

Zero to Federated at the Speed of Jenkins

A Case Study of Success in DevOps

Richard Mills June 18, 2015



**Jenkins
User Conference**



Zero to Continuous in 90 Days

A Case Study of Success in DevOps

Richard Mills June 18, 2015



**Jenkins
User Conference**



Who is this guy?



- *Me*: Mad Software Developer turned Mad Software Engineer turned DevOps Solution Lead. Particular focus on tools and automation. CI, CD, DevOps ... what's next?
 - PS: Thanks for inventing the term “DevOps” to describe what I like to do.
- *Pays my bills*: Coveros helps organizations accelerate the delivery of secure, reliable software using agile methods.
 - Agile transformations, development, and testing
 - DevOps implementations
 - Training course in Agile, DevOps, Application Security
- *Keeps me intrigued*: SecureCI
 - Open-source DevOps product
 - Integrated CI stack with security flavor



...to tell you a little about some successes we had introducing Continuous Delivery during a client engagement with a major media and advertising company.

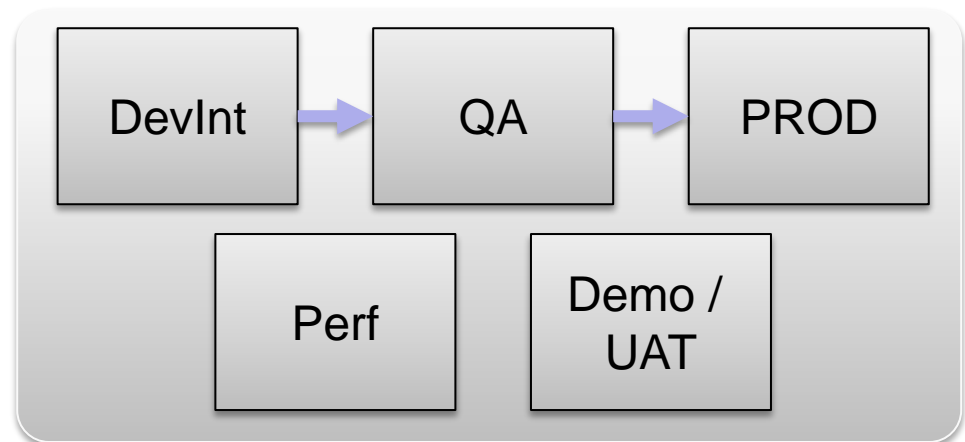
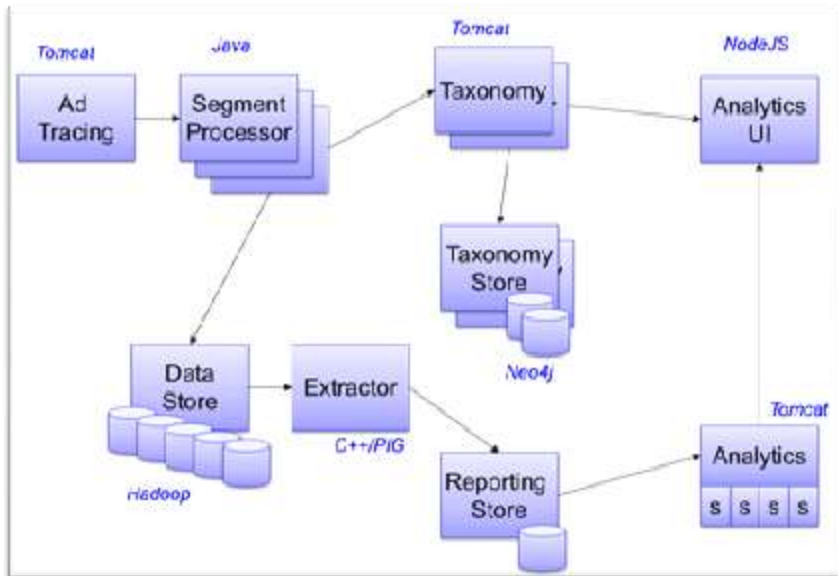


- Jenkins for continuous build
- Manual downstream activities
 - Testing and test environments managed by QA teams
 - Isolated siloes of development, QA, tech-ops groups
 - “... takes us 6 weeks to roll out completed software releases”
 - “... major pain configuring fleet of servers”
 - Long, hard-to-integrate release branches
 - **“We want CI/CD”**
- Two years of failed attempts to introduce Continuous Delivery (CD) into their development processes
- Green-field project to re-engineer the internal data management platform for processing advertising data
 - Approx 50 people distributed between US and Europe



What were we working with?

- Large, federated system with about 18 independent system components
 - Front-end UI, Tomcat REST web services, Java-based distributed data pumps, Hadoop back end
 - Installed into 3-5 environments ranging from ~15-40 servers each

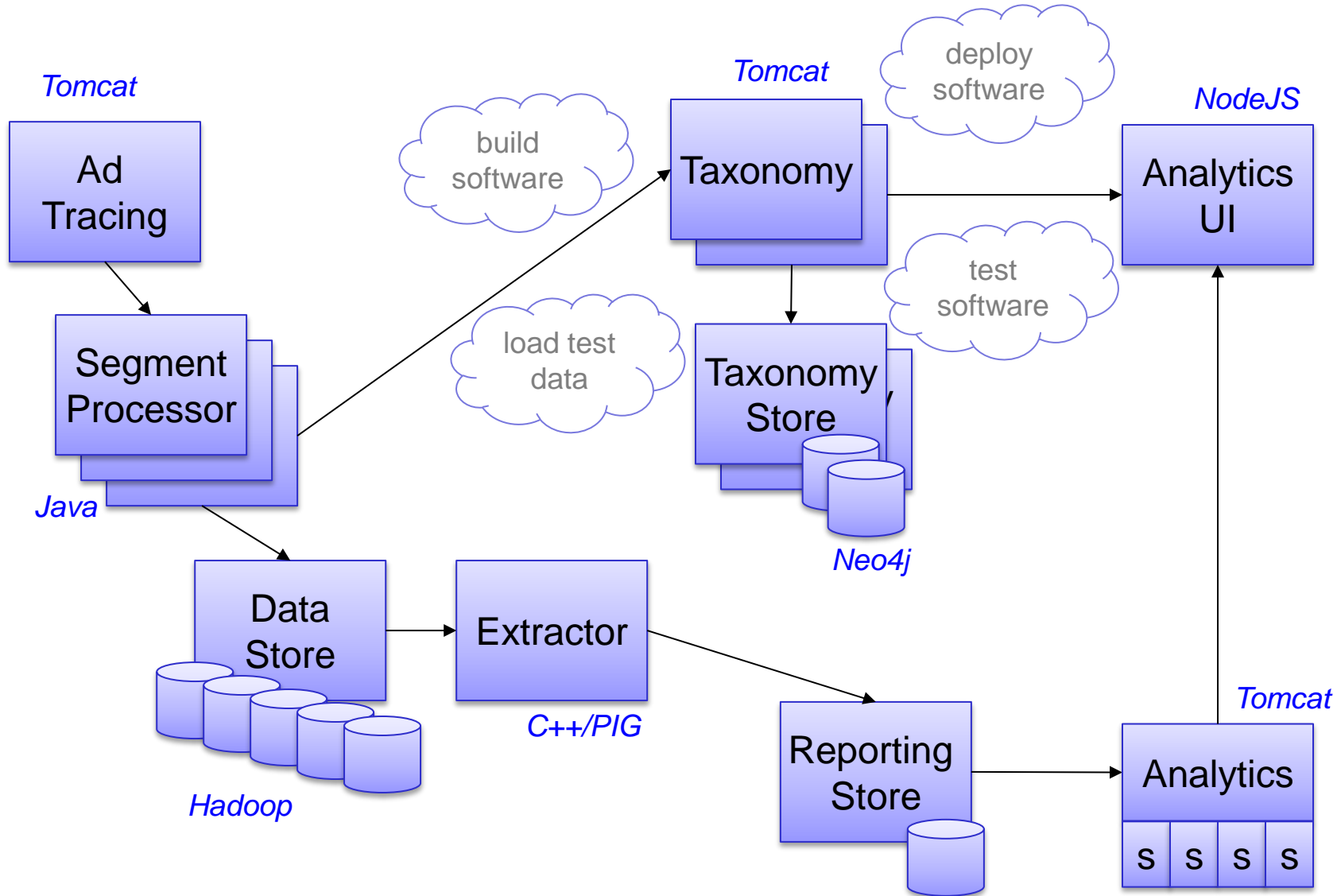


- Continuous build, test, delivery of most of the systems through multiple environments
 - Automated Unit, Deploy, Smoke, Integration, Functional, Performance testing
 - Automated configuration (provisioning) of new hosts into the ecosystem
 - Push-button deployment to Production
- Benefits
 - Software delivered to DEV in minutes, TEST within an hour, PROD upon push-button approval
 - Developers able to develop and test across latest components
 - Developers rapidly identify problems, deliver functionality to PROD
 - 1-2 weeks from requirements to delivery to production

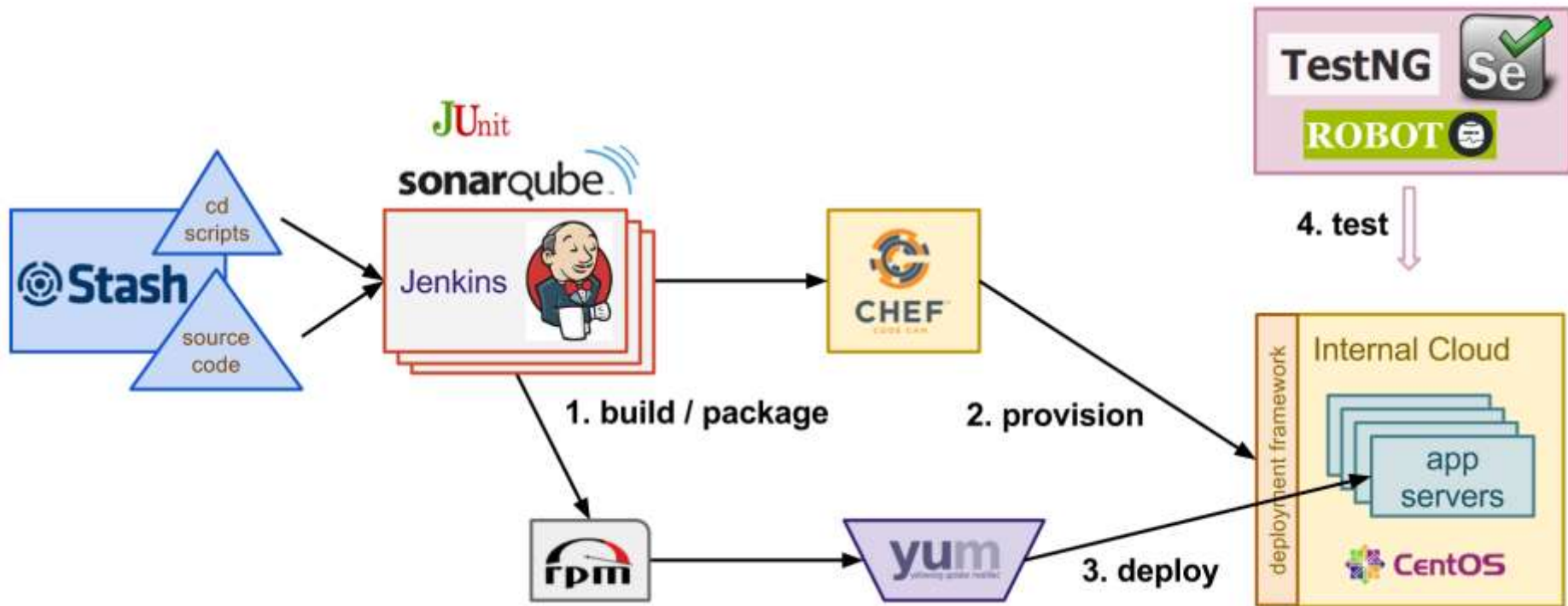


So... what were the challenges and how did we do that?

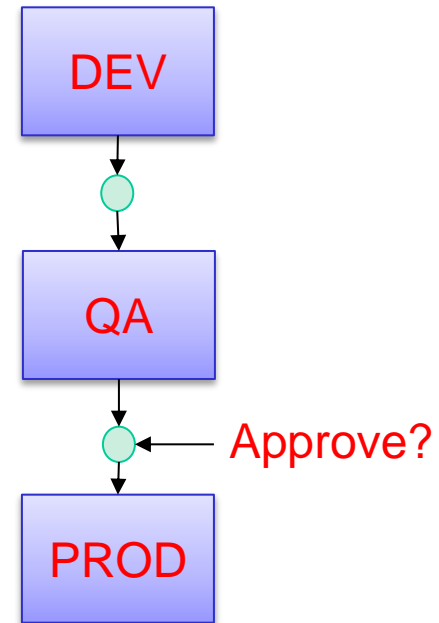
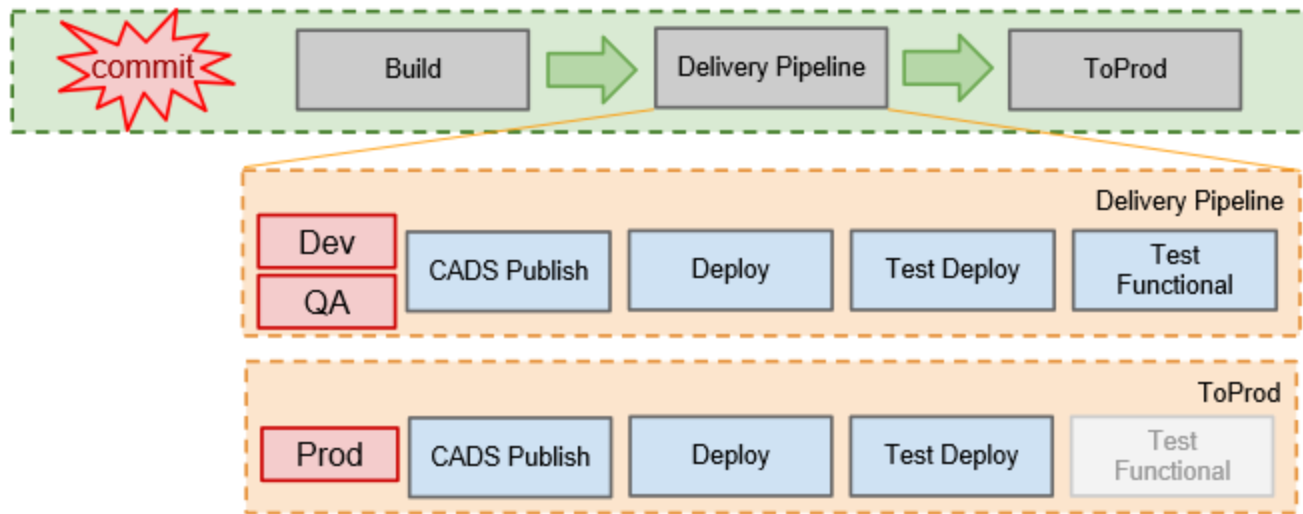
Federations Make for Complicated Delivery



- Pipeline using Git, Jenkins, Sonar, Chef to deliver (mostly) Java-based solutions onto internally managed CentOS Linux cloud environment



- A pattern emerged as we built independent pipelines for our federated system components

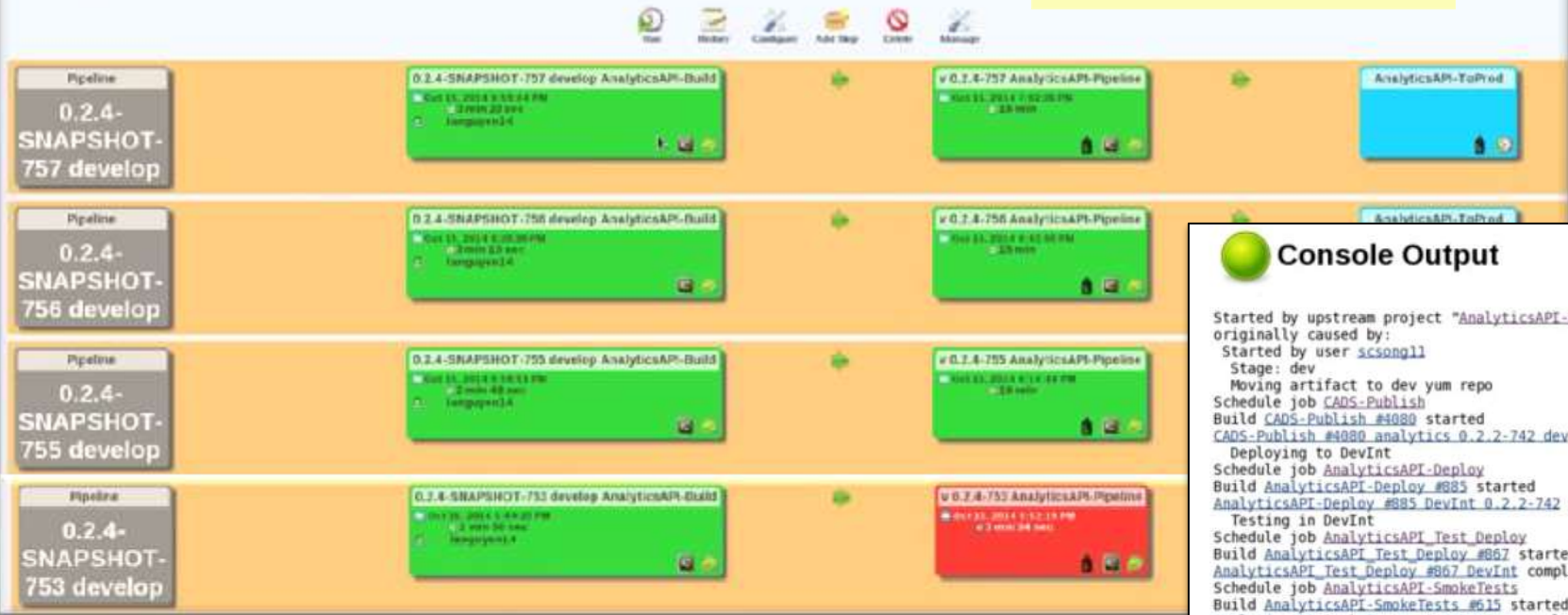


Pipeline Implementation in Jenkins



Build Pipeline View

Build Pipeline



```
[ env: "qaInt", lc: "qa" ]
envs.each() {
  print " Stage: " + it.get('lc') + "\n"
  print " Moving artifact to " + it.get('lc') + " yum repo \n"
  if ( it.get('env') == "DevInt" ) {
    build( "CADS-Publish", ARTIFACT_NAME: "analytics", ARTIFACT_VERSION: params["ARTIFACT_VERSION"], ARTIFACT_VERSION_SUFFIX: "dev" )
  } else {
    build( "CADS-Promote", ARTIFACT_NAME: "analytics", ARTIFACT_VERSION: params["ARTIFACT_VERSION"], ARTIFACT_VERSION_SUFFIX: "qa" )
  }
  print " Deploying to " + it.get('env') + "\n"
  build( "AnalyticsAPI-Deploy", DMP_ENVIRONMENT: it.get('env'), ARTIFACT_VERSION: params["ARTIFACT_VERSION"], ARTIFACT_VERSION_SUFFIX: "qa" )
  print " Testing in " + it.get('env') + "\n"
  build( "AnalyticsAPI-Test_Deploy", DMP_ENVIRONMENT: it.get('env') )
  build( "AnalyticsAPI-SmokeTests", DMP_ENVIRONMENT: it.get('env') )
  build( "AnalyticsAPI-FunctionalTests", DMP_ENVIRONMENT: it.get('env') )
}
```

Console Output

```
Started by upstream project "AnalyticsAPI-Build" build number 742
originally caused by:
Started by user scsongll
Stage: dev
Moving artifact to dev yum repo
Schedule job CADS-Publish
Build CADS-Publish #4080 started
CADS-Publish #4080 analytics_0.2.2-742_dev completed
Deploying to DevInt
Schedule job AnalyticsAPI-Deploy
Build AnalyticsAPI-Deploy #885 started
AnalyticsAPI-Deploy #885 DevInt 0.2.2-742 completed
Testing in DevInt
Schedule job AnalyticsAPI-Test_Deploy
Build AnalyticsAPI-Test_Deploy #867 started
AnalyticsAPI-Test_Deploy #867 DevInt completed
Schedule job AnalyticsAPI-SmokeTests
Build AnalyticsAPI-SmokeTests #615 started
AnalyticsAPI-SmokeTests #615 DevInt completed
Schedule job AnalyticsAPI-FunctionalTests
Build AnalyticsAPI-FunctionalTests #91 started
AnalyticsAPI-FunctionalTests #91 DevInt completed
Stage: qa
Moving artifact to qa yum repo
Schedule job CADS-Promote
Build CADS-Promote #1548 started
CADS-Promote #1548 analytics_0.2.2-742 completed
Deploying to qaInt
Schedule job AnalyticsAPI-Deploy
Build AnalyticsAPI-Deploy #886 started
AnalyticsAPI-Deploy #886 qaInt 0.2.2-742 completed
Testing in qaInt
Schedule job AnalyticsAPI-Test_Deploy
Build AnalyticsAPI-Test_Deploy #868 started
AnalyticsAPI-Test_Deploy #868 qaInt completed
Schedule job AnalyticsAPI-SmokeTests
Build AnalyticsAPI-SmokeTests #616 started
AnalyticsAPI-SmokeTests #616 qaInt completed
Schedule job AnalyticsAPI-FunctionalTests
Build AnalyticsAPI-FunctionalTests #92 started
AnalyticsAPI-FunctionalTests #92 qaInt completed
Finished: SUCCESS
```

Build Flow Plugin

- Often, the hardest part of CD is orchestrating the configuration of multiple sets of servers
- We used Chef “environments” to map a host name to a Chef tag and Chef role
 - Jenkins use Chef **knife** script to read the environment data
 - Jenkins used Chef **knife** to assign tags and roles during one-time “provisioning” phase
 - Jenkins used Chef **knife ssh** commands to execute **chef-client** based on a tag search to execute assigned roles/recipes
 - Jenkins used **knife** to write versions and configuration info into Chef data bags



Attribute	Value
analytics-ut.Production.date	2014-10-06
analytics-ut.Production.version	1.1.0-626
analytics.Production.date	2014-10-06
analytics.Production.version	0.2.2-742
analytics_ut.Production.version	1.0-695
comment	This is a dynamically defined data bag to record software versions that are installed in various environments. It is updated programmatically when software is installed.
component.Production.date	2014-09-18
component.Production.version	1.2.3-857
ibred-tag-cache.Production.date	2014-10-15
ibred-tag-cache.Production.version	1.1-111

HTML Navigation

Taxonomy API CD Pipeline

Pipeline for building, deploying, and testing the Taxonomy API component of DMP.

Stack group preparation jobs

- [TaxonomyStack-Provision](#) - assign and tag all hosts with their Chef roles based on stackgroups/hostmap entries in Chef environment (Devint, QAint, Production)
- [TaxonomyStack-Deploy](#) - install all software needed to run Taxonomy API on it's stack group. This includes Neo4j, Tomcat, and the DMP software.
- [TaxonomyStack-Neo4jMigrateData](#) - migrate data from one environment to another
- [TaxonomyStack-Neo4jBackupData](#) - capture Neo4j data from an environment and upload to file store
- [TaxonomyStack-Neo4jRestoreData](#) - restore data from file store into Neo4j cluster in an environment

Delivery pipeline jobs. Refer to [Taxonomy Delivery Pipeline](#) view for visual presentation. Uses the Taxonomy Pipeline jobs [Taxonomy-Pipeline](#) job to actually orchestrate the installation and testing.

Utility jobs that do individual work to support the pipeline

- [Taxonomy-Build](#) - compile latest Taxonomy API source, run unit tests, and publish to Yum repository. This also kicks off the CD pipeline into Devint.
- [Taxonomy-Deploy](#) - run Chef to install the Taxonomy API on to servers in a specific environment.
- [Taxonomy-Test-Smoke](#) - run deployment smoke tests to ensure that the software installed and is responsive
- [Taxonomy-Test-Functional](#) - run functional system tests against the API using Test NG
- [Taxonomy-Test-Integration](#) - run code-based integration tests against the API using JerseyTest
- [Taxonomy-Profile](#) - Build flow job to execute all jobs together through Devint and QAint
- [Taxonomy-Prod](#) - Short pipeline build flow job to promote and execute job through Production

Maven project Taxo

Build DMP Taxonomy API code and docu
For Sonar analysis, view [Sonar Taxonomy](#)
For latest Swagger API documentation, view [Swagger](#)
For latest Javadoc documentation, view [Javadoc](#)

Taxonomy Common [Jenkins](#)
Taxonomy Embedded [Jenkins](#)

Workspace

Last Successful Builds

- taxonomy-0.2.9-1052-develop-rpm 50.07 MB [view](#)
- taxonomy-wit 56.01 MB [view](#)

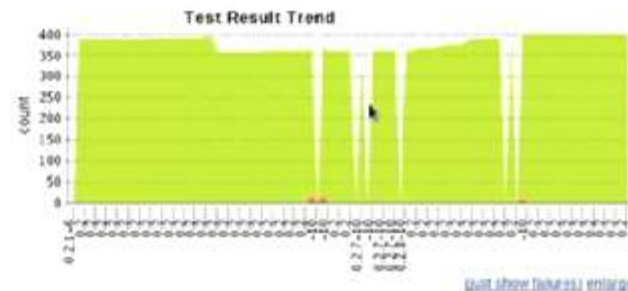
Recent Changes

Latest Test Result (no failures)

0.2.9-SNAPSHOT-1075	develop	Oct 1
0.2.9-SNAPSHOT-1074	develop	Oct 1
-1073	develop	Oct 1
-1072	develop	Oct 1
0.2.9-SNAPSHOT-1071	develop	Oct 1
-1070	develop	Oct 1
0.2.9-SNAPSHOT-1069	develop	Oct 1
0.2.9-SNAPSHOT-1054	develop	Oct 9
0.2.9-SNAPSHOT-1053	develop	Oct 8, 2014 7:54:39 PM
0.2.9-SNAPSHOT-1052	develop	Oct 8, 2014 2:56:51 PM
0.2.8-1051	develop	Oct 8, 2014 2:53:24 PM
DMP v0.2.5 2014.10.08		
0.2.8-SNAPSHOT-1050	develop	Oct 7, 2014 9:44:39 PM
0.2.7-1049	master	Oct 7, 2014 9:33:12 PM
0.2.8-SNAPSHOT-1048	develop	Oct 7, 2014 8:59:39 PM

Project disk usage information + trend graph

Disk Usage: Workspace 439978632, Builds (all=2144823806, locked=461689411), Job directory 2364934878



Maven Build Jenkins Plugin
M2 Release Jenkins Plugin
jGit-Flow Maven Plugin

Build Name Setter

- Digesting a multi-node Jenkins CD system with LOTS of jobs

Welcome to the DMP Jenkins Continuous Delivery Pipeline

This is responsible for building, integrating, testing, and deploying the Data Management Platform components.
GUIDELINES: For information on creating and maintaining jobs for your teams, visit the [DMP Jenkins Job Guidelines](#) Google doc.

- Chel Server
- Source dashboard
- Production artifact versions and install data

All CD Pipelines Components DMP Metrics DMP Offline Segmenter Dashboard DevOps Pipeline Dashboard Tests v0.1 Edit

Dashboard View

Nested View

Latest Builds

Job	Build	Time
ELSAgent-Monitor	#4415	Oct 17, 2014 8:53:45 PM
ELSAgent-Monitor	#4414	Oct 17, 2014 7:53:45 PM
Taxonomy_Test_Functional	#1275 Devint	Oct 17, 2014 7:38:43 PM
Taxonomy_Test_Deploy	#1538 Devint	Oct 17, 2014 7:38:28 PM
Taxonomy_Deploy	#1461 Devint 0.2.9-1092	Oct 17, 2014 7:32:58 PM
Nexus-Publish	#4413 taxonomy	Oct 17, 2014 7:31:13 PM
CADS-Publish	#4465 taxonomy 0.2.9-1092 dev	Oct 17, 2014 7:31:13 PM
Taxonomy_Test_Functional	#1274 Devint	Oct 17, 2014 7:26:58 PM
Taxonomy_Test_Deploy	#1537 Devint	Oct 17, 2014 7:26:43 PM
Taxonomy_Deploy	#1460 Devint 0.2.9-1091	Oct 17, 2014 7:19:38 PM

Unstable Builds

Job	Build	Time
AdaptVingestor_Build		
AnalyticsUI_Build		
AnalyticsUI-SmokeTests		
offline-kettle-wrapper_deploy		
offline_consumer_ui		
Pixel-Streamer_Test		
Segmentindexer_Build		
Taxonomy_Test_Functional		
UserKeyStore-Deploy		

Scene: At the end of a well-played demo of the brand new UI by the product team after a week long “hack-a-thon” with 20 Dev/QA/Ops guys in a room...

CTO: “That was an awesome demo ... and it’s all being done with CICD, right?
<awkward laughing>”

Me: “Yes. Of course.”

CTO: “Really?!”

Me: “Yes. We built the automation with the dev team while they were writing the code.”

CTO: <shocked look> “Damn. That really IS awesome.”

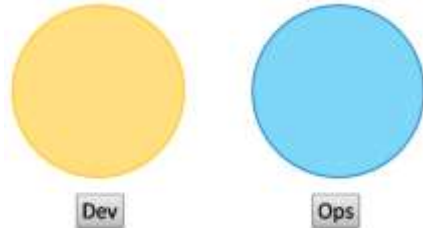


- Integrated, dedicated personnel
 - Ops ... machine configuration, admin skills
 - Dev ... code ownership, unit tests, configuration
 - QA ... test code, rapid feedback
- DevOps “boot-strap” skillset (fill the gap)
 - Integrated DevOps mentoring
- Remove barriers: get people what they need when they need it (machines, tools, network access, whatever...)
- Automated testing is critical
 - Build confidence in your application
 - Continuous and incremental improvement: ratchet up quality goals that can't be hit on day one

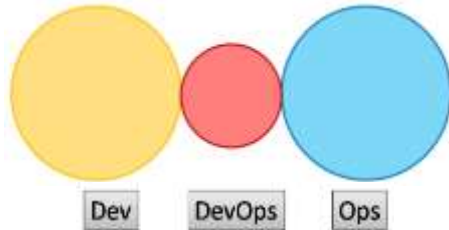


Problematic

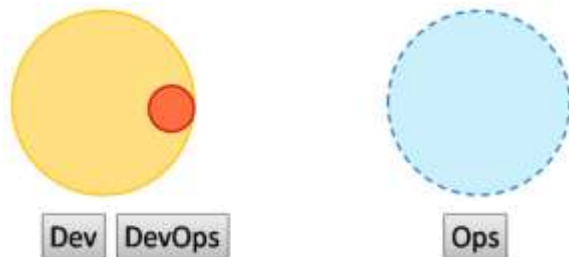
- Separate silos



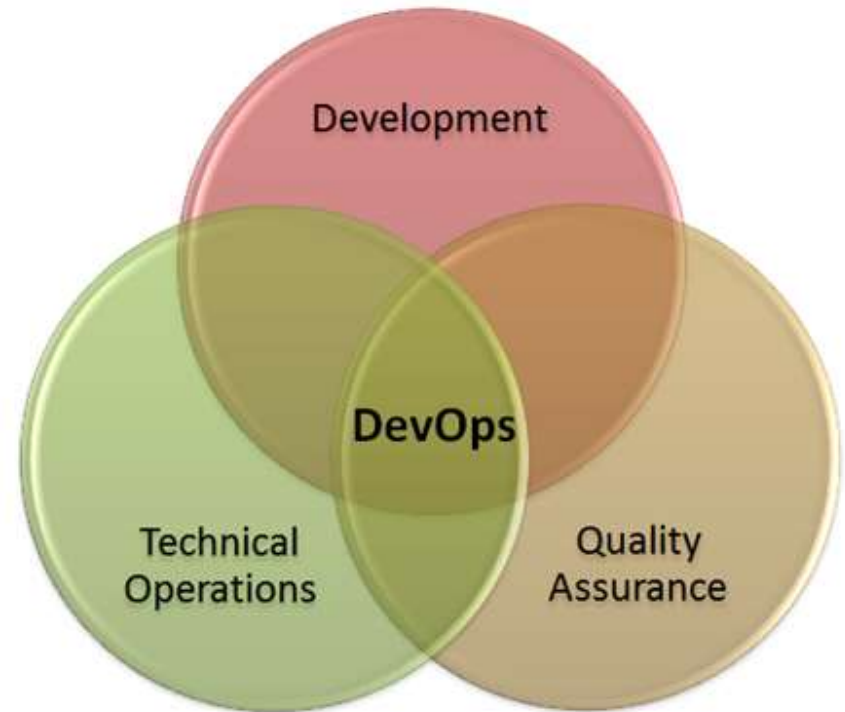
- Dev, Ops, DevOps silos



- No Ops



Desired



- Automate the most frequent (or painful) thing(s) first
 - Compile
 - Unit test
 - Deploy/configure your app
 - Test your deployment
 - Deploy/configure middleware
 - Configure new servers
 - Launch/harvest dynamic servers
- Don't forget automated testing!
- Use your build server as the center of everything (initially)
- Eventually: figure out how to “engineer” your pipeline



Thank you for your time!

Questions?

Contact Information:

Richard Mills
rich.mills@coveros.com
703.585.8961
@armillz



www.coveros.com