White Paper(Draft)

Continuous Integration/Delivery/Deployment in Next Generation Data Integration

Ranjith Krishnamoorthy
Informatica Professional services – May 2017

Contents

Introduction **3**
Challenges **3**
Continuous Methodology Steps **3**
Continuous Integration 4
Code Build 4
Code versioning 5
Automated Code review and Analysis 5
Automated Testing 5
Automated Test Results Alerting 6
Automated Artifact Storage 6
Continuous Delivery 6
Automated Code deployment 6
Automated Regression/Integration testing 7
Continuous Deployment 7
Integrating all the steps through a CI platform 7
Reference architecture for Continuous Deployment on Informatica PowerCenter DI platform **8**
Use case 8
Pre-requisites 8
Solution Architecture 9
Benefits 10
Summary **11**
Further Reading **11**

Ranjith Krishnamoorthy
Informatica Professional services – May 2017

Introduction

As enterprises demand faster go to market and reduced costs in doing so, Enterprise IT is also adapting to it by turning to Automation in many areas. The most prominent trend is the automation of code build and deployments. Continuous integration (CI) process and tools encourage IT teams integrate software code modules and build the entire application package more frequently. This helps in identifying code integration issues faster. Continuous Delivery/Deployment (CD) process and tools help IT team run multiple tests on the built code and deploy to higher environments in an automated fashion. This helps in delivering applications to production faster hence helping businesses in their outcomes.

This white paper explains the challenges encountered in implementing a continuous methodology on a Data Integration tool based development environments and examines the capabilities an Enterprise grade Data Management should have to overcome these challenges. Further the white paper introduces a Solution Architecture with Informatica's industry leading Data Management products and deliver more than 50% reduction in application development/deployment time for certain use cases.

Challenge

The Continuous Integration/Delivery/Deployment practices are very common in the industry now and is probably safe to say that it will become de facto standard in the near future. Both concepts mentioned above were developed mainly for a Computer programming language based application development environment (Java, .Net etc). When it comes to packaged tools in the Data Integration domain achieving CI/CD needs a little bit more effort and planning. Below are the major challenges in implementing a CI/CD process

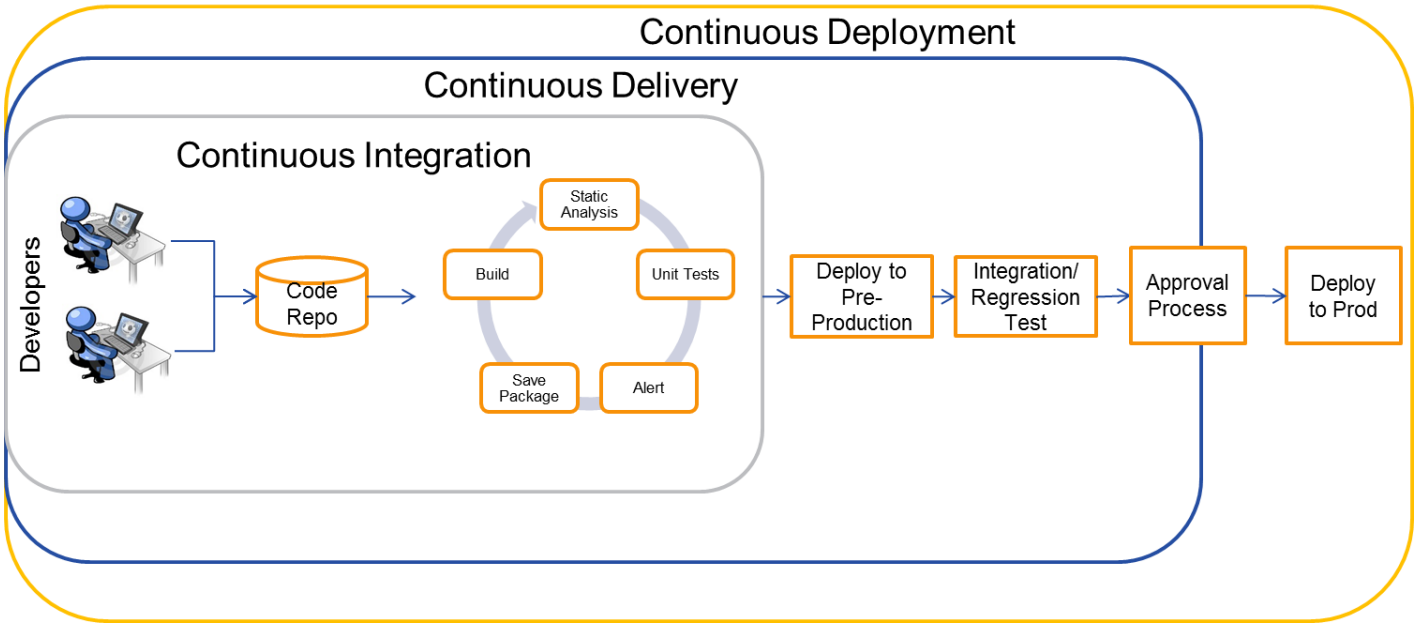
- Data Integration tools have their own code repository and version control. These may not be readily integrated with the CI/CD tools available in the market
- The repository structure and versioning features don't support a Branching/ Merging strategy which is very essential for a clean code integration when multiple developers are working on a single application

Even with these challenges CI/CD is still possible with a DI tool for certain use cases albeit with some one-time Custom coding to deliver certain functionalities.

Continuous Methodology Steps

Below figure shows the steps involved in a CI/CD environment. Each outer block represents the level of maturity achieved with automation of the development process

Ranjith Krishnamoorthy
Informatica Professional services – May 2017



A detailed review various steps involved in Data Integration development reveals challenges involved in these steps and solutions for each of these

Continuous Integration

Code Build

Automation of build is probably the most complex step in the Continuous methodology when It comes to DI platforms. However, it is possible if the platform used comes with certain key capabilities

Enterprise grade Data integration tools are mostly GUI based drag and drop tools. This is their biggest selling point which avoids hand coding and helps IT teams in delivering DI applications more efficiently while future proofing them. So implementing an automated build is not straight forward and varies greatly depending on use cases. Designing pre-defined design patterns that can be replicated over and over again is the fundamental requirement for any code automation solution. Below are some options available to automate code build

Build Trigger

Build trigger is typically a commit to a Source Code Management (SCM)repository. CI platforms offer hooks on SCM tools to polling for commits and trigger builds. In a DI tool based environment this is difficult and may be impossible. An alternative approach would be to query tool repository contents for changes. Now in case build is completely automated, they need extensive input attributes (metadata) and custom code may be needed to accept and process this input.

Templating Tools

If the DI tool deliver templating capability leveraging third-party tools like Microsoft Visio to create templates automation of build is very easy. Coupling the template file with a parameter

file multiple DI pipelines can be built programmatically. Creating and publishing template is a onetime activity for each design pattern.

Platform SDK

If the DI tool include SDK's in language like Java it can be used to generate DI code in an automated manner. They could either directly create objects in the tool repository or create files (mostly XML) that can be imported in to the repository. Using SDKs is probably the best option available which safeguard against version update changes.

Generating code end to end in programmatically readable format like XML

Alternate approaches to automated build involves generating the entire DI code in a standard format like XML and importing them to the Tool repository. Although theoretically possible the challenges involved in generating then entire code structure programmatically is considerably high. This approach also comes with some amount of risk that between version to version xml file structure may vary and any code written to parse/create the xml files may need to be re-written.

Code generated in any of the approaches mentioned above can be imported in to the Tool repository programmatically using Command Line features available in the tool. Practically a combination of Templating tool, and a data processing engine that can process the generated code file presents a better chance of succeeding in build automation

Code versioning

Enterprise quality DI tools come with internal code versioning feature. Mostly they are maintained in a relational database repository which may not have hooks available to integrate with CI platforms. However, if they deliver capability to export the code contents in formats that are programmatically readable like XML, they can be versioned separately in SCM tools like Git, Subversion etc which are more readily integrated with CI/CD tools

Automated Code review and Analysis

In a CI workflow this step is important as well where newly built code is scanned programmatically to identify coding standard violations and other issues like code duplication etc. A complete data management vendor should have products/modules delivered to cover this functionality as well. Depending on the maturity and popularity of the Platform, Vendor marketplace may have partner tools which can do an automated code review after deployments.

Automated Testing

Running pre-built test cases in an automated manner to do unit testing gives feedback to developers quickly and help deliver better quality code in to higher environment faster. Enterprise data management platforms should deliver options to create data validation test cases using GUI. If they come with an option to import/export these test cases in programmatically readable file formats like XML, test cases can be generated automatically with the metadata used for automating the build. For test cases which can be repeated

multiple times, even creating these manually would give considerable time savings and can be integrated in to the CI workflow. One hidden sub step here is to execute the Data Pipelines in an automated manner. DI platform's Command Line Interface (CLI) can be used to accomplish this step quite easily.

Integration with Test management tools would also be beneficial for centralized tracking if the Testing tool provides functionality to do so

Automated Test Results Alerting

Test automation/Test management tools should provide standard functionality to do alerting and generate reports based on the test results. Custom code could be written to fetch test results stored in the testing tool. Depending on the easiness of implementation one of the above option can be chosen

Automated Artifact Storage

Artifact repositories help IT organizations store and share binaries/libraries amongst developers and also help in compliance adherence by looking at license information of third party binaries. They also help in reducing SCM repository bloat. Since most of the artifacts generated from the DI tools are in non-binary format, in an ETL/Data Integration development environment this step may be mostly irrelevant. This step could how ever be replaced by Code check in step where a logical bundle of code in tool repository can be exported and stored in the SCM repository itself.

Continuous Delivery

Once above mentioned CI process is matured more benefits can be achieved by implementing a Continuous Delivery methodology. The journey towards Continuous methodologies could start at this point at well if automation possibilities mentioned above do not exist. In the absence of code branching strategy, alternative approaches do exist to get multiple pieces of mutually exclusive code developed by various developers. DI platforms should offer capabilities to isolate each developers coding workspace within the tool repository. Code developed can then be combined in to the master workspace

Automated Code deployment

In packaged DI platforms, deployments can be done in multiple ways. They mostly come with deployment utilities which helps moving code artifacts from one repository to another. Although this might be the most common approach for many IT organizations, it might not be very easy to automate deployments and integrate with CI/CD tools. An obvious alternative to that approach is to use the exported code files from the automated artifact storage step discussed above. These files can be imported in to the target repository programmatically using DI tool's CLI

Automated Regression/Integration testing

This step is an extension of the automated unit testing discussed in the Continuous integration. Typically for data integration applications testing at this stage focusses largely on Data Testing with a data set that resembles production or a sample that covers all the data scenarios expected in production. One of the major challenges faced by IT organizations is getting a valid data sample in the test environment in a timely manner without violating compliance requirement. Investing in a Test Data Management (TDM) package would address this challenge. An enterprise grade TDM tool would also have capabilities to automate the data set up process once its defined and setup. At this step new and existing Data Pipelines which might have impacted by a change can be executed and tested. Test alerts and reports generated from this step gives feed back to the developers/Business Users in an automated manner

Continuous Deployment

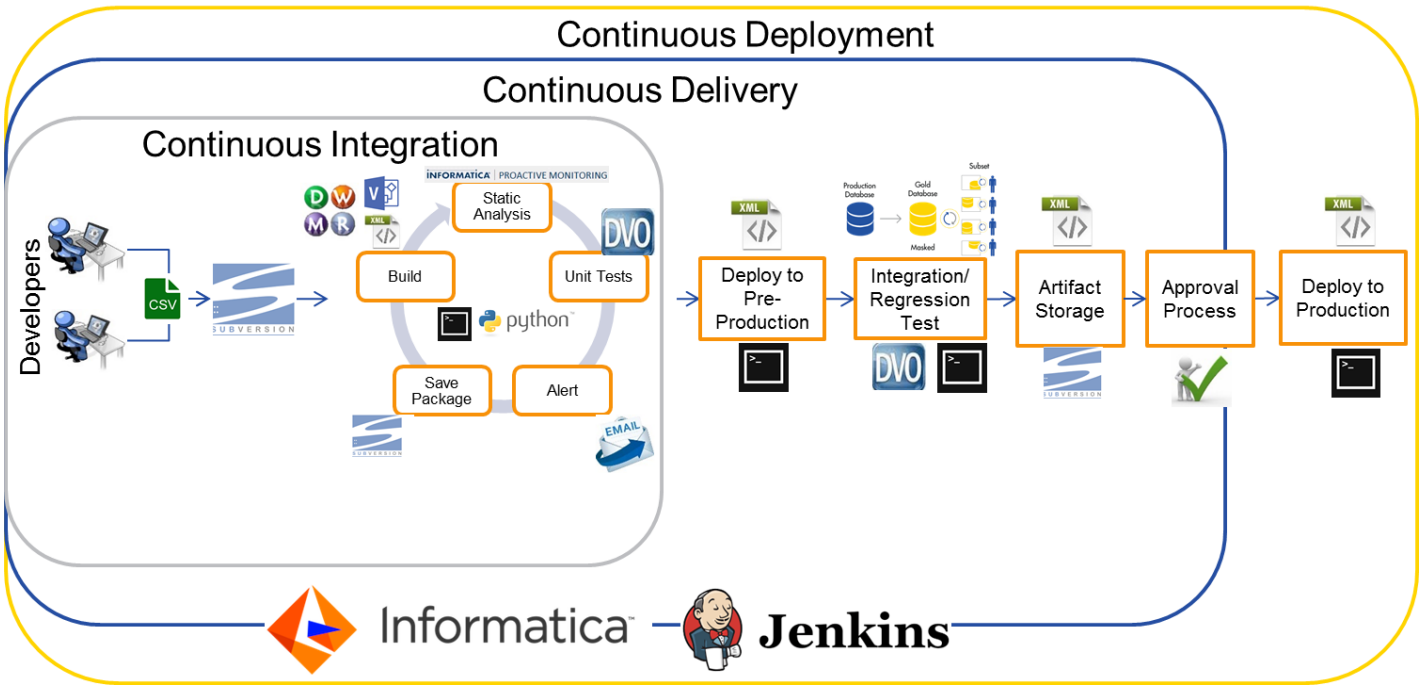
Once the Continuous Delivery process is matured the outcome is high quality deployable code in the pre-production delivered really fast compared to traditional approaches. Now a review is done by stake holders through a Change Management process for the approval of deployment and a manual deployment is done to get functionality to production. Automating this deployment process to production completes the Continuous Deployment pipeline. This step is just an extension of the deployment process discussed in the automated deployment step in the Continuous Delivery above, with many modules and code re-used.

Integrating all the steps through a CI platform

The entire Continuous Deployment workflow is knitted together in an enterprise grade CI platform. Choices of CI platforms are plenty, ranging from open source to licensed proprietary. Integration points with these platforms mostly would be through Command Line utilities. Wrapper scripts written in scripting languages like Python/Bash may be necessary to integrate complex sub steps within each step. Some of these platforms have marketplace third-party plugins which would reduce the time to configure and setup the CI tasks.

Reference architecture for Continuous Deployment on Informatica PowerCenter DI platform

Below diagram shows a reference Continuous deployment architecture used by Informatica professional services for a major customer in the Semiconductor manufacturing industry for



Use case

Implement a continuous deployment pipeline to do data ingestion to an ODS layer in a Data Warehousing appliance.

Pre-requisites

For an automated build and delivery process below are the key pre-requisites

Process: Create design patterns upfront with smaller simpler steps that can be easily automated

People/Technology: Along with informatica expertise experience, knowledge of xml, Informatica data transformation, scripting -Python/Bash, Microsoft Visio

Infrastructure: A windows VM with ample memory CPU and disk capacity with required software's

Ranjith Krishnamoorthy
Informatica Professional services – May 2017

Solution Architecture

For the use case Jenkins is used as the CI platform. CI pipeline is created knitting together the various steps involved in the process. Each of those steps are explained below

Automated build - Here the concept of Continuous integration is diluted and the frequent integration of individual changes followed by an automated build trigger of code build is replaced by a fully automated code build based on a set of input attributes. Informatica's Data integration platform delivers capabilities to do templating using Mapping Analyst for Visio and automation capabilities with Command Line tools. These is used along with a Python programming language to build the mapping from scratch. Below are the high level steps

1) Take attributes needed to build mapping in a CSV
2) Developer commits CSV to Subversion SVN repository
3) A Jenkins hook detects this change and triggers build process
4) Python scripts creates PowerCenter source and target definitions using Database library views
5) Python scripts processes the input CSV and uses PowerCenter CLI, Mapping Analyst for Visio Mapgen utility and xml libraries to build mapping/session/workflows and imports in to PowerCenter repository
6) Python scripts creates parameter files needed and checks it in to SVN

Steps 4,5,6 above can also be achieved using Informatica Java SDK.

Automated Code review* - Informatica's Data integration platform Proactive Monitoring (PMPC) add on is used to do code review in an automated manner. The Complex event processing engine behind this option responds to events and do static code analysis. The tool comes with pre-built rules which improves productivity. In the example architecture code analysis is invoked in response to code check in events

Automated Test Case generation - PowerCenter Data Validation Option (DVO) is an add on to Informatica's DI platform which provides a GUI to develop data validation test cases. The tool comes with command line tools and abilities to export/import code in serializable file formats. In the reference architecture Python scripts are used to generate the DVO code files using metadata provided in the input file and imported in to DVO repository

Automated Execution of Jobs - Command Line Interface available in the PowerCenter platform is used to launch jobs programmatically

Automated Test results alerting - DVO test cases created in the previous step is invoked programmatically using DVO command line. On completion of the test cases DVO has out of the box capabilities to send alerts and generate validation reports on a web browser. The results are also available on a repository database view which can be used to build customized alerting and reporting solution as well

Automated Package storage - Informatica PowerCenter has Team based development option with inbuilt repository versioning. It also provides capabilities to export the artifacts in xml format which is used for versioning in an SCM platform like Subversion (SVN). Contents are exported programmatically using Command Line Interface

Automated deployment to Pre-production – Contents exported and stored in SVN are picked up by Python scripts and imported in to the target repository using CLI

Test Data Generation*– Informatica's Test Data Management suite is used to create data subsets in the pre-production environment. The Data refresh can be done periodically or also on demand within the CI pipeline. This ensures excellent test coverage and helps deliver data pipelines at a higher quality. Also productivity is enhanced compared to ad hoc/ manual approaches to data setup

Automated Integration testing* – More test cases if needed are generated programmatically in DVO for better test coverage and is basically an extension of the test cases generated in the CI unit test step. Feedback loop is again a replica of what achieved in Unit testing phase

Automated Artifact Storage – Since the deployable contents in this use cases are non-binary files; no particular benefit is seen in using an Artifact storage repository like Nexus/Artifactory/Apache Archiva. This step is substituted by export of Pre-production, Data tested PowerCenter workflows back in to SVN. This serves as the golden copy that's ready to be deployed in production. CLI and python scripts are used to complete this step

Automated deployment to Production – This is achieved using the same code components written to deploy to pre-production which accepts parameters to control the Source and Target for deployment.

* Future enhancements to architecture that are in backlog yet to be incorporated at the time of document authoring

Benefits

Implementing a continuous deployment pipeline in this manner delivers many tangible and non-tangible benefits

- 90% Reduction in delivery time to production
- 90% reduction in code bugs
- Strict adherence to coding standards
- Frees up valuable Developer hours which can be utilized doing more important tasks that impacts ROI directly

Summary

Continuous methodologies deliver considerable benefits when implemented in the right manner for the use case in hand. It can also be achieved in IT organizations using DI packaged tools. The choice of platform is an important part and Informatica delivers industry leading DI platform with all the capabilities to achieve a fully automated Continuous Deployment environment in IT organization.

Further Reading

https://www.informatica.com/content/dam/informatica-com/global/amer/us/collateral/white-paper/preparing-next-generation-data-integration_white-paper_2538.pdf

Ranjith Krishnamoorthy
Informatica Professional services – May 2017