From messy XML to wonderful YAML and pretty Job DSL

An in-Jenkins migration story

Piotr Bielak
piotr.bielak@corp.ovh.com

Szymon Datko
szymon.datko@corp.ovh.com

13th November 2018

About us

Piotr Bielak

Passionate Python dev

Clean code enthusiast

128% IT nerd

Szymon Datko

DevOps & local Bash wizard

Open Source software lover

`makepkg`, not war

What is this mysterious Jenkins thing?

- One of the most popular automation servers.
- Powerful, Open Source, written in Java.
- Easy to start, configure, manage and use.
- Heavily extensible - plenty of plugins available.
- Widely used by the top IT companies!

redhat NOKIA Microsoft OVH rackspace

DELL facebook NASA NETFLIX cloudbees SPACEX

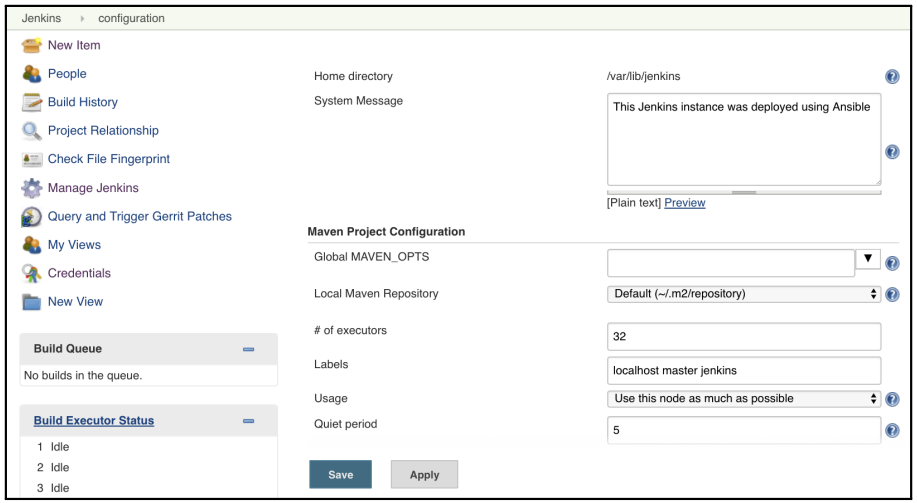
LinkedIn LINEAGE SONY openstack ECLIPSE FOUNDATION

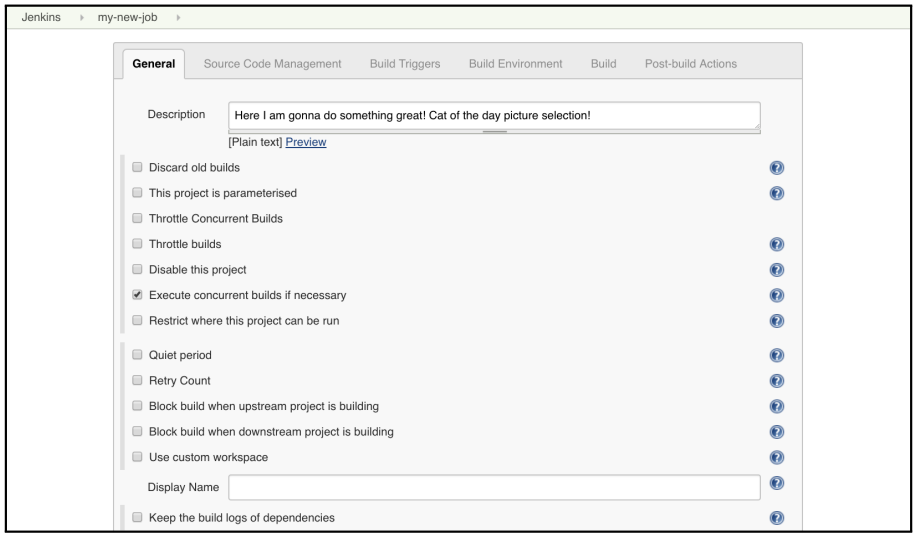
... and many, many more!

Sources: https://wiki.jenkins.io/pages/viewpage.action?pageId=58001258, https://stackshare.io/jenkins.

The traditional way for Jenkins configuration (1)

Click, click, click, ...





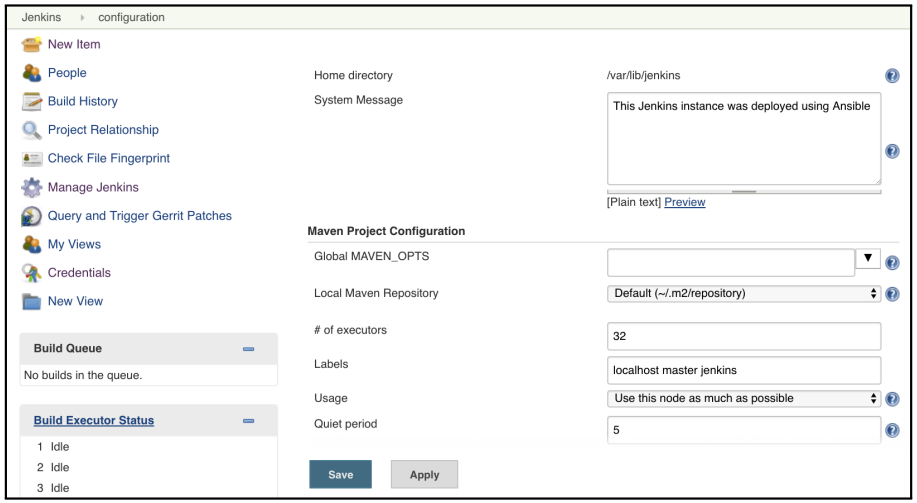
Typical steps:

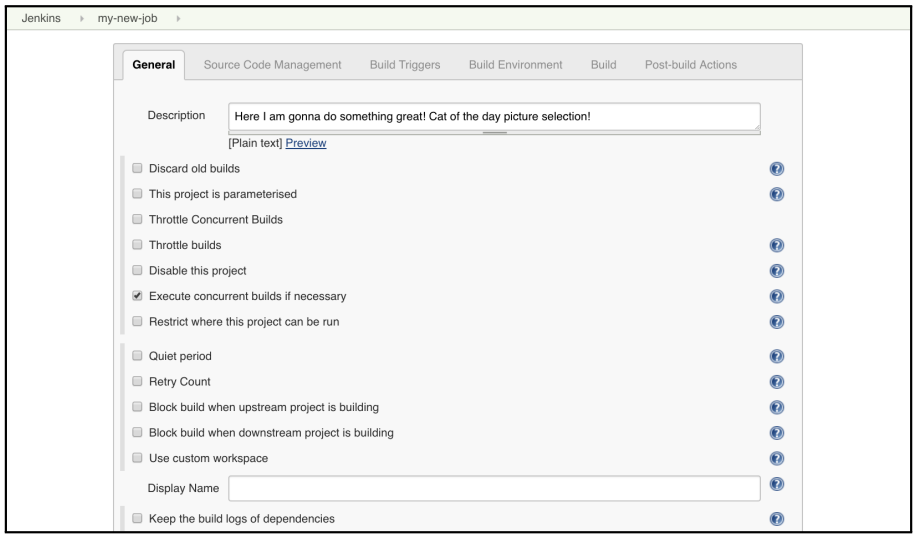
1. Start Jenkins server.
2. Open web browser.
3. First-launch setup.
4. Sign in as admin.
5. Configure the rest:
   a. adjust global settings,
   b. add some jobs.

OVH

The traditional way for Jenkins configuration (2)

Click, click, click, ... ⇒ Generated XML file






```
<?xml version='1.1' encoding='UTF-8'?>
<flow-definition plugin="workflow-job@2.21">
  <actions>
    <org.jenkinsci.plugins.pipeline.modeldefinition.act
    <org.jenkinsci.plugins.pipeline.modeldefinition.act
      <jobProperties/><triggers/><parameters/>
    </org.jenkinsci.plugins.pipeline.modeldefinition.ac
  </actions>
  <description>Some example job.</description>
  <properties>
    <org.jenkinsci.plugins.workflow.job.properties.Pipe
      <triggers>
        <hudson.triggers.TimerTrigger>
          <spec>H * * * *</spec>
        </hudson.triggers.TimerTrigger>
      </triggers>
    </org.jenkinsci.plugins.workflow.job.properties.Pip
  </properties>
  <definition class="org.jenkinsci.plugins.workflow.cps
    <scm class="hudson.plugins.git.GitSCM" plugin="git@
      <configVersion>2</configVersion>
      <userRemoteConfigs>
        <hudson.plugins.git.UserRemoteConfig>
          <name>origin</name>
          <url>ssh://user@my.repo.url.ovh/project-name<
          <credentialsId>repo-access-key</credentialsId
        </hudson.plugins.git.UserRemoteConfig>
      </userRemoteConfigs>
  (...)
```

The traditional way for Jenkins configuration (3)

Issue - XML is not human-friendly:

- hardly readable,
- a lot of redundancy,
- non-necessary things inside,
- bad formatting.

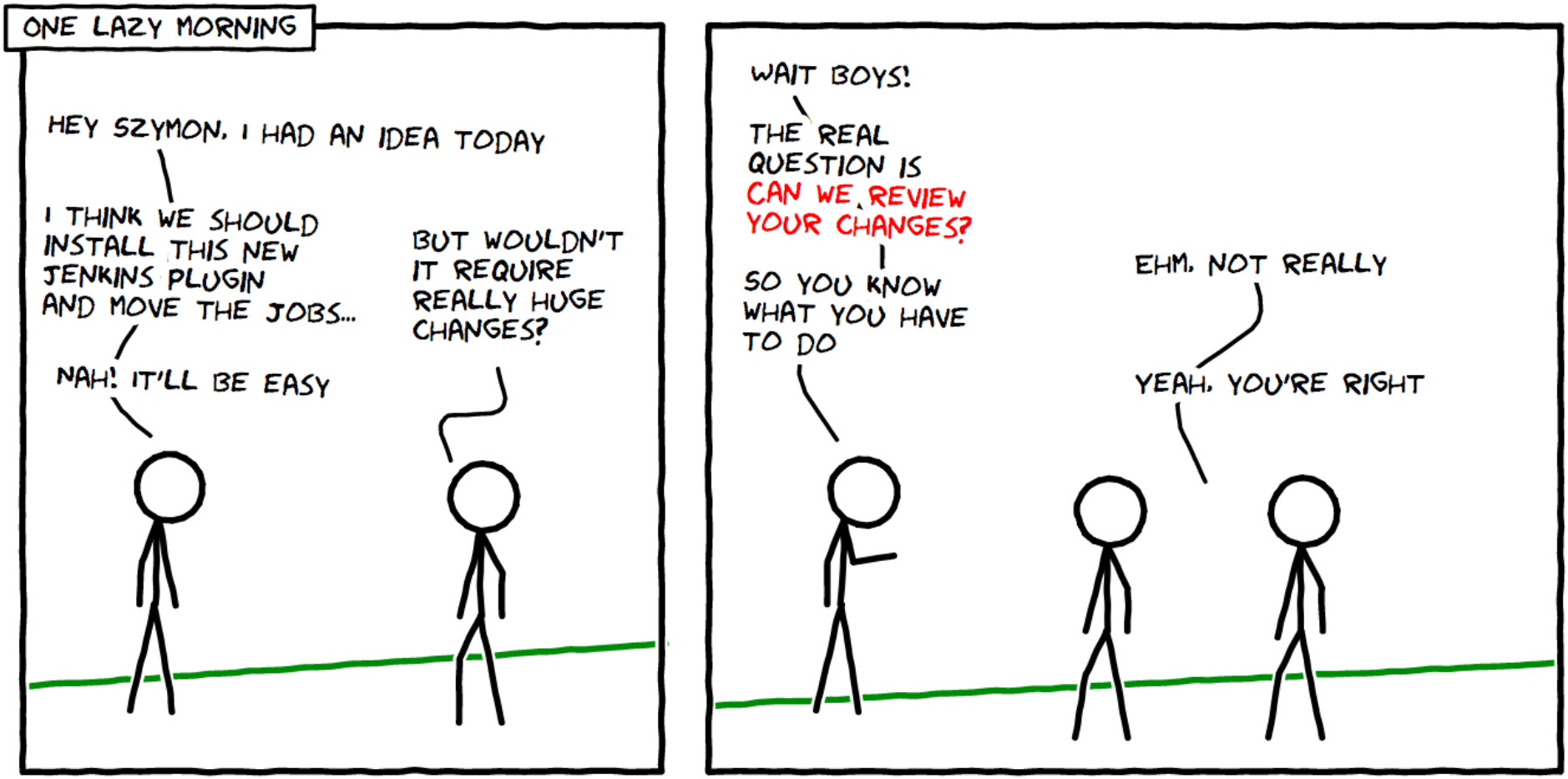
Therefore:

- versioning is difficult,
- review is inconvenient.

Generated XML file

```
<?xml version='1.1' encoding='UTF-8'?>
<flow-definition plugin="workflow-job@2.21">
  <actions>
    <org.jenkinsci.plugins.pipeline.modeldefinition.act
    <org.jenkinsci.plugins.pipeline.modeldefinition.act
      <jobProperties/><triggers/><parameters/>
    </org.jenkinsci.plugins.pipeline.modeldefinition.ac
  </actions>
  <description>Some example job.</description>
  <properties>
    <org.jenkinsci.plugins.workflow.job.properties.Pipe
      <triggers>
        <hudson.triggers.TimerTrigger>
          <spec>H * * * *</spec>
        </hudson.triggers.TimerTrigger>
      </triggers>
    </org.jenkinsci.plugins.workflow.job.properties.Pip
  </properties>
  <definition class="org.jenkinsci.plugins.workflow.cps
    <scm class="hudson.plugins.git.GitSCM" plugin="git@
      <configVersion>2</configVersion>
      <userRemoteConfigs>
        <hudson.plugins.git.UserRemoteConfig>
          <name>origin</name>
          <url>ssh://user@my.repo.url.ovh/project-name<
          <credentialsId>repo-access-key</credentialsId
        </hudson.plugins.git.UserRemoteConfig>
      </userRemoteConfigs>
  (...)
```

Our goal (1)

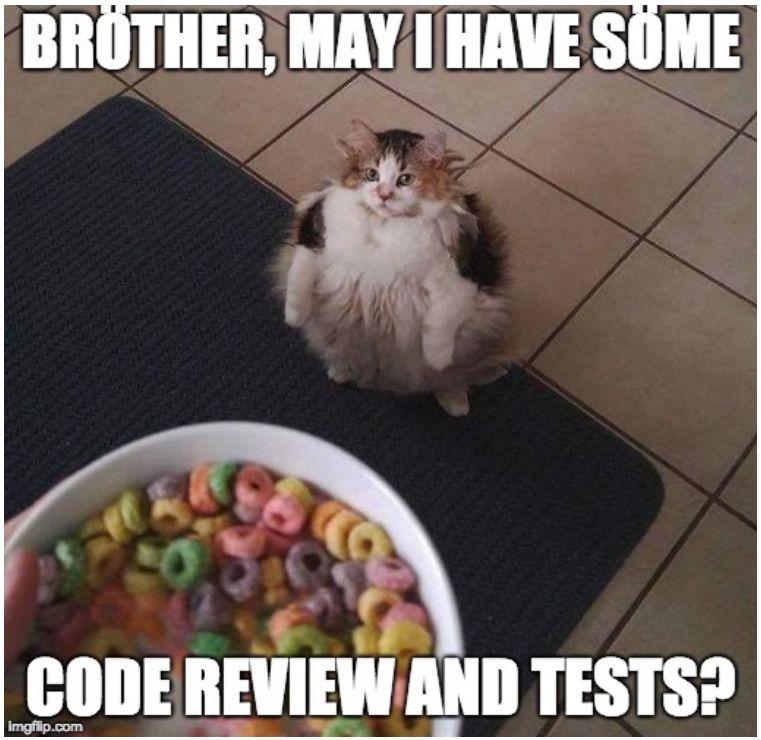


How it actually started - Piotr's perspective

Our goal (2)

Main desires:

- Not everyone has to be Jenkins pro-user to improve its configuration.
- We would like to have regular review of configuration changes.



How it actually started - Szymon's perspective

OVH

The mighty solution for (nearly) all your problems

There are three plug-ins that do come in handy...

Configuration as Code

Job DSL

Job Pipelines

(Jenkinsfiles)

```
pipeline {
  agent docker:'maven:3.3.3'
  stages {
    stage('build') {
      steps {
        sh 'mvn --version'
        sh 'mvn install'
      }
    }
  }
}
```

OVH

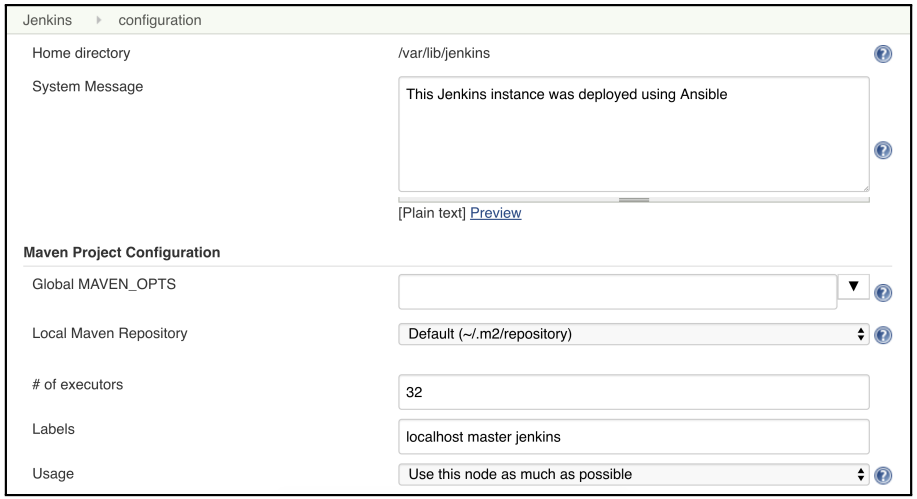
Configuration as Code (1)

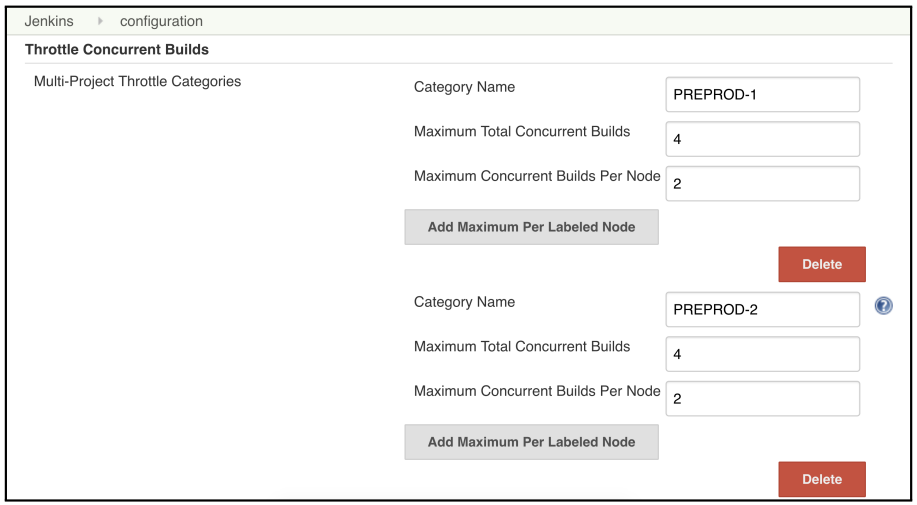
- **JCasC** – *Jenkins Configuration as Code*.
- Relatively young project, but fast growing:
  - initial release (0.1-alpha) on 23.02.2018,
  - stable version 1.0 released on 12.09.2018.
- Responsible for Jenkins general configuration.
- No more UI clicking needed to adjust things!
- Utilizes nice and tidy YAML files.

OVH

Configuration as Code (2)

UI view





YAML file

```yaml
jenkins:
  systemMessage: "This Jenkins instance was (...)"
  numExecutors: 32
  labelString: "localhost master jenkins"
  (...)
```

```yaml
unclassified:
  throttlejobproperty:
    categories:
      - categoryName: "PREPROD-1"
        maxConcurrentPerNode: 4
        maxConcurrentTotal: 2
      - categoryName: "PREPROD-2"
        maxConcurrentPerNode: 4
        maxConcurrentTotal: 2
```

OVH

Configuration as Code (3)

A regular job defined using JCasC? It is possible...

However, we recommend to define only one, special *seed job*.

```
jobs:
  - script: >
      folder('System') {}
  - script: >
      job('System/Create-jobs') {
        steps {
          dsl {
            text(new File("/var/lib/jenkins/job-definitions/main.groovy").text)
            removeAction('DELETE')
            removeViewAction('DELETE')
          }
        }
      }
```

OVH

Job DSL (1)

- DSL – *Domain Specific Language*.
- Built on top of Groovy.
- Used for defining basic job properties:
  - name and description, triggers, steps, etc.,
  - most options from UI mapped to functions,
  - also allows to modify XML DOM directly.
- Pretty mature - developed for few years.
- One special job type introduced: *seed job*.

OVH

Job DSL (2)

```
utils.makeBasicJob(
    dsl: dsl,
    location: 'Review/',
    jobName: 'Unit-tests',
    jenkinsfilePath: 'review/unit-tests.Jenkinsfile'
).with {
    triggers {
        gerrit {
            project(
                'reg_exp:^OpenStack\\/(?!.+-debianize)',
                'ant:**'
            )
            buildStarted(0, 0)
            buildSuccessful(1, 0)
            buildFailed(-1, 0)
            configure { gerritTrigger ->
                utils.setGerritServer(gerritTrigger, 'my.repo.url.ovh')
                utils.triggerOnPatchsetCreated(gerritTrigger)
                utils.triggerOnCommentContainsRegex(gerritTrigger, '(?i:recheck)')
            }
        }
    }
}
```

OVH

Job DSL (2)

```
utils.makeBasicJob(
    dsl: dsl,
    location: 'Review/',
    jobName: 'Unit-tests',
    jenkinsfilePath: 'review/unit-tests.Jenkinsf
).with {
    triggers {
        gerrit {
            project(
                'reg_exp:^OpenStack\\/(?!.+-debi
                'ant:**'
            )
            buildStarted(0, 0)
            buildSuccessful(1, 0)
            buildFailed(-1, 0)
            configure { gerritTrigger ->
                utils.setGerritServer(gerritTrigger, 'my.repo.url.ovh')
                utils.triggerOnPatchsetCreated(gerritTrigger)
                utils.triggerOnCommentContainsRegex(gerritTrigger, '(?i:recheck)')
            }
        }
    }
}
```

Image source: https://knowyourmeme.com/memes/wait-a-sec-are-u-trying-to-cheat-me-again.

OVH

Job DSL (3)

We defined custom functions to make the Job DSL API more user-friendly.

```groovy
def makeBasicJob(Map args) {
    DslFactory dsl = args.getAt('dsl')
    String location = args.getAt('location')
    String jobName = args.getAt('jobName')
    (...)
    def job = dsl.pipelineJob(location + jobName) {
      logRotator(60, -1, -1, -1)
      definition {
        cpsScm {
          scm {
            git {
              branch(repoBranchName)
              extensions {
                wipeOutWorkspace()
              }
              remote {
                credentials(CREDENTIALS_KEYNAME)
                name('origin')
                url(CI_TOOLS_REPOSITORY_URL)
              }
            }
          }
          scriptPath(jenkinsfilePath)
    } } }
    (...)
    return job
}
```

```groovy
def makePeriodicallyTriggeredJob(Map args) {
  String cronSpec = args.getAt('cronSpec')
  def job = makeBasicJob(args)
  job.with {
    triggers {
      cron(cronSpec)
    }
  }
  return job
}
```

```groovy
def setGerritServer(Node trigger, String serverURL) {
  trigger.appendNode('serverName', serverURL)
}
```

```groovy
def makeFoldersFromList(Map args) {
  DslFactory dsl = args.getAt('dsl')
  List<String> folderNames = args.getAt('folderNames')
  String folderParent = args.get('folderParent', '')
  folderNames.each { folderName ->
    dsl.folder(folderParent + folderName) {}
  }
}
```

OVH

Jenkins Pipeline (1)

```
pipeline {
  agent docker:'maven:3.3.3'
  stages {
    stage('build') {
      steps {
        sh 'mvn --version'
        sh 'mvn install'
      }
    }
  }
}
```

- Pipeline DSL and completely new kinds of job.
- Provides a new way to define jobs' steps:
  - use text to specify actions and parameters,
  - can be stored in files called *Jenkinsfiles*,
  - may be located outside Jenkins itself.
- Two syntax variants: *scripted* and *declarative*.
- Pipelines allow re-usability between jobs.
- More and more plugins are compatible now!

OVH

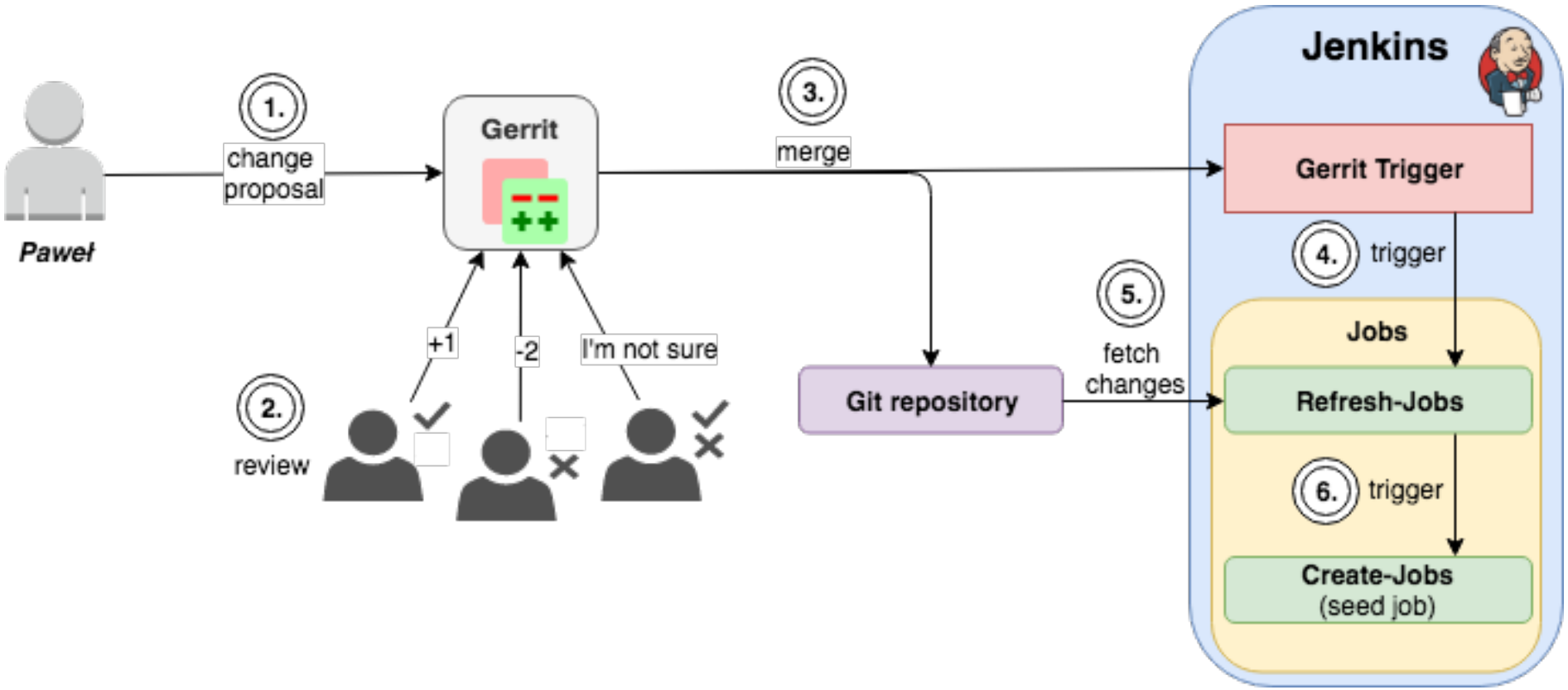
Jenkins Pipeline (2)

```
pipeline {
  agent any
  stages {
    stage('Virtual Machine') {
      steps {
        sh script: '''
          ./common/create-instance.sh \
          --key-name='jenkins' \
          --image='Ubuntu 16.04' \
          --flavor='c2-30'
        '''
        script {
          env.VMIP = readFile('.VMIP').trim()
          env.VMID = readFile('.VMID').trim()
        }
      }
    }

  stage('Environment') {
    steps {
      sshagent([(...]) {
        sh script: '''
          scp (...) common/*.sh "ubuntu@${VMIP}:~/."
          ssh (...) "ubuntu@${VMIP}" "
            (...)
            ./setup-instance-for-unit-tests.sh
          "
        '''
      }
    }

  stage('Tests') {
    steps {
      sshagent([(...]) {
        sh script: '''
          ssh (...) "ubuntu@${VMIP}" "
            (...)
            ~/launch-unit-tests.sh
          "
          '''
        }
      }
    }
  }

  post {
    always {
      sh script: '''
        VMID=$(cat '.VMID' | tr -d '[:space:]')
        nova delete "${VMID}"
      '''
    }
  }
}
```

Workflow overview

All together,

the introduced plugins lead us to automated updates of Jenkins configuration!



Time for some examples...

Let us examine a few use cases in comparison to classical approach:

1. Where to look in order to change global Jenkins configuration?
2. What to do when it is needed to create some new jobs?
3. How can an existing job be modified?

Examples - change global configuration

XML file

```
 <?xml version='1.1' encoding='UTF-8'?>
 <hudson>
   <disabledAdministrativeMonitors>
     <string>
     jenkins.security.s2m.MasterKillSwitchWarning
     </string>
     <string>OldData</string>
   </disabledAdministrativeMonitors>
   <version>2.128</version>
   <installStateName>RUNNING</installStateName>
-  <numExecutors>32</numExecutors>
+  <numExecutors>64</numExecutors>
   <mode>NORMAL</mode>
   <useSecurity>true</useSecurity>
 (...)
   <buildsDir>${ITEM_ROOTDIR}/builds</buildsDir>
   <jdks/>
   <clouds/>
   <quietPeriod>5</quietPeriod>
   <scmCheckoutRetryCount>0</scmCheckoutRetryCount>
   <primaryView>all</primaryView>
   <slaveAgentPort>0</slaveAgentPort>
   <label>localhost master jenkins</label>
   <nodeProperties/>
   <globalNodeProperties/>
   <noUsageStatistics>true</noUsageStatistics>
 </hudson>
```

YAML file

```
 jenkins:
    systemMessage: "This Jenkins instance was (...)"
-   numExecutors: 32
+   numExecutors: 64
    labelString: "localhost master jenkins"
    scmCheckoutRetryCount: 0
    mode: "NORMAL"
    noUsageStatistics: True
```

OVH

Examples - add new jobs (1)

XML file

```xml
<?xml version='1.1' encoding='UTF-8'?>
<flow-definition plugin="workflow-job@2.21">
  <actions>
    <org.jenkinsci.plugins.pipeline.modeldefinition.actions.DeclarativeJobAction
    plugin="pipeline-model-definition@1.3"/>
    <org.jenkinsci.plugins.pipeline.modeldefinition.actions.DeclarativeJobPropertyTrackerAction
    plugin="pipeline-model-definition@1.3">
      <jobProperties/>
      <triggers/>
      <parameters/>
    </org.jenkinsci.plugins.pipeline.modeldefinition.actions.DeclarativeJobPropertyTrackerAction>
  </actions>
  <description></description>
  <logRotator class="hudson.tasks.LogRotator">
    <daysToKeep>60</daysToKeep>
    <numToKeep>-1</numToKeep>
    <artifactDaysToKeep>-1</artifactDaysToKeep>
    <artifactNumToKeep>-1</artifactNumToKeep>
  </logRotator>
  <keepDependencies>false</keepDependencies>
  <properties>
    <org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty>
      <triggers>
        <hudson.triggers.TimerTrigger>
          <spec>H H * * *</spec>
        </hudson.triggers.TimerTrigger>
      </triggers>
    </org.jenkinsci.plugins.workflow.job.properties.PipelineTriggersJobProperty>
  </properties>
```

OVH

Examples - add new jobs (2)

XML file - further part

```
30|  <definition class="org.jenkinsci.plugins.workflow.cps.CpsScmFlowDefinition" plugin="workflow-cps@2.53">
31|    <scm class="hudson.plugins.git.GitSCM" plugin="git@3.9.1">
32|      <configVersion>2</configVersion>
33|      <userRemoteConfigs>
34|        <hudson.plugins.git.UserRemoteConfig>
35|          <name>origin</name>
36|          <url>ssh://my.repo.url.ovh/OpenStack/nova.git</url>
37|          <credentialsId>repo-access-key</credentialsId>
38|        </hudson.plugins.git.UserRemoteConfig>
39|      </userRemoteConfigs>
40|      <branches>
41|        <hudson.plugins.git.BranchSpec>
42|          <name>master</name>
43|        </hudson.plugins.git.BranchSpec>
44|      </branches>
45|      <doGenerateSubmoduleConfigurations>false</doGenerateSubmoduleConfigurations>
46|      <gitTool>Default</gitTool>
47|      <extensions>
48|        <hudson.plugins.git.extensions.impl.WipeWorkspace/>
49|      </extensions>
50|    </scm>
51|    <scriptPath>review/unit-tests.Jenkinsfile</scriptPath>
52|    <lightweight>false</lightweight>
53|  </definition>
54|  <disabled>false</disabled>
55|</flow-definition>
```

OVH

Examples - add new jobs (3)

Job DSL – add single job

```
utils.makePeriodicallyTriggeredJob(
    dsl: dsl,
    location: 'Review/Unit-tests/nova/',
    jobName: 'master',
    cronSpec: 'H H * * *',
    jenkinsfilePath: 'review/unit-tests.Jenkinsfile'
)
```

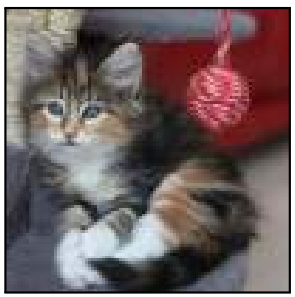
Image source: https://placekitten.com/.

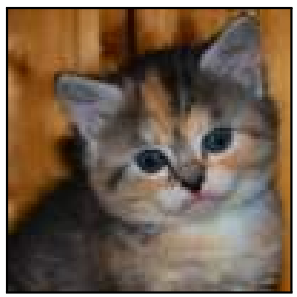
OVH

Examples - add new jobs (4)

Job DSL – add multiple jobs

```
1|def branches = ['master', 'rocky', 'queens', 'pike', 'ocata', 'newton']
2|branches.each { branchName ->
3|    utils.makePeriodicallyTriggeredJob(
4|        dsl: dsl,
5|        location: 'Review/Unit-tests/nova/',
6|        jobName: branchName,
7|        cronSpec: 'H H * * *',
8|        jenkinsfilePath: 'review/unit-tests.Jenkinsfile'
9|    )
10|}
```


Images source: https://placekitten.com/.

OVH

Examples - modify existing job

XML file

```
 <?xml version='1.1' encoding='UTF-8'?>
 <project>
   (...)
   <triggers>
     <hudson.triggers.TimerTrigger>
       <spec>H * * * *</spec>
     </hudson.triggers.TimerTrigger>
   </triggers>
   <builders>
     <hudson.tasks.Shell>
       <command>apt-get update
-apt-get install python3-dev</command>
+apt-get install python3-dev
+pip install 'tox'</command>
     </hudson.tasks.Shell>
+    <hudson.tasks.Shell>
+      <command>tox -v -e py36,pep8</command>
+    </hudson.tasks.Shell>
     <hudson.tasks.Shell>
       <command>dpkg-buildpackage -b</command>
     </hudson.tasks.Shell>
   </builders>
   <publishers>
     <hudson.tasks.ArtifactArchiver>
       <artifacts>*.deb</artifacts>
       (...)
     </hudson.tasks.ArtifactArchiver>
   </publishers>
 </project>
```

Jenkinsfile

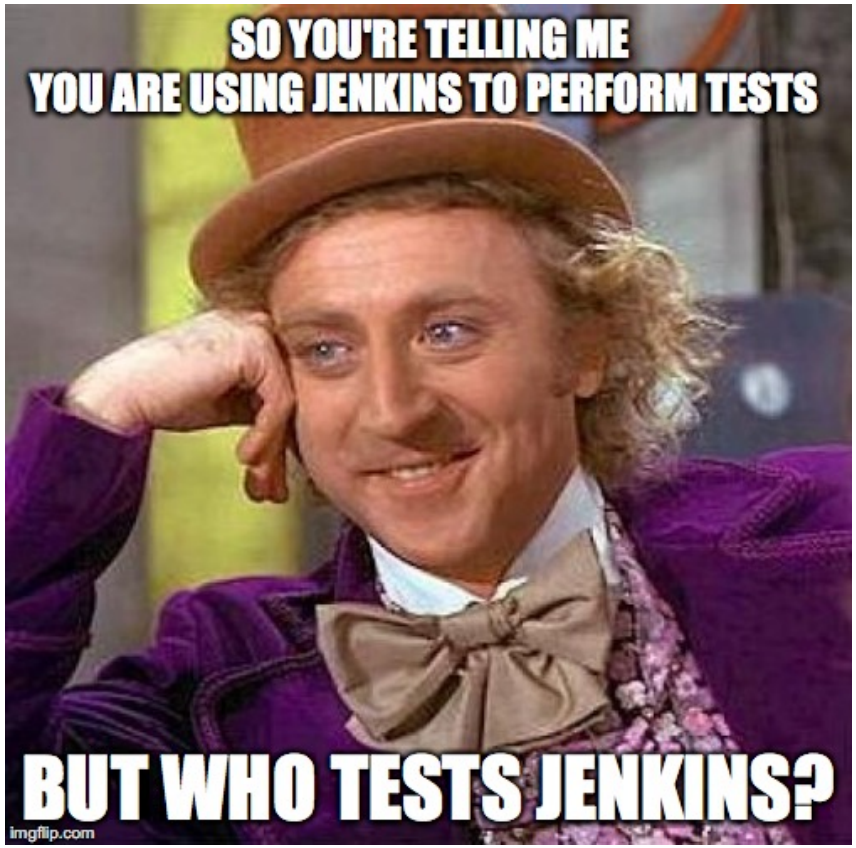
```
 pipeline {
   agent any

   stages {
     stage('Build package') {
       steps {
         sh script: '''
           apt-get update
           apt-get install python3-dev
+          pip install 'tox'
           '''
+        sh script: 'tox -v -e py36,pep8'
         sh script: 'dpkg-buildpackage -b'
       }
     }
   }

   post {
     always {
       archiveArtifacts artifacts: '*.deb'
     }
   }
 }
```

OVH

But wait a second here...

Since we can have clean review of changes now, the whole process starts to reassemble a regular way you (should) test your projects.

There is still, however, a field for necessary improvement...

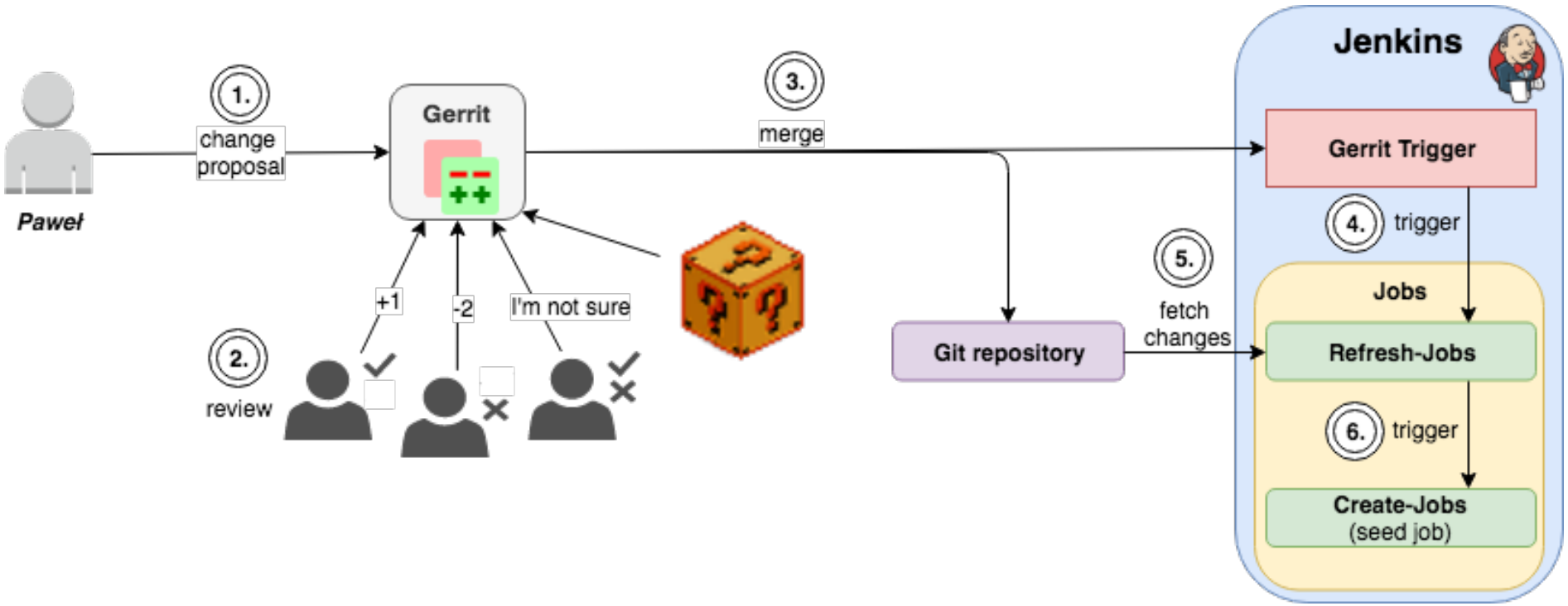


OVH

The missing piece (1)

The human-review process does not guarantee the build will be successful.

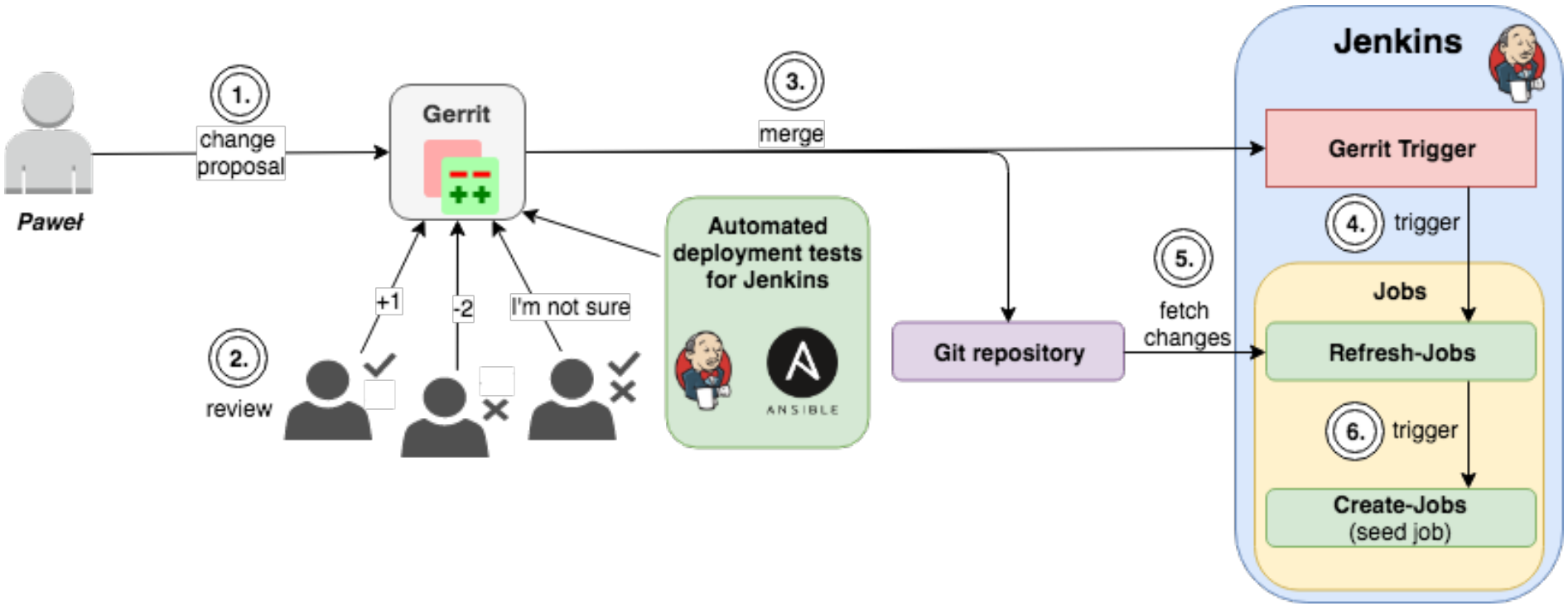
We do miss something important here...



The missing piece (2)

There shall be a non-human review component present as compulsory part.

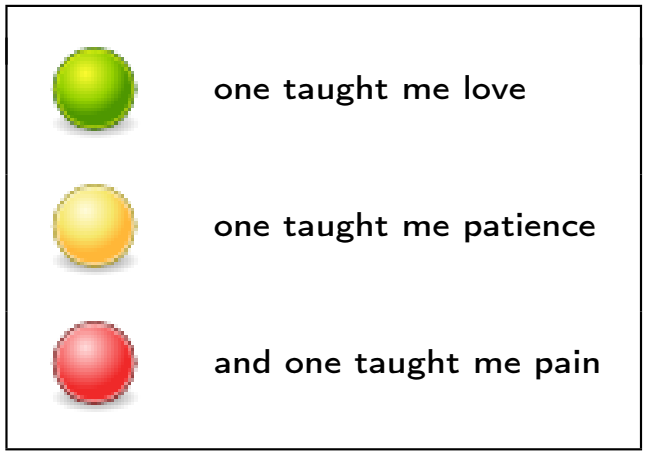
Having a configuration stored in versioned code repository makes it easier!



Conclusions

To sum up:

- Jenkins - open and powerful,
- easy to start and configure,
- three helpful plugins to checkout:
  - Jenkins Configuration as Code,
  - Job DSL,
  - Job Pipelines;
- they allow and simplify:
  - meaningful code reviews,
  - proper workflow for changes.



OVH

Thank you for your attention!

The slides are available: `http://datko.pl/OS-Berlin.pdf`

OVH

From messy XML to wonderful YAML and pretty Job DSL

An in-Jenkins migration story

Piotr Bielak
piotr.bielak@corp.ovh.com

Szymon Datko
szymon.datko@corp.ovh.com

13th November 2018