Job scheduling @Helpshift with Jenkins

RootConf, 2018; Bangalore
Vineet Naik
@naiquevin

About this talk

What?

Target Audience

Overview

How we built a distributed job scheduling platform

Leveraging Jenkins and it’s plugin ecosystem

To solve the problems with our earlier job scheduling approach

About this talk

What?

Target Audience

Overview

A general understanding of,

- Batch jobs
- Master-slave architecture
- Domain specific languages (DSL)

About this talk

What?

Target Audience

Overview

- Our use cases
- Old approach & its problems
- Problem statement
- Why Jenkins?
- New, Jenkins based approach
    - Arch & Implementation
    - Benefits
    - Known issues
    - Future plans

Our use cases

Batch jobs

Semi-automated workflows

Background tasks eg. data crunching & aggregation, backups, cleanups etc.

Periodically scheduled eg. every 15 mins, every 4 hours, once a day, once a week..

Jobs to run semi-automated workflows on demand

Old approach

Quartzite scheduler

Problems

Disclaimer

Jobs (mainly) written in Clojure

Quartzite, a Clojure wrapper for **Quartz** library in Java

Jar is deployed on a node

Long running process

- ➔ Scheduler initialized at startup
- ➔ Jobs scheduled in separate threads

http://clojurequartz.info/
http://www.quartz-scheduler.org/

Old approach

Quartzite scheduler

Problems

Disclaimer

Release requires a restart

Single process running scheduler and jobs

During release, process is restarted

- ➔ Interruption of in-progress jobs
- ➔ Chance of jobs getting skipped during restart window

Impact: Possibility of SLA breach

Old approach

Quartzite scheduler

Problems

Disclaimer

Overshooting jobs

Job duration > Frequency

Job #	Start time	Duration	Comments
#1	10:30 am	20 mins	
#2	11:00 am	27 mins	
#3	11:30 am	34 mins	
~~#4~~	~~12:00 pm~~	-	👎 Skipped
#5	12:30 pm	...	

Impact: High chance of SLA breach

Old approach

Quartzite scheduler

Problems

Disclaimer

Other problems

- Continuously running processes
- Cannot scale horizontally
- No on-demand job execution
- Lack of visibility
  - Currently running jobs, job history, upcoming jobs etc.
- Interspersed logs
- Only specific to Clojure/Java

And so on..

Old approach

Quartzite scheduler

Problems

Disclaimer

Quartz(ite) is not the problem; It’s the approach

It’s a sufficiently advanced scheduler

Can be configured and extended to solve some of the problems

But, still Jenkins makes a better platform (more later)

Problem Stmt.

What were we looking for?

Prevent SLA breaches

Distributed execution of jobs; Horizontal scalability

Job Pipelines

UI for running jobs on-demand

Common functionality provided

Easy to write & onboard jobs

Not just limited to Clojure/Java

Why Jenkins?

Automation Platform

Our prior experience

Our philosophy

- Generic automation platform
- Much more than just CD/CI
- Built-in job scheduler
- Active community
- Matured plugin ecosystem

Why Jenkins?

Automation Platform

Our prior experience

Our philosophy

Already running another Jenkins cluster for CD/CI

> 500 jobs

~ 20 slaves

~ 4 years

Why Jenkins?

Automation Platform

Our prior experience

Our philosophy

Invest → Reuse → Standardize

Build on top of existing work

Ship faster

New approach

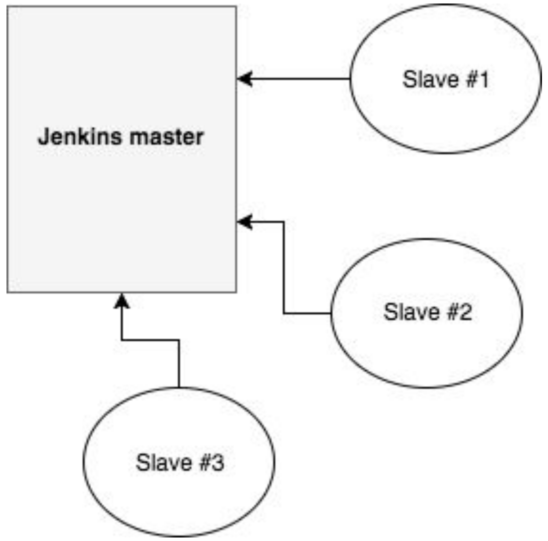
Jenkins

Job wrapper

Code

Job definitions as code

Release Integration



Jenkins cluster running in **master-slave** configuration

New approach

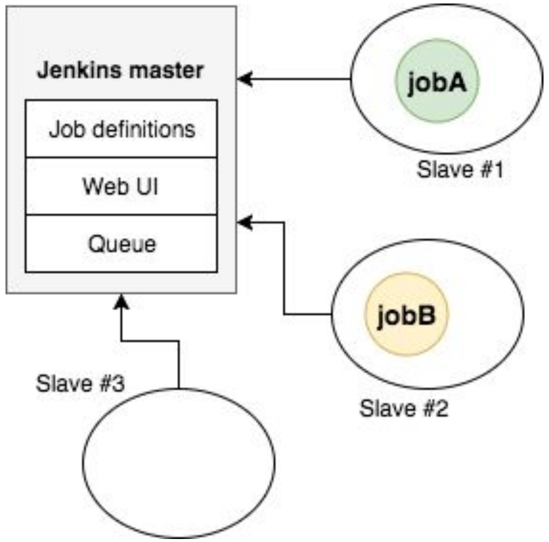
Jenkins

Job wrapper

Code

Job definitions as code

Release Integration



Master: stores job definitions, schedules jobs; provides web UI
Slaves: connect to master; run jobs

New approach

Jenkins

Job wrapper

Code

Job definitions as code

Release Integration

Python script, installed on slaves

Layer between scheduler & the code written by devs where we can plug-in common functionality

Provides retries, timeouts, monitoring

Does all the reusable heavy lifting so that jobs can focus on business logic

Owned by the OPS team

New approach

Jenkins

Job wrapper

Code

Job definitions as code

Release Integration

Code that encapsulates business logic to process the task

Can be written in any language

Should run like command line script, exiting with the correct code

zero for success; *non-zero* for failure

Owned by developers

New approach

Jenkins

Job wrapper

Code

Job definitions as code

Release Integration

Written using groovy based Pipeline DSL (more on it later)

Checked into the git repo along with source code

Owned by developers

New approach

Jenkins

Job wrapper

Code

Job definitions as code

Release Integration

Build: package source code + groovy scripts into an artifact (tarball)

Pre-deploy: Prepare nodes to join master as slaves

Deploy: copy the artifact to nodes

Post-deploy: Trigger a special job called “seed job” on master that translates DSL scripts into jenkins jobs

Jenkins Plugins

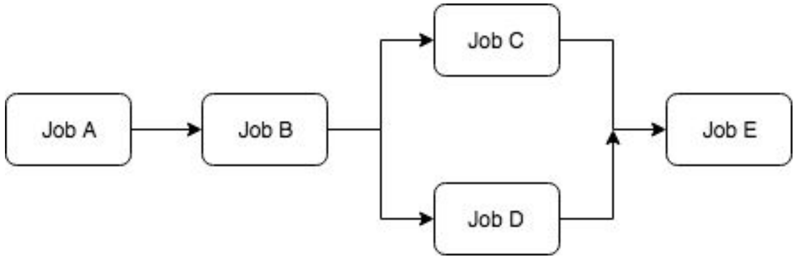
Pipelines + DSL

Job DSL

Jenkins Swarm Slaves

Metrics

Multi stage jobs



Jobs can run on different slaves, written in any language by different teams

https://jenkins.io/doc/book/pipeline/

Jenkins Plugins

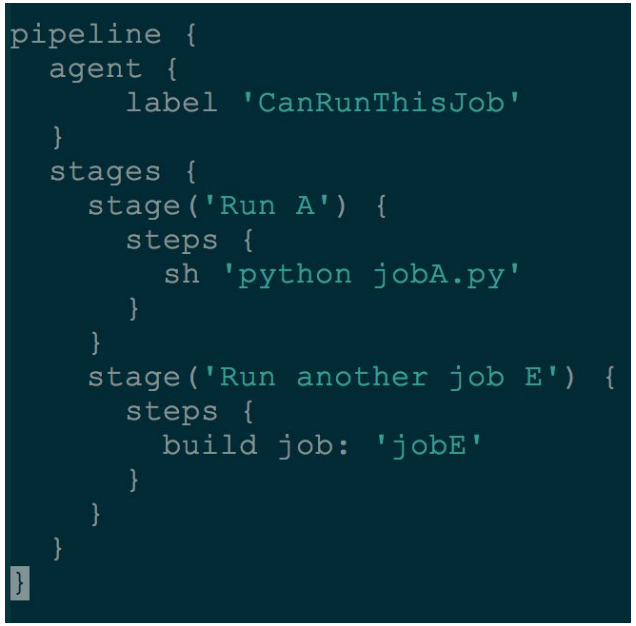
Pipelines + DSL

Job DSL

Jenkins Swarm Slaves

Metrics

Groovy based DSL to define jobs



```
pipeline {
  agent {
      label 'CanRunThisJob'
  }
  stages {
    stage('Run A') {
      steps {
        sh 'python jobA.py'
      }
    }
    stage('Run another job E') {
      steps {
        build job: 'jobE'
      }
    }
  }
}
```

https://jenkins.io/doc/book/pipeline/

Jenkins Plugins

Pipelines + DSL

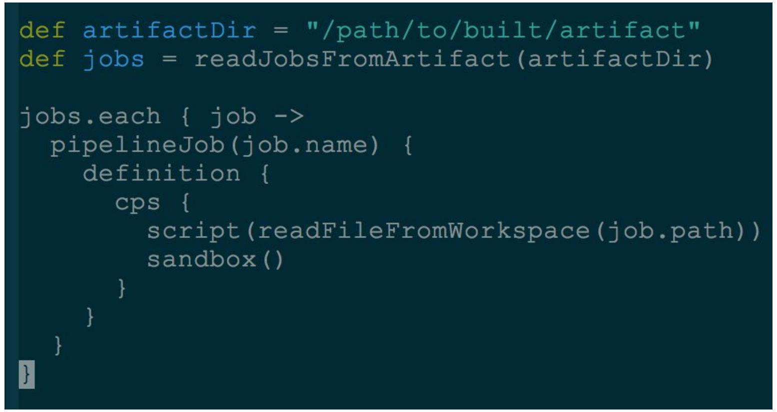
Job DSL

Jenkins Swarm Slaves

Metrics

Seed jobs

Job that creates other jobs
Groovy based DSL to describe jobs



```
def artifactDir = "/path/to/built/artifact"
def jobs = readJobsFromArtifact(artifactDir)

jobs.each { job ->
  pipelineJob(job.name) {
    definition {
      cps {
        script(readFileFromWorkspace(job.path))
        sandbox()
      }
    }
  }
}
```

https://plugins.jenkins.io/job-dsl

Jenkins Plugins

Pipelines + DSL

Job DSL

Jenkins Swarm Slaves

Metrics

Distributedness & Auto-scaling

Slaves initiate connection to master

Master doesn’t need to know about slaves in advance

Easier to auto-scale

Helps in Jenkins master HA (more later)

https://plugins.jenkins.io/swarm

Jenkins Plugins

Pipelines + DSL

Job DSL

Jenkins Swarm Slaves

Metrics

Monitoring

Provides Dropwizard metrics API

Contracts for health checks

API consumed by a sensu plugin that emits alerts

https://plugins.jenkins.io/metrics
http://metrics.dropwizard.io/4.0.0/

Benefits

Releases don't affect jobs

Overshooting jobs queued

Horizontally scalable

Other

Each job runs in a separate process

No restart needed

Creation/updation of jobs happens on master and is independent of the in-progress jobs running on slaves

Impact: No SLA breaches during releases

Benefits

Releases don't affect jobs

Overshooting jobs queued

Horizontally scalable

Other

Overshooting jobs are queued on master until they can be started

Job #	Start time	Duration	Comments
#1	10:30 am	20 mins	
#2	11:00 am	27 mins	
#3	11:30 am	34 mins	
#4	12:04 pm	...	👍 Queued
#5	...	...	

Impact: Reduced SLA breaches

Benefits

Releases don’t affect jobs

Overshooting jobs queued

Horizontally scalable

Other

Jobs are **distributed** across slaves

Swarm Slaves make **auto-scaling** possible

https://plugins.jenkins.io/swarm

Benefits

Releases don’t affect jobs

Overshooting jobs queued

Horizontally scalable

Other

Web UI to run jobs on-demand

Common functionality provided by the platform

Easy to write and onboard jobs

Better visibility

Better logs

RESTful API, ACL etc. for free

In Production

Current Status

High availability

Monitoring

Running in production for a few months

32 Jobs

13 slaves

>15k job runs so far

On average per day running time of ~100 hours

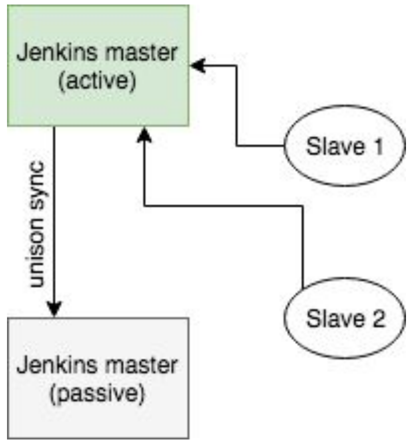
We’ve named this project *Igor*

In Production

Current Status

High availability

Monitoring



Active/Passive setup

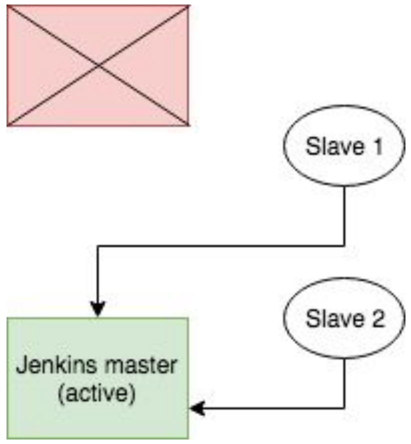
Passive node is hot standby - continuously syncs files from active using a tool called *unison*

In Production

Current Status

High availability

Monitoring



If active goes down, switch-over

Swarm slaves will reconnect to new (active) master by re-resolving DNS

In Production

Current Status

High availability

Monitoring

Jobs: Wrapper script sends alerts based on exit code and metrics such as job duration

Master: Process checks + health checks exposed by metrics plugin

Slaves: Process and health checks for swarm client process

Known issues

HA for master is not *real* HA

At present active/passive switch-over is manual

Auto-scaling not implemented yet

Limited to use cases where the load is predictable

Future plans

Better HA with automated switch-over

Auto-scaling of swarm slaves

State passing between pipeline stages

(May be) Rewrite the python wrapper script in Java and package it as a Jenkins plugin

Thank You!

@naiquevin

https://naiquevin.github.io

https://www.helpshift.com

https://engineering.helpshift.com

Questions?