Build Trust in Your
Build-to-Deployment Flow!
Fred Simon, JFrog
About me

✓ Fred Simon
✓ Chief Architect @JFrog
✓ Trying to tell machine to behave for the last 20 years
✓ From Consulting to DevOps in the last 13 years
Agenda

✓ The cloud silver bullet
✓ The right tool for the job
✓ Binaries all the way
✓ The black magic of releasing

LOTS OF LIVE DEMOS AHEAD!
The New Silver Bullet

EVERYTHING *aaS

www.jfrog.com
What’s good about *aaS?

*aaS features Continuous Delivery
Continuous Delivery FTW

✔ Advantages for the user:
  > Always on the latest version

✔ Advantages for the ISV:
  > Agile
  > Rapid feedback
  > Users are the best beta-testers
  > No long-term support

✔ Everybody Wins?
Almost, except DevOps

- Very frequent releases
- More than one version in production
- Complicated access levels
Almost, except DevOps

- Very frequent releases
- More than one version in production
- Complicated access levels

DevOps Borat @DEVOPS_BORAT
For job security in devops make of sure you advocate Continuous Delivery and implement by manual procedure of 45 step!
Almost, except DevOps

- Root cause analysis
  - Trace from binaries to sources
  - Reproduction abilities
- Promotions
  - Status changes
Almost, except DevOps

- **Root cause analysis**
  - Trace from binaries
  - Reproduction abilities

- **Promotions**
  - Status changes

In startup we welcome advocate of continuous delivery by put them on pager. Next they advocate quarterly release.

DevOps Borat @DEVOPS_BORAT
Sounds Familiar?

- Agile principles applied for DevOps
- We have good tooling for Agile development
  - Version Control
  - Unit Testing (and coverage)
  - Build Servers
  - Hot Swap tools
- What’s up with tooling for Agile DevOps?
Agile Tooling for DevOps Checklist

✓ Versioning
✓ Access control
✓ Traceability
✓ Promotions
✓ Tags and annotations
✓ Search
How Do I Know?

✓ JFrog SaaS offering
  > Artifactory Online
  > Gradle, Grails, SpringSource, Typesafe, Jenkins, etc.

✓ We build, release and eat our own dog food
  > Continuously
The Right Tool for the Job

HERE COMES BINARY REPOSITORY

www.jfrog.com
Here Comes Binary Repository

✓ E.g. Artifactory
✓ Main feature - Smart Storage
✓ Much more than passive storage
✓ Critical for CI and ALM
Tooling Chain

**Tools and Logos**
- *** git
- **Subversion**
- *** Perforce
- **Maven**
- **Ivy**
- **NuGet**
- **Yum**
- **TC**
Artifactory in DevOps Ecosystem
Meet Artifactory

DEMO TIME!

www.jfrog.com
 Binaries All the Way

- From some point product in your lifecycle, all you care about is binaries
- Lots of things to do after the software is built
The Release Pipeline

Source: Agile ALM, Michael Hüttermann, Manning Publications Co.
Passing the software to QA

- Different access rights
- Different physical location
- Ability to annotate
Staging and Preproduction

- Replication of Production environment
  > Lock versions of dependencies and artifacts
- Allow access to set of users
Going to Production

- Convert staging binaries to production
- Allow public access
- Change settings
- Tag
Traceability

✓ Binaries should be traceable in every stage
  > Sources
  > Dependencies
  > Environment details
  > Tags

✓ Where’s the information?
  > Version Control System
  > Build Server
WHAT MY FRIENDS THINK I DO

DevOps

www.jfrog.com
What Others Think I Do
What I Think I Do
What I Really Do
What I should Do

LAUNCH
Target: Automation

✓ It’s impossible to release frequently with manual procedures
  > While maintaining quality
✓ Use your binaries storage to release
Put your repository to work

THE MAGIC OF RELEASE

www.jfrog.com
Release With Release Candidate

✓ Your next build is release-candidate
✓ Once successfully built and tested, click the button
  > Automatic versions switch
    › From integration to release
  > Right place to put your binaries
    › Move from Staging to Public
  > Automatic VCS tagging for the release
Release With Release Candidate

✔ Process:

1. Produce and build snapshots until satisfied
2. Once satisfied, build release candidate
3. Stage RC, check and verify
4. Once checked, release
Release With Artifactory: Mechanics

✔️ : The Artifactory Jenkins Plugin
  > Gathers build information
  > Uploads artifacts in bulk
  > Uploads build information
  > Provides bi-directional linking

✔️ Release Management
  > Changes versions in build script
  > Allows to Choose repository to deploy to
  > Creates a VCS tag/branch
Release With Release Candidate

DEMO TIME!

www.jfrog.com
OOTB Release Management

✓ Pros

> Out of the box
> Supports the “by the book” release cycle
> Supports majority of the tools

✓ Cons

> Not flexible
> Not extensible
> May not suite your case
We Know: We Don’t Know Better

✓ YMMV (great deal)  ✓ Write your own release logic
✓ Pre and post build deploy hooks
Flexible Release

✅ Code your release strategy
  > Versioning scheme
  > VCS (tagging, branching, commit comments)
  > Promotion hook (copy/move, comments, status)

✅ Available by REST
Controlling Versioning Scheme

✓ Classic versioning scheme:
  > Release version
    › 2.0.3
  > Integration version
    › 2.0.4-SNAPSHOT

✓ YMMV
  > Write your own strategy for versioning
  > Dynamic Groovy code
Example: Promotion of Snapshots

✔ Sometimes the build takes long time...

✔ But that’s the silly reason
Release With Release Candidate

✓ Process:

1. Produce and build snapshots until satisfied
2. Once satisfied, build release candidate
3. Stage RC, check and verify
4. Once checked, release
Process:
1. Produce and build snapshots until satisfied
2. Once satisfied, build release candidate
3. Stage RC, check and verify
4. Once checked, release
Release With Release Candidate

✓ Lots of things can go wrong during one more build
✓ If we won’t build it, we won’t screw it
✓ Process:
  1. Produce and build snapshots until satisfied
  2. When satisfied, check and verify
  3. Once checked, release
Example: Promotion of Snapshots

✓ Choose existing build to become a release
✓ Using REST API without build server
✓ Invoke Promotion plugin
  > Convert to next version
  > Tag, branch, etc.
  > Promote (copy/move)
Plugin What?

CODE TIME!

www.jfrog.com
Plugin Code

✓ Groovy goodness
✓ Executed directly in Artifactory
✓ Uses Public API
  > Search for artifacts
  > Search for builds
  > Copy/Move artifacts
  > Manipulate files
    › E.g. change versions in descriptors
Manipulating Version Control Systems

```java
vcsConfig = new VcsConfig()
vcsConfig.useReleaseBranch = false
vcsConfig.createTag = true
vcsConfig.setTagUrlOrName = "gradle-multi-example-\${releaseVersion}"
vcsConfig.setTagComment = "[gradle-multi-example] Release version \${releaseVersion}"
vcsConfig.setNextDevelopmentVersionComment = "[gradle-multi-example] Next development version"
```
Plugin Code

// Iterate over modules list
modules.each {item ->
    // Find project inner module dependencies
    def match = []
    def dependenciesList = item.getDependencies()
    dependenciesList.each {dep ->
        def res = stageArtifactsList.asList().find {sal -> sal.g
            if (res != null) match << res
    ```

✓ Manipulating BuildInfo object
Plugin Code

```python
artifactsList = item.getArtifacts()
artifactsList.eachWithIndex { (art, index) ->
    def stageRepoPath = getStageRepoPath(art, stageArtifactsList)
    def releaseRepoPath = null
    if (stageRepoPath != null) {
        releaseRepoPath = getReleaseRepoPath(targetRepository, stageRepoPath, stageVersion)
    } else {
        missingArtifacts << art
        return
    }

    def releasedArtifact = null
    // Return type of status is different coming from deploy and copy. I know it is ugly
    def status = null
    // If ivy.xml or pom then create and deploy a new Artifact with the fix revision, status
    if (art.getType() == 'ivy') {
        status = generateAndDeployReleaseIvyFile(stageRepoPath, releaseRepoPath, match)
        if (status.isError()) rollback(releaseArtifactsSet, status.getException())
    } else if (art.getType() == 'pom') {
        status = generateAndDeployReleasePomFile(stageRepoPath, releaseRepoPath, match)
        if (status.isError()) rollback(releaseArtifactsSet, status.getException())
    } else {
        status = repositories.copy(stageRepoPath, releaseRepoPath)
    }
```
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=/14|
targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14|
targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14&
targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14|
targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14|targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14|targetRepository=gradle-release-local
Calling REST API With CURL

http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?
params=snapExp=d14|targetRepository=gradle-release-local

Artifactory Server
Plugins API
Plugin Name
Build Name and Number
Param: Versioning Scheme
Target repository for release
Recap: Promotion of Snapshots

- Choose existing build to become a release
- Using REST API without build server
- Invoke Promotion plugin
  - Convert to next version
  - Tag, branch, etc.
  - Promote (copy/move)
DEMO TIME!

Release by Snapshot Promotion

www.jfrog.com
4 Commandments of DevOps

- Automate everything
- Version everything
- Trace everything
- Report/Log/Feed back everything
4 Commandments of DevOps

- Automate everything
- Version everything
- Trace everything
- Report/Log/Feed everything
Thank You!