Pimp your Continuous Delivery Pipeline with Jenkins workflow

Cyrille Le Clerc, CloudBees

About Me

@cyrilleleclerc

Xebia

CTO

CloudBees

Product Management

Cyrille Le Clerc

Open Source

jmxtrans

The Apache Software Foundation
http://www.apache.org/

DevOps, Infra as Code,
Continuous Delivery

About you

Jenkins? CI? CD? Code Jenkins plugin?

Agenda

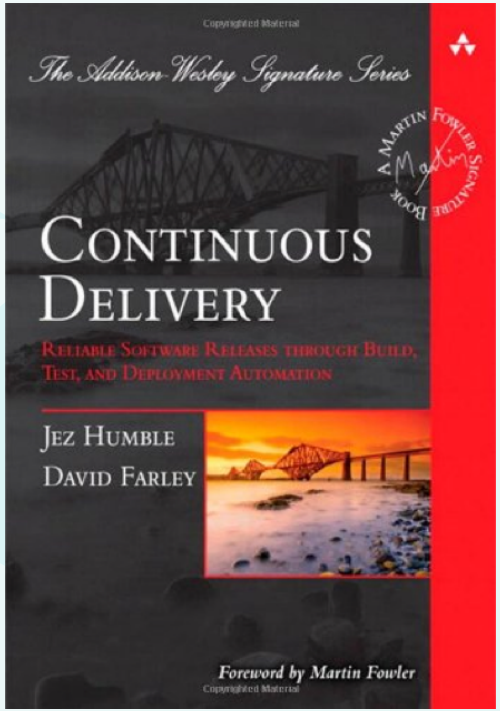
- Continuous Delivery
- Jenkins & Continuous Delivery
- Workflows in Jenkins today
- The new Jenkins Workflow Engine
- Jenkins Workflow Syntax Card
- Possible future

Continuous Delivery

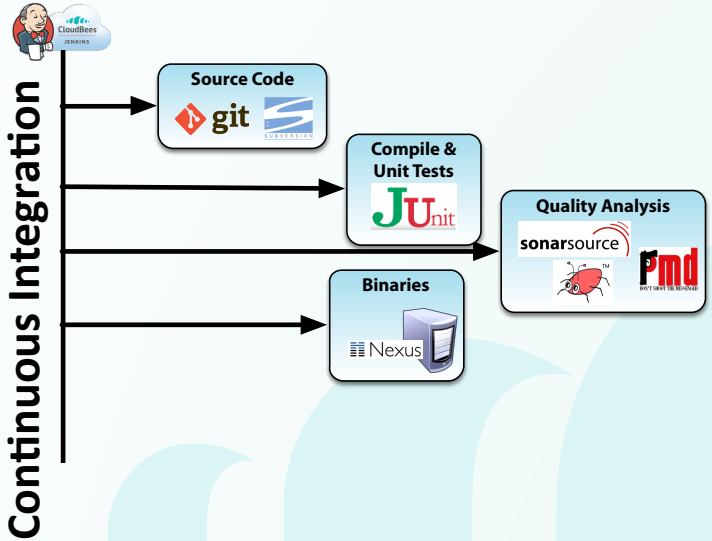
Continuous Delivery

- *"Reliable Software Releases through Build, Test, and Deployment Automation"*

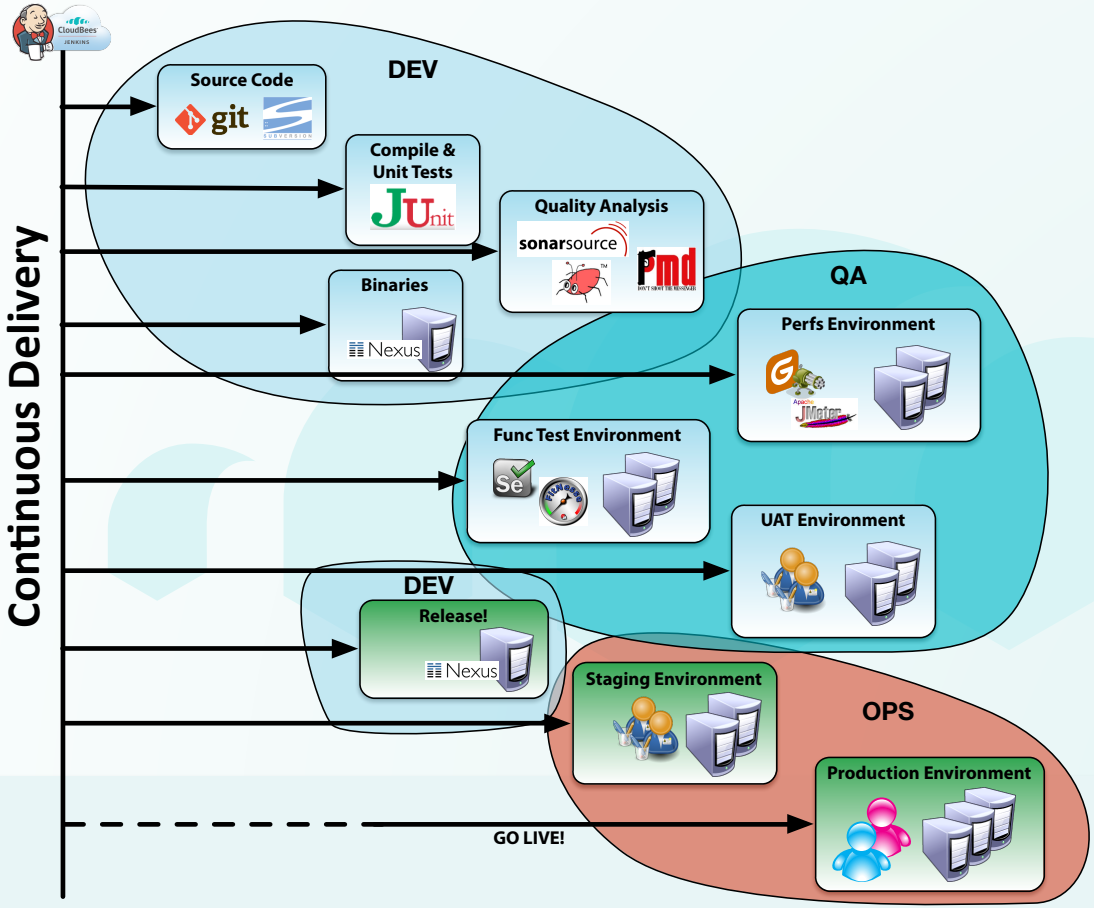
- DONE = Shippable into Production



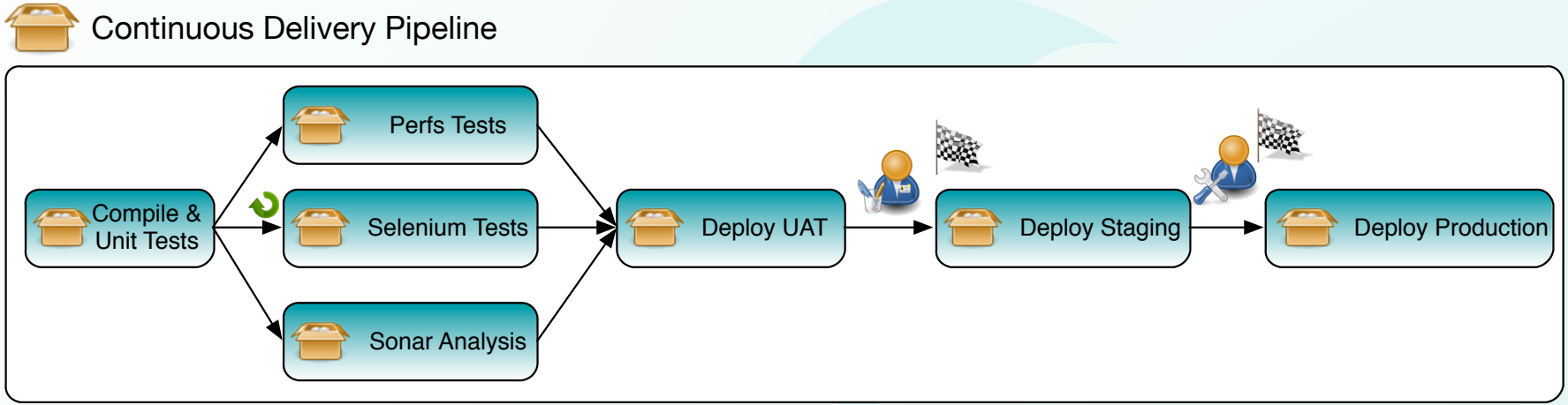
Continuous Integration Steps



Continuous Delivery Steps

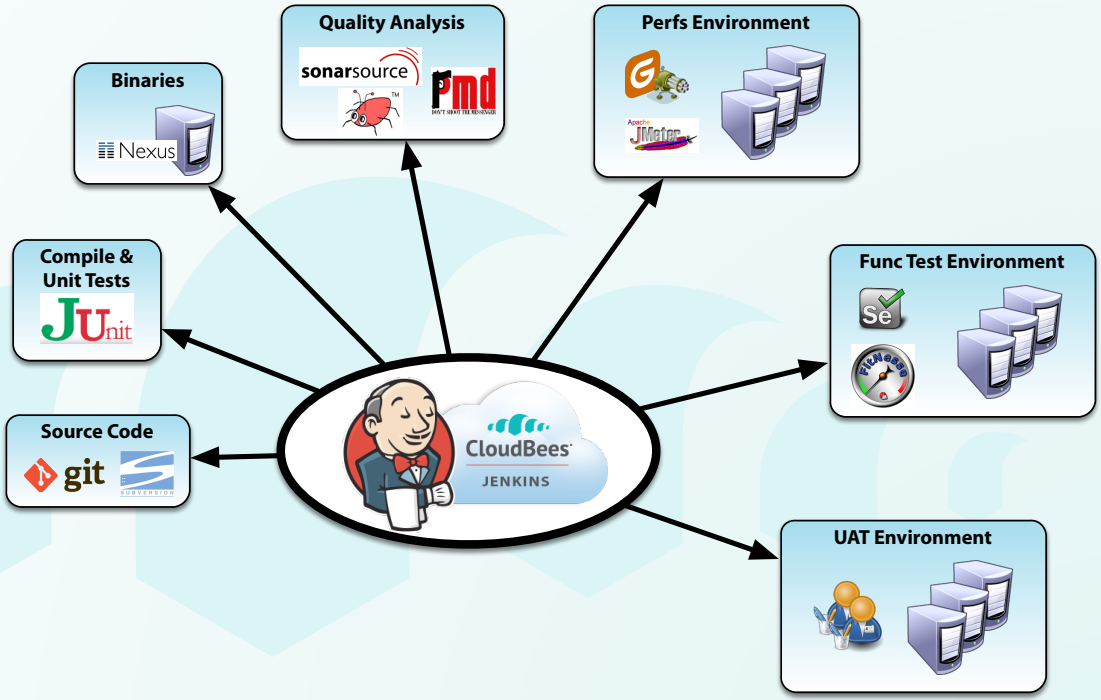


Continuous Delivery Pipeline

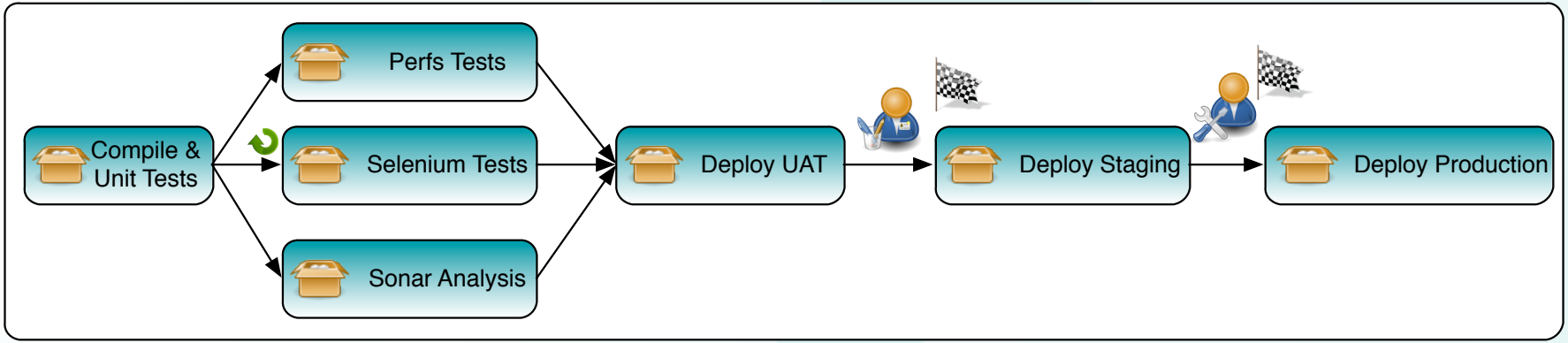


Jenkins: Hub of Continuous Delivery

- Connect the dots
- It is the process



CD Pipeline ➔ Workflow

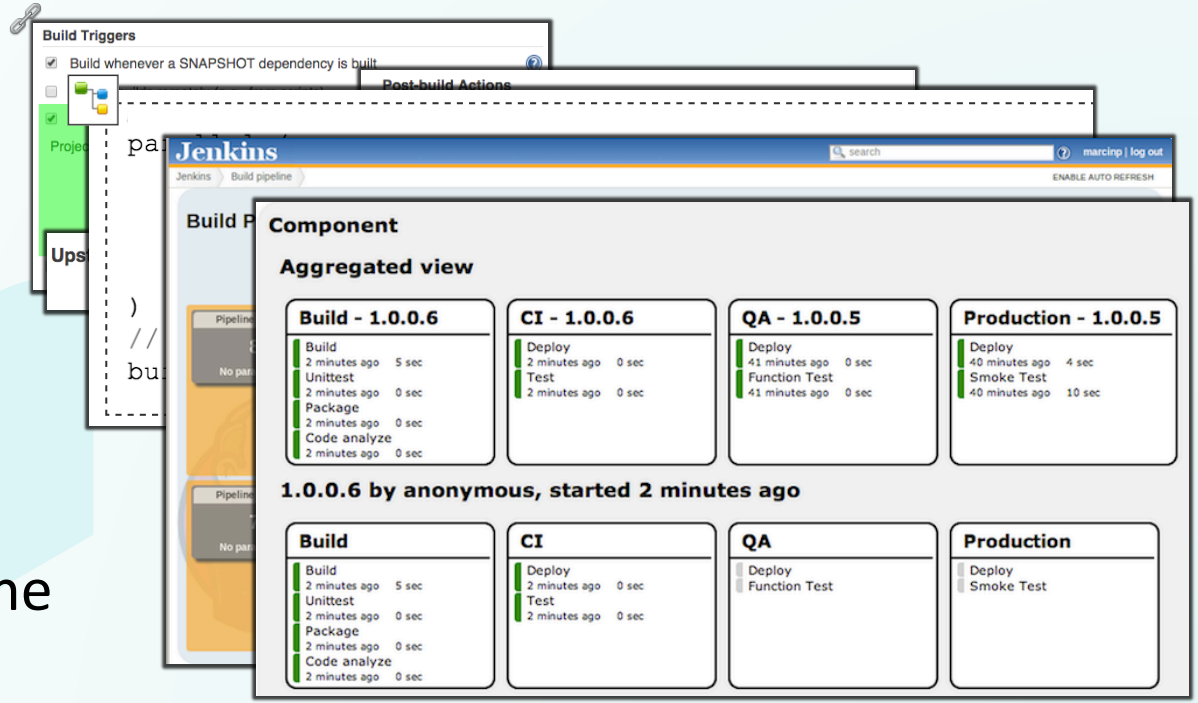


What we need

- **Complex pipelines** involving multiple stages
- **Non-sequential logic** such as loops, forks …
- **Long-running builds** must survive outages
- **Interactions with humans** including pauses, input
- **Restartable builds** in case of transient errors
- **Reusable definitions** to avoid duplication
- **Comprehensible scripts** with one clear definition

Workflow until today

- Job chaining
- Build Flow Plugin
- Build Pipeline Plugin
- Build Delivery Pipeline



Workflow until today

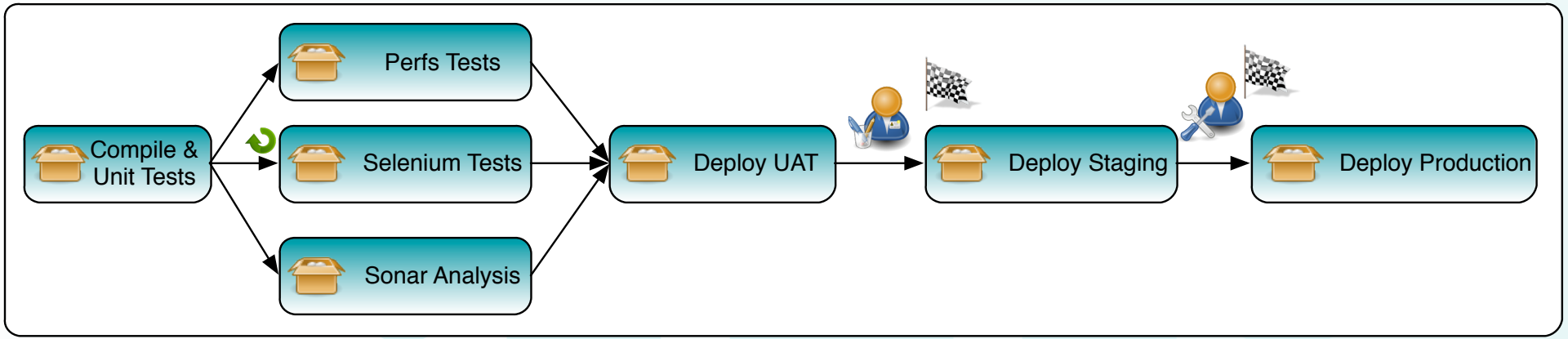
- Many atomic jobs
- Hard to share variables/state between jobs
- Limited logic
- Mix build triggers, parameterized build …

Build Flow Plugin

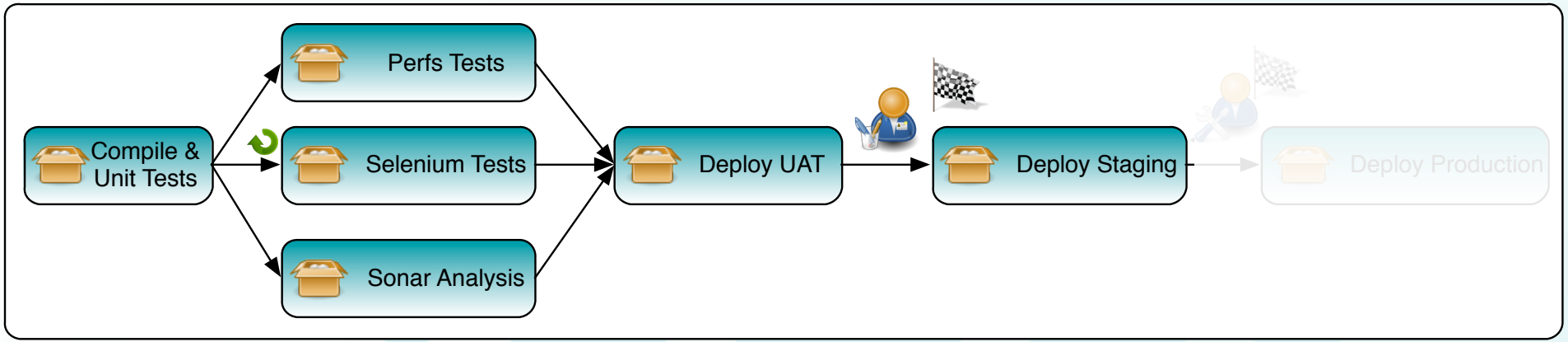
- Did have scriptability and extensibility
- Did not address configuration “sprawl”
- Disjointed view of what really ran
- No ability to survive restarts
- Almost good enough but could not go further

Jenkins Workflow Engine

Simplifying our sample workflow



Simplifying our sample workflow



What we need

- **Complex pipelines** involving multiple stages
- **Non-sequential logic** such as loops, forks …
- **Long-running builds** must survive outages
- **Interactions with humans** including pauses, input
- **Restartable builds** in case of transient errors
- **Reusable definitions** to avoid duplication
- **Comprehensible scripts** with one clear definition

New Workflow Syntax

```
stage 'DEV'
node('linux') {
    // COMPILE AND JUNIT
    git url: 'https://github.com/cyrille-leclerc/spring-petclinic.git'
    sh 'mvn -o clean package'
    archive 'target/petclinic.war, src, pom.xml'
    step $class: 'hudson.tasks.junit.JUnitResultArchiver', testResults: 'target/surefire-reports/*.xml'
}

parallel(qualityAnalysis: {
    // RUN SONAR ANALYSIS
    node('linux') {
        stage name: 'QUALITY_ANALYSIS', concurrency: 1
        unarchive mapping: ['src/': '.', 'pom.xml': '.']
        sh 'mvn -o sonar:sonar'
    }
}, performanceTest: {
    // DEPLOY ON PERFS AND RUN JMETER STRESS TEST
    node('linux') {
        stage name: 'PERFS', concurrency: 1
        unarchive mapping: ['src/': '.', 'pom.xml': '.', 'target/petclinic.war': 'petclinic.war']
        deployApp 'petclinic.war', perfsCatalinaBase, perfsHttpPort
        sh 'mvn -o jmeter:jmeter'
        shutdownApp(perfsCatalinaBase)
    }
})

checkpoint 'ENTER QA'
input message: "Deploy to QA?", ok: "DEPLOY TO QA!"
// DEPLOY ON THE QA SERVER
node('linux') {
    stage name: 'QA', concurrency: 1
    unarchive mapping: ['target/petclinic.war': 'petclinic.war']
    deployApp 'petclinic.war', qaCatalinaBase, qaHttpPort
}
```

Key features

- Entire flow is one concise Groovy script
  - For loops, try-finally, fork-join …
- Can restart Jenkins while flow is running
- Allocate slave nodes and workspaces
  - As many as you want, when you want
- *Stages* throttle concurrency of builds
- Human input/approval integrated into flow
- Standard project concepts: SCM, artifacts, plugins

Groovy DSL vs. Graphical Workflow

- Familiar control flow construction
- Familiar “tools” for building abstractions

  Functions, classes, variables, …
- Workflow in version control
- As opposed to:

  Graphical workflow designer

Project setup

- One workflow is defined as a job
- Single script for all steps
- Build triggers & parameters like regular projects
- Each workflow execution is a regular Jenkins build displayed in regular Jenkins views
- Graphical visualization of actual build possible

  Not of visualization job definition but of build execution

Non-sequential logic

```
parallel(qualityAnalysis: {
    // RUN SONAR ANALYSIS
    // ...
}, performanceTest: {
    // DEPLOY ON PERFS AND RUN JMETER STRESS TEST
    // ...
})
```

```
for (def hostname: hostnames) {
    sh "ssh $hostname -c /opt/tomcat/bin/shutdown.sh"
}
```

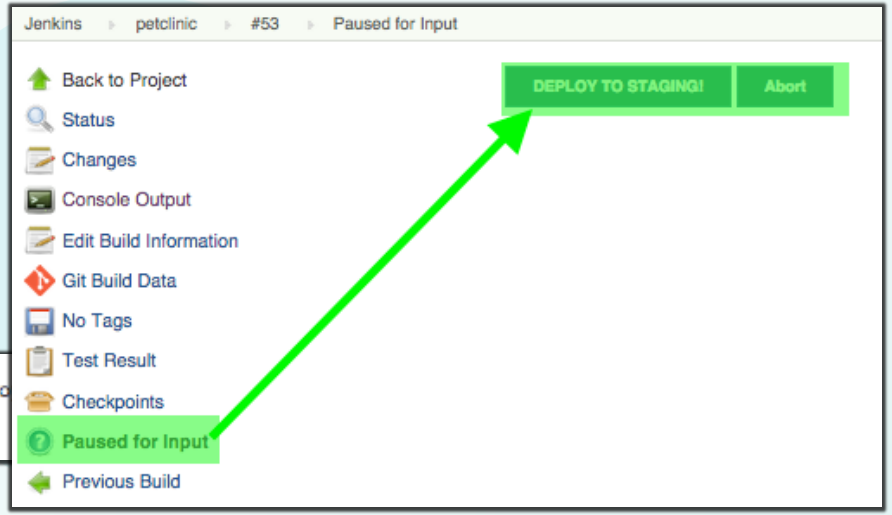
try {} catch {}

```
retry(count: 5) {
    sh "sleep 5 && curl http://localhost:$httpPort/health-check.jsp"
}
```

Interaction with humans

```
42  input message: "Does staging app http://localhost:$qaHttpPort/ look good? " ,
43                 ok: "DEPLOY TO STAGING!", submitter: "petclinic-qa-group"
```

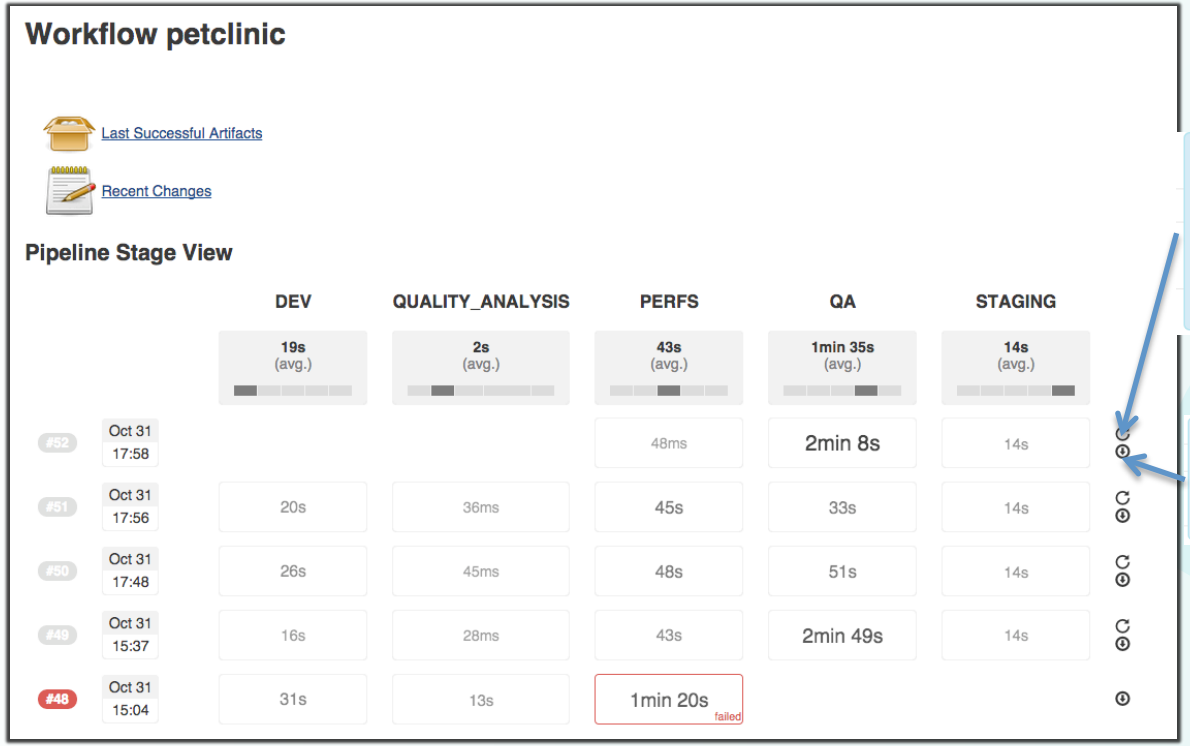
```
Running: Input
Does staging app http://localhost:8081/ look good? If yes, we deplo
DEPLOY TO STAGING! or Abort
```



Restartable build / checkpoint


```
36
37  checkpoint 'CHOOSE TO ENTER STAGING'
38
39  input message: "Does staging app http://localhost:$qaHttpPort/ look good? " +
40                 "If yes, we deploy on staging.", ok: "DEPLOY TO STAGING!"
```

Visualization



This Workflow run can be restarted from the following Checkpoint(s):

Delete	Restart	ENTER QA
Delete	Restart	CHOOSE TO ENTER STAGING

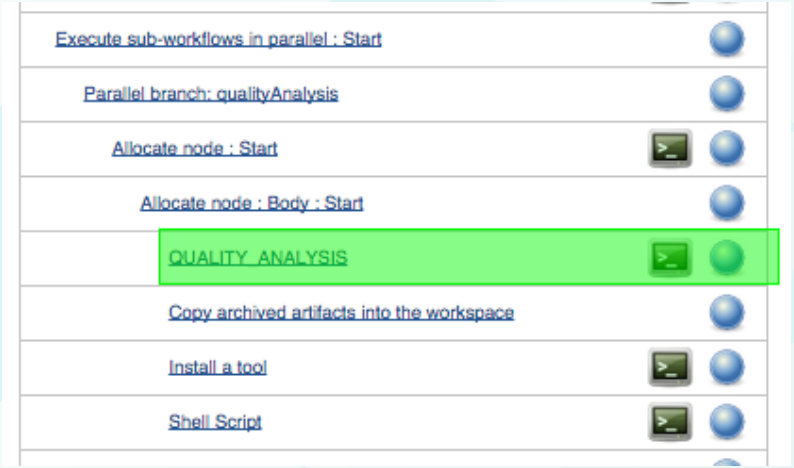
Produced the following 2 artifact(s):

src.tar	(417.50KB)
petclinic.war	(28.04MB)

Stages

```
stage name: 'QUALITY_ANALYSIS', concurrency: 1
unarchive mapping: ['src/': '.', 'pom.xml': '.']
sh 'mvn -o sonar:sonar'
```

- Semaphore
- Visualization



Comprehensible scripts


```groovy
/**...*/
def deployApp(war, catalinaBase, httpPort) {
    sh "${catalinaBase}/bin/shutdown.sh || :" // use "|| :" to ignore exception if server is not started
    sh "rm -rf ${catalinaBase}/webapps/ROOT"
    sh "rm -rf ${catalinaBase}/webapps/ROOT.war"
    sh "cp -rf ${war} ${catalinaBase}/webapps/ROOT.war"
    sh "${catalinaBase}/bin/startup.sh"
    echo "$catalinaBase server restarted with new webapp $war, see http://localhost:$httpPort"
    retry(count: 5) {
        sh "sleep 5 && curl http://localhost:$httpPort/health-check.jsp"
    }
}

/**...*/
def shutdownApp(catalinaBase) {
    sh "${catalinaBase}/bin/shutdown.sh || :" // use "|| :" to ignore exception if server is not started
    echo "$catalinaBase server is stopped"
}
```

Demo time!

Workflow Syntax Card

Workflow Syntax Card

- **`stage`**: Enter a new stage
- **`node`**: Allocate node
- **`ws`**: Allocate workspace
- **`parallel`**: Execute sub-workflow in parallel
- **`retry`**: Retry the body up to N times
- **`catchError`**: Catch error and continue
- **`input`**: Input / human interaction
- **`load`**: Evaluate a Groovy source file into the workflow script

Workflow Syntax Card

- ***step***: General build step
- ***sh***: Shell script
- ***bat***: Windows batch script

Workflow Syntax Card

- **`archive`**: Archive artifacts
- **`unarchive`**: Copy archived artifact into the workspace
- **`echo`**: Print message
- **`dir`**: Change directory
- **`readFile`**: Read file from workspace
- **`writeFile`**: Write file to workspace

Workflow Syntax Card

- ***`git`***: Git
- ***`svn`***: Subversion
- ***`scm`***: General SCM
- ***`tool`***: install a tool
- ***`build`***: Build a job

Workflow Syntax Card

- ***`checkpoint`***: capture the state of the execution so that it can be restarted later

Possible Future

Possible Future: probably OSS

- More plugin interoperability
- Multi-branch project (& SCM API) integration
- More advanced SCM
- In-IDE editing support
- Concise syntax

Possible Future: probably Jenkins Enterprise

- More visualizations
- Validated merge integration
- Deployment, incl. blue/green with rollback
- Freestyle and/or Build Flow import
- Templates integration

Get the code

- https://gist.github.com/cyrille-leclerc/796085e19d9cec4a71ef
  - workflow.groovy