Build Trust in Your Build-to-Deployment Flow

Yoav Landman, JFrog
About me

✓ Yoav Landman
✓ Creator of the Artifactory Binary Repo
✓ CTO at JFrog
✓ @yoavlandman
Agenda

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

EVERYTHING *aaS

www.jfrog.com
Why We Need *aaS?

✓ *aaS features Continuous Delivery
Continuous Delivery FTW

✓ User advantages:
  > Latest version/features
  > No upgrades/maintenance

✓ Developer advantages:
  > 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 procedures
Almost, except DevOps

✓ Root cause analysis
  > Tracing from binaries to source
✓ Version tracking
✓ Not everyone is ready for CD
Almost, except DevOps

✓ Root cause analysis
  > Tracing from binaries to source
✓ Version tracking
✓ Not everyone is ready

DevOps Borat  @DEVOPS_BORAT

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

✓ Agile principles applied for DevOps
✓ We have good tooling for Agile development
  > Version control
  > Unit testing and code coverage
  > CI servers
  > Hot swap tools
✓ What’s up with tooling for agile DevOps?
Agile tools for DevOps - checklist

✓ Versioning
✓ Access control
✓ Traceability
✓ Promotion
✓ Tags and annotations
✓ Search
Feeling the pain

✓ JFrog SaaS offering
  > artifactoryonline.com
  › SpringSource, Grails, Jenkins plugins, etc.

✓ We build and release continuously
The Right Tool for the Job

HERE COMES BINARY REPOSITORY

www.jfrog.com
Here Comes Binary Repository

✓ E.g. Artifactory
✓ Proxy
✓ Smart storage
  > Much more than a passive space
✓ Critical for CI/CD and ALM
Artifactory in DevOps Ecosystem
Meet Artifactory

DEMO TIME!

www.jfrog.com
Binaries all the way

✓ From some point in the release 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.
Traceability

✓ Binaries should be traceable at every stage
  > Sources
  > Dependencies
  > Environment details
  > Tags
✓ Where’s the information?
  > Version control system
  > Build server
Traceability w. the Artifactory plugin

✓ Gathers build information
✓ Uploads artifacts in a bulk
✓ Uploads build information
✓ Maintains bi-directional links
Tracing Artifacts

DEMO TIME!

www.jfrog.com
THE MAGIC OF RELEASING

Put your repository to work

THE MAGIC OF RELEASING

www.jfrog.com
Release candidates

✓ Your next build is a release-candidate
✓ Once successfully built and tested, click a button
  › Automatic versions switch
    › From integration to release
  › Right place to put your binaries
    › Move from Staging to Public
  › Automatic VCS tagging
Releasing with release candidates

✓ Process:

1. Produce and build snapshots until satisfied
2. Once satisfied, build a release candidate
3. Stage RC, check and verify
4. Once verified, release
Releasing w. the Artifactory plugin

✓ Changes versions in build script
✓ Allows choosing a target deploy repository
✓ Creates a VCS tag/branch
Releasing with Release Candidates

DEMO TIME!
Staging based release management

✅ Pros

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

✅ Cons

> Limited extensibility
> May not fit your requirements
Releasing with release candidates

✓ Process:

1. Produce and build snapshots until satisfied
2. Once satisfied, build a release candidate
3. Stage RC, check and verify
4. Once checked, release
Releasing with release candidates

✓ Process:

1. Produce and build snapshots until satisfied
2. **Once satisfied, build release candidate**
3. Stage RC, check and verify
4. Once checked, release
Releasing with release candidates

✓ 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
Target: automation

✓ It’s impossible to release frequently with manual procedures
  > While maintaining quality
✓ Use your binaries storage to release
A more flexible release

✓ Code your release strategy
  > Versioning scheme
  > VCS (tagging, branching, commit comments)
  > Target repo
  > Promotion hooks (copy/move, comments, status)

✓ Automated with REST
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 PAPI
  - Search for artifacts
  - Search for builds
  - Copy/move artifacts
  - Manipulate files
    - E.g. change versions in descriptors
Plugin Code

```java
vcsConfig = new VcsConfig()
vcsConfig.useReleaseBranch = false
vcsConfig.createTag = true
vcsConfig.tagUrlOrName = "gradle-multi-example-${releaseVersion}"
vcsConfig.tagComment = "[gradle-multi-example] Release version ${releaseVersion}"
vcsConfig.nextDevelopmentVersionComment = "[gradle-multi-example] Next development version"
```

✓ Manipulating version control
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.
        if (res != null) match << res
    }

✓ Manipulating the BuildInfo object
Plugin Code

```java
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

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

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
Release by Snapshot Promotion

DEMO TIME!

www.jfrog.com
Strive for minimal input

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

1. Automate everything
2. Version everything
3. Trace everything
4. Report/Log/Feedback everything
Thank You!