End to End Automation On Cloud with Build Pipeline: The case for DevOps in Insurance Industry

Continuous Integration, Continuous Testing, and Continuous Delivery

Mitesh Soni
IGATE
Gandhinagar, India
mitesh.soni@igate.com

***Abstract*— In modern environment, delivering innovative idea in a fast and reliable manner is extremely significant for any organizations. In the existing scenario, Insurance industry need to better respond to dynamic market requirements, faster time to market for new initiatives and services, and support innovative ways of customer interaction. In past few years, the transition to cloud platforms has given benefits such as agility, scalability, and lower capital costs but the application lifecycle management practices are slow with this disruptive change.**

DevOps culture extends the agile methodology to rapidly create applications and deliver them across environment in automated manner to improve performance and quality assurance. Continuous Integration (CI) and Continuous delivery (CD) has emerged as a boon for traditional application development and release management practices to provide the capability to release quality artifacts continuously to customers with continuously integrated feedback. The objective of the paper is to create a proof of concept for designing an effective framework for continuous integration, continuous testing, and continuous delivery to automate the source code compilation, code analysis, test execution, packaging, infrastructure provisioning, deployment, and notifications using build pipeline concept.

Keywords— Cloud Computing; DevOps; Continuous Integration; Continuous Testing; Configuration Management; Continuous Delivery; Automation

I. INTRODUCTION

Application development phases include coding, building, integrating, testing, troubleshooting, infrastructure provisioning, configuration management, setting up runtime environment, and deploying applications in different environments. It can be a repetitive, time consuming, manual, and complex process. Additionally, enterprise applications are so diverse and composed of so many technologies, open source and commercial tools that it is becoming difficult to manage application delivery lifecycle for organizations while dealing with these complexities.

Usually, application is deployed in the production environment when development team has finished all its development part and Operations team creates and configures deployment environment for the application independently. Most of the time there is hardly any interaction between development and operations team. In addition to it, in traditional environment transition of latest application build across environments lasts over weeks. Hence, there is enormous waste of time in a typical IT environment with people waiting for resources or they are stuck solving the same problems over and over again. Execution steps are manual and hence it is likely to create problems because of manual errors. To succeed in a world where technologies, requirements, ideas, tools, and timelines are constantly changing; information must be accurate, readily available, easily found, and ideally delivered constantly, in real-time to all team members [1].

DevOps is an emerging culture in which development, testing, operations teams collaborate to deliver outcome in a continuous and effective manner DevOps represents a significant opportunity for organizations to gain market place in the comparison of their competition and build better applications; thus opening the door for increased benefits and improved customer experiences. The basic philosophy behind the concept is, "faster you fail, faster you recover". It enables the organizations to quickly grab opportunities and reduce the time to include customer feedback into new feature development or a big fix. The end goal of DevOps is to reduce the time between the initial concept and the end result after implementation part in the form of production ready application.

Agile practices and DevOps culture results into faster time to market and an efficient outcome. Organizations can perform continuous integration of its code and verify the integrity of the code in early stages of development. Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day [2]. Teams can also perform unit testing of customer requirements with development. Continuous Delivery is a software development discipline where you build a deployment pipeline in such a way that the software can be released to production at any time [3]. This instant feedback helps to ensure that whatever is being produced will meet the

978-1-4673-8566-4/16 $31.00 © 2016 IEEE
DOI 10.1109/CCEM.2015.29

needs of an end user. Thus it is helpful to show the customer, in almost real time, what developers are delivering.

II. CHALLENGES FACED BY INSURANCE INDUSTRY

Considering requirements to improve visitor acquisition, online marketing campaigns, search engine optimization, mass customization and diverse products, etc. business agility is now a necessity for Insurance Industry and not an option. Insurance Industry needs to improve its time to market. Idea of business ability is either supported by quick response to new opportunities or it can be obstructed by traditional or manual processes that take its own time to respond. Additionally, with changes coming at a rapid pace to deliver new services and products; application delivery lifecycle is pertinent to the success of business. There is a huge gap exists in requirements and application delivery with the traditional approach and hence it results into delays and rift between Dev and Ops team which results into business loss.

Additionally, Insurance industry is looking forward to create new ways to span business and to bring innovation aspect at the level of feasibility. The main reason behind to go for the disruptive innovation such as Cloud computing is to bring real time intelligence and to manage periodic peak loads.

In short, Insurance industry requires a more streamlined application delivery model, elastic resources, automation of manual processes for build, integration, test, and deployment. The major challenges most insurance organizations face is to improve the speed and quality of their builds and shorten their application delivery lifecycle while reducing the time spent on fixing problems. Some of the challenges in the current processes are:

- Customer expectations are continuously changing and if unfulfilled on specific time, it results into unhappy customer
- Increased rate of production releases due to Agile philosophy
- Slow Process to integrate customer feedback
- Manual build process, which requires regular fixes, is a big show stopper in deploying code more frequently
- Non-standard deployment scripts across environments
- Lack of verification and validation of existing build and quality during toll gates
- Slow and Error prone Application release process
- Deployment artifacts or configuration are migrated and implemented using different methods
- Manual intervention for rollback
- An urgent need to introduce new processes and tools covering build integration, test, deployment, and end to end deployment orchestration
- To develop and deploy efficient applications in cost effective manner
- To meet internal business unit demands for more resources to support increasing number of users and new application development
- To improve low resource utilization to boost RoI and minimize electricity bills as well as to implement a disaster recovery plan
- To adopt innovative and interactive marketing ways, Insurance industry requires an agile, highly available, cost-effective, fault tolerant, dynamically scalable, highly utilized and manageable infrastructure to support its growing digital existence

The need of an hour is to adopt innovative models to satisfy dynamic and rapid demands of customers to remain effective in the competitive market. Multiple diverse applications with different environments in application delivery lifecycle that needs elastic resource usage require new approach to satisfy end-users. Combination of DevOps Culture and Cloud models can serve as an effective way out. Establish new processes for software build, integration, deployment, and control. DevOps practices enable faster and safer releases because they are smaller, less complex, and more thoroughly tested. Automating provisioning and deployment further reduces configuration errors and speeds up delivery [4].

Cloud environments provide agile and scalable resources to analysis and predict customer behavior to develop innovative applications or end-product for marketing campaigns. With agility of cloud, it is possible to deploy innovative applications rapidly without delay of provisioning resources. Cloud technologies can help to identify fraud and undesirable behavior by providing resources to manage analytics. The scalable computing power in the cloud can support more sophisticated risk modelling to improve financial planning and risk management.

The delivery of an innovative application can be much faster and less error prone due to deployment pipeline automation. Additionally, customer feedback can be integrated in application much faster in DevOps culture that makes customer more engaged, satisfied, and happy. To realize the feasibility and benefits of DevOps and Cloud combination, we present a proof of concept for end to end automation to accelerate delivery of an application.

III. END TO END AUTOMATION: BUILD PIPELINE

Automation is the key enabler for effective communication and collaboration between development and operations teams. In the given proof of concept, we have focused on open source tools to achieve end to end automation. Based on the type of tools in the technology stacks, automation approaches may differ in terms of pull or push mechanisms of artifacts from CI to CD.

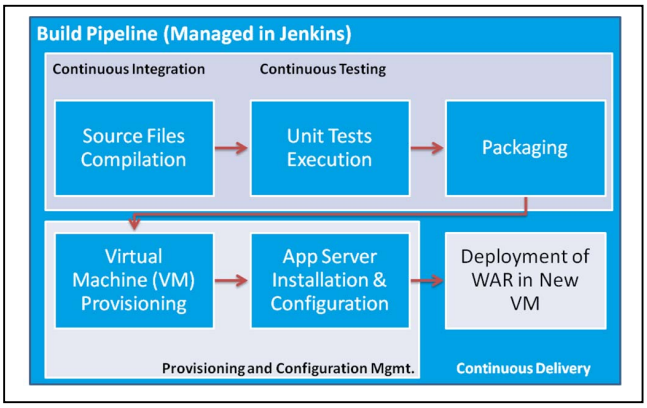

Fig. 1. Continuous Integration, Continuous Testing, Continuous Delivery

Application delivery is orchestrated with the use of build pipeline to model different aspects of DevOps Culture, different environments, and quality gates. Quality gates helps to manage orchestration across build pipeline. Ideally, if one step fails to execute successfully then continuous delivery or next step must not take place. Quality gates ensure the same. We validate the end to end automation approach with build pipeline by a prototype implementation to showcase its practical feasibility.

Build Pipeline is a View of upstream and downstream connected jobs; each one triggers build to quality assurance steps. It has ability to define manual triggers for jobs that require intervention prior to execution such as approval process. It provides orchestration of the promotion of a version of application through quality gates and into different environments.

The following use cases are implemented as a part of the PoC:

- On-boarding of applications
- Integration of CI Server and Source Code Repositories
- Run build and deployment job automatically when the code changes
- Plug-in installation and configuration for CI Server
- Creating a Build pipeline for Orchestration and Quality Gates creation
- Integration of CI Server and Static Code Analysis Tool
- Integration of CI Server and Binary Repository
- Integration of CI Server and Configuration management Tool

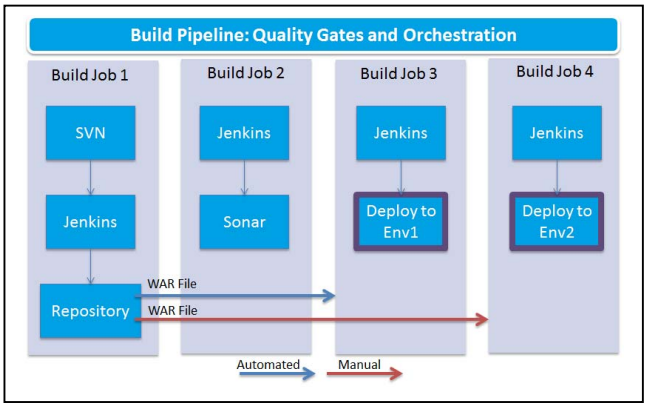

Fig. 2. Build Pipeline for Orchestration

- Application assessment for Cloud environment - Feasibility Analysis
- Infrastructure Provisioning using Configuration management Tool in Cloud Environment
- Configuration of runtime environment
- Deploy the application to Application server
- Run deployments multiple times

Every commit into source code repository by a developer is verified by automated build process using continuous integration server. Chain of build jobs that constitute build pipeline with upstream and downstream build jobs in continuous integration server will ensure that next steps are followed based on job configuration. Build pipeline can be configured to process automatically or having minor manual intervention in case authorization or approvals are required. Static code analysis and automated test case execution can be managed by CI server and notification of successful or failed build execution can be sent to configured stakeholders. Once continuous integration is successful, binary artifact can be archived or can be stored in the binary repository for deployment across different environment.

Cloud computing is a revolutionary paradigm that allows to acquire pay per use and seemingly infinite resources on demand. Together with agile methodologies and DevOps culture the cloud can help organizations to innovate and quickly respond to market demand for continuous delivering of services. To verify Cloud suitability to the application, we used Cloud Acceleration Program or CAP framework to leverage the power of Cloud Computing for cost efficiency and business benefit. CAP is useful to carry out requirement analysis, technical feasibility analysis, and cost-benefit analysis—using our frameworks—of applications for transitioning to Cloud. Deliverables of the CAP comprise clear recommendations on the applications to be migrated to Cloud, a Cloud model, the vendor technology for their deployment, and the transition roadmap. Fully automated provisioning, configuration management, and deployment into different environment such as QA, pre-prod, prod, etc. in order to manage the costs for managing applications are some of the

most essential requirements to make use of the benefits of Cloud computing.

Technology Stack	Description
Integrated Development Environment	**Eclipse:** Development environment used by developers to create an application
Source Code repositories	**SVN, Git, StarTeam:** Centralized location to keep updated source code based on authentication
Build Tools	**Ant, Maven:** Build automation tools to automate the process of compilation of source code
Continuous Integration Server	**Jenkins:** Verify integrated code with automated build and notify status of build execution to stakeholders
Automated Testing / Continuous Testing	**Junit:** Automated test cases: Unit test execution on integrated source code
Static Code Analysis Tool	**Sonar, FindBug:** Manage code quality: Comments, Coding rules, Potential bugs, Complexity, Unit Test, Duplications,
	Architecture & Design
Binary Repository	**Artifactory:** Versioning of Application Binaries into repository to manage, host and control the flow of binary artifacts
Configuration Management - Provisioning	**VMware vSphere, VMware vCloud Director, Amazon Web Services, Microsoft Azure:** Infrastructure Provisioning in Cloud Environment
Configuration Management - Preparing Runtime Environment	**Chef:** Application Runtime Environment Creation by creating various cookbooks to configure runtime environment.
Deployment Tool / Script	**Deployment Plugins or Shell scripts:** To deploy final artifact to deployment server in different environment
Monitoring / Notifications	**Jenkins Plugins:** To monitor various environments as well as complete deployment pipeline process to get notification on any failures

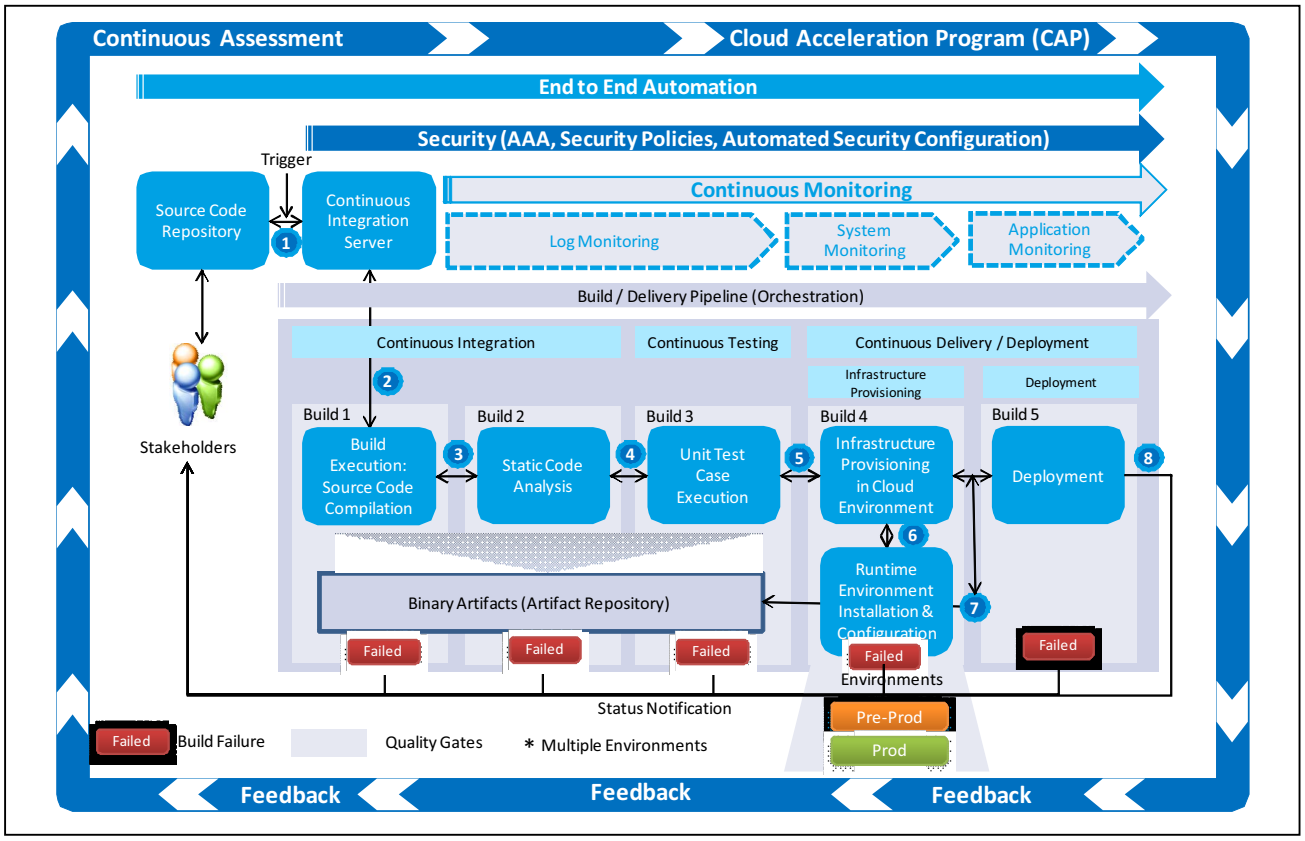

Fig. 3. End to End Automation

With the use of configuration management tools such as Chef, provision virtual machine in public cloud, private cloud or VMware based virtualization environment. Cookbooks create required runtime environment for the application and configuration management is managed from central location. Once all Quality gates are successfully crossed in build pipeline, final deployment can take place which is parameterized and approval based. Deployment plugins, shell scripts or commands can be used for deployments. For this proof of concept, we used shell commands directly for the deployment.

Benefits achieved:

- Automated deployments and standardized production environments are outcome of agility of business and IT collaboration

- Faster delivery of builds, features, and bug fixing thereby creating a continuous build pipeline
- Accelerated customer feedback cycles and collaboration results into high quality for feature/function delivery
- Continuous innovation because of continuous development of new ideas
- End-To-End Automation and Configuration Management for deployment pipeline using open source tools in days rather than in weeks
- Rapid delivery in Dev, Test, Pre-production and Production environments: Elimination of manual and repeatable processes based deployment
- Orchestration: End to end deployment workflows governed through build pipeline
- Quality gates for process verification and orchestration
- Improved application turn-around time across environments
- Streamlining of deployment process and improve quality of scripts and integrations
- Automated Infrastructure provisioning in Cloud Environment: Integration and Configuration across Public, Private and Hybrid Cloud – IT environment
- Versioning of Runtime Environments in the Configuration management system
- Automated application deployment in web/application servers
- Monitor health, configuration and status of connected systems before, after and during deployment
- High availability of resources: Low downtime
- No resource bottlenecks

IV. Conclusion and Future Work

The implemented proof of concept showcases various benefits of end to end automation in application delivery lifecycle using combination of DevOps culture and cloud environment.

Future work needs to be focused on monitoring, security, quantifying benefits of end to end automation for continuous delivery and introducing integration of open source application life cycle management tools (ALM Tools).

Monitoring: Infrastructure and application performance monitoring with tools such as Nagios, New Relic, etc.

Security: Centralized security policy configuration, Continuous security policy deployment, Automate logging and audit trails for all compute resources.

ALM Tools: Application Lifecycle Management tools to manage entire lifecycle of application; manage various methodologies such as agile and waterfall, requirements management, continuous integration, release management, change management, maintenance, governance, etc.

Acknowledgment

Author would like to thank Jyoti, Namjoshi, Vishwajit Pandit, and Cloud team of IGATE, for their insightful feedback, creative inputs, encouragement, and support during exploration and proof of concept execution.

References

[1] Cois, C.A. ; Yankel, J. ; Connell, A., "Modern DevOps: Optimizing software development through effective system interactions," IEEE, Pittsburgh, PA, 2014

[2] Martin Fowler, Continuous Integration, [Online]. Available: http://www.martinfowler.com/articles/continuousIntegration.html

[3] Martin Fowler, Continuous Delivery, [Online]. Available: http://martinfowler.com/bliki/ContinuousDelivery.html

[4] Forrester, The New Software Imperative: Fast Delivery With Quality, 2014