# Building Pipelines at Scale

Julien Pivotto
inuits.eu

25th June 2014

#jenkinsconf

## Building Pipelines at Scale

## Julien Pivotto

- Open-Source Consultant at inuits.eu
- Linux Systems Administrator
- FOSS Defender for many years
- ♡ DevOps
- Puppet user since 2011
- Doing also some Ruby & Python

## inuits

- Open-Source Consultancy company
- 50+ people in 4 countries (.be, .ua, .nl & .cz)
- Doing both Development & Systems Administration
- A lot of domains: Linux, Web, Databases, Monitoring...
- Early DevOps practitioners

## MediaMosa

- Drupal-based Digital Asset Management system
- Open-source
- Store assets
- Transcode videos
- Create, extract and manage metadata using open standards: Dublin Core, Qualified DC, IEEE/LOM, CZP
- Webservice oriented

## MEDIΛSΛLSΛ: MediaMosa As a Service

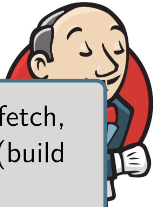
- A turn-key MediaMosa solution
- A few environments:
  - Dev/Uat/Prod environments
  - Customer-specific environments
- One backend per environment
- Multiple frontends per backend

= A lot of Drupal sites

## Definition of pipelines

A **pipeline** is a chain of Jenkins jobs that are run to fetch, compile, package, run tests and deploy an application (build pipeline plugin).

## Definition of pipelines

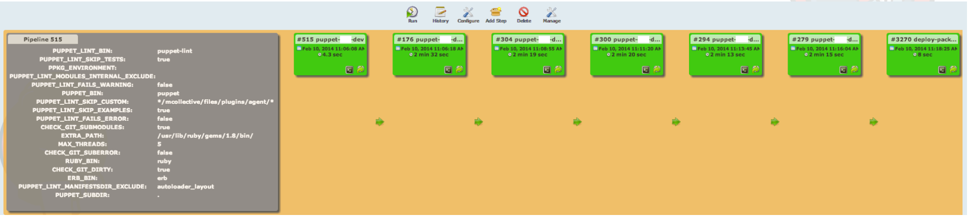


A **pipeline** is a chain of Jenkins jobs that are run to fetch, compile, package, run tests and deploy an application (build pipeline plugin).

Pipeline *vs* large (or independent) jobs:

- Separation between different tasks
- Overview of tests, etc...
- Deployment depends on testing

## Example: a puppet pipeline



Pipeline 515

|  |  |
|---|---|
| PUPPET_LINT_BIN: | puppet-lint |
| PUPPET_LINT_SKIP_TESTS: | true |
| PPKG_ENVIRONMENT: |  |
| PUPPET_LINT_MODULES_INTERNAL_EXCLUDE: |  |
| PUPPET_LINT_FAILS_WARNING: | false |
| PUPPET_BIN: | puppet |
| PUPPET_LINT_SKIP_CUSTOM: | */mcollective/files/plugins/agent/* |
| PUPPET_LINT_SKIP_EXAMPLES: | true |
| PUPPET_LINT_FAILS_ERROR: | false |
| CHECK_GIT_SUBMODULES: | true |
| EXTRA_PATH: | /usr/lib/ruby/gems/1.8/bin/ |
| MAX_THREADS: | 5 |
| CHECK_GIT_SUBERROR: | false |
| RUBY_BIN: | ruby |
| CHECK_GIT_DIRTY: | true |
| ERB_BIN: | erb |
| PUPPET_LINT_MANIFESTSDIR_EXCLUDE: | autoloader_layout |
| PUPPET_SUBDIR: | . |

#515 puppet-[redacted]-dev
Feb 10, 2014 11:06:08 AM
4.3 sec

#176 puppet-[redacted]-d...
Feb 10, 2014 11:06:18 AM
2 min 32 sec

#304 puppet-[redacted]-d...
Feb 10, 2014 11:08:55 AM
2 min 19 sec

#300 puppet-[redacted]-d...
Feb 10, 2014 11:11:20 AM
2 min 20 sec

#294 puppet-[redacted]-d...
Feb 10, 2014 11:13:45 AM
2 min 13 sec

#279 puppet-[redacted]-d...
Feb 10, 2014 11:16:04 AM
2 min 15 sec

#3270 deploy-pack...
Feb 10, 2014 11:18:25 AM
8 sec

## Inside the pipeline

- Checkout
- Style
- Syntax
- Compile
- Unit tests
- Integration tests
- Packaging
- Regression tests
- Deployment / Delivery
- ...

## Ok, let's use pipelines!

- Let's make a pipeline for our puppet code

## Ok, let's use pipelines!

- Let's make a pipeline for our puppet code
- ...and one for our puppet code in dev environment
- ...and one for our puppet code in uat environment

## Ok, let's use pipelines!

- Let's make a pipeline for our puppet code
- ...and one for our puppet code in dev environment
- ...and one for our puppet code in uat environment
- Let's make a pipeline for backend app

## Ok, let's use pipelines!

- Let's make a pipeline for our puppet code
- ...and one for our puppet code in dev environment
- ...and one for our puppet code in uat environment
- Let's make a pipeline for backend app
- ...and one for backend app in dev environment
- ...and one for backend app in uat environment

## Ok, let's use pipelines!

- Let's make a pipeline for our puppet code
- ...and one for our puppet code in dev environment
- ...and one for our puppet code in uat environment
- Let's make a pipeline for backend app
- ...and one for backend app in dev environment
- ...and one for backend app in uat environment
- Let's make 5 pipelines for frontend apps
- ...and 5 for frontends apps in dev environment
- ...and 5 for frontends apps in uat environment

## How to scale?

- First attempt: clone jobs in the ui
  - Dozens of jobs to clone & edit
  - Human intervention = mistakes
- Second attempt: clone jobs xml files
  - grep & cp & sed...
  - Crappy bash scripting is crappy
- Mmmh...we need something else...

## Let's change the pipelines now!

## Let’s change the pipelines now!

click

## Let’s change the pipelines now!

click click

## Let's change the pipelines now!

click click click click click

## Let’s change the pipelines now!

click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click click

## Let's change the pipelines now!

click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click click click click click click click click click click click
click click click

## Puppet

## Infrastructure as Code

Puppet/Chef/Cfengine/...same fight:

- Modelize your infrastructure
- Reproducable platforms
- Fast and reliable
- Disaster recovery for "free"

## What to puppetize

Usually you puppetize:

- OS installation & configuration
- Apps installation & configuration
- DB installation & configuration

## What to puppetize

Usually you puppetize:

- OS installation & configuration
- Apps installation & configuration
- DB installation & configuration

But you do not automate:

- Application data

Are Jenkins jobs to be puppetized?

## What to puppetize

Usually you puppetize:

- OS installation & configuration
- Apps installation & configuration
- DB installation & configuration

But you do not automate:

- Application data

Are Jenkins jobs to be puppetized?
...we tried...

## Modules structure

- Using the Jenkins upstream module
- And creating a specific *jenkinsjobs* module
  - Difficult to share
  - Mainly templates

## Structure of the module

- manifests
  - job.pp (definition; file)
  - packages.pp (packages needed by tests)
  - dashboard.pp (augeas xml to modify jenkins config)
  - pipeline/frontend.pp (definition; jobs)
  - pipeline/backend.pp (definition; jobs)
  - service.pp (exec to reload configuration)

## `job.pp`

- Takes a template as parameter
- Creates xml file and reload jenkins configuration
- It has a lot more arguments (project name, next job, etc...)
- Adds the job to the view (augeas)

## `service.pp`

- Reload the Jenkins service

## service.pp

- Reload the Jenkins service

```
java -jar /var/cache/jenkins/war/WEB-INF/jenkins-cli.jar
-s    http://127.0.0.1:8080/    reload-configuration
--username "${username}" --password "${password}"
```

- It makes Jenkins unavailable

## pipeline.pp

## pipeline.pp

### Defaults for jobs

```
Jenkinsjobs::Job{
  ensure         => $ensure,
  git_repository => $git_repository,
  customer       => $customer,
  assigned_node  => $debian_slave,
  dashboard_view => $dashboard_view,
  package_name   => "frontend-${customer}",
  vhost_docroot  => $_vhost_docroot,
  custom_modules => 'sites/all/modules/custom',
  job_type       => 'drupal',
}
```

## pipeline.pp

### Defaults for promotions

```
Jenkinsjobs::Promotion{
  ensure          => $ensure,
  customer        => $customer,
  assigned_node   => $debian_slave,
  packaging_job  => "frontend-build-${customer}",
  package_name    => "frontend-${customer}",
  vhost_docroot   => $_vhost_docroot,
  job_name       => "frontend-promo-${customer}",
}
```

## pipeline.pp

### Jobs definitions

```
jenkinsjobs::job{
  "frontend-checkout-${customer}":
    job_template   => 'jj/jobs/checkout.erb',
   next_job     => "frontend-syntax-${customer}",
    start_pipeline => true,
}

jenkinsjobs::job{
  "frontend-syntax-${customer}":
    job_template => 'jj/jobs/syntax.erb',
    next_job     => "frontend-style-${customer}",
}
```

## pipeline.pp

### Job with promotions

```
jenkinsjobs::job{
  "frontend-promo-${customer}":
    job_template => 'jj/jobs/promote.erb',
    promotions   => [
      "deploy-${customer}-front-to-uat",
      "deploy-${customer}-front-to-prod",
      "deploy-${customer}-front-to-com-prod",
    ],
}
```

## pipeline.pp

### Promotions

```
if 'uat' in $targets {
  jenkinsjobs::promotion {
    "deploy-${customer}-front-to-uat":
      template     => 'jj/promotion/uat.erb',
      debian_repo  => 'mediamosa-uat',
      targets      => $targets['viaa-uat'],
  }
}
```

## dashboard.pp

### Augeas example

```
  augeas {
    "dashboard view $title":
      lens    => 'Xml.lns',
      incl    => '/var/lib/jenkins/config.xml',
      context => '/files/var/lib/jenkins/config.xml',
      changes => [
        "set ${augeas_prefix}[last()+1]/owner/#attribute/class \"hudson\"",
        "set ${augeas_prefix}[last()]/owner/#attribute/reference \"../../..\"",
        "set ${augeas_prefix}[last()]/includeStdJobList/#text \"true\"",
        "set ${augeas_prefix}[last()]/name/#text \"${title}\"",
      ],
      onlyif  =>
"match ${file_prefix}/hudson/views/*/name/#text[.=\"${title}\"] size == 0",
      notify  => Exec['reload-jenkins'],
  }
```

## Advantages of setting pipelines in Puppet

- Easy to deploy new pipelines

## Advantages of setting pipelines in Puppet

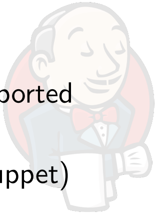
- Easy to deploy new pipelines
- Deploy pipelines in the same time as applications (exported resources)
- Define pipelines at the same place as applications (puppet)

## Advantages of setting pipelines in Puppet

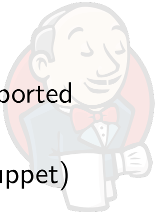
- Easy to deploy new pipelines
- Deploy pipelines in the same time as applications (exported resources)
- Define pipelines at the same place as applications (puppet)
- Track history of pipeline changes

## Advantages of setting pipelines in Puppet

- Easy to deploy new pipelines
- Deploy pipelines in the same time as applications (exported resources)
- Define pipelines at the same place as applications (puppet)
- Track history of pipeline changes
- Having separate jobs for each project
- Reuse puppet template functionality

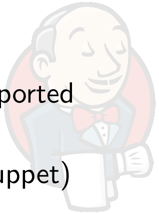


## Advantages of setting pipelines in Puppet



- Easy to deploy new pipelines
- Deploy pipelines in the same time as applications (exported resources)
- Define pipelines at the same place as applications (puppet)
- Track history of pipeline changes
- Having separate jobs for each project
- Reuse puppet template functionality
- Deploy new Jenkins server in minutes with all jobs
- Keep several Jenkins (test, prod) in sync

## Advantages of setting pipelines in Puppet



- Easy to deploy new pipelines
- Deploy pipelines in the same time as applications (exported resources)
- Define pipelines at the same place as applications (puppet)
- Track history of pipeline changes
- Having separate jobs for each project
- Reuse puppet template functionality
- Deploy new Jenkins server in minutes with all jobs
- Keep several Jenkins (test, prod) in sync
- Add ssh keys, git config, etc...in Puppet too
- Create test databases & vhosts in Puppet (exported resources)

## Disadvantages

- Play with Augeas & xml (wrong feeling)
- Makes Jenkins unavailable on changes
- Every small change ends up in Puppet

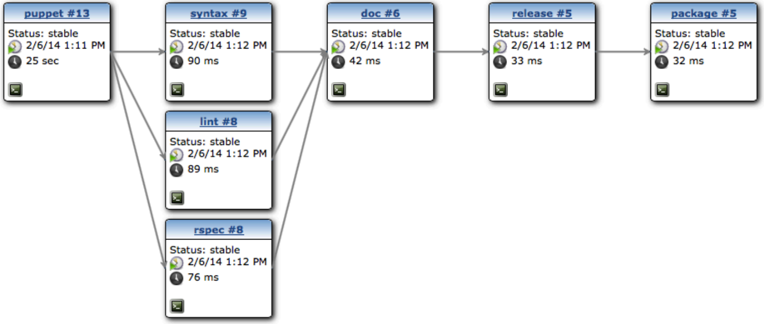
## Plugins

## A lighter approach: Jenkins plugins

- Remove the Puppet complexity
- Still having Pipelines in sync
- Minimize human intervention
- Speed-up the pipelines
- Do not restart Jenkins each time

## Plugin #1: Build flow

- Creates Pipelines
- Uses Groovy scripts
- Can run jobs in parallel
- Several other plugins do the same



puppet #13
Status: stable
2/6/14 1:11 PM
25 sec

syntax #9
Status: stable
2/6/14 1:12 PM
90 ms

lint #8
Status: stable
2/6/14 1:12 PM
89 ms

rspec #8
Status: stable
2/6/14 1:12 PM
76 ms

doc #6
Status: stable
2/6/14 1:12 PM
42 ms

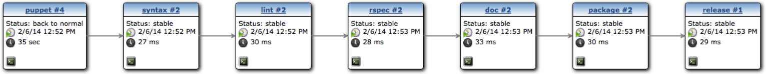
release #5
Status: stable
2/6/14 1:12 PM
33 ms

package #5
Status: stable
2/6/14 1:12 PM
32 ms

## Build Flow: Groovy scripts

### Pipelines

```
build ( "syntax" )
build ( "lint" )
build ( "rspec" )
build ( "doc" )
build ( "package" )
build ( "release" )
```



puppet #4
Status: back to normal
2/6/14 12:52 PM
35 sec

syntax #2
Status: stable
2/6/14 12:52 PM
27 ms

lint #2
Status: stable
2/6/14 12:52 PM
30 ms

rspec #2
Status: stable
2/6/14 12:53 PM
28 ms

doc #2
Status: stable
2/6/14 12:53 PM
33 ms

package #2
Status: stable
2/6/14 12:53 PM
30 ms

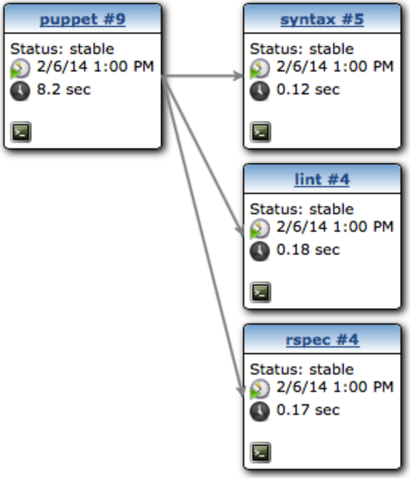
release #1
Status: stable
2/6/14 12:53 PM
29 ms

## Build Flow: Groovy scripts

### Parallel jobs

```
parallel (
    {
        build ( "syntax" )
    },
    {
        build ( "lint" )
    },
    {
        build ( "rspec" )
    },
)
```

## Build Flow: Groovy scripts

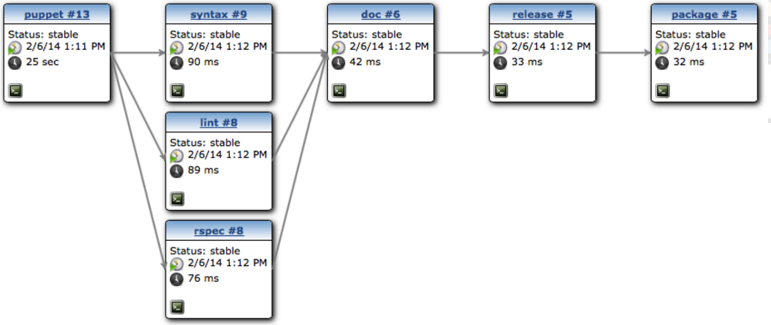


## Build Flow: Groovy scripts

### Conditionals

```
if ( params["RELEASE"] == "true" ) {
    switch ( params["DEPLOY_METHOD"] ) {
        case "librarian":
            build ( "librarian" )
        case "r10k":
            build ( "r10k" )
        case "package":
            build ( "package" )
    }
}
```

## Build Flow: Groovy scripts

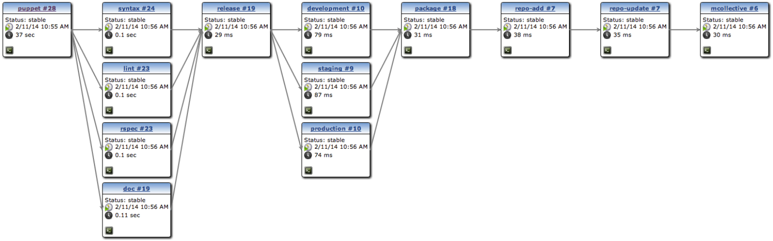


## Build Flow: Groovy scripts

### Build parameters

```
build ( "gitsubmodules",
    NAME: params["NAME"],
    ENVIRONMENT: params["ENVIRONMENT"]
)
```

## Build Flow: Groovy scripts



## But also...

- A "retry" function to try the failing jobs 2, 3, 4 time
- A "rescue" function to launch a jobs if another fails
- The first job fails if one of the pipeline jobs fails!
- And a lot of other stuffs

## Plugin #2: Job Generator

- Jenkins jobs generator
- Creates jobs on the fly
- Jobs do not need to be created at the start of the (flow) pipeline
- Also updates existing jobs
- Allows to keep jobs per project (per project history and graphes)
- Takes special parameters

## Build Flow Plugin + Job generator

- One starting job (orchestrator)
- Calls some jobs or some generators, in parallel
- Ends with a nice, up-to-date Pipeline

## Plugin #3: JobConfigHistory Plugin

- Keeps track of the changes in the jobs
- Alternative to keep the xml in git

## Conclusion

## Other bottlenecks/problems at scale

- Load of the Jenkins server

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves
- SCM pulling

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves
- SCM pulling: Think Gerrit or Trigger

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves
- SCM pulling: Think Gerrit or Trigger
- Jobs are slow

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves
- SCM pulling: Think Gerrit or Trigger
- Jobs are slow: Optimize, Divide them, Monitor them (graphite)
- Notifications

## Other bottlenecks/problems at scale

- Load of the Jenkins server: Think slaves
- SCM pulling: Think Gerrit or Trigger
- Jobs are slow: Optimize, Divide them, Monitor them (graphite)
- Notifications: Think XMPP, IRC, mails, Gerrit...

## Pipelines at scale...

- Two different approaches
  - application-level
  - infrastructure-level
- Pick the one you prefer (or invent your own)
- Do not hesitate to separate Jobs
- Monitor your jobs/pipelines (graphite)

**Thank you!**

Thanks to my colleague @tomdevylder for the Build Flow slides

## Contact

Julien Pivotto
julien@inuits.eu
@roidelapluie

Inuits BVBA
Belgium
+32 473 441 636
https://inuits.eu

## Thank You To Our Sponsors

### Platinum

CloudBees

### Gold

zend The PHP Company

MidVision™ Release the innovation

codecentric

### Silver

XebiaLabs the deployment automation company

SOASTA Test Faster. Release Sooner.

### Corporate

cloudera®

### Community

Java User Group Berlin Brandenburg

Lightweight Java User Group Munich