suyati
access to fluid cx

Continuous Integration in Salesforce using Salesforce DX

suyati
access to fluid cx

TABLE OF CONTENTS

INTRODUCTION

CONTINUOUS INTEGRATION

Change is the only constant in the software ecosystem. However, changes also mean the risk of introducing new or functionalities that could impact existing code. Testing, to exclude bugs, is a laborious, time-consuming and resource-intensive process. Running extensive tests everytime a change is proposed is not practical.

Enter continuous integration. Continuous Integration (CI) is a software development approach of automating the test suite, to activate all integration and unit tests run on every merge or pull request, automatically. The approach pre-empts problems post-merge, by unearthing it upfront. Early fixes save time, money, and effort, and ensure robust code which makes customers happy.

HOW DOES CONTINUOUS INTEGRATION WORK?

Continuous Integration automates the test process and makes it simple, making it viable to run tests every time a change is implemented.

- Continuous Integration works by triggering an automated process every time the developer introduces new code to an application. The automated process residing in the CI server grabs the code from the shared repository, integrates the new build, and tests the software for problems.

The essential tools to get started with Continuous Integration are:

- Salesforce DX: Developers commit changes locally in Salesforce CX, and upload it to Travis CI or any other continuous integration server used for the testing process.
- GitHub, or the code hosting platform for collaboration and version control.
- A continuous integration server which integrates well with GitHub. Travis CI is the most popular CI server for Salesforce, though other CI servers work just as well. The script to trigger and run the automated testing process resides in the Continuous Integration Server.
- The master code resides in the GitHub repository, accessible to all developers. Each developer "checks out" code to their sandbox in Salesforce DX, makes the required changes and pushes the revised code back into the repository. When such new or updated code enters the shared repository, it triggers the automated CI process, running the integration tests. The tests make sure the newly inducted code does not break another part of the software or renders the legacy code unworkable. If the tests turn unsuccessful, the CI server generates the details of the failure, pinpointing where errors occurred. If there are no errors, or when developers rectify the errors, the updated software may be released manually or even scheduled for automatic release. DevOps teams may take automation to its logical conclusion by having the system deploy and deliver the product automatically.

- Such a process is easy to run and the process gets over fast, making it viable to run it multiple times in a day. The code is only merged once, after the build passes testing, eliminating the chance of breaking the master code. The process runs in the background, allowing developers to carry on with other tasks.

SET-UP

There is no fixed or specified set of software for implementing Continuous Integration. However, implementing Continuous Integration requires a methodological approach

Step 1:

- Decide on the Approaches and Models
  An important decision to take upfront is the environment strategy or selecting the right environment for the right task.

- Set the development environment, test environment and production environment upfront. While traditionally testing is done in a sandbox, adopting Scratch org in the developer environment and testing environment allows testers to start in the same environment as a developer, with no difference in the template or any other variables. For large and complex projects, having a branching strategy that aligns with the development environment makes life a lot easier for the developer.

- There are two broad models to run Continuous Integration. The preferred model depends on the development approach. The package development model is the best option for teams using scratch orgs and unlocked packages to deploy their changes. The org development model is the better option for teams using sandboxes and Metadata API commands to deploy their changes. Salesforce DX adopts scratch org, making the org development model a better option.

Step 2:

- Select and Deploy any Continuous Integration Tool of Choice

- Continuous Integration runs through a Continuous Integration tool that connects with the version control system. The tool contains the script that triggers the tests when the CI process is invoked from the shared repository. Travis CI is the most popular tool among Salesforce developers. However, other tools from different vendors also work fine. The development team may select any Continuous Integration tool of choice, and place it in the Continuous Integration server.

- A version control repository is an optional inclusion, but an important safety net if things go wrong.

Step 3:

- Set Rules and Select Tests

- Having selected the Continuous Integration tool, build rules specifying the tests to run on triggering Continuous Integration. The main tests are feature testing, integration testing and performance testing. The order of tests is not set in stone and depends on the nature of the app.

- The test is run when committing the code to scratch org. The automated server pulls changes into source control, and run the tests. Salesforce DX Command Line Interface (CLI) delivers breakthrough integration and automation capabilities, making the process of automated tests very easy and stable.

- Create a test delivery pipeline, specifying different tests for different situations, and running the test when it should be run. The success of the entire approach depends on selecting the right tests, to run at each stage.

Step 4:

- Make Changes

- Having deployed a Continuous Integration tool and set the rules, developers may work normally, introducing changes as and when they like.

- Using traditional sandbox allows each developer to maintain their sandbox, and committing the code to the version control system, where the CI script is triggered. All tests run in developer sandbox. When the developer introduces a change to the repository or does something to trigger the test, the Continuous Integration tool runs a script in the background. The script configures the environment, deploy the changes, and run the test suite.

- On completion of the test each time, the tool generates a log containing details on the tests passed and failed. Configuring the pull request rules to ensure changes are not merged until all builds pass pre-empts the introduction of new bugs into the code. After each test is done, the scratch org, no longer needed, is retested.

3RD PARTY APPS

- Several vendors such as AppVeyor, Bamboo, BitPucket, CircleCI, GitLabs Pipelines, Jenkins, and Travis CI offer depositories. Each vendor offers subtle differences and nuances. For instance, CircleCI, one of the most common tools, usable either as cloud-based or on-premises, integrates with the existing version control system to push incremental updates to the specified environments. The Lightning platform supports any CI tool, though Travis Ci seems to be the preferred choice among developers.

- Each repository offers a readme offering instructions on setting up the CI vendor's tool with the SFDX project and the version control system. Most vendors also offer a sample CI configuration file, making it easy for uninitiated developers to get started with Continuous Integration.

- Vendors charge either by Docker image or for minutes used. Minutes used may be a cheaper option when running fewer tests per build. However, when the number of tests increases, paying per Docket image may become a cheaper option. Some vendors even offer a free business plan, including free hosting, and public version control repositories. While such plans may offer limited functionality, it is a good choice when getting started.

BEST PRACTICES

- Continuous Integration has increased in popularity because it works. Developers may optimize the effectiveness of the approach by adopting some proven best practices:

- The essential requirement for the success of Continuous Integration is maintaining a single source repository with easy access for developers and making sure every build occurs on an integration container or VM.

- Using the Continuous Integration script along with a Version Control System (VCS) allows retaining the legacy code if for some reason the development team wants to revert to an older version. Deploying version control, though not essential, is an important safety net, allowing developers to fall-back on an earlier version if something goes wrong. Version Control allows switching back and forth between versions.

- Deploying parallel connectors and virtual machines allow automating and running tests and build quickly, for even long-running processes.
- Continuous Integration servers may be hosted either on-premises or in the cloud. Unless the enterprise has the required IT resources, skillset and budget to access private network resources, opting for cloud-hosted servers is the better option, any day.
- Perform testing in a test environment, such as a sand-box or scratch.org. Creating a disposable environment such as a scratch org to test specific code or changes is preferable to introducing changes to a shared org.
- Salesforce CLI makes it possible to automate the creation of scratch orgs as part of the CI process. With Salesforce CLI, it is possible to completely script all tasks in the CI configuration file, regardless of adopting org-based or package-based development
- An easy way to get started quickly with Continuous Integration is by cloning a sample repository from any vendor of choice.
- For large and complex projects or geographically dispersed teams, setting a branching strategy may help reduce complexity. Different branches allow each developer to makes changes and run tests in their branch. The Continuous Integration process takes over after each branch runs its testing.

CONCLUSION

Continuous Integration is increasingly the integration method of choice. The popularity is fuelled largely by the speed at which it fixes bugs and enables the release of new features. Also, Continuous Integration spares developers the hassles of having to backtrack to identify code breaks, in the process saving considerable time and resources. Developers get to know where exactly problems occur, facilitating a much faster response and repair. Continuous Integration in Salesforce CX results in 20% savings on code reviews and 40% fewer bugs on average, minimal hotfixes, and no extra time required to deploy code. The continuous testing process also means programmers never work on out-of-date products or fall to the temptation of pushing changes hurriedly to meet deadlines or demand.

Write To Us

REFERENCES

- https://developer.salesforce.com/blogs/2019/05/continuous-integration-with-salesforce-dx.html
- https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_ci.htm
- https://trailhead.salesforce.com/en/content/learn/modules/sfdx_travis_ci
- https://www.youtube.com/watch?v=wUc1l5keYmo
- https://www.youtube.com/watch?v=lLfTPLGEwYI
- https://www.youtube.com/watch?v=VLl1uUPF97g
- https://www.youtube.com/watch?v=8obwIwvzmMw
- https://www.appdynamics.com/blog/engineering/continuous-integration-works-big-benefit-no-one-talks/
- https://developer.salesforce.com/blogs/2019/05/continuous-integration-with-salesforce-dx.html
- https://developer.salesforce.com/docs/atlas.en-us.sfdx_dev.meta/sfdx_dev/sfdx_dev_ci.htm

RELATED EBOOKS:

- Getting the Best Out of Salesforce through Customization
- 7 Questions to ask before a Salesforce lightning
- Salesforce Workflow: Why your business needs it

suyati
access to fluid cx

Founded in 2009, Suyati Technologies partners with clients to engineer great experiences for digital customers. We collaborate with businesses to strategize and implement impactful digital initiatives that position our clients ahead of the competition. We are digital-first and we focus on delivering great customer experiences that accelerate exponential growth.

Our custom technology solutions ensure that you win stakeholder support, secure early wins through competitive advantage, and transform your business for future growth. And our tailor-made platform, Mekanate, helps you discover your business DNA from your passive and active data, and use it to initiate, integrate and achieve operational efficiency.

With our niche and rich expertise in a wide range of technologies and services - CMS, CRM, e-commerce and Marketing Automation. We help companies across the globe leverage their best on web and cloud platforms.

Learn more: www.suyati.com

Get in touch: services@suyati.com