SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Emerging Trends in Continuous Integration and Continuous Deployment

CI/CD Across Data and Integration

A Technical Paper prepared for SCTE by

Rohit Kuruppath
Lead Architect
Cox Communications Inc
Atlanta
+1 (770) 878-0567
Rohit.K@cox.com

Table of Contents

List of Figures

1. Introduction

Continuous Integration and Continuous Deployment (CI/CD) procedures have been extensively used across software development life cycles specifically around application development. The CI/CD implementation around data engineering tools was challenging within the UI centric tool stack. The delivery requirements were not agile without the need of paired programming and common code base. In most cases, the versioning capabilities of the data tools itself was utilized without utilizing the enterprise CI/CD capabilities.

This paper discusses how data engineers can take advantage of the new flexibility that comes with the deployment of software across different data and integration tools, more quickly and securely. It will provide an overview of CI/CD, with a specific focus on how it is being applied across on-premises, cloud and hybrid data environments.

Over the years, real-time data integration and automation have been highly critical for all organizations to meet their analytic needs and providing better insights and capabilities to customers and business leaders. With the huge amount of data to be collected and processed from different systems across the network, before building analytics models around it, there is a pressing need to adopt DevOps guidelines to deliver quality data outputs.

This paper will describe how Cox Communications leveraged the existing CI/CD tool capabilities and expanded upon the configuration tools to support Agile DevOps delivery for building a hybrid data lake. Data Engineers and analysts could leverage a stable pipeline flow and common set of automation tools for delivering the solutions without much onboarding and transitioning into these CI/CD processes and tools.

2. Data at Scale

The transformation of network devices and the possibility of different data points at higher frequency to derive analytical and business insights have been tremendously growing during the last few years and it will continue to evolve throughout decades to come. When exploring the possibilities of big data and cloud technologies, enterprise teams started collecting data points from all the different devices in and out of network. The understanding of 5v's of Big Data – velocity, veracity, volume, value, and variety, allowed data scientists to derive more value from their data while also allowing organizations to become more customer centric, while being proactive to changes in the network.

With the advancement of analytical capabilities in Cox, all the analytics teams across Cox's Technology and Operations organizations were centralized back in 2017. Analytic initiatives around network and customer service health is solely driven by the data across the network and is focused on four key areas.

1. Deep understanding of data that describes quality and use of service by our customers
2. Ability to predict future service impacts and mitigate or prevent them
3. Machines executing the right actions to take, at scale
4. Continuous learning capabilities to improve over time

Different flavors of data from the edge of the network and across the elements of the network, sampled at different intervals, must be accommodated. Formats must be collected, cleansed and processed for both real time and historical trending. Analytics are required to have more accuracy and lesser development

turnaround time. Hybrid data lake environments coupled with traditional reporting and analytical solutions have resulted in a diverse tool stack for the developers to work with.

The traditional extract, transform, and load (ETL) tools, along with the data warehousing technologies and relational databases, which used to be the primary tool kit for the data engineers, took second place with the new advancement in big data and cloud technologies. Data engineers ended up building and supporting solutions across different technologies and programming languages like the software programming application development. Data engineers are using traditional ETL tools like Informatica, Alteryx for the structural data transformations, along with big data technologies like Apache Spark and using Scala, Python to process unstructured data sets from sources and cloud services for their hybrid solutions.

3. CI/CD Deployment Pipeline

CI/CD are a series of steps that must be performed in order to deliver a new version of software. A pipeline is a process that drives software development through a path of building, testing, and deploying code. The objective is to reduce human error and maintain a steady process to release software through automation. Figure 1 below depicts the continuous integration and continuous deployment pipeline flow here at Cox for a traditional software development application life cycle.

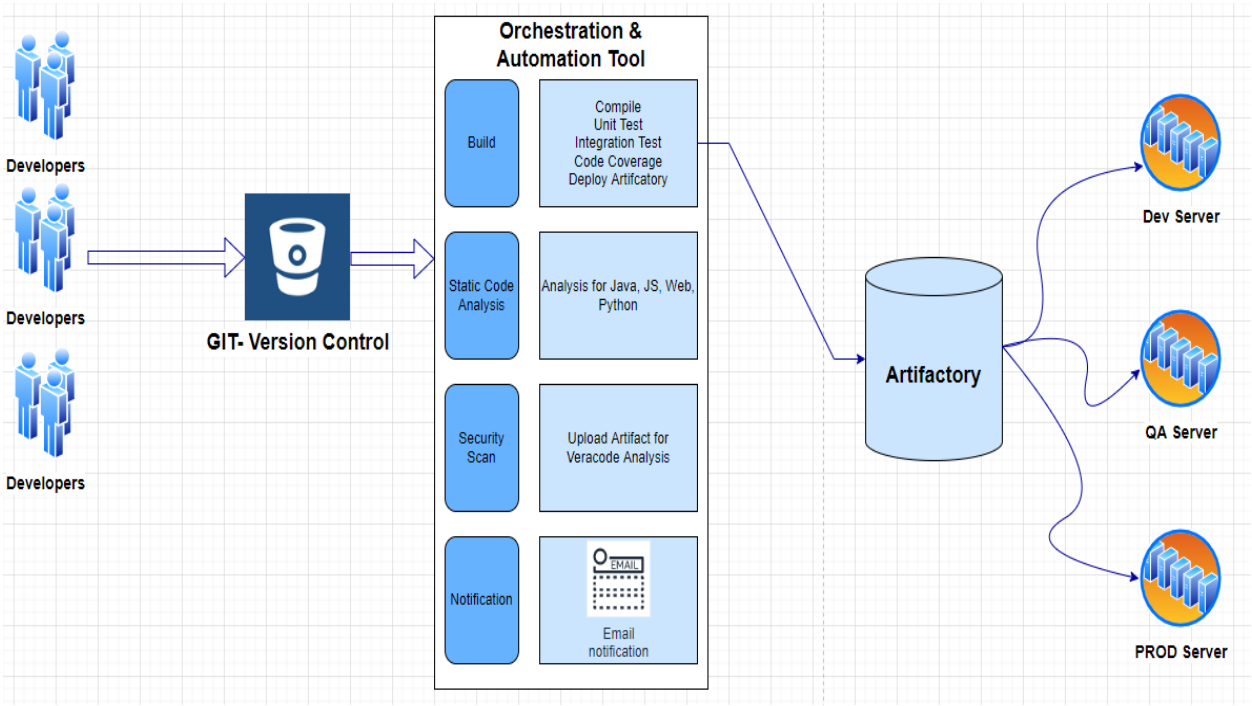

Figure 1 – Deployment Pipeline

In a GitOps-driven deployment process, all activities emanate from the version-controlled code repository. Here in Cox, we drive the deployment process directly within the code repository or version control service BitBucket. Storing the deployment instructions in a central version control repository provides a single source of truth for all deployment activities. Using the repository as a single source of

truth provides reliable change management and auditing capabilities. Version control and access security are also built into the service.

Orchestration tools play the role of managing the deployment process, which can range from a single task – such as running a unit test case against a source code – to a complex deployment process embodying many tasks. When an automation is triggered, it will refer to the playbook in the deployment repository stored within the version control and execute the tasks. Artifactory's build integration to orchestration tool allows our build jobs to deploy artifacts and resolve dependencies to and from Artifactory. Artifactory is a centralized binary management system providing management binary artifacts generated and used by the organization.

4. Hybrid Data Lake and Network Data Ecosystem

Data from different source systems gets loaded into the operational data store either through database (DB) replication or ETL loads built by data engineers over the years. These were built in different tool sets including Informatica, Oracle and loaded into relational databases. With the onset of big data technologies and possibilities of analyzing unstructured data assets, more data points at the least possible frequency started getting streamed (using streaming applications like Apache Kafka, Spark Streaming), pulled from sources on a specified interval (using tools like Apache Sqoop, Spark Batch) and even written into Hadoop Distributed File System (HDFS) for further processing and consumption.

Data engineers started working across multiple tools to build this ecosystem leveraging on-prem ETL processes to relational databases and big data lake for the unstructured data assets for analytical insights. There are different production solutions to meet both real-time and batch reporting needs in place now that are built across different tool stacks that consume data from systems like ticketing, polling platforms and so on. Before the expansion into on-prem big data lake and onto the cloud data lake, we realized the need to have a stable automated deployment process with a common CI/CD tool stack to support most of our needs and deliver in an agile fashion.

Traditional batch loads to meet the enterprise reporting through an enterprise data warehouse (EDW) which have been a stable process for more than a decade with 1000s of daily loads handling data assets from billing, network alarms, construction workflows, ticketing, customer calls, chats, geospatial assets, and many more. At the very beginning of defining the requirements for a more agile delivery and more frequent data needs, the team invested on setting up and configuring an automated CI/CD pipeline, as well for the traditional ETL tool and databases working with the enterprise software configuration management (ESCM) team.

Within the last four years, as Cox Communications started the data cloud journey, most of the on-prem ETL, big data deliveries are managed through CI/CD pipelines that provide more visibility into the source code. The CI/CD pipeline is auditable for security scanning and vulnerabilities and provides the developers the ability to follow a common deployment process for easy handoff for reviews and deployment across all the different tool stacks. End-to-end release cycles starting from the source to the very end at the data lake, are all getting managed through CI/CD pipelines that are configured and managed through individual application and data teams. This makes any changes seamless and controllable, even for an easy rollback in the case of faulty deployments.

For example, the polling data from the modems through a CMTS/CCAP devices goes through all different layers including streaming applications, an on-prem data lake, cloud services for analytics/machine learning models, and the traditional warehouse for enterprise reporting and integration. A change to a device element or the addition of new pollers, would have been a very intensive software

development life cycle (SDLC) process, if the end-to-end architecture, including all data spaces, are not able to support CI/CD processes. With the cloud journey, data is getting processed across multiple cloud accounts based on the department needs for different use cases. The codebases involving streaming applications to ingest raw data using Spark Streaming and landing them into file systems, ETL processing and analytical solutions using Python, PySpark and machine learning models are all stored in an on-prem code repository for orchestration for their respective environments. This provides the ability for development leads and architects across respective organizational units to have a better view of the solutions, and an ability to review and commit to the codebase for production deployments through a single CI/CD pipeline.

Because different servers are involved across the tool stack for ETL, big data processes also pushed the need of automation for the environment life cycle management to include upgrades and server additions. This reduced the need for downtime during the data tool stack upgrades and server additions. This is critical for the proactive service and health analytical solutions that are built on top of these data assets where data unavailability over a time period will have impacts to the model outputs and customer notifications. Figure 2, at a high level, depicts the network data assets and their data processing through a relational database, big data streaming applications, and ETL tools. Enterprise cloud data lake accounts for the hybrid data lake for enterprise consumption, storage and processing serves the downstream consumer accounts for their analytical models and outputs.

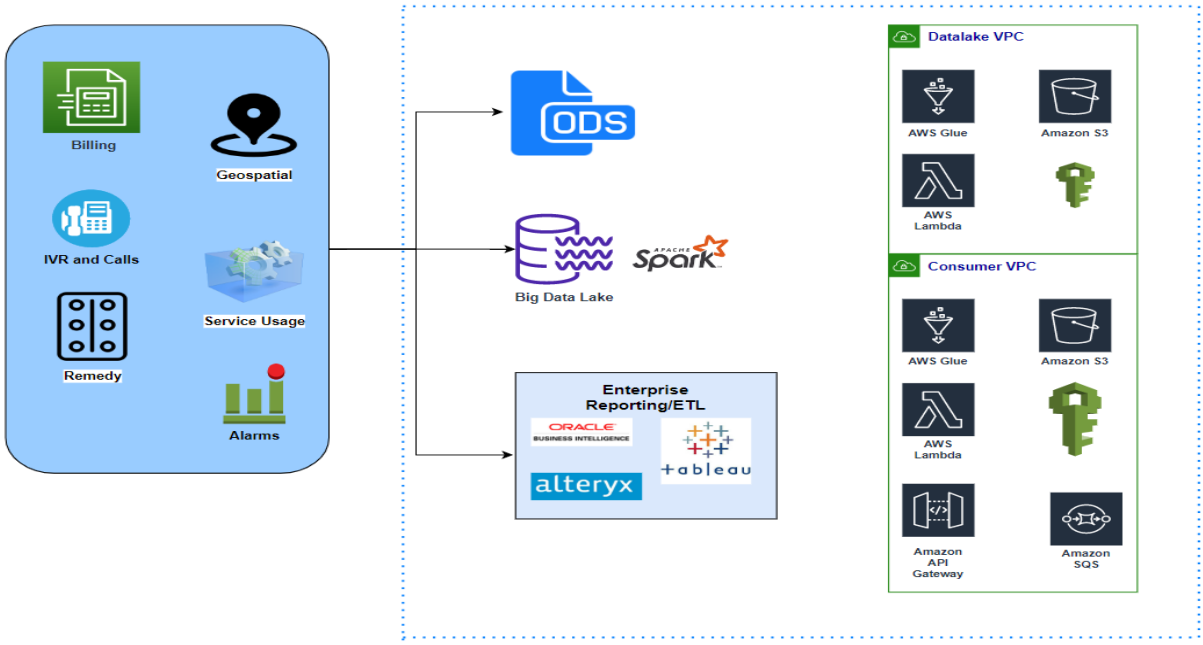

Figure 2 – Network Data Ecosystem

5. Data and Integration CI/CD Lifecycle

At Cox, we have an ECSM team to provide automation services that increase the velocity of development and operations teams across Cox to drive value to the business. DevOps teams across Cox collaborate with the ECSM team to define a deployment pipeline and support the right set of tools for delivering a CI/CD process to help the development teams to deliver the software or infrastructure as a code easily and

securely. Compared to the earlier days of infrastructure automation and traditional application development, the ESCM team plays a pivotal role in identifying, configuring and supporting different automation tools to meet the needs of engineers across data, automation, integration, application development, databases, and infrastructure.

Working with the ESCM team, the first step towards this journey was the identification of a unified platform and pipeline configuration for data engineers that provided minimal overhead for developers and did not impact the end users or customers. Starting with the ETL tool stack, where we had a built-in tool for deployment automations and validations, we tried reusing that stable process orchestrated through BitBucket and Jenkins with the goal of making it easy to use for data engineers new to the programming world and CI/CD tools. CI/CD tools in this case acted as a wrapper process to invoke ETL built in code validations, test scripts, and deployment processes, while also setting up a repository and pipeline flow for unified deployments.

The approach of building an initial prototype and establishing the pipeline configuration was followed during the big data and cloud journey to identify the right set of CI/CD tools and process that is supportable and extensible for different services within these cloud providers. The upfront design and setup time spent with the ESCM team helped the ongoing development teams to reuse the templates, Ansible scripts, and pipeline flows to build their own ETL solutions with minimal changes. User adoption is a key factor for a successful CI/CD model along with the DevOps enablement for agile program delivery. Individual teams will be responsible to maintain the Git projects along with their code base and delivery, while the ECSM team supports the tools and the environments to make this happen for DevOps teams. This allows enterprise teams to reuse the CI/CD tool stack, environments, and to follow standard guidelines along with the enterprise security checks on the codebase and processes.

6. Orchestration Architecture

Enterprise teams at Cox Communications have been using Bitbucket and Jenkins for code source control and automation/orchestration for the CI/CD flow as explained in Figure 3. Ansible is used for automating operations and building templates for repetitive and reusable functions. Figure 3 shows the high-level architecture and tools involved for an AWS Cloud deployment pipeline flow.

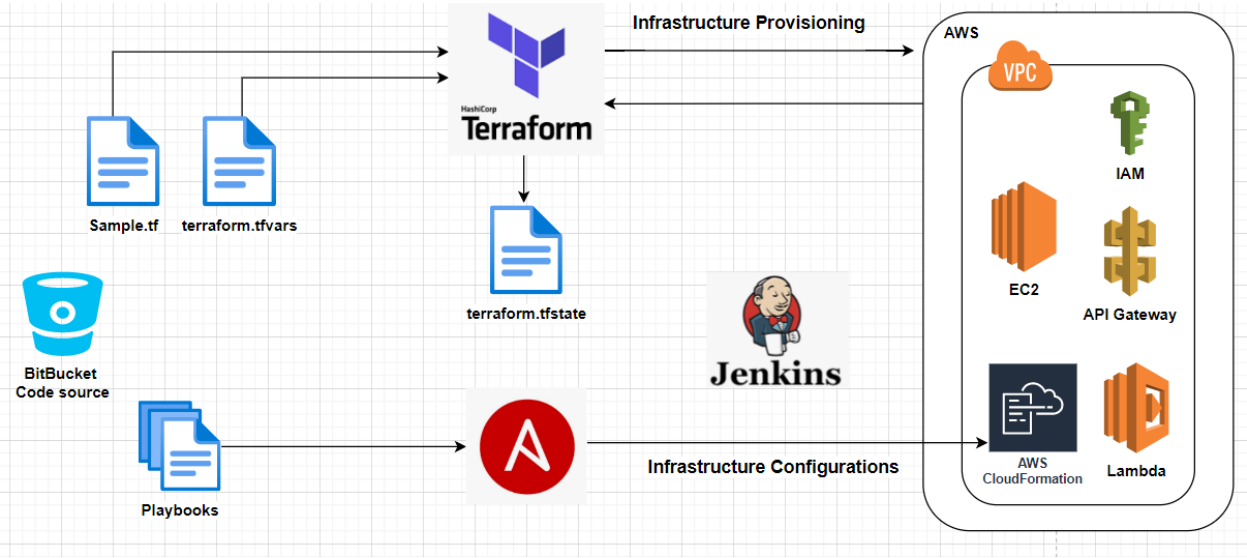

Figure 3 – Reference Architecture

Bitbucket is the source code controller for developers to commit their code. We are following the terraform deployments for infrastructure provisioning in the cloud, including the role provisioning, user accounts. Terraform variable files, modules, and the Ansible playbooks are all stored in the source control tool. Jenkins orchestrates the deployment automation process by invoking the cloud formation templates through Ansible scripts for serverless application deployments such as AWS Glue and AWS Lambda. These serverless processes are heavily used for the integration and ETL processing. Terraform records the state files and compares them against the requested resources while running the plan and deploy the infrastructure as a code to the requested AWS instance.

For the on-prem deployments, we have the built in Informatica power center command shells that manage the deployment through the enterprise supported CI/CD tools Bitbucket and Jenkins. This helps the developers to maintain a common codebase and repository while reusing the deployment pipelines wherever possible. Similarly, the big data applications across different services and tools like Spark Streaming, batch Spark, HBase, Python applications, are all deployed following the above architecture. The exception to this is the on-prem Hadoop file systems that are orchestrated through Ansible automations instead of AWS Terraform and AWS CloudFormation. This provides the developer a simplified and well-established deployment automation pipeline that is configured upfront and can be extended based on the specific use cases.

7. Conclusion

The journey into expanding a culture of Continuous Integration and Continuous Deployment into the Data Integration space has been one that mirrors the actual process of continuously iterating and deploying. Adoption of existing capabilities into the data space, and the partnership with enterprise configuration management teams to extend the capabilities for different software development tools, helped us to build a robust and useful CI/CD process.

People, process and technology are three pillars of a successful delivery model. Rolling out CI/CD into data space was done without jeopardizing the customer experience with frequent releases and lesser overhead on software developers to adopt the process across different tool stacks. Zero down time server upgrades, what would have once been a unique phenomenon, is now a common practice for data platforms with the introduction of standby clusters. Those lessons are now able to be transferred into other sides of engineering, to help drive improvements at a wider scale and without the need of constant manual intervention.

I would like to take the time to thank those who lent their time and expertise into the completion of this paper and presentation – Lisa Pinkey Coleman, Kesha Heard, Greg Garrison and Chandrasekhar Anne.

Abbreviations

AWS	Amazon web services
CI/CD	continuous integration and continuous deployment
DB	database
DevOps	development and operations
EDW	enterprise data warehouse
ESCM	enterprise software configuration management
ETL	extract, transform, and load
HDFS	Hadoop distributed file system
SDLC	software development life cycle
UI	user interface

Bibliography & References

1. Informational interview, Chandrasekhar Anne, Lead Systems Engineer, Cox Communications Inc
2. Informational interview, Kesha Heard, Lead Architect, Cox Communications Inc
3. “Multi Region Deployment in AWS with Terraform”, by Yadav Lamichhane
4. “Terraform: Beyond the Basics with AWS”, by Josh Campbell and Brandon Chavis

