EXTERNAL SOURCE CONTROL & PENTAHO

One-button export, formatting and standardization, commit, and deploy from separate environments.

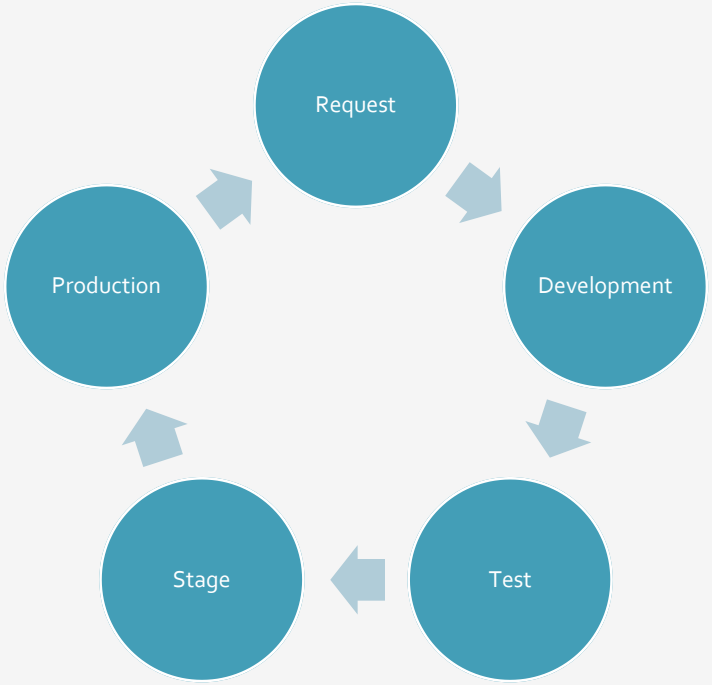
NEXTGEAR CAPITAL

About NextGear Capital and Nathan Hart

- NextGear Capital
  - Formed in 2013 with the merger of Dealer Services Corporation and Manheim Automotive Financial Services
  - Part of Cox Automotive Inc (Manheim Auto Auctions, AutoTrader, Dealertrack, Kelly Blue Book, …)
  - Over 22k clients (mostly independent auto dealers) across US, CA, and UK
- Me
  - Started at NextGear in March 2015, first introduction to PDI
  - Working in BI since June 2009

NEXTGEAR CAPITAL

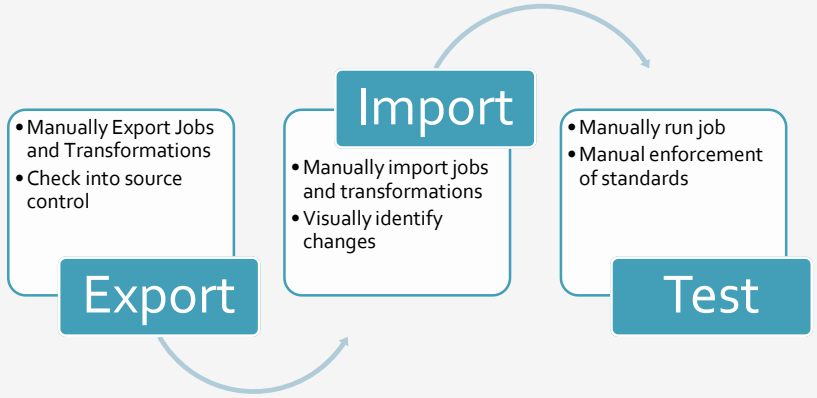
Environments and Configuration



- Environments – Unix & Postgres
  - Development
  - Test
  - Stage/Pre-Prod
  - Production
- Team Build
  - Three devs (up to five)
  - Two QA
- Stats
  - 93 production jobs, 118 active schedules
  - 75 common utilities and test benches
  - 35 unique partners / “families”

NEXTGEAR CAPITAL

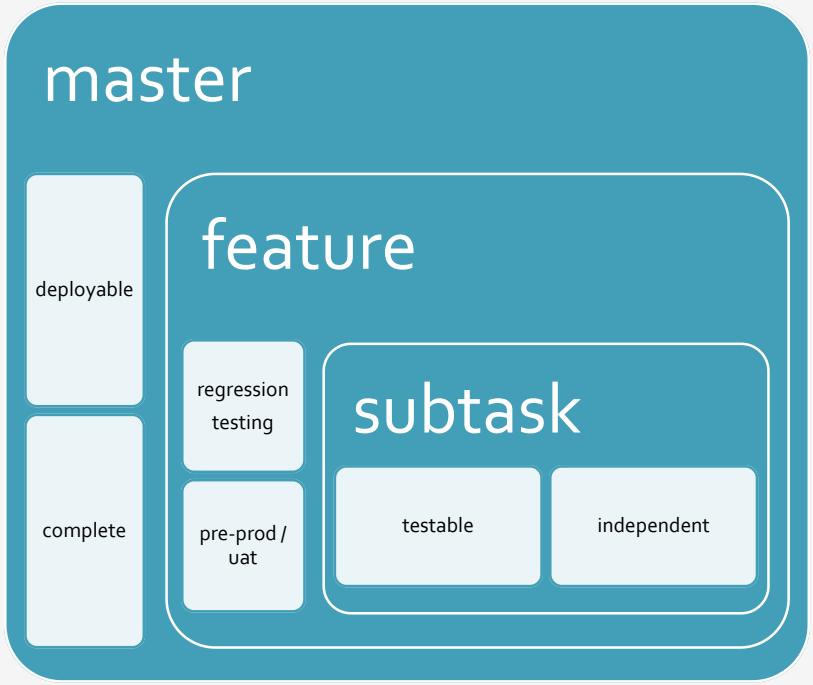
Challenges



- Manual Heavy
  - Exports / Imports are all manual
  - Open to human error (missed utilities)
  - Slow deployments
- Difficult to Test
  - Harder to determine what changed
  - Standards are manually enforced, or missed
- Source of Truth
  - Where is the "true" version of a give job or transformation
  - Track same file throughout development cycle
  - Difficult to shelve changes

NEXTGEAR CAPITAL

Solution? Git!



Benefits of External Source Control

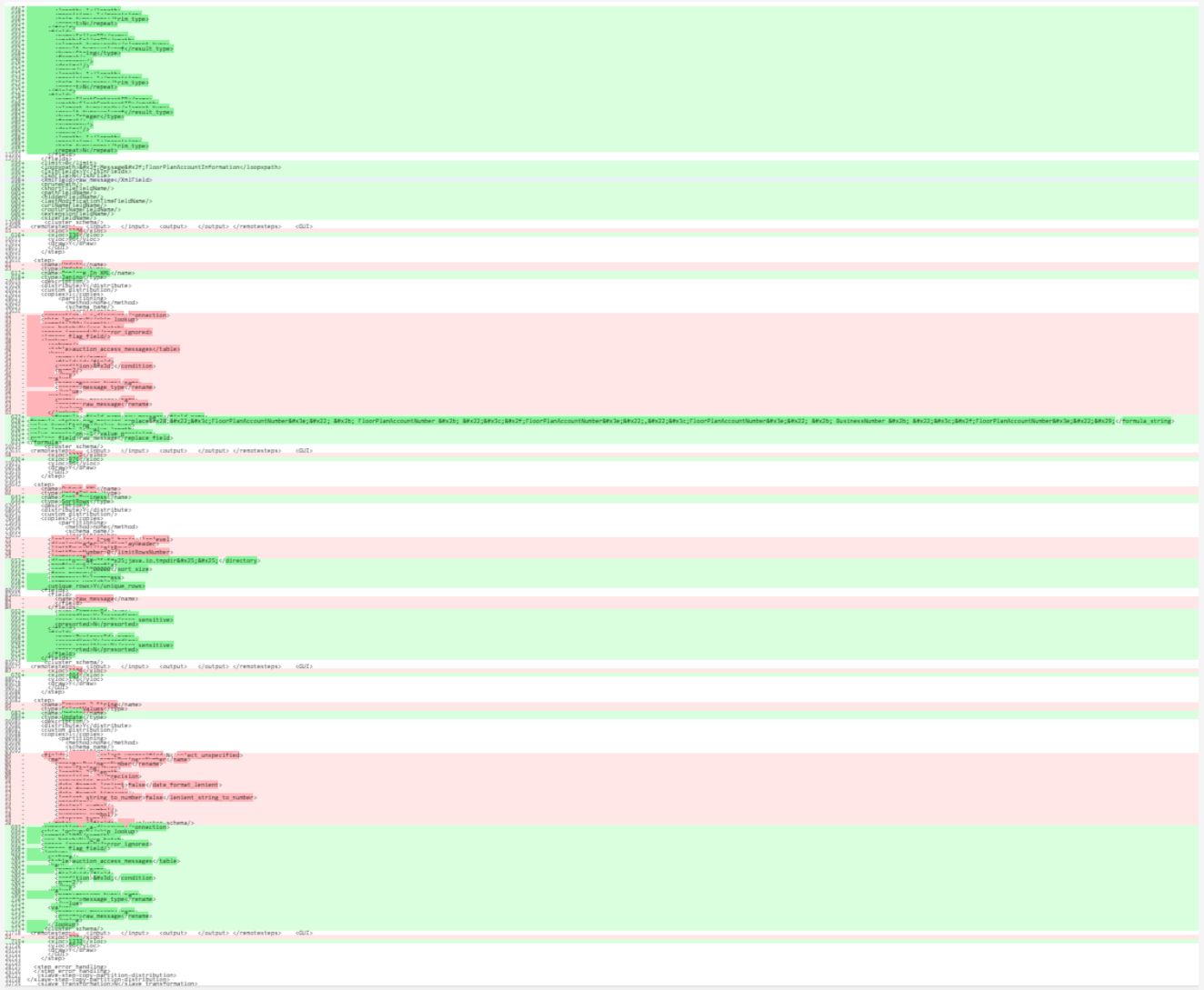
- Allows for automated build and deploys
  - Upon check-in, scripts to validate files against standards
  - Auto-deploy to next environment when appropriate
- Change Identification
  - Can provide list of changes / checklist for deployment
- Improved Testing
  - Allows smaller changes to be promoted and tested while development continues
- Production Support
  - Deployable copy of production to any environment from "true" copy

NEXTGEAR CAPITAL

Jobs and Xfrms in Git

The XML that makes up the ktr and kjb files can be fairly fluid when comparing version over version after export. This makes tracking changes and differentials very difficult.

Solution? Alphabetize the XML on export!



NEXTGEAR CAPITAL

Tools and Requirements

- Data Integration
  - Kitchen
  - Import
- CLI
  - curl
  - git
- Liquibase

Export

- curl –X GET
- http://{pentaho-server:port}/pentaho/api/repo/files/{path2file}/download
- Returns zip of individual file
- Unzip, parse for additional resources. Export and repeat at necessary.

Commit

- Split XMLs and sort alphabetically
- Determine unchanged utilities, remove
- Commit and push to branch

Validate

- Check variable usage against configurations
- Check naming convention

Import

- import script that comes with client version (Import.bat / import.sh)
- -rep –user –pass –dir –comment –norules –replace –file
- Push configurations
- Deploy database changes

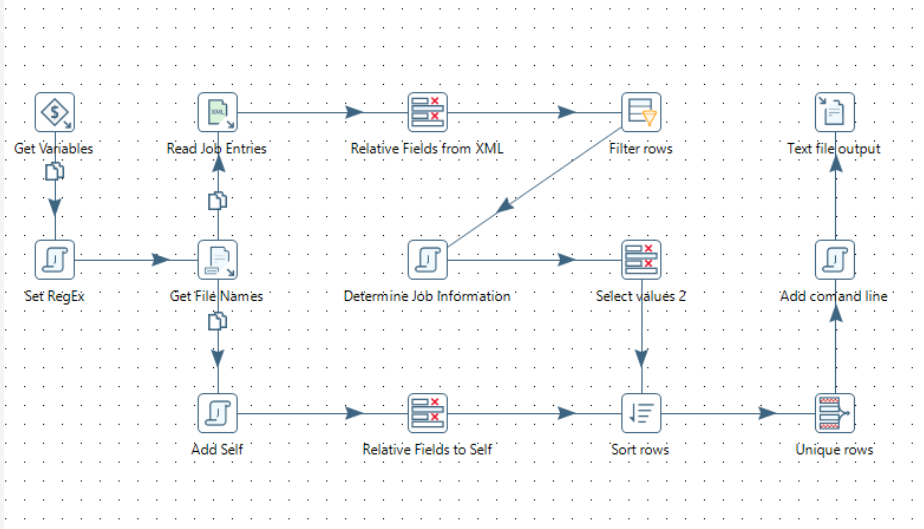
Test

- Kick off execution
- Post results

NEXTGEAR CAPITAL

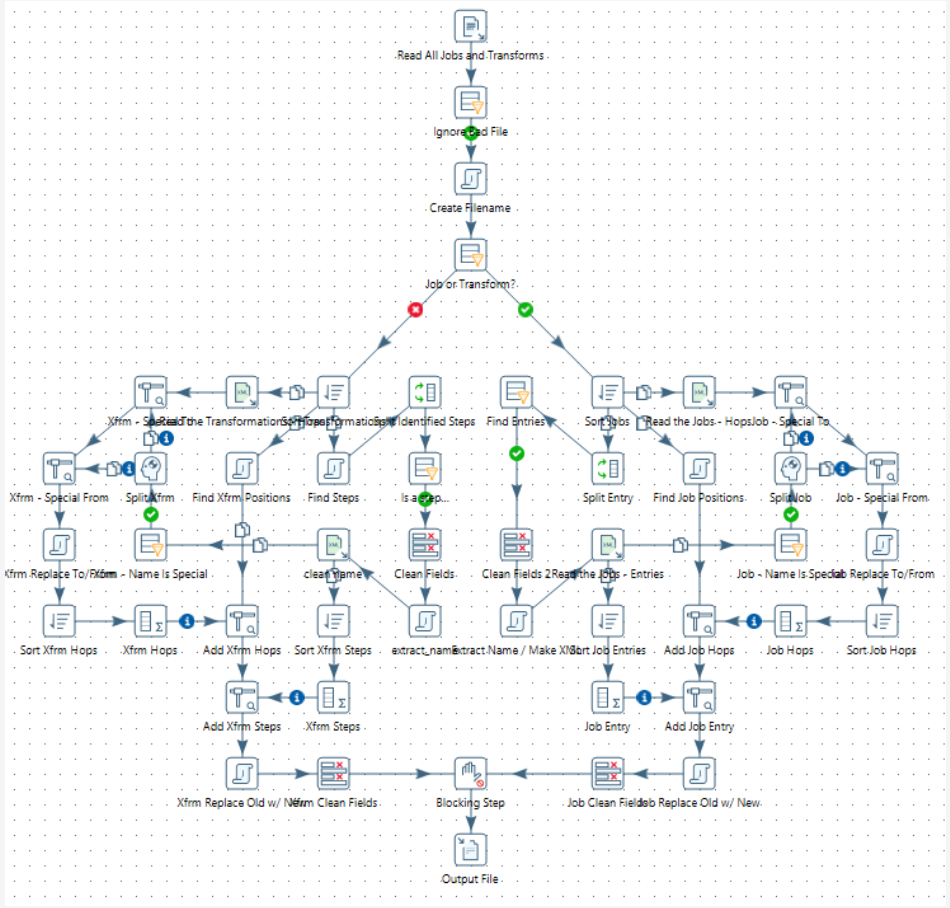
Export

1. curl -X GET -u ${username}:${pass} http://${pentaho-server}:${port}/pentaho/api/repo/files/public/${jobFamily}/${jobname}.kjb/download > /shared/jobs/files/tmp/${jobname}.zip
2. Use kitchen to call "Get References by Job" utility (custom job)
    a. Unzips resulting file to "export" directory
    b. Xfrm to parse XML for references to subjobs and transformations -> export into "export" directory; move to "clean" directory
    c. Recurse through "export" directory until empty



NEXTGEAR CAPITAL

Clean and Commit



1. Use kitchen to call "XML Manipulation" (custom job)
   - Run against "clean" directory
2. Compare cleaned files against existing branch, remove unchanged
   - Move rest to "commit" directory
3. Copy files into git structure and commit*
   - Move committed to "import" directory

* Git must already be on desired branch

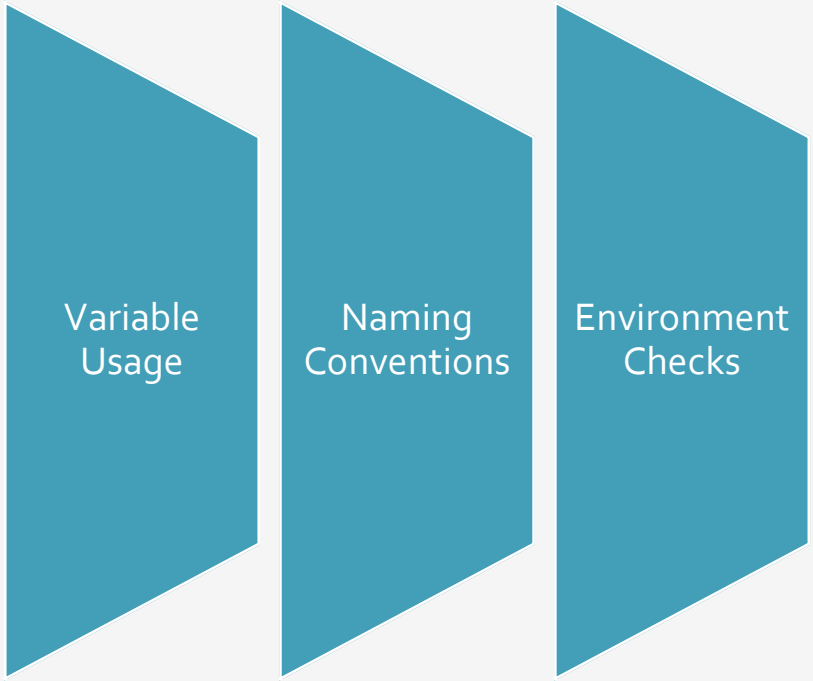
XML Manipulation (custom job)

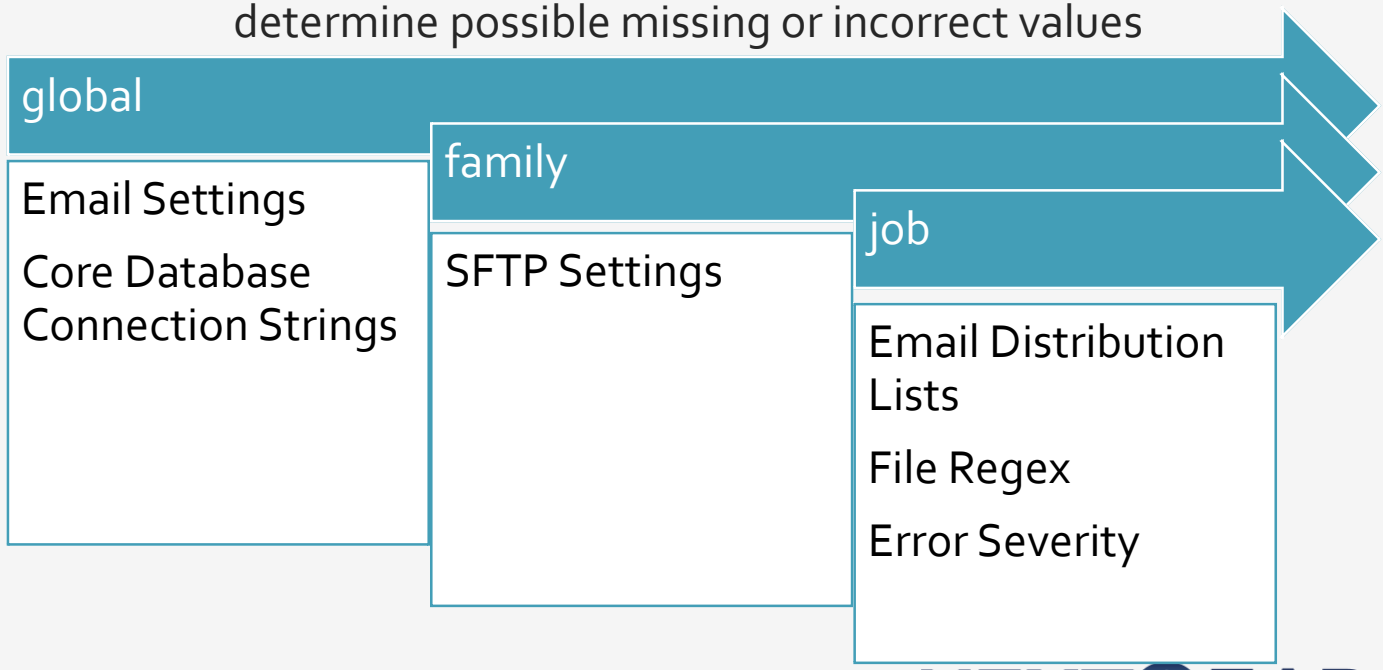
- Enable Database Logging
- Check Database Connections
- Purge Slave Servers, Partitions and Clusters
- Standardize Email Steps
- Check Utility Paths
- Confirm Utility Variables
- Alphabetize XML
- Set Variable Scope

NEXTGEAR CAPITAL

Validate

- Checking against standards
  - Parsing XML for used variables and comparing against configurations
  - Comparing job name and location against rules
  - Checking configurations for each environment to determine possible missing or incorrect values





NEXTGEAR CAPITAL

Import

```
#######################################################################
# execute the push of files
# we do it this way to allow for concurrency of pushes to the server regardless
# of user id (personal or pentaho)
#######################################################################

if { $entireStructure == "TRUE" } {
    # not going to happen at this time - too dangerous until we discuss this with the team
    # swap src and dst if using below line
    #exec scp -r $pentahoUser$serverName:/shared/\* $dataPullDestination/shared/.
    puts "ERROR: you have requested that the entire /shared file structure be pushed ... Sorry not at this time."
    exit 1
} else {

    # establish the base direcrtories
    exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/config
    exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/sshkeys/
    exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/shellScripts/
    exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/files
    exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/resources

    if { $globalPropertiesUsage == TRUE } {
        # not going to happen at this time - too dangerous until we discuss this with the team
        exec scp $sourceRoot/config/global.properties $pentahoUser$serverName:$remoteTmpRoot/config/.
        exec scp $sourceRoot/config/global_$environment.properties $pentahoUser$serverName:$remoteTmpRoot/config/.

        exec scp -r $sourceRoot/shellScripts $pentahoUser$serverName:$remoteTmpRoot/.

        exec scp -r $sourceRoot/sshkeys $pentahoUser$serverName:$remoteTmpRoot/.

        # this is sucky - there is no pattern to what other directories or files might exist at family level -
        # no way to guess other job names and their associated directories thus unlike properties, this is all or nothing
        # all files for all jobs in family instead of perferred family resource dirs/files and just file dir for the job
        # same goes for global
        exec scp -r $sourceRoot/files $pentahoUser$serverName:$remoteTmpRoot/.
        exec scp -r $sourceRoot/resources $pentahoUser$serverName:$remoteTmpRoot/.

    } else {
    }

    if { $familyPropertiesUsage == TRUE } {
        # push the family generic and only the environmental file that will be used ####
        # on that server ###########################################################
        exec ssh $pentahoUser$serverName mkdir -p $remoteTmpRoot/config/$family
        exec scp $sourceRoot/config/$family/$family.properties $pentahoUser$serverName:$remoteTmpRoot/config/$family/.
        exec scp $sourceRoot/config/$family/$family\_$environment.properties $pentahoUser$serverName:$remoteTmpRoot/config/$family/.

        # We use a find since a bash [ -e ] is not possible within the confines of expect/tcl
        set familyShellScriptPathExists [ exec find $sourceRoot/shellScripts/ -name $family -print ]

        if { $familyShellScriptPathExists eq "" } {
            puts "INFO: no $family shell script path."
        } else {
            # We use a -A count to verify that non-'.' files exist
            set familyJobShellScriptFileNPathCount [ exec ls -A $sourceRoot/shellScripts/$family/ | wc -l ]

            if { $familyJobShellScriptFileNPathCount == 0 } {
                puts "INFO: no $family/job shell script path nor files in $family."
            } else {
```

- Push configuration files to target server
  - Create missing directories
  - Upload necessary resources (templates, starting data)
  - Execute liquibase changesets
- ./import.sh -rep=${repo} -user=${username} -pass=${pass} -dir=/ -comment="${comment}" -norules -replace=Y -file="/shared/jobs/files/tmp/import/${filename}"
- curl -X POST -u ${username}:${pass} http://${pentaho-server}:${port}/pentaho/kettle/executeJob?job=/public/${jobFamily}/${jobname}/${jobname}&rep=${repo}&level=Detailed&user=${username}&pass=${pass}

NEXTGEAR CAPITAL

Test

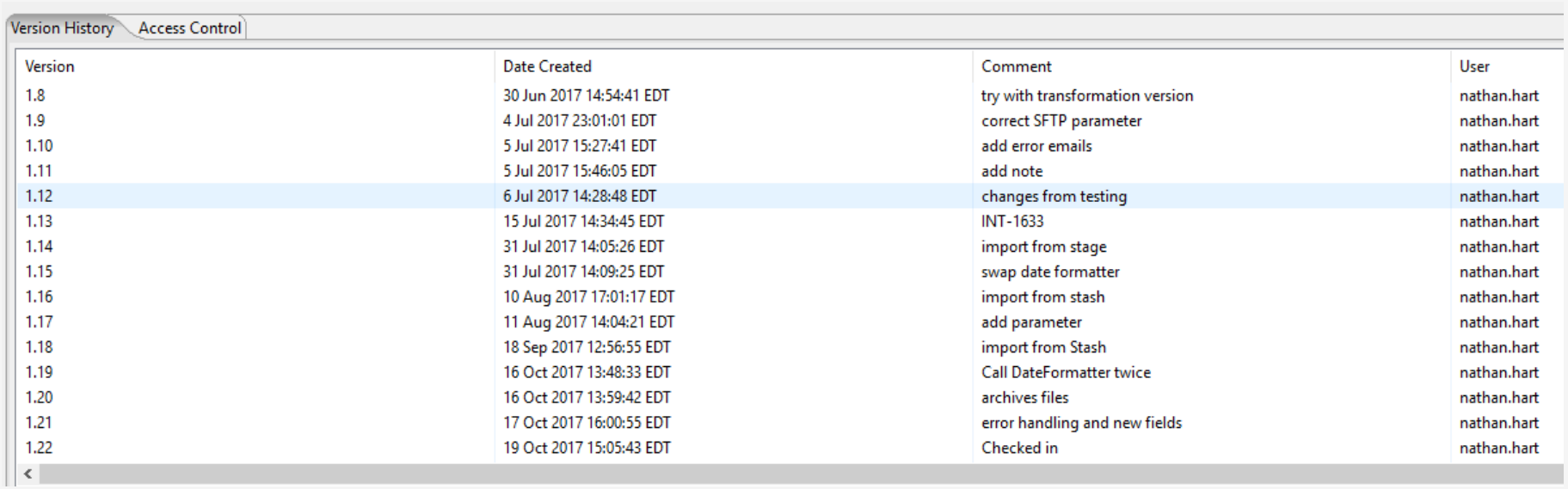
Parse execution results

- Did it run out of the box?
- Did it follow the happy path?
- Run post-execution validation script

NEXTGEAR CAPITAL

Still using Pentaho Repository

- Offers a history of deployments to each environment
- Allows restoring/rollbacks when necessary
- Provides a more visual history than Git
- Can easily flip between branches in single environment



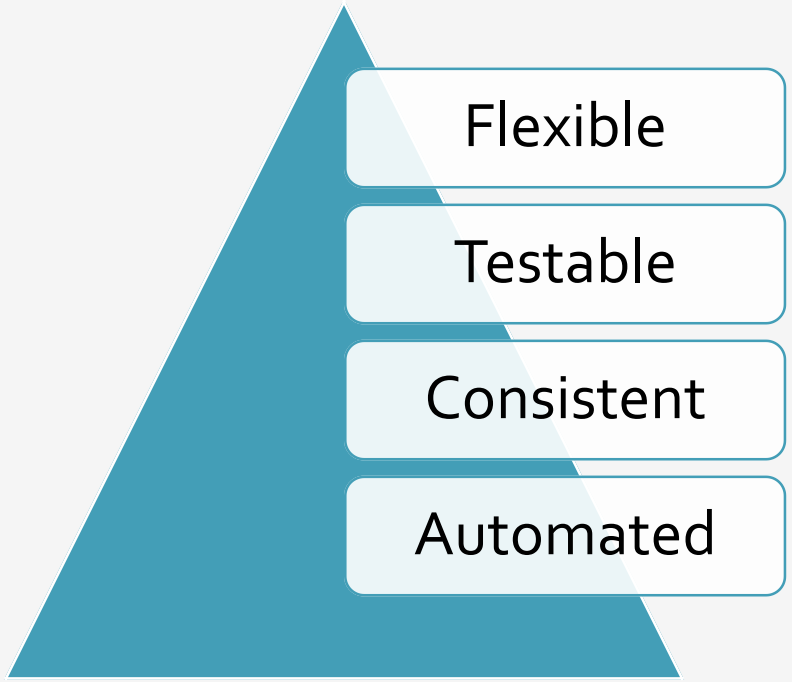
Version History | Access Control

Version	Date Created	Comment	User
1.8	30 Jun 2017 14:54:41 EDT	try with transformation version	nathan.hart
1.9	4 Jul 2017 23:01:01 EDT	correct SFTP parameter	nathan.hart
1.10	5 Jul 2017 15:27:41 EDT	add error emails	nathan.hart
1.11	5 Jul 2017 15:46:05 EDT	add note	nathan.hart
1.12	6 Jul 2017 14:28:48 EDT	changes from testing	nathan.hart
1.13	15 Jul 2017 14:34:45 EDT	INT-1633	nathan.hart
1.14	31 Jul 2017 14:05:26 EDT	import from stage	nathan.hart
1.15	31 Jul 2017 14:09:25 EDT	swap date formatter	nathan.hart
1.16	10 Aug 2017 17:01:17 EDT	import from stash	nathan.hart
1.17	11 Aug 2017 14:04:21 EDT	add parameter	nathan.hart
1.18	18 Sep 2017 12:56:55 EDT	import from Stash	nathan.hart
1.19	16 Oct 2017 13:48:33 EDT	Call DateFormatter twice	nathan.hart
1.20	16 Oct 2017 13:59:42 EDT	archives files	nathan.hart
1.21	17 Oct 2017 16:00:55 EDT	error handling and new fields	nathan.hart
1.22	19 Oct 2017 15:05:43 EDT	Checked in	nathan.hart

NEXTGEAR CAPITAL

Conclusion

By using an external source control solution with multiple Pentaho environments, we can greatly simplify the workflow for our developers and especially QA. Automating the import/export process as well as standardizing the output gives greater consistency in our codebase, making it easier to identify outliers. This also provides an additional layer of transparency to our work and seeing feature progress and movement throughout the development process. Using external build and deploy tools open us up for more automating testing and future enhancements. This allows us to get the most of our the existing Pentaho Repository structure without the limitations of multiple environments / parallel development cycles.



Questions?

NEXTGEAR CAPITAL