Consultingwerk
Continuous Deployment of OpenEdge Applications
Daniël van Doorn
Senior Consultant
Agenda

- About us
  - Introduction Continuous Deployment
  - Based on a real life case
  - Philosophy / Goals
  - Process
  - Build pipeline
  - Deployment pipeline
  - Deployment per item
  - Tips and best practices
Consultingwerk Software Services Ltd.

- Independent IT consulting organization
- Focusing on **OpenEdge** and related technology
- Located in Cologne, Germany, subsidiaries in UK and Romania
- Customers in Europe, North America, Australia and South Africa
- Vendor of developer tools and consulting services
- Specialized in GUI for .NET, Angular, OO, Software Architecture, Application Integration
- Experts in OpenEdge Application Modernization
Daniel van Doorn

- Senior Consultant
- Started developing Progress in 1997 with version 7.
- Experience working in several different roles varying from developer / R&D / technical application manager / “The DevOps guy” to lead architect
- Spent the last 10 years on R&D and building development factories.
Continuous Deployment

"THE TESTER IS NOT AT HIS DESK!
YOU CAN DEPLOY, HURRY UP"

Here is what they always forget to tell new testers in the job interview
- **Continuous Delivery**
  - The complete iterative process of
    - Development
    - Version Management
    - Building
    - Quality control
  - Software package ready to deploy at any moment

- **Continuous Deployment**
  - Really deploy at any moment
  - Automatic (!)
  - Deploy an environment to an “empty” system
Based on a real life case

Starting conditions:

Almost no version control (Round table / SVN only trunk)

Monolithic application(s) from various periods in the history of OE

OpenEdge TTY Linux app, Dynamics, SmartComponent Library, ADM/2, webspeed

Manual “build” manual deployment

300 patches in prod of unknown origin

Not always the most elegant solution, but a real/practical solution based on available budgets, etc.
Choices and decisions

- Build in Jenkins with Ant and PCT
- We need Ant and PCT to deploy on the servers.
  - Platform independent
  - We need the database generation and schema diff upon install
- Deployment using Jenkins with slaves
- The OpenEdge webclient executables are deployed by company wide published applications
Philosophy

- The truth is always in SCM for configuration and sources
- All development / build / deployment properties in one location
- Commits in must be always done with an issue tracker ID (JIRA, etc)
- Continuous integration build to give the developer feedback within 5 minutes
- Nightly builds with more checks, always fresh check-out, invoke unit tests
- Builds deliver their binaries to a software store / depot
- From any release to any release (downgrade must be possible)
(primary) Goals

✓ Easy upgrade process, eliminate manual labour (= bugs)
✓ Provable software
✓ Predictable software
✓ Install from scratch
✓ Reduce downtime
✓ Reduce TTM
✓ Prevent (the need) to access (production) servers
BUILD

Source Code Repository
Developer commits code to source code repository, for example Subversion, GIT, etc.

Automated Builds

Software Repository
A software repository, such as Artifactory or Nexus, holds the output of each build, enabling each release to be deployed to any environment as required.

RELEASE

Automated Package Deployment

Automated Package Configuration

SIT

UAT

PROD
Prerequisites

- All items must be under version control
  - Sources
  - DF’s
  - Scripts
  - Tools
  - Configuration
One configuration file to rule them all

- The ultimate goal should be to have all configuration for all actions from development to deployment in one properties file.
- Use java properties format
- Nicely name your properties
- Write parsers to use the properties in all used technologies

```java
Product.Appserver.BatchAppserver.Parameter.debuggerEnabled=1
```
Build pipeline

- No we don’t do that here, there are enough sessions on CD/CI
- Just produce an archive with the application release
  - The archive should contain:
    - OpenEdge library
    - All configuration needed to deploy and execute
    - Installation tools
    - The DF(s) with the latest DB schema
    - Release bound data
    - Audit information
Deployment pipeline

- Create the deployment jobs with Ant and PCT
  - (Should be just re-using parts from development and build scripts)
- Download the binaries from the software store
- Update all databases and both on primary and fail over appservers / webservers, scripts and services in parallel
Offline deployment

Pre deployment tasks
- Start
  - Extract archive, Build temp databases
  - Generate delta DF and Data
  - Determine what to update
- Batches
- Webservers
- Appservers
- Watch dogs

Shutdown
- Databases
- Appservers
- Webservers
- Batches
- Watch dogs
- Unix TTY
- Scripts
- Configuration

Deploy
- Appservers
- Databases
- Webservers
- Batches
- Watch dogs
- Unix TTY
- Scripts
- Configuration

Start up
- Monitor
- Feed back
- Cleanup
- Undo

Post deployment tasks
Online deployment

Pre deployment tasks

- Extract archive, Build temp databases
- Generate delta DF and Data
- Determine what to update

Shutdown tasks

- Start
- Batches
- Databases
- Web servers
- Unix TTY
- Scripts
- Configuration

Deploy tasks

- Start
- Batches
- Databases
- Web servers
- Unix TTY
- Scripts
- Configuration

Post deployment tasks

- Start up
- Deploy
- Monitor
- Feedback
- Cleanup
- Undo

Start up

- Appservers
- Databases
- Batches

Undo
Database

- Deploy using Ant and PCT
- Build temporary database
- Use PCT to generate a delta
- Use own tool to generate delta data
- Always re-deploy the control scripts for starting / stopping / replication / etc.
Database update - Offline strategy

- For each database in production, build a temp-database
  - With a special .st file to keep it as small as possible.
  - Load the DF’s and release bound master data
  - Diff the schema’s and the data with the prod instance.
  - Dump the delta
  - Shut down databases
  - Update all the schema of all databases
  - Start up all databases
  - Load the release bound master data changes
Database update - Online strategy

- It’s not hard
  - Educate the developers to not delete or change database objects
    - Create a new ticket and leave the deletion for a major release update
    - PCTLoadSchema can handle online schema changes
  - (More to come in OE V12 !)

(https://knowledgebase.progress.com/articles/Article/P79975)
Progress libraries

- Put a date and or hash in the file name
  - (but start with something human readable, to be nice to your colleagues)
  - Enables you to place a new library next to the old one
  - Let old sessions naturally die
PAS Appservers

- Re-deploy all configuration from the properties file
- Deploy all files part of the application not in the library
- Put the new library in place
- Trim appserver agents
Classic Appservers / Webspeed brokers

- Add the new pl.
- Update configuration files
- Update ubroker.properties
- Trim appserver agents
Parsing the ubroker.properties file

- Load the file in a longchar
- Break up the lines by chr(10)
- Find the line with [UBroker.AS/WS.&1]
- Add all lines in the section to a variable
- Then find the next empty line.
- Do a replace of the block by the new configuration from the properties file
- Keep the ident(!) and don’t forget the empty line
- Containing o.a. the PROPATH with the new library
Web Client

- A client consists of three packages:
  - Boot Loader
  - Installer
  - Client

- Use C:\Users\<User>\Appdata\MyProduct to install the application
- The only thing going in the company .msi factory is 1 .r an icon and shortcut
- The boot loader starts a new OpenEdge process with the installer
Web client II

- Upon install the release hash is stored in the local history.
- When the hash differs from the server hash a new version will be downloaded and installed.
- Check if there is already an instance of the application running for this environment
  - Tip! Put the environment name in the command line parameter
- We did not want incremental updates
- And however it’s a nice product, we did not have budget for Riverside PDO ([http://riverside-software.fr/pdo.html](http://riverside-software.fr/pdo.html))
UNIX TTY

- Add the new pl
- Redeploy the configuration files
- Redeploy the application start script with new PROPATH
The installer checks if the app is already running

(Tip always add the environment name to the shortcut starting the app)

```csharp
oProcess = System.Diagnostics.Process:GetProcessesById( piProcessID);
objManScope = NEW System.Management.ManagementScope("\root\cimv2":U);
objManScope:CONNECT();

objManObjSearcher = NEW System.Management.ManagementObjectSearcher(
    SUBSTITUTE(
        "SELECT CommandLine FROM Win32_Process WHERE ProcessId = &1":U,
        INTEGER(piProcessID)
    )
).

cmdEnumerator = CAST (objManObjSearcher:Get(), System.Collections.IEnumerable):GetEnumerator().

DO WHILE cmdEnumerator:MoveNext() ON ERROR UNDO, THROW:
    ASSIGN
        cmd = CAST(cmdEnumerator:Current, ManagementObject).
    RETURN STRING (cmd:GetProperty("CommandLine":U)).
END.

FINALLY:
    GarbageCollector:DeleteObject(oProcess).
    GarbageCollector:DeleteObject(oStartInfo).
```
Windows service wrapper for never ending processes

- Windows Service wrapper [https://github.com/kohsuke/winsw](https://github.com/kohsuke/winsw)
- Executable with start stop install and uninstall options

```xml
<configuration>
  <id>[id]</id>
  <name>Batch Scheduler Service ([version])</name>
  <description>Scheduler Service</description>
  <executable>[prowinexe]</executable>
  <arguments>[parameters]</arguments>
  <log mode="none"/>
  <logpath>[logpath]</logpath>
</configuration>

<!-- Build the service configuration -->
```

```xml
<replace file="${Product.Target.Root}/Bin/BatchScheduler.xml" token="[version]" value="${install.version}"/>
<replace file="${Product.Target.Root}/Bin/BatchScheduler.xml" token="[DLC_BUILD]" value="${DLC_BUILD}"/>
<replace file="${Product.Target.Root}/Bin/BatchScheduler.xml" token="[logPath]" value="${Product.Target.Root}/Log"/>
```
Docker

- Docker is slowly getting adopted in the OpenEdge world
- It will be available shortly for PAS appservers
- Configure the docker image in the build
- Place image in a software store / depot (i.e. https://hub.docker.com/)
- Moves the configuration from deployment to build
Correction programs / conversion programs

- Just keep those out of the automated deployment.
  - It makes deployment over complicated and almost certainly incremental
- Create a special correction product to deploy and execute corrections
Feedback loop

- Have something monitor your production.
  - Log files (** / Error / invalid / unable / mismatch / SYSTEM-ERROR )
  - Database status / replication status
  - Configure it!
  - Set alerts
  - Try to get the errors really down to 0
- Ignored errors are worse than no DevOps tool at all
How to get there / road map
Tips and best practices

- Read Java properties in OpenEdge

```java
INPUT STREAM sIn FROM VALUE(FILE-INFO:FULL-PATHNAME).
REPEAT:
    IMPORT STREAM sIn UNFORMATTED cLine.
    cLine = TRIM(cLine).
    IF SUBSTRING(cLine,1,1) <> "#" AND
        SUBSTRING(cLine,1,1) <> "[" AND
        cLine MATCHES "*=*" THEN
    DO:
        CREATE ParameterObject.
        ASSIGN
            ParameterObject.Name = ENTRY (1,cLine,"=")
            ParameterObject.cValue = SUBSTRING (cLine, INDEX (cLine,"=") + 1)
            ParameterObject.oParameter = NEW PARAMETER (ParameterObject.Name,
                                                        ParameterObject.cValue).
    END.
END.
```
Tips and best practices

- Create an Ant library with makros
- Re-use those macros
- Keep your product build and deployment scripts as small as possible
- Do not put configuration in an Ant script, put it in the properties file
- Use “TargetProducts” csv in property file
  - Use contrib to do a for each over the products
- Execute all possible items in parallel
- Do not store passwords or environment data in the properties file
  - Inject those properties during deployment from an alternate source
Tips and best practices II

- Make sure all production like environments have the same architecture
  - Ports / file / folder locations
- All parts of the deployment pipeline must fully function stand alone
- If you really must, create a pipeline to deliver some .r’s as patch
  - Always clean up patches automatically on deployment(!)
- Use Windows service wrapper to run the Jenkins slave as a service
- Use crontabl to run the Jenkins slave on UNIX
  - Not happy with the java process? Only activate the service in time window
Tips and best practices III

- Tag / Name your release with a suffix
  - _unstable / _unstable_nightly
  - _Beta (Beta version -> SIT system)
  - _RC (Release Candidate -> UAT system)
  - _GA (General Available -> PROD system)

- Create nodes / slaves in Jenkins and bind build jobs to a specific node
- Use property expansion, never repeat a property value
- Parallelize the deployment jobs using jenkinsfile for pipeline
- Parallelize as much in a job as possible
Tips and best practices IV

- Everything that uses the same DB schema must be in one branch / stream
  - Don’t create a dependency hell when everything is based on the same crc
- Build deploy (combined) products
  - Database package + database tools
  - Framework Client + Installer + Application Client
  - Framework Appserver + Application Appserver + Appserver tools and scripts
Tips and best practices V

- Keep Ant references to OpenEdge tools platform independent
  
  ```xml
  <condition property="asbman" value="asbman.bat">
    <os family="windows" />
  </condition>
  <condition property="asbman" value="asbman">
    <os family="unix" />
  </condition>
  ```

- Tell Jenkins not to kill all child processes when the job is finished
  - BUILD_ID=dontKillMe
Questions
Dank u!

Go raibh maith agat
非常感谢！

Thank you!

Merci beaucoup!

Danke!

شكرا

Mutțumesc foarte mult

Большое спасибо!

τεşekkür ederim!

Qatlho!

Grazie mille!!

ขอบคุณมาก

Ευχαριστώ πολύ!

© 2018 Consultingwerk Software Services Ltd. All rights reserved.