Trusting Your Build-to-Deployment Flow with JenkinsCI

Yoav Landman
JFrog
www.jfrog.com

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 Jenkins Artifactory plugin
- The black art of release management

The New Silver Bullet

EVERYTHING *aaS

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

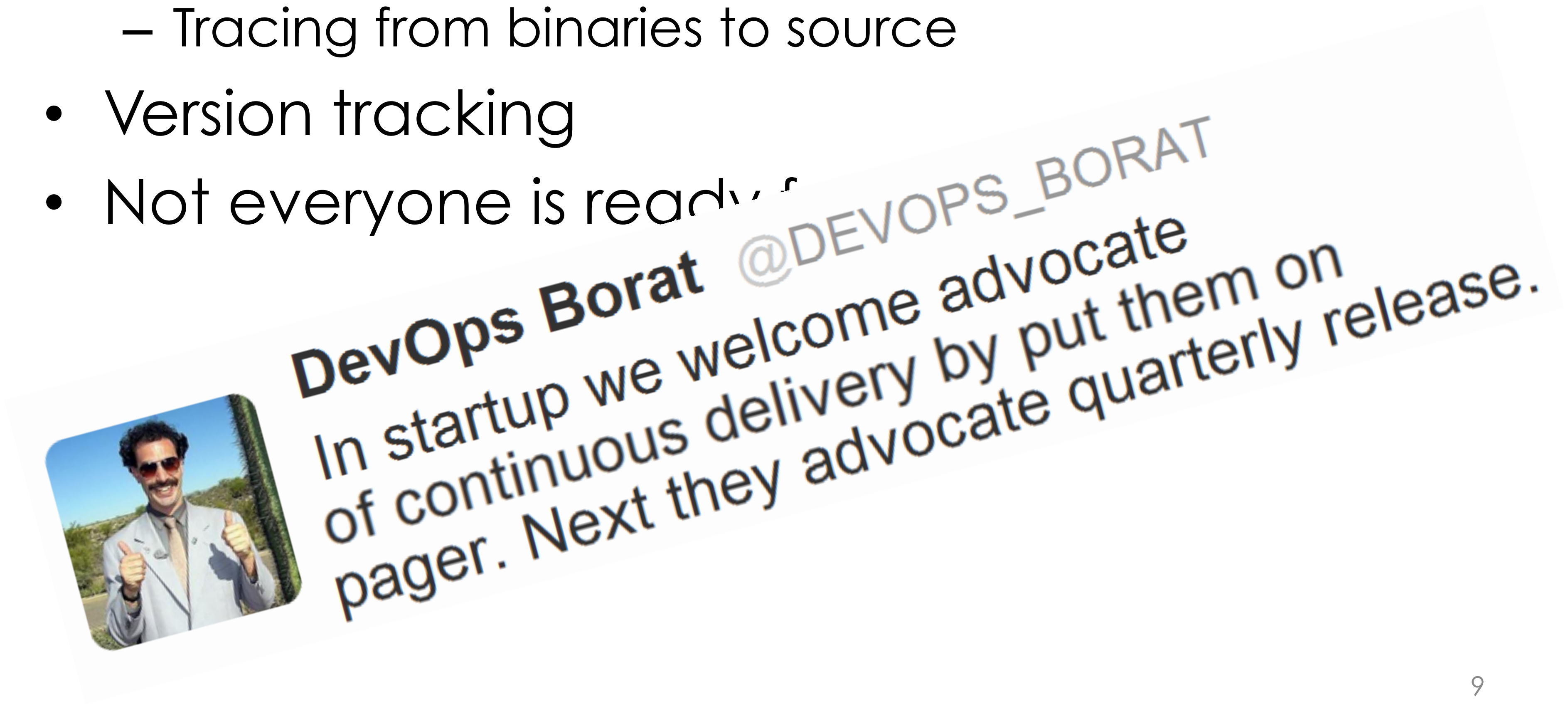
- Very frequent releases
- More than one version in production
- Complicated procedures

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 f



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

Agile developer tools

- 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

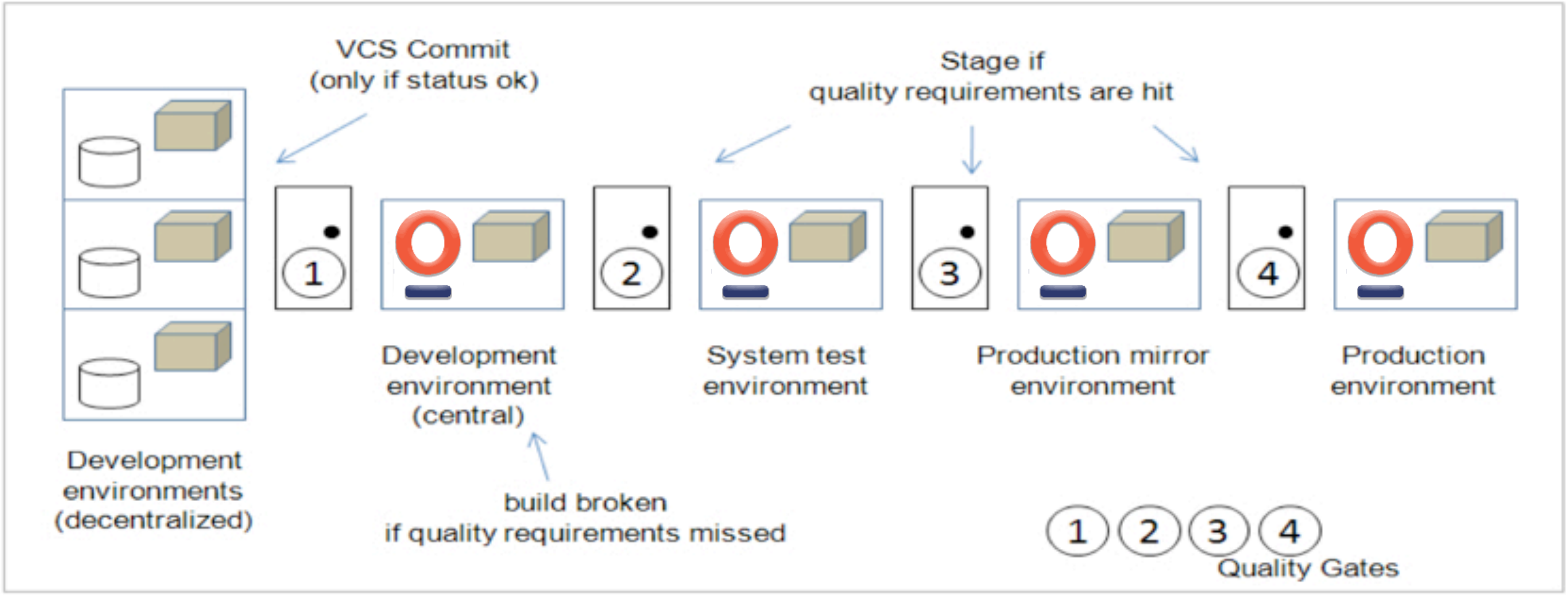
Feeling the pain

- JFrog SaaS offering
  - artifactoryonline.com
    - SpringSource, Grails, Jenkins plugins, etc.
- We build and release continuously

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


VCS Commit
(only if status ok)
Stage if
quality requirements are hit
1
2
3
4
Development environments (decentralized)
Development environment (central)
System test environment
Production mirror environment
Production environment
build broken
if quality requirements missed
1 2 3 4
Quality Gates

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
  - Issue tracker

The Right Tool for the Job

HERE COMES BINARY REPOSITORY

Here comes binary repository

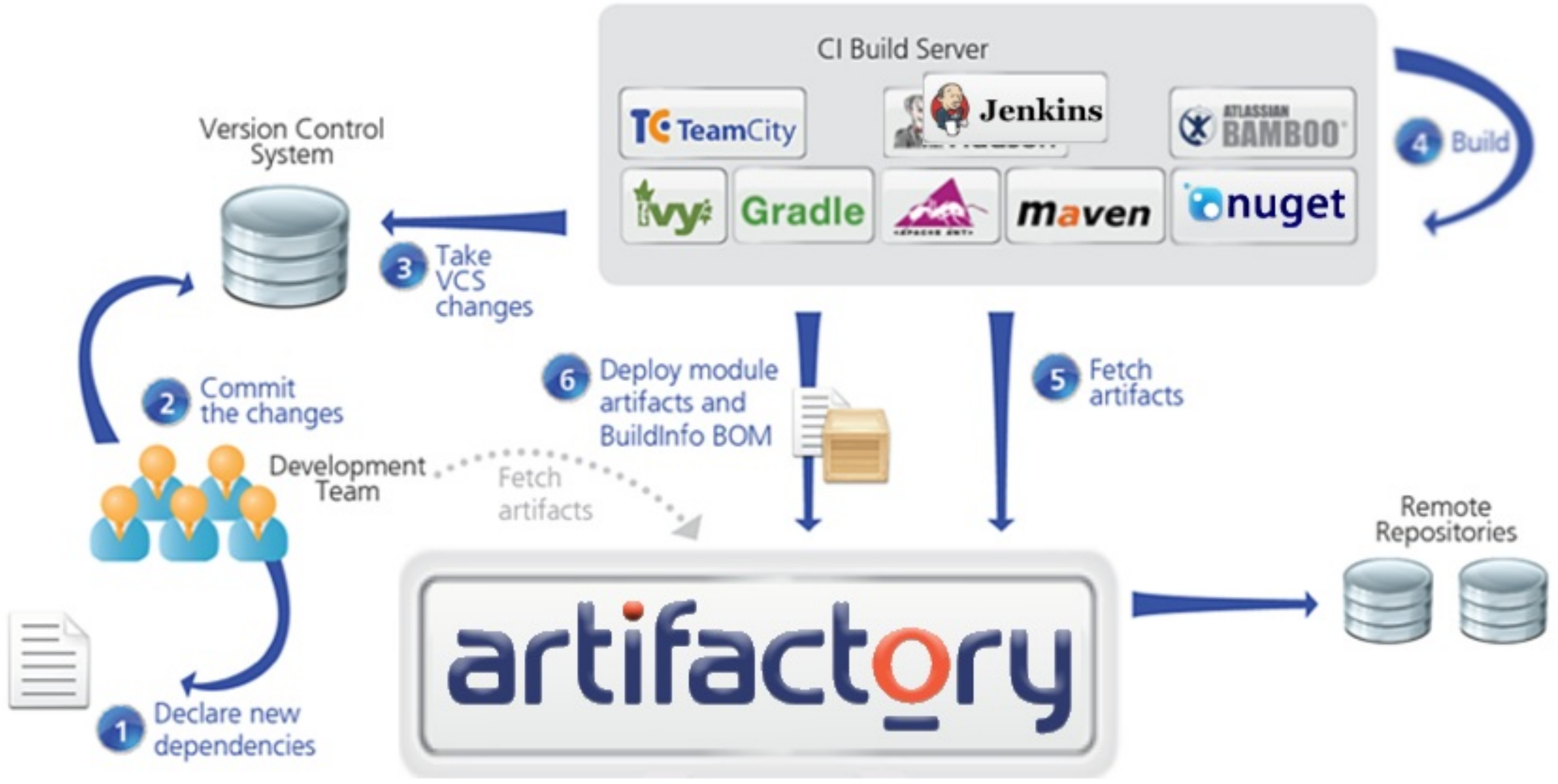
- E.g. Artifactory
- Proxy
- Smart storage
  - Much more than a passive space
- Critical for CI/CD and ALM

Talks to the standard tool stack

git
SUBVERSION
PERFORCE

ivy
maven
yum
yellowdog updater modified
nuget

Artifactory in DevOps Ecosystem

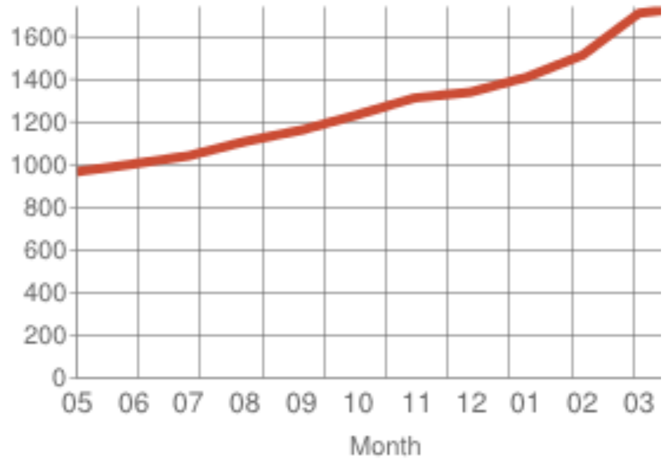


Meet Artifactory

DEMO TIME!

The Jenkins Artifactory plugin

- 1st Release – 14 Dec 2009
- Installations ~1,800, May 2012
- Integrates with other plugins:
  - Maven
  - Gradle
  - Ivy
  - Subversion
  - Git
  - Perforce
  - JIRA



Traceability w. the Artifactory plugin

- Gathers build information
  - And build-related
- Uploads artifacts in a bulk
- Uploads build information
- Maintains bi-directional links
- Powerful staging and promotion

Tracing Artifacts

DEMO TIME!

Put your repository to work

THE ART OF RELEASE MANAGEMENT

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

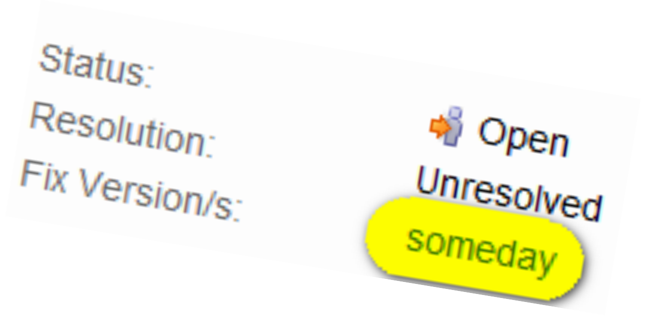
DEMO TIME!

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

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



Status: Open
Resolution: Unresolved
Fix Version/s: someday

Example: using the latest build

```
BuildRun latestReleaseOrLatestBuild(
                                      List<BuildRun> buildRuns) {
     BuildRun[] allReleasedBuilds =
            buildRuns.findAll { buildRun ->
                    (buildRun.releaseStatus == 'released') }
     if (allReleasedBuilds) {
          buildRuns = allReleasedBuilds
     }
     buildRuns.max {buildRun -> buildRun.startedDate }
}
```

Your own release strategy

DEMO TIME!

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

Can you spot the problem?

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

Redundant build

Releasing with release candidates

- Lots of things can change 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: snapshot promotion

- Choose existing build to become a release
- Using REST - no UI
- Invoke promotion plugin
  - Convert to next version
  - Tag, branch, etc.
  - Promote (copy/move)

Plugin What?

CODE TIME!

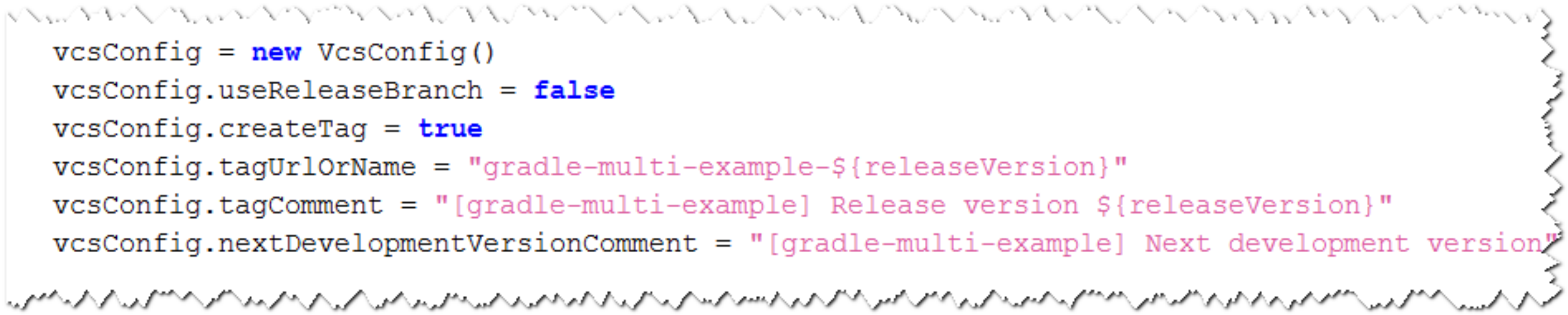
Plugin Code

- Groovy goodness
- Executed directly in Artifactory
- Uses PAPI
  - Searches
  - Artifacts
    - E.g. change versions in descriptors
  - Builds
  - REST execution extensions
  - Jobs

https://github.com/JFrogDev/artifactory-user-plugins

Plugin Code

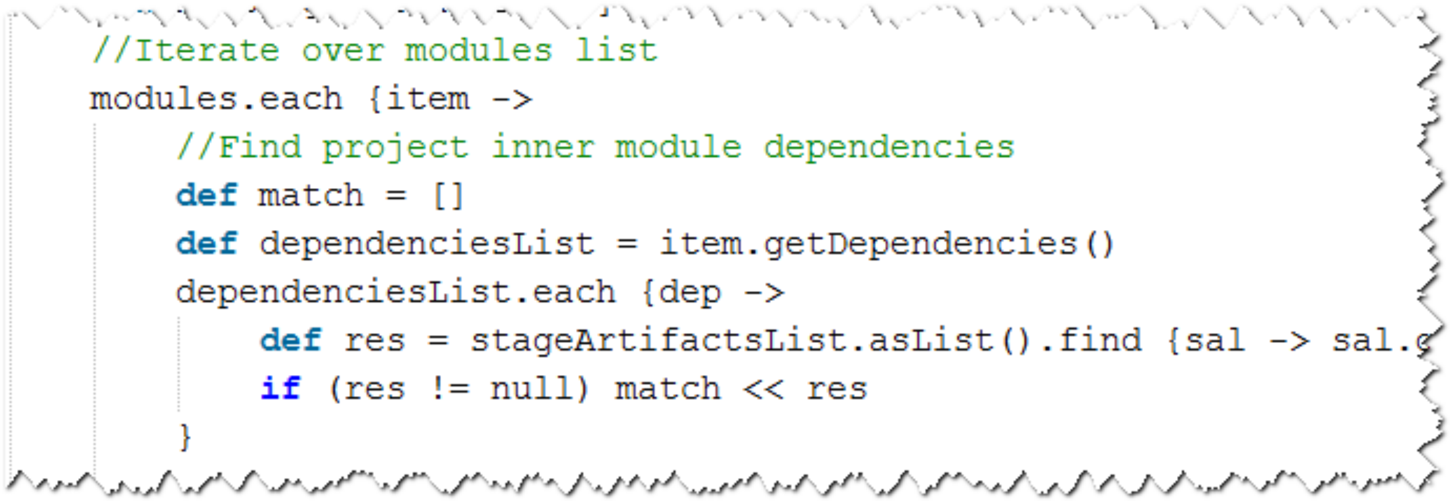
- Manipulating version control



```
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"
```

Plugin Code

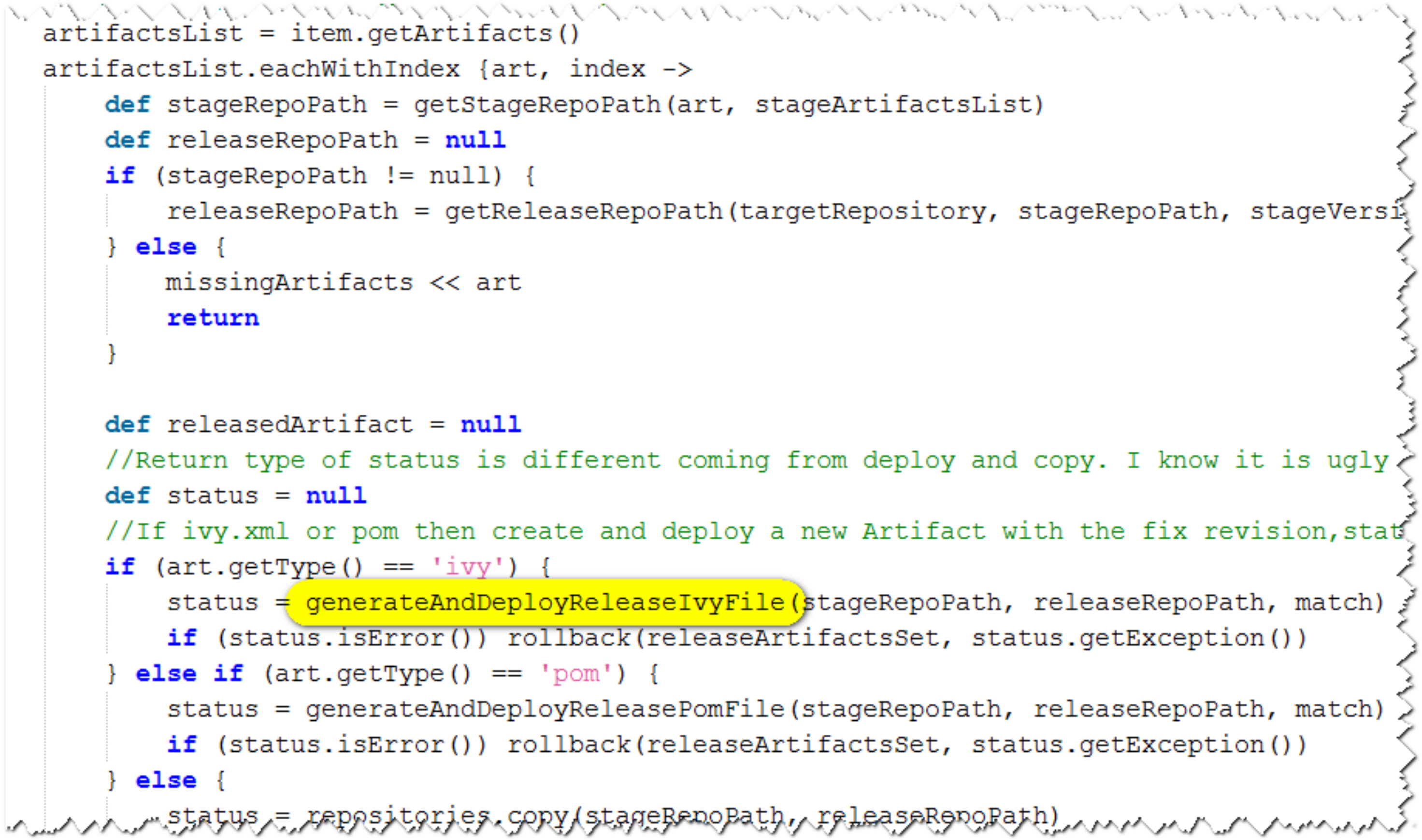
- Manipulating the BuildInfo object



```
//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
    }
```

Plugin Code

- Creating and replacing artifacts



```
artifactsList = item.getArtifacts()
artifactsList.eachWithIndex {art, index ->
    def stageRepoPath = getStageRepoPath(art, stageArtifactsList)
    def releaseRepoPath = null
    if (stageRepoPath != null) {
        releaseRepoPath = getReleaseRepoPath(targetRepository, stageRepoPath, stageVersi
    } 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,stat
    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)
```

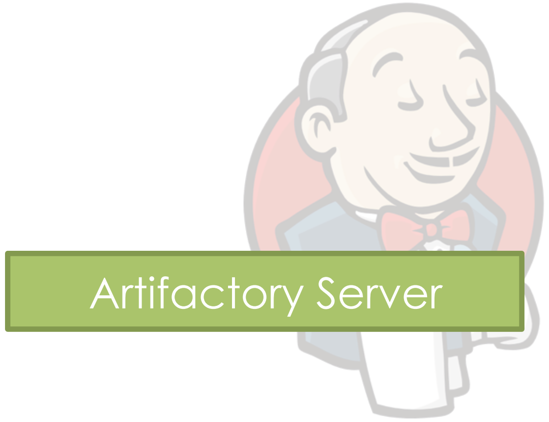
Calling REST API With CURL

```
http://repo-demo:8080/artifactory/api/plugins/build/promote/snapshotToRelease/gradle-multi-example/1?params=snapExp=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
```

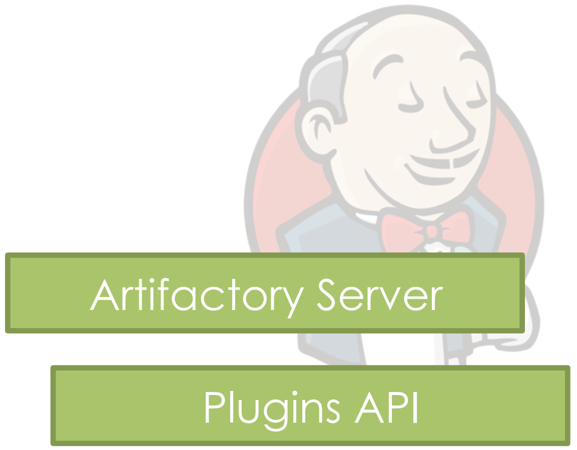
Calling REST API With CURL



```
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/snapshotToRelea
gradle-multi-example/1
?params=snapExp=d14|
targetRepository=gradle-release-local
```

Artifactory Server

Plugins API

Plugin Name

Calling REST API With CURL

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

Artifactory Server

Plugins API

Plugin Name

Build Name and Number

Calling REST API With CURL

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

Artifactory Server

Plugins API

Plugin Name

Build Name and Number

Param: Versioning Scheme

Calling REST API With CURL

```
http://repo-demo:8080/
artifactory/api/plugins/
build/promote/snapshotToRelea
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

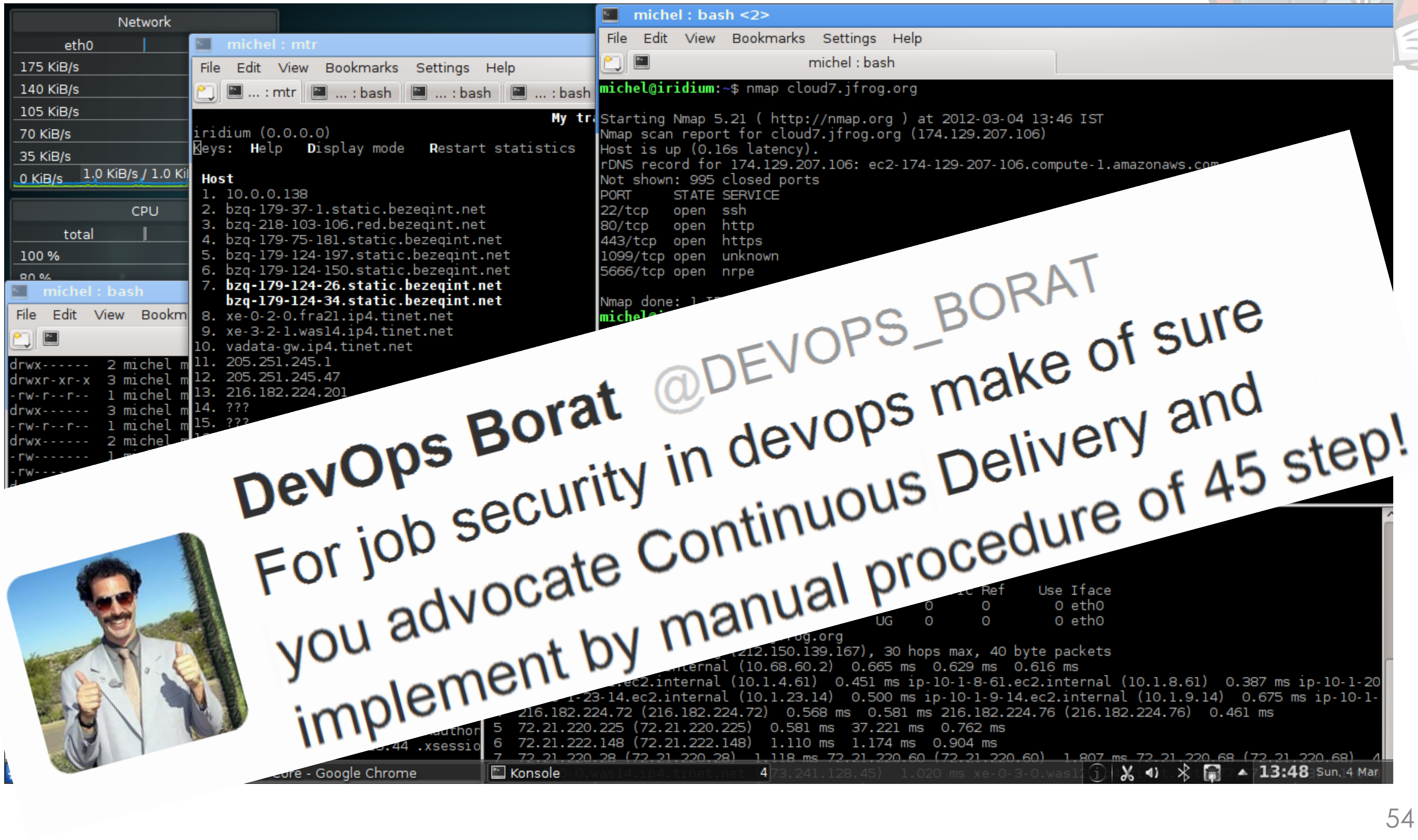
Recap: Promotion of Snapshots

- Choose existing build to become a release
- Using the REST API without building
- Invoking the promotion plugin
  - Convert to next version
  - Tag, branch, etc.
  - Promote (copy/move)

Release by Snapshot Promotion

DEMO TIME!

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!

```
michel@iridium:~$ nmap cloud7.jfrog.org

Starting Nmap 5.21 ( http://nmap.org ) at 2012-03-04 13:46 IST
Nmap scan report for cloud7.jfrog.org (174.129.207.106)
Host is up (0.16s latency).
rDNS record for 174.129.207.106: ec2-174-129-207-106.compute-1.amazonaws.com
Not shown: 995 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
1099/tcp open  unknown
5666/tcp open  nrpe

Nmap done: 1 I
michel
```

```
                                                    My tra
iridium (0.0.0.0)
Keys:  Help   Display mode   Restart statistics

 Host
 1. 10.0.0.138
 2. bzq-179-37-1.static.bezeqint.net
 3. bzq-218-103-106.red.bezeqint.net
 4. bzq-179-75-181.static.bezeqint.net
 5. bzq-179-124-197.static.bezeqint.net
 6. bzq-179-124-150.static.bezeqint.net
 7. bzq-179-124-26.static.bezeqint.net
    bzq-179-124-34.static.bezeqint.net
 8. xe-0-2-0.fra21.ip4.tinet.net
 9. xe-3-2-1.was14.ip4.tinet.net
10. vadata-gw.ip4.tinet.net
11. 205.251.245.1
12. 205.251.245.47
13. 216.182.224.201
14. ???
15. ???
```

```
drwx------  2 michel m
drwxr-xr-x  3 michel m
-rw-r--r--  1 michel m
drwx------  3 michel m
-rw-r--r--  1 michel m
drwx------  2 michel m
```

```
                                           Ref     Use Iface
                                  0        0         0 eth0
                         UG       0        0         0 eth0
              og.org
  (212.150.139.167), 30 hops max, 40 byte packets
  ternal (10.68.60.2)  0.665 ms  0.629 ms  0.616 ms
  .ec2.internal (10.1.4.61)  0.451 ms ip-10-1-8-61.ec2.internal (10.1.8.61)  0.387 ms ip-10-1-20
  -1-23-14.ec2.internal (10.1.23.14)  0.500 ms ip-10-1-9-14.ec2.internal (10.1.9.14)  0.675 ms ip-10-1-
  216.182.224.72 (216.182.224.72)  0.568 ms  0.581 ms 216.182.224.76 (216.182.224.76)  0.461 ms
5  72.21.220.225 (72.21.220.225)  0.581 ms  37.221 ms  0.762 ms
6  72.21.222.148 (72.21.222.148)  1.110 ms  1.174 ms  0.904 ms
```

Achieving the 4 DevOps "commandments"

1. Automate everything
2. Version everything
3. Trace everything
4. Report/Log everything

Designed by Jessica Allen on Dribbble.com

THANK YOU

Thank You To Our Sponsors

Platinum Sponsor	CloudBees
Gold Sponsors	JFrog CLOUDANT
Silver Sponsors	LIFERAY SendGrid Email Delivery. Simplified. CLOUDSMITH
Bronze Sponsors	WAN DISCO