

Jenkins Pipelines

Steffen Gebert (@StGebert)

DevOps Meetup Frankfurt, 20.10.2016

About Me



**Researcher / PhD Student
(Software-based Networks)**

2011 - 2016

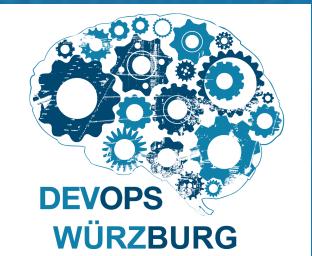


Core Team Member

2010 - 2013

Server Admin Team Member

since 2011



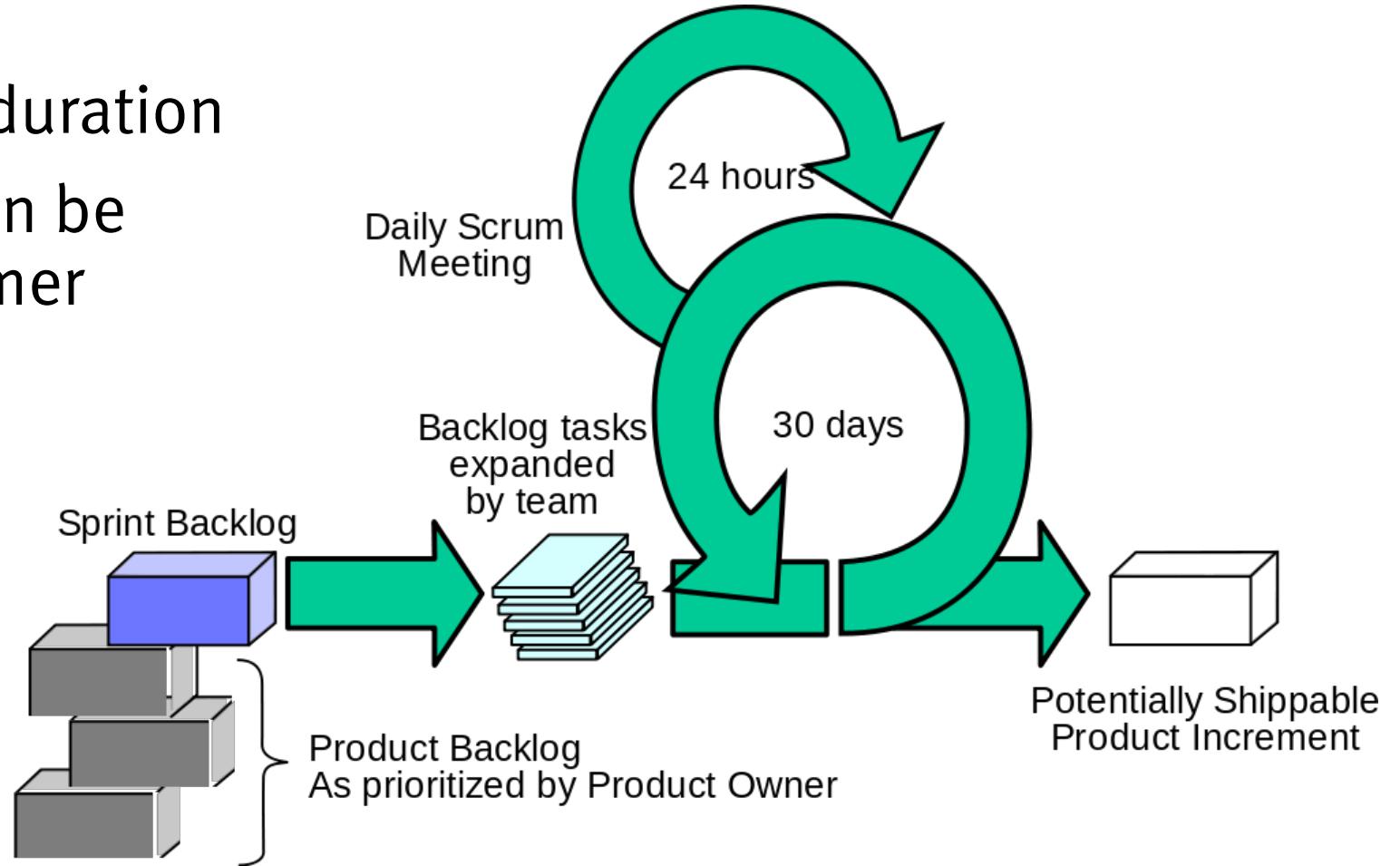
Co-Organizer DevOps Meetup Würzburg

since last week

Continuous Delivery

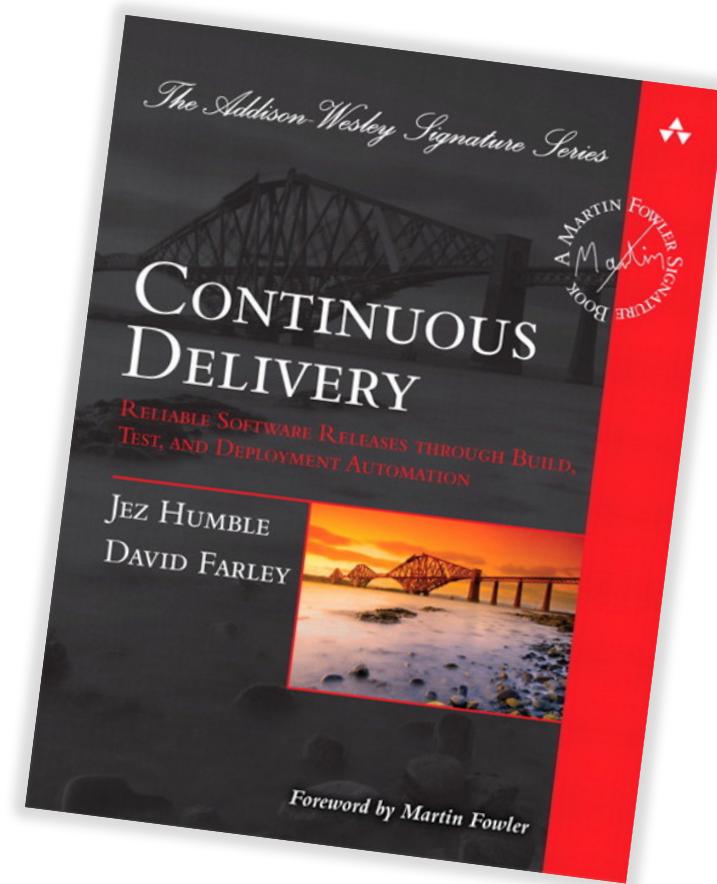
Agile Development

- Sprints of ~1-4 weeks duration
- Result: Product that can be shown/given to customer
- Return: Feedback



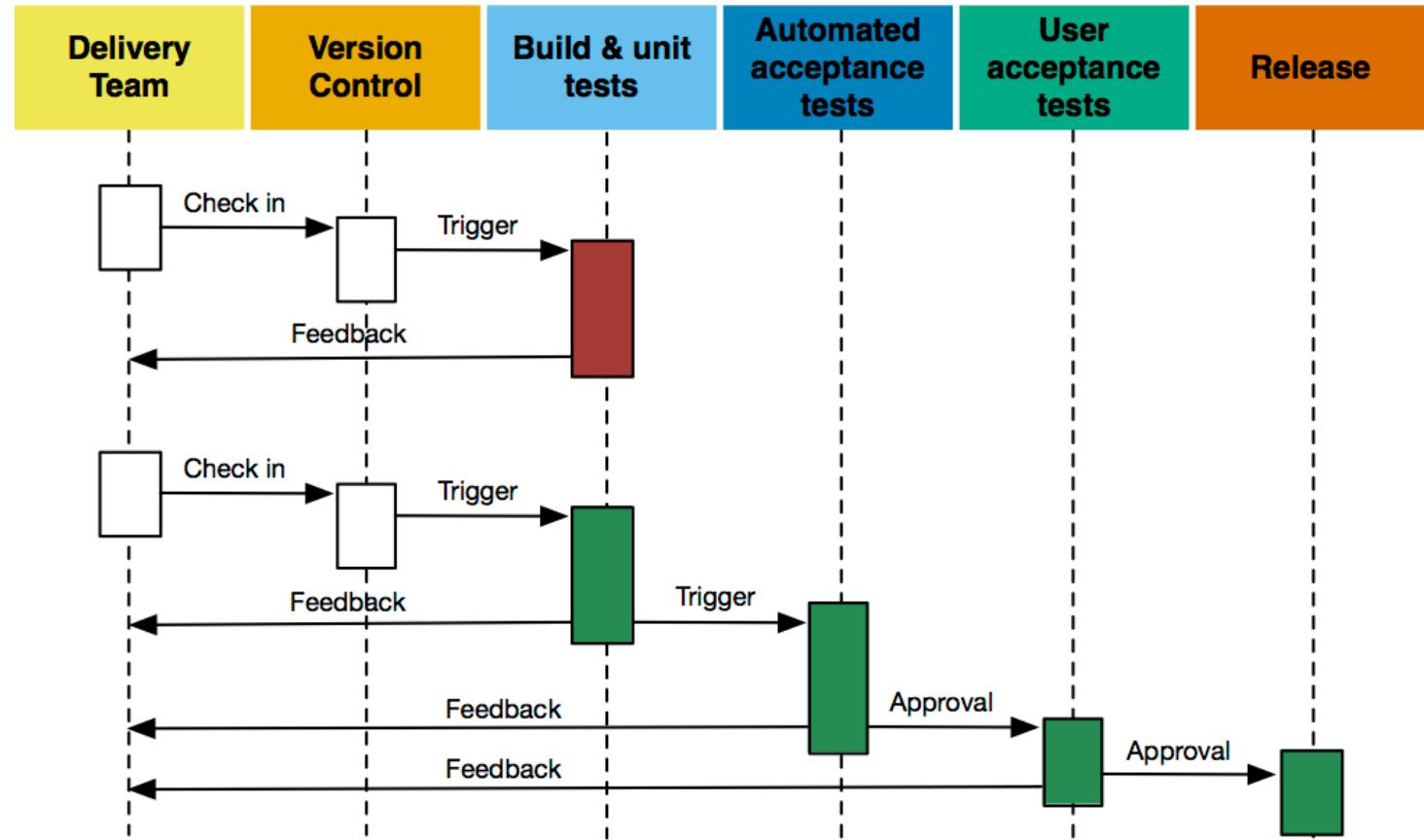
Continuous Delivery

- Reduce cycle times (even more)
 - cf. “DevOps is not enough” talk
 - *“How long does it take to release a one-line change?”*
- Potentially deliverable code with every commit
 - Automated tests decide about acceptance
 - You don’t *have to* release/deploy it, but you *could*
→ Continuous Deployment: Deploy every successfully tested commit automatically



Pipelines

- Every check-in triggers pipeline execution
- Feedback to the team in every stage
 - “*Bring the pain forward*”
 - “*Fail fast, fail often*”
- Minimize execution time
- Always aware of latest *stable* release



CI & CD Tools

CI/CD Tools

- On-premise
 - *Jenkins*
 - Thoughtworks Go
 - Gitlab CI
- SaaS
 - TravisCI
 - CircleCI
 - AppVeyor
 - Codeship
 - Visual Studio Team Services

Why (I like) Jenkins



- Established open source project
- On premise installation
- Thousands of plugins
- Integrates with many tools/services
- Tailored to CI/CD

Jenkins

search log in

ENABLE AUTO REFRESH

People Build History Job Priorities

Build Queue

No builds in the queue.

Build Executor Status

Icon: S M L

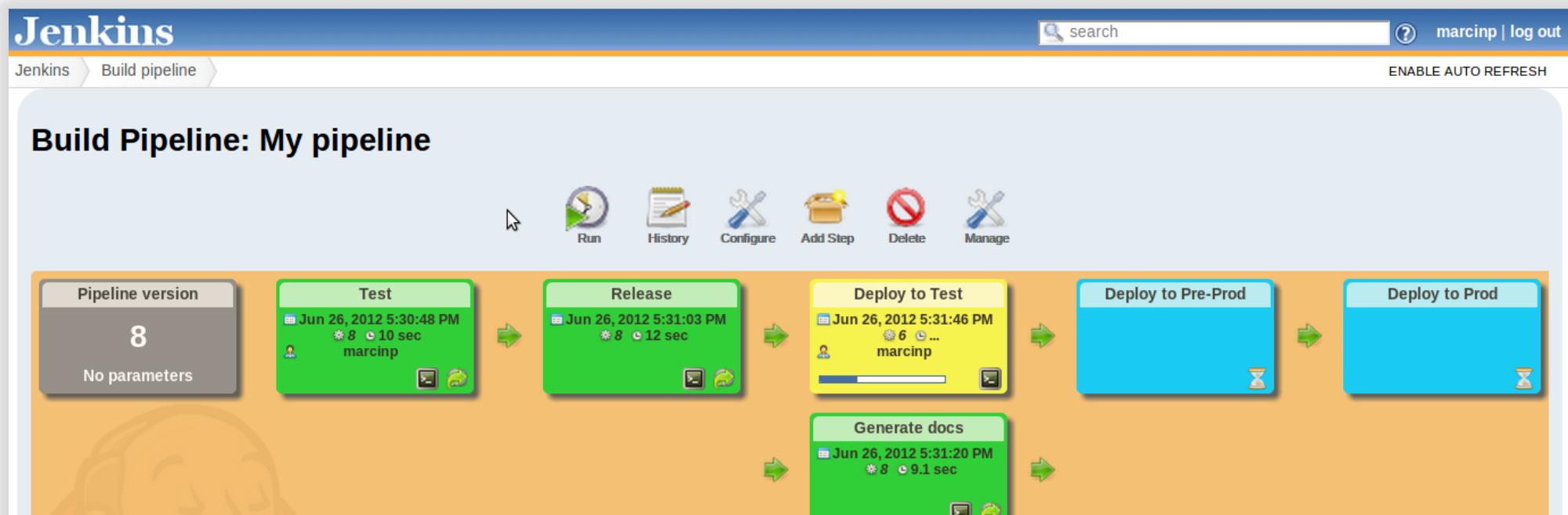
All

S	W	Name ↓	Last Success	Last Failure	Last Duration
green	sun	Main Chef repo	3 days 23 hr - #42	3 mo 18 days - #14	1 sec
green	sun	Seed: Chef Main Repo	1 mo 28 days - #2	N/A	1.3 sec
orange	rain	TYPO3's Chef Cookbooks	N/A	N/A	N/A

Legend RSS for all RSS for failures RSS for just latest builds

History of CI/CD with Jenkins

- Downstream Jobs
 - Job A triggers job B triggers job C triggers job D trig...
 - If one job fails, fail all
- *Build Pipeline View*



... and what I want instead

But that's awkward

Configuration as Code

- Define jobs/pipelines as code
 - Avoid point & click
 - **In version control**
 - Can live with the application
 - **Scales better**
- Example: `.travis.yml` (from TravisCI)
- Similar to GitlabCI

```
language: php

services:
- redis-server

before_script:
- composer install

script:
- ./bin/phpunit -c ...

notifications:
slack:
  ...
  ...
```

Code-driven Approaches in Jenkins

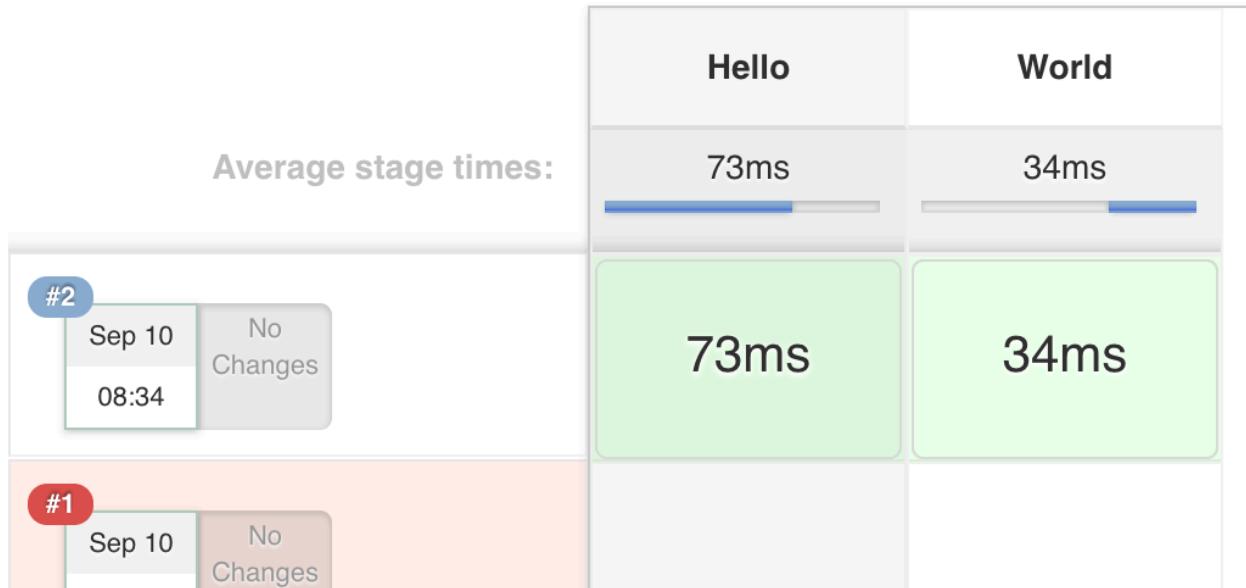
- Jenkins Job Builder
 - Python / YAML based, from the OpenStack project
- Job DSL plugin
 - Groovy DSL!
(e.g. query Github API and create jobs for all branches)
 - Great thing!
 - But creates single jobs

```
job('my-project-main') {  
    scm {  
        git('https://github.com/...')  
    }  
    triggers {  
        scm('H/15 * * * *')  
    }  
    publishers {  
        downstream('my-project-unit')  
    }  
}
```

Jenkins Pipeline Plugins

Jenkins Pipeline Plugins

- Whole suite of plugins (10+), open-sourced earlier this year
- Shipped with Jenkins 2.0
- Formerly commercially available by CloudBees, called *Workflow*
- Define pipeline as code (again Groovy DSL)



```
stage("Hello") {  
    echo "*Hello*"  
}  
stage("World") {  
    echo "*World*"  
}
```

Pipeline Example



Console Output

```
Started by user anonymous
[Pipeline] node
Running on master in /var/lib/jenkins/jobs/T3CM/jobs/Example/workspace
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
[workspace] Running shell script
+ echo Could run composer intall now
Could run composer intall now
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Unit)
[Pipeline] sh
[workspace] Running shell script
+ echo Could run phpunit now
Could run phpunit now
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

```
node {
    stage("Build") {
        sh "echo Could run 'composer install' now"
    }
    stage("Unit") {
        sh "echo Could run 'phpunit' now"
    }
    // ...
}
```

Pipeline Execution

- node step allocates executor slot (“heavyweight executor”)
 - As other Jenkins jobs
 - Filter node by labels (i.e. `node('php7')`, `node('master')`)
 - Required for heavy lifting
 - Avoid many sequential allocations (slows down pipeline progress)
- Code outside of node
 - Flyweight executor
 - Running on master, “for free”
- Pipeline execution survives Jenkins restarts (CPS)

Pipeline DSL Steps

- Shellout
 - *nix systems: sh
 - sh("make"), sh("rm -rf /")
 - Windows systems: bat
 - bat("C:\Program Files\...")
- SCM
 - checkout("https://github.com/..git")
- File handling
 - readFile("my.config")
 - writeFile(file: "README.md", text: "Hello World")
 - fileExists

Pipeline DSL Steps (2)

- Build another job:

```
build("jobname")
build("jobname", wait: false, propagate: false)
Build(job: 'jobname', parameters: [
    [$class: 'StringParameterValue', name: 'target', value: target]
    [$class: 'ListSubversionTagsParameterValue', name: 'release', tag: release],
    [$class: 'BooleanParameterValue', name: 'update_composer', value:
        update_composer.to_boolean()]])
```

- Will not go further into detail ☺

Pipeline DSL Steps (3)

- Copy workspace to next executor

```
stage("build") {  
    node {  
        sh("make")  
        stash("my-workspace")  
    }  
}  
stage("release") {  
    node('magic-release-tool') {  
        unstash("my-workspace")  
        sh("release")  
    }  
}
```

Pipeline DSL Steps (4)

- Specify custom environment variables

```
withEnv(['MYTOOL_HOME=/usr/local/mytool']) {  
    sh '$MYTOOL_HOME/bin/start'  
}
```

```
env.FOO = "bar"  
sh 'echo $BAR'
```

- Use variables provided by Jenkins

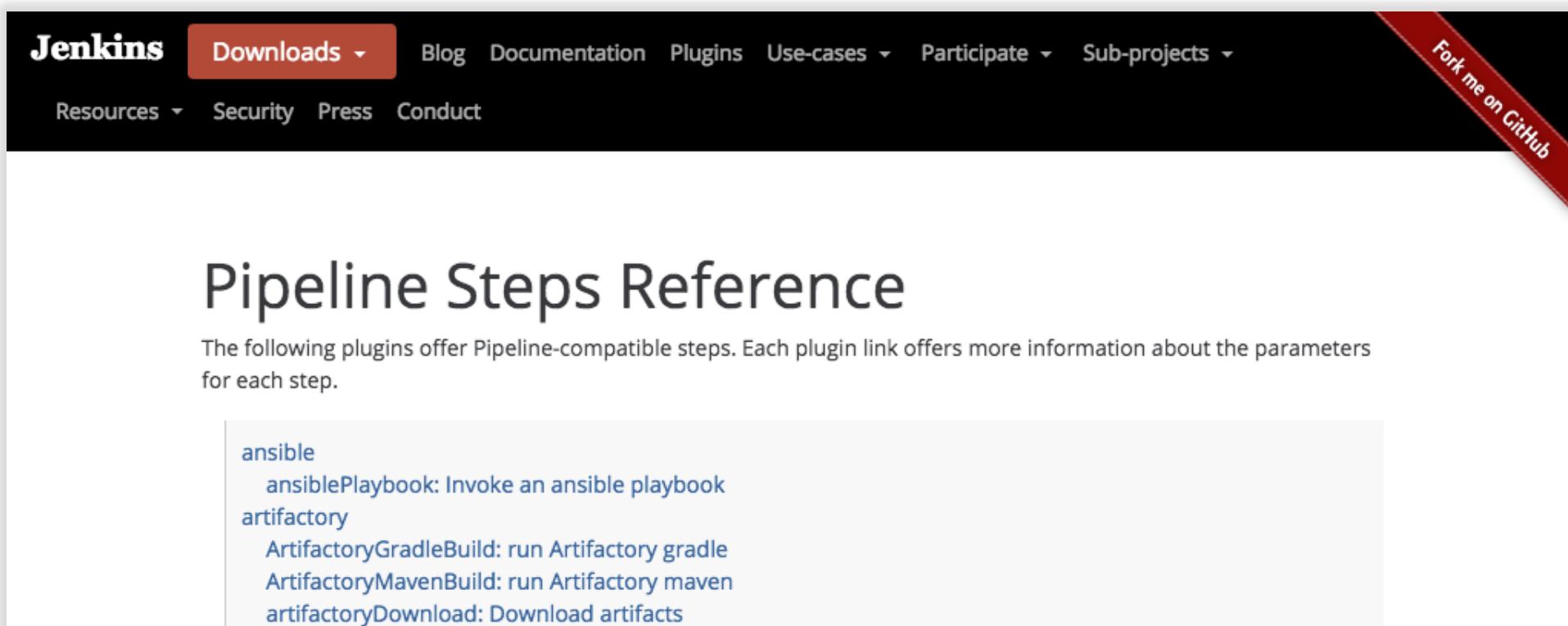
```
sh 'echo $BUILD_NUMBER > version.txt'
```

- Global Groovy variables

```
currentBuild.result = 'FAILED'
```

Even More Pipeline Steps

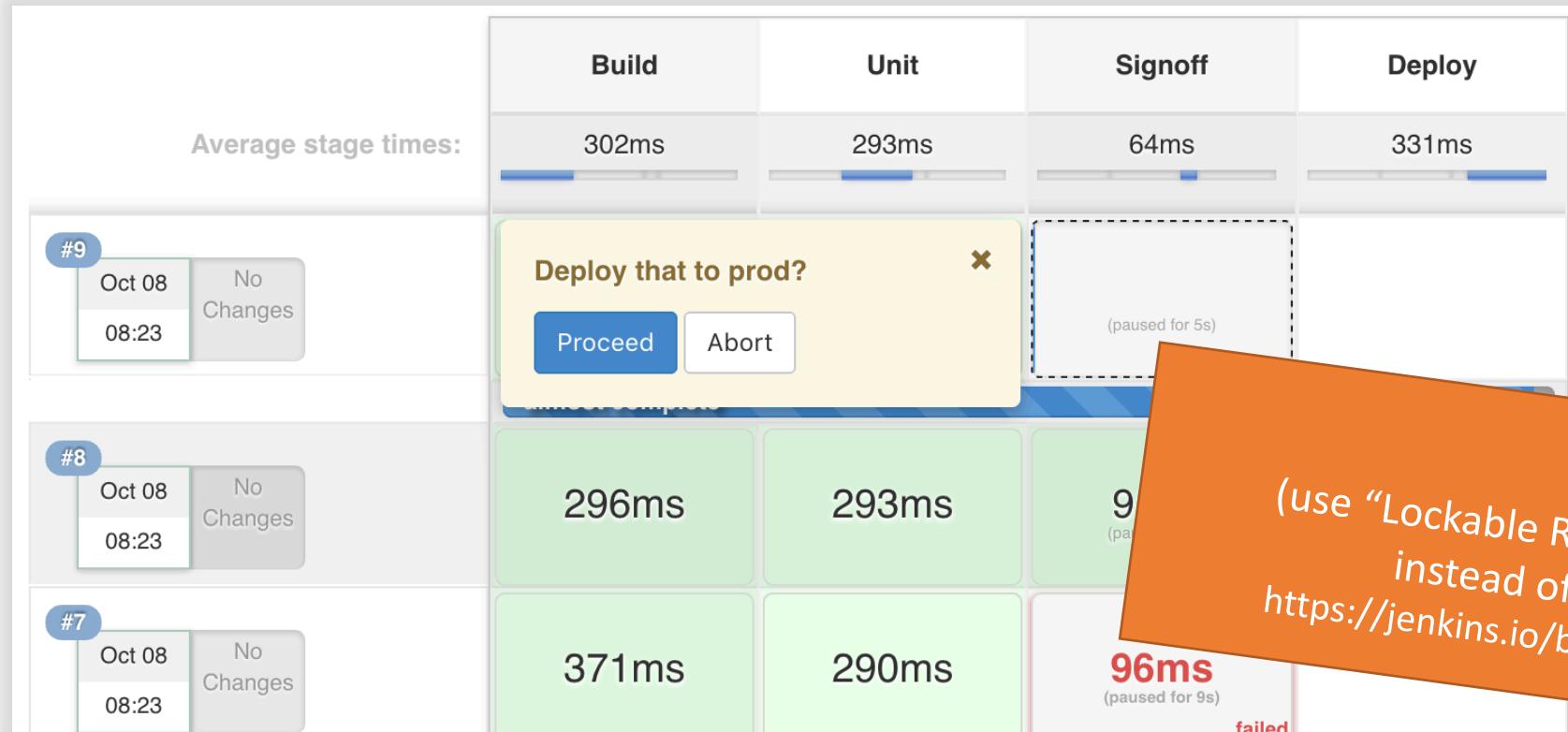
- Plugins contribute additional steps
 - Steps available in this Jenkins instance via  Pipeline Syntax
 - Online reference: <https://jenkins.io/doc/pipeline/steps/>



The screenshot shows the Jenkins Pipeline Steps Reference page. The top navigation bar includes links for Jenkins, Downloads, Blog, Documentation, Plugins, Use-cases, Participate, Sub-projects, Resources, Security, Press, Conduct, and a GitHub fork button. The main content area features a large title "Pipeline Steps Reference". Below it, a paragraph states: "The following plugins offer Pipeline-compatible steps. Each plugin link offers more information about the parameters for each step." A list of plugins is provided, each with a link to its documentation:

- ansible
- ansiblePlaybook: Invoke an ansible playbook
- artifactory
- ArtifactoryGradleBuild: run Artifactory gradle
- ArtifactoryMavenBuild: run Artifactory maven
- artifactoryDownload: Download artifacts

Example: Manually Confirm Deployment



Deprecated!
(use “Lockable Resources” plugin and its lock step
instead of concurrency parameter)
<https://jenkins.io/blog/2016/10/16/stage-lock-milestone/>

```
stage("Signoff") {  
    input("Deploy that to prod?")  
    milestone()  
}  
stage(name: "Deploy", concurrency: 1) { node {...} }
```

Docker

- Run build jobs within Docker containers
 - No need to install software on Jenkins master/slave
 - Use multiple versions of the same tool
- Containers can be existing ones or built on demand
- .. and Kubernetes

```
stage("run in docker") {  
    node {  
        withDockerContainer("php:7-fpm") {  
            sh "php -v"  
        }  
    }  
}
```

Snipped Editor & Docs

- Because first steps are hard..
- Auto-generated DSL documentation
(*Pipeline Syntax → Step Reference*)

Steps

Sample Step `writeFile: Write file to workspace`

File path in workspace `README.txt`

Text to write `Hello World`

Character encoding

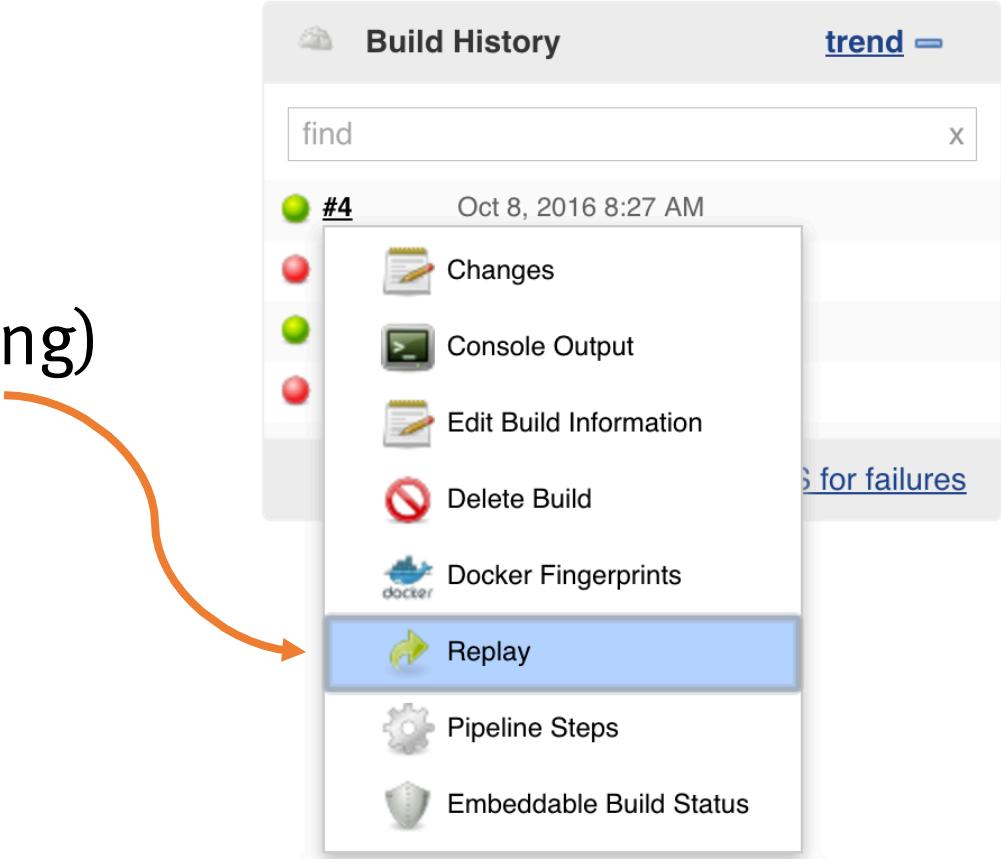
Generate Groovy

```
writeFile file: 'README.txt', text: 'Hello World'
```

Where to put that config?

Pipeline Configuration

- Paste code into job config (fine for testing)
- Create pipelines via JobDSL
- Commit it to your repo
 - File called `Jenkinsfile`
 - It evolves with the application
 - It is versioned
 - Everybody can read (and modify) it
 - You can throw away your Jenkins master at any time



Multibranch & Organization Folder Plugins

- Scans a complete GitHub/Bitbucket organisation for Jenkinsfile
- Triggered by Webhook and/or runs periodically
- Automatically adds pipeline jobs per repo/branch/PR

The screenshot shows the Jenkins dashboard for the organization "TYPO3's Chef Cookbooks". The left sidebar includes links for Up, Status, Re-scan Organization, People, Build History, and GitHub. The main content area displays a "TYPO3's Chef Cookbooks" folder with the folder name "TYPO3-cookbooks". Below it, a "Repositories" table lists five entries:

S	W	Repository ↓	Description
	⚡	backuppc	Chef cookbook for BackupPC
	⚡	backuppc » develop	
	⚡	gerrit	Chef cookbook for the Gerrit review system.
	⚡	gerrit » develop	
	⚡	gerrit » master	

At the bottom right of the dashboard, the number "28" is displayed.

DRY: Jenkins Global Library

- Provides shared functionality available for all jobs
- Stored on Jenkins master, available to all slaves
 - Available via Jenkins-integrated Git server
 - Can be loaded from remote Git repos, specified in global configuration and `Jenkinsfile`

```
@Library("https://github.com/..")  
node { deployToProd() }
```

Real-World Example

Jenkins Pipelines for Chef Cookbooks

Chef CI/CD at TYPO3.org

- Code that runs the *.typo3.org infrastructure, chef-ci.typo3.org
- Objective: Chef cookbooks
 - Server provisioning (installs packages, configures services)
 - Code: github.com/TYPO3-cookbooks



Many Cookbooks, Many Pipelines

- Scans our GitHub organization *TYPO3-cookbooks*
 - Automatically adds/removes pipelines for branches and pull requests*
 - Triggered via Webhooks
 - Contents of Jenkinsfile:

```
def pipe = new org.typo3.chefci.v1.Pipeline()  
pipe.execute()
```

* Currently suspicious to arbitrary code execution

Jenkins Global Library

- Pipelines implemented in Global Library

[TYPO3-infrastructure/jenkins-pipeline-global-library-chefci](#)

Branch: master ▾ [jenkins-pipeline-global-library-chefci](#) / [src](#) / [org](#) / [typo3](#) / [chefci](#) / [v1](#) /

 StephenKing [BUGFIX] Forgotten statement to delete tempfile	..
ArchiveArtifacts.groovy	Feels like I'm learning Groovy..
BerkshelfInstall.groovy	Don't use try/catch
BerkshelfUpload.groovy	Really upload cookbook
Lint.groovy	Feels like I'm learning Groovy..
Pipeline.groovy	Not clear to me, why it clones to workspace@script/
SlackPostBuild.groovy	Add SlackPostBuildAction
TidyUp.groovy	TRAVISIM1_E... - User-defined build step

Parallelize Integration Tests

- Run *Test-Kitchen* (integration test for Chef cookbooks)
- Run all instances in parallel (by Jenkins)

```
$ kitchen status
```

Instance	Driver	Provisioner	[..]	Last Action
default-debian-78	Docker	ChefZero		<Not Created>
default-debian-82	Docker	ChefZero		<Not Created>
physical-debian-78	Docker	ChefZero		<Not Created>
physical-debian-82	Docker	ChefZero		<Not Created>
production-debian-78	Docker	ChefZero		<Not Created>
production-debian-82	Docker	ChefZero		<Not Created>

Parallelize Integration Tests (2)

- Goal: Extract instance list, run kitchen commands in parallel
- Expected result:

```
parallel(  
  'default-debian-82': {  
    node {  
      unstash('cookbook-tk')  
      sh('kitchen test --destroy always default-debian-82')  
    }  
  },  
  'physical-debian-82': {  
    node {  
      unstash('cookbook-tk')  
      sh('kitchen test --destroy always physical-debian-82')  
    }...  
  }
```

Parallelize Integration Tests (3)

- Grab list of instance names

```
def ArrayList<String> getInstances(){
    def instanceNames = []
    node {
        def lines = sh(script: 'kitchen status', returnStdout: true).split('\n')
        for (int i = 1; i < lines.size(); i++) {
            instanceNames << lines[i].tokenize(' ')[0]
        }
    }
    return instanceNames
}
```

Parallelize Integration Tests (4)

```
def Closure getNodeForInstance(String instanceName) {
    return {
        // this node (one per instance) is later executed in parallel
        node {
            // restore workspace
            unstash('cookbook-tk')
            sh('kitchen test --destroy always ' + instanceName)
        }
    }
}

for (int i = 0; i < instanceNames.size(); i++) {
    plNodes[instanceName] = this.getNodeForInstance(instanceNames.get(i))
}
parallel plNodes
```

Failure Notification

- Pipeline stops, when any step fails
- But.. I want that info in Slack!

```
def execute() {  
    this.prepare()  
    this.run(new Lint())  
    this.run(new BerkshelfInstall())  
    this.run(new TestKitchen())  
    this.run(new ArchiveArtifacts())  
    // erm... git flow is a bad idea  
    if (env.BRANCH_NAME == "master") {  
        this.run(new BerkshelfUpload())  
    }  
}
```

```
def run(Object step){  
    try {  
        step.execute()  
    } catch (err) {  
        this.postBuildNotify  
        failTheBuild("Build failed")  
    }  
}
```



A new Jenkins UI

Blue Ocean & Other News

Blue Ocean

- Tailored to the pipeline plugins

Screenshot of the Blue Ocean dashboard showing a list of pipelines.

The dashboard has a blue header with tabs: Pipelines, Administration, Dashboard, and New Pipeline. The New Pipeline tab is highlighted with a white background and black border.

Name	Health	Branches	Pull Requ...
jenkins / chef-repo		-	-
jenkins / chef-repo-seed		-	-
jenkins / Pipeline Examples / Docker		-	-
jenkins / Pipeline Examples / Example		-	-
jenkins / Pipeline Examples / Hello World		-	-
jenkins / Pipeline Examples / Manual Deploy		-	-
jenkins / Pipeline Examples / Parallel Docker		-	-

Blue Ocean

- Tailored to the pipeline plugins

jenkins / Pipeline Examples / Parallel Docker #3

Branch Parallel Docker
No changes

a few seconds
a few seconds ago

Pipeline Changes Tests Artifacts

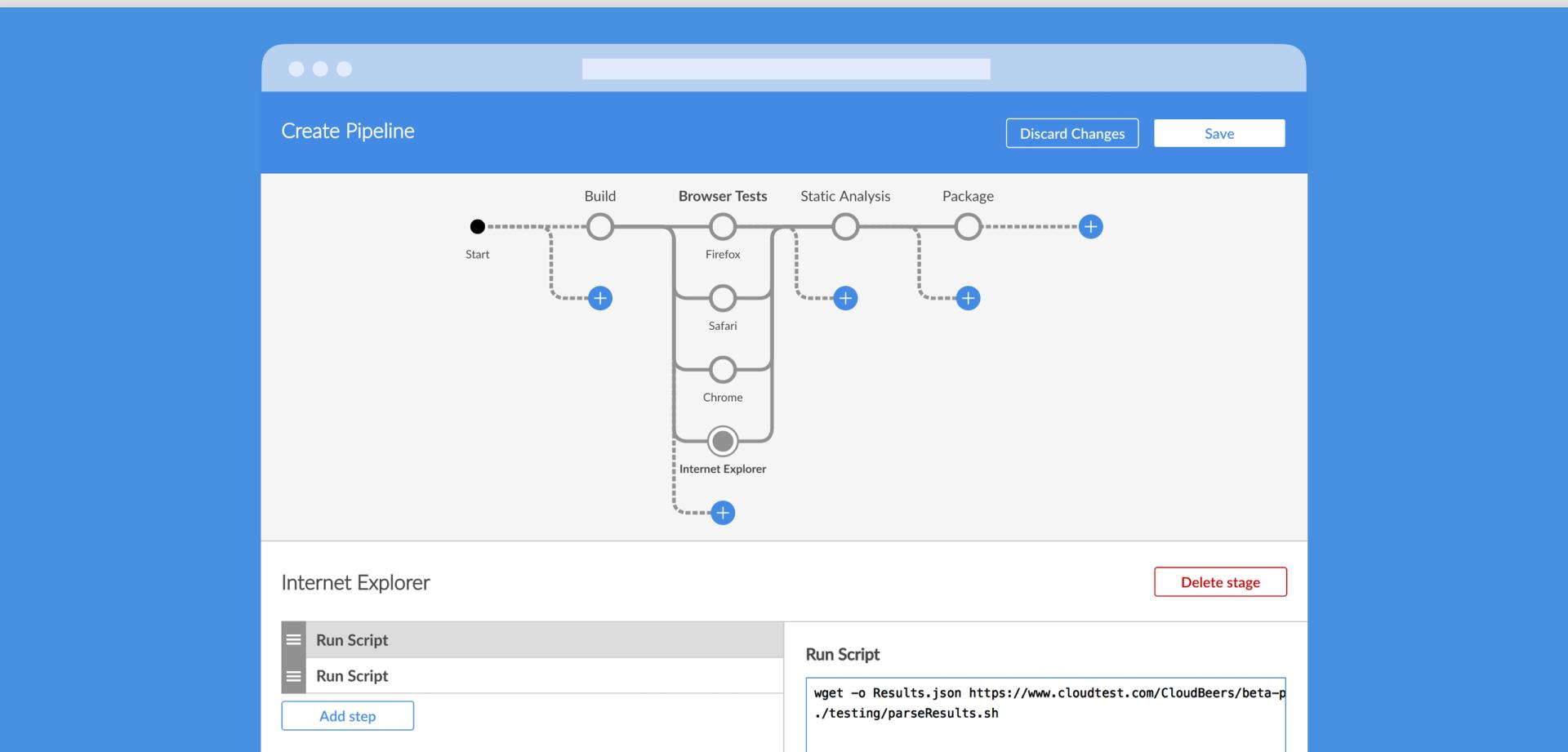
```
graph LR; whatever --> runInDocker[run in docker]; runInDocker --> deploy[deploy]; subgraph runInDocker [run in docker]; php7 --> php5[php5]; end
```

Steps - deploy

✓ > Print Message

Pipeline Editor*

* Recently announced, not yet available



Declarative Pipelines

- Sorry, the code I've shown might be already outdated.. ☺
- Declarative approach solves some issues (failure handling, post-build steps)

```
pipeline {  
    agent docker:'node:6.3'  
    stages {  
        stage('build') {  
            sh '...'  
        }  
        stage ('test') {  
            sh 'npm test'  
        }  
    }  
    postBuild {  
        always {  
            sh 'echo "This will always run"'  
        }  
        failure {  
            sh 'echo "This will run only if failed"'  
        }  
    }  
}
```

Summary

- Jenkins pipeline suite modernizes its CI/CD capabilities
 - CloudBees, Inc. is very actively pushing development
 - Many Jenkins plugins already compatible
- Pipeline defined as code
 - Versioned
 - Doesn't mess up Jenkins jobs
 - Code sharing
- Automated pipeline creation based on GitHub/Bitbucket APIs
- Blue Ocean refreshes Jenkins' UI
- Still couple of rough edges (failure handling, frequent changes)

Further Reading

- Pipeline Tutorial:
<https://github.com/jenkinsci/pipeline-plugin/blob/master/TUTORIAL.md>
- Getting started with pipelines:
<https://jenkins.io/pipeline/getting-started-pipelines/>
- Jenkins World 2016 Wrap Up - Pipelines:
<https://jenkins.io/blog/2016/09/24/jenkins-world-2016-wrap-up-pipeline/>
- Step documentation:
<https://jenkins.io/doc/pipeline/steps/>
- Pipeline global library:
<https://github.com/jenkinsci/workflow-cps-global-lib-plugin/blob/master/README.md>
- Docker in Jenkins pipelines:
<https://jenkins.io/blog/2016/08/08/docker-pipeline-environments/>
- Notifications (Mail, Slack, etc.):
<https://jenkins.io/blog/2016/07/18/pipline-notifications/>
- Parallel execution:
<https://jenkins.io/blog/2016/06/16/parallel-test-executor-plugin/>
- Extending pipeline DSL:
<https://jenkins.io/blog/2016/04/21/dsl-plugins/>
- Controlling the Flow with Stage, Lock, and Milestone:
<https://jenkins.io/blog/2016/10/16/stage-lock-milestone/>
- TYPO3's Chef CI:
<https://chef-ci.typo3.org>

*Slides will be available at
st-g.de/speaking*

DevOps Würzburg Mainfranken

Startseite Mitglieder Fotos Diskussionen Mehr

Mitglied werden!



Würzburg,
Deutschland
Gegründet 26. Apr 2015

Über uns...

DevOps 74

Anstehende
Meetups 1

Unser Kalender



Organisatoren:

Featured Meetup

Erstes DevOps Meetup in Würzburg

Montag, 7. November 2016
19:00

Uni Würzburg, Informatikgebäude
Am Hubland, Würzburg ([Karte](#))

Herzlich Willkommen beim ersten DevOps-Meetup in Unterfranken. Um alle Teilnehmer bei ihrem aktuellen Wissensstand abzuholen, möchten wir das erste Meetup mit zwei kurzen, einführenden Vorträgen beginnen. Hierdurch hoffen wir, ein breites Publikum vom Studenten bishin zum DevOps-Guru zu erreichen.

Vortrag 1 (Lenz Weber, mayflower)
DevOps - neue Antworten auf neue Herausforderungen

Vortrag 2 (Steffen Gebert, Uni Würzburg)
Continuous Delivery - Mit dem Konzept "Continuous Delivery"

Nimmst Du
teil?

Beitreten und
RSVPen

14 nehmen teil



Steffen G.
Co-Organisator,
Event-Koordinator



Andreas R.
Co-Organisator,
Event-Koordinator



Sebastian K.
Technical Evangelist at Microsoft.
I'm on Twitter: http://twitter.com/sebastian_k