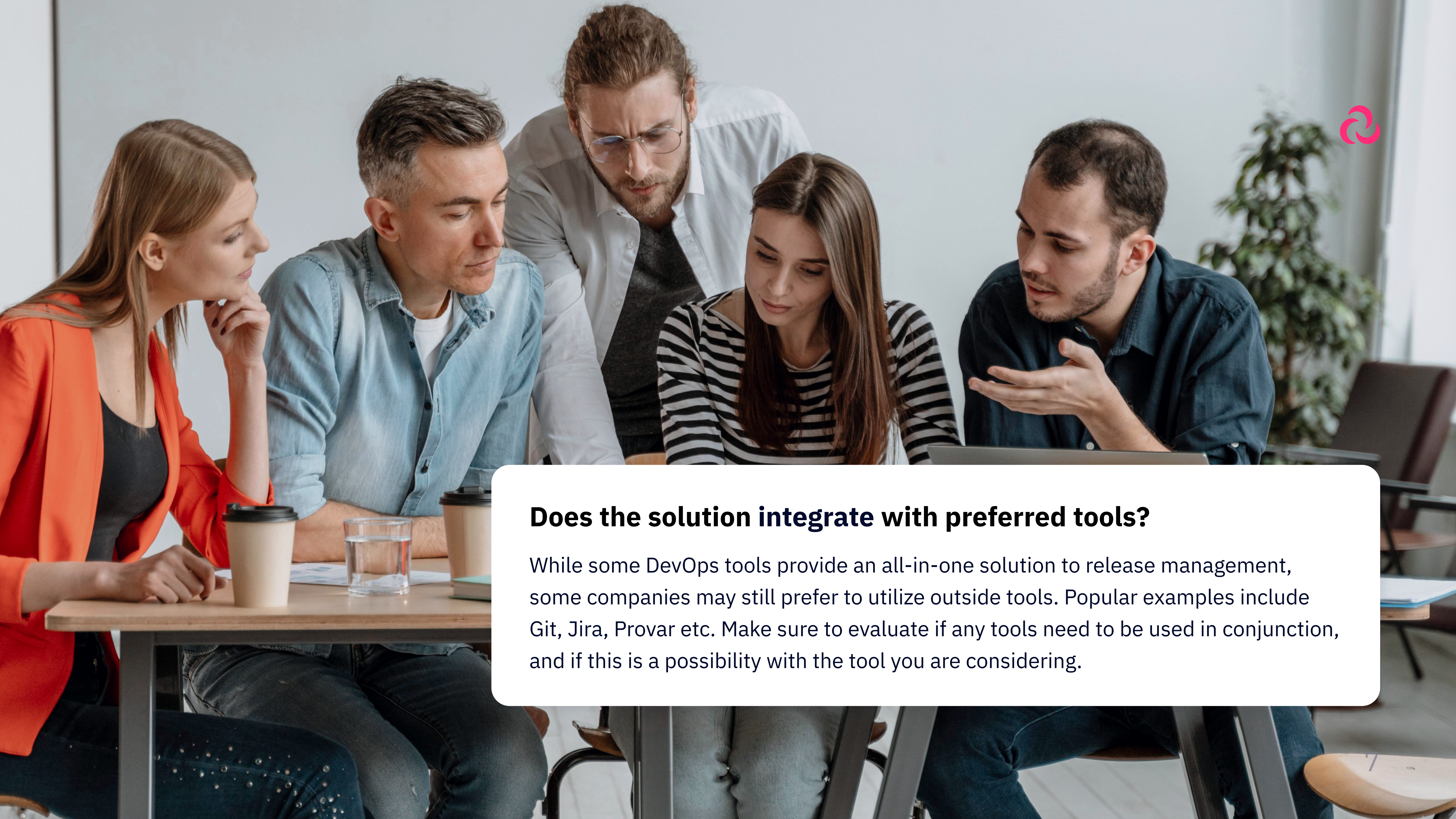
Does the tool require backdoor access to your production org? Do you need to whitelist IP addresses in order for the tool to perform?

Salesforce application development and testing is a multifaceted process and traditionally many tools were used to cover different aspects such as environment management, data migration, conflict management, code repository, regression testing etc. However, because these solutions are separate and essentially “duct taped” together they are sometimes inefficient and incompatible.

- How many tools will be needed to complete my DevOps Cycle? For example, if I go with this Release Management tool will I need a separate Version Control Solution or Merge Editor?
- Are these tools able to easily integrate with each other?

Furthermore, the force.com platform is very unique in terms of architecture. It is completely cloud based and rather than developers writing code on their desktop and then committing it to Git, all development takes place directly in the development sandbox. And while certain tools are built to compliment this unique architecture, covering the entire process, some are meant for other types of development and cannot handle these unique needs. When evaluating which release management tool(s) are right for your process it is helpful to consider the following:

- What is the added cost of a multiple tool set vs. an all-in-one solution?
- How much training is required for your team to be able to utilize all of these separate tools?



Does the solution integrate with preferred tools?

While some DevOps tools provide an all-in-one solution to release management, some companies may still prefer to utilize outside tools. Popular examples include Git, Jira, Provar etc. Make sure to evaluate if any tools need to be used in conjunction, and if this is a possibility with the tool you are considering.

Extendability & Customization



Is the tool extendable?

Each organization is unique in the way their DevOps environment functions, and each has their own set of rules that govern their procedures. For example, one organization may only deploy changes into production on Fridays between 5-7pm, while another can only deploy certain components into production. It is important to determine if the tool you are looking at can accommodate those types of needs.

What is the level of expertise required to make customizations to the tool?

Solutions which are built on AWS or Heroku cannot be customized directly by users because they are written behind the scenes on the java platform. Some cannot be customized at all.

What is the cost to make customizations?

Be sure to note if any customizations needed are included with the list price or if you will be charged for them separately.

Solution Usability & Ease of Adoption

Is the solution easy to use for all team members? Is knowledge of Git required?

A DevOps tool should be useful for the entire team. Most development teams include citizen developers who are able to make declarative changes, but are non-technical in nature. These users cannot commit their changes to Git or utilize many of the Git based functionalities. Not only does this end up creating more work, but also leads to team members overwriting each other's changes. Furthermore, if your team utilizes external consultants, it is important to find a solution that allows for quick adaptation, as these team members are often coming and going.

Does the solution support the size of your development team?


Because of the limitations of some tools, they are not scalable to enterprise level organizations. If you are part of a large organization, make sure to ask about the solutions scalability and how their solution caters to large development teams.

Is the User Interface user friendly?

When evaluating a solution it is a good idea to take a close look at the User Interface. Some solutions have built their own, while some utilize Salesforce's UI. Because all team members utilize Salesforce it is generally an easy transition, while custom built UI's can become more difficult to navigate.

Specific Functionality to Consider

There are certain areas within the DevOps world that have historically caused major pain points for development teams. When evaluating the last few tools that are on your list, we suggest doing a trial of each one and paying careful attention to these areas. How does the tool handle these problems? How complicated is the strategy? How many steps are involved? How long does it take to complete?

 Following is the list of areas we suggest considering.



Merging Conflicts

Merging conflicts can be an extremely painstaking and time consuming process for developers, and **many tools cannot handle the merge of many of the declarative components that Salesforce utilize such as lightning components, static resources and aura definition bundles**. Here are some helpful questions to consider when evaluating a tools merge functionality:

- Does the solution provide a merge editor?
- How tedious is it to merge branches? How many steps are involved?
- Must the conflicts be resolved manually?
- Can the solution do declarative changes - for example can it merge lightning components, static resources and Auradefinition bundles? If so, how?

Branching Strategy

Keeping branches in synch is another painstaking and time consuming process for developers. Here are some helpful questions to consider when evaluating a tools merge functionality:

- How much time is required for your team to maintain the branching strategy?
- What happens if the branches are out of synch? How long will it take to re-synch the branches?
- How many merges/syncs are required to move code from the developer's sandbox from QA to UAT to production?
- How many environments must be maintained in order to keep code in synch?

Sandbox **Syncing**

Development sandboxes need to be in sync with what is in production so that developers can work on the latest code. Consider the following:

- Does the tool you're evaluating provide a solution for this?
- What is the process like?

Out of Sequence **Deployments**

There are times when developers may want to deploy their commits out of synch. Say for example a developer has commits 5 - 8, but only wants to deploy commit 6 & 7. Some tools branching strategies will support this, others will not.

Data **Migrator**

The process of migrating data downstream from the production org into sandboxes or upstream is a necessary process for Salesforce developers.

This allows developers to work with actual data to test their code functionality. And while some tools offer data migration, they do not maintain **hierarchical relationships when moving it**, which makes the process extremely painstaking. Consider the following:

- Does this solution include a tool to migrate data?
- What is the process like? Are hierarchical relationships maintained?
- Does your data move to a third party platform when migrated? (As we mentioned before, any solutions that offer Data Migration actually move the data from your production org and into the third party platform they are hosted on and then back into the desired target org, which presents a huge security risk).
- Can the solution scramble and mask migrating data for added security?

Implementation & Support



DevOps tools vary greatly in the **amount of service that is provided**. While some tools have 24x7 support, others are limited to certain hours in certain countries.

Some are “self-serve” and do not offer live support at all. It is important to know what this looks like when evaluating a tool. Here are some questions to consider:

→ How often will the solution check in to make sure we are successful?

- What are the hours which I will be able to reach the support team?
- Will we have a dedicated support person for our team to guarantee our success?
- What does the solution’s support model look like?
- What is the implementation and training process?
Is this included when we purchase the product or is there a separate fee?

Cost



Cost is **always a factor** when evaluating any solution for any business. When evaluating this area, be sure to consider the following:

→ What is my total cost of ownership for the tool?

Think not only of the solution and licensing, but be sure to consider maintenance, training, server hosting etc. as well.

→ What is included in my purchase? Are there hidden fees? Do I have to pay extra for provisioning of licenses, implementation, support, customization etc.?

→ How many functions does this tool provide?

→ If I need to purchase multiple tools what is the cost of them added together vs. an all-in-one solution? If solution(s) require Git, what will be the cost to train the entire team on Git?

→ What is the maintenance cost? total cost of ownership-licensing, training, server, hosting?

→ How will this tool impact the cost of business - Will I save money in increased developer productivity? Will it help reduce my compliance cost?

Extendability & Customization



Is the tool extendable?

Each organization is unique in the way their DevOps environment functions, and each has their own set of rules that govern their procedures. For example, one organization may only deploy changes into production on Fridays between 5-7pm, while another can only deploy certain components into production. It is important to determine if the tool you are looking at can accommodate those types of needs.

What is the level of expertise required to make customizations to the tool?

Solutions which are built on AWS or Heroku cannot be customized directly by users because they are written behind the scenes on the java platform. Some cannot be customized at all.

What is the cost to make customizations?

Be sure to note if any customizations needed are included with the list price or if you will be charged for them separately.

Technology



Does the tool require backdoor access to your production org? Do you need to whitelist IP addresses in order for the tool to perform?

Salesforce application development and testing is a multifaceted process and traditionally many tools were used to cover different aspects such as environment management, data migration, conflict management, code repository, regression testing etc. However, because these solutions are separate and essentially “duct taped” together they are sometimes inefficient and incompatible.

- How many tools will be needed to complete my DevOps Cycle?
For example, if I go with this Release Management tool will I need a separate Version Control Solution or Merge Editor?
- Are these tools able to easily integrate with each other?

Furthermore, the force.com platform is very unique in terms of architecture. **It is completely cloud based and rather than developers writing code on their desktop and then committing it to Git, all development takes place directly in the development sandbox.** And while certain tools are built to compliment this unique architecture, covering the entire process, some are meant for other types of development and cannot handle these unique needs. When evaluating which release management tool(s) are right for your process it is helpful to consider the following:

- What is the added cost of a multiple tool set vs. an all-in-one solution?
- How much training is required for your team to be able to utilize all of these separate tools?