Jenkins: To infinity and beyond the small team

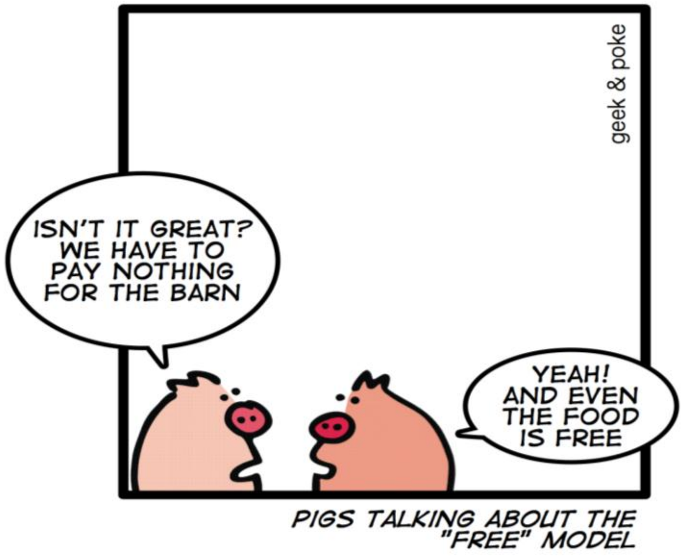
James Nord
Cisco Systems, Inc
http://www.cisco.com/

@jenkinsconf

Our assumptions and philosophy

- Reproducible builds are king
  - Infrastructure should not change the build behaviour
  - Standardized build (Maven / base pom)
- CI is vital – as it is used to make releases
- No full time "build guy"
- Trust users not to be malicious
- Don't trust users not to do daft things
  - or read documentation, or to have well behaved unit tests
- We have multiple branches of the same product under development at once (for support, feature branches)
- Multi region (developers and SCM location)

Hardware isn't free


geek & poke
ISN'T IT GREAT? WE HAVE TO PAY NOTHING FOR THE BARN
YEAH! AND EVEN THE FOOD IS FREE
PIGS TALKING ABOUT THE "FREE" MODEL

"Just use more hardware – it's cheap

Kohsuke Kawaguchi – Jerusalem, Israel. October 2010

image © (CC by 3.0) source http://geek-and-poke.com/geekandpoke/2010/12/21/the-free-model.html

Jenkins setup evolution

early 2007

1 Team
1 Old server
1 Hudson master (tomcat windows)
1 Project, 6 jobs

2008

More teams
More servers
More Hudson masters
more projects/jobs

2009

Divisional adoption & consolidation
1 master per region (SCM location)
standalone winstone

2010

Hudson / VMware integration

early 2011

Hardware performance issues
(new hardware)

2011

Jenkins!
Job Templates

2012

Job Templates go live!

2013

Performance
Automated Acceptance Tests
Build Flow

Take 1 (2009)

- Solaris NFS server (6TB zfs) as datastore
- 3 x 1RU Server
  - dual Intel E5420 Xeon
  - shared with other VMs
- Initially worked very well
- After 2 years with more jobs started to strain
- jobs took ~30% longer than on dev box
- master slow to respond
- general VMs became slower impacting unrelated services

Realization

- Storage (NFS) was limitation.
- Investigated many solutions many rejected as £££££
- Jenkins has a need for different types of storage

1. Large (Cheap)
   1. Build artifacts
2. Fast
   1. Build workspaces
   2. Job configuration
   3. Build Reports
3. Redundant (backed up HA etc)
   1. Job configuration
   2. slave templates
   3. Build Reports

Take 2 (2011)

- NetApp as redundant datastore (but not for master or build slaves!)

  Doesn't need to be uber fast – just reasonable and backed up.

  Used existing corporate Filer – purchased a new set of disks (also used for other VMs on different servers)

- 2 * 2RU Server – for master and slave VMs

  2 x Intel X5690 for slaves and masters)
  144 GB RAM
  16 x 300GB SAS disks
  1GB FBWC on controller

- 1 * 2RU Server – for acceptance test VMs
  - 2 x Intel E5649
    144 GB RAM
    2 x 300GB SAS disks (for ESXi only)
    256MB BBWC on controller

Take 3a (early 2013)

Take 3b (2013?)

- All Flash Array backend for master.
- *Custom "pluggable artifact storage (JENKINS-17236)" implementation*
- *Prevent code review (Gerrit) builds recording Fingerprints for Maven2/3 builds*
- New hardware (Cisco USC blades)
- *Builds in RAM disk with custom workspace archiver*

Automated Acceptance Tests

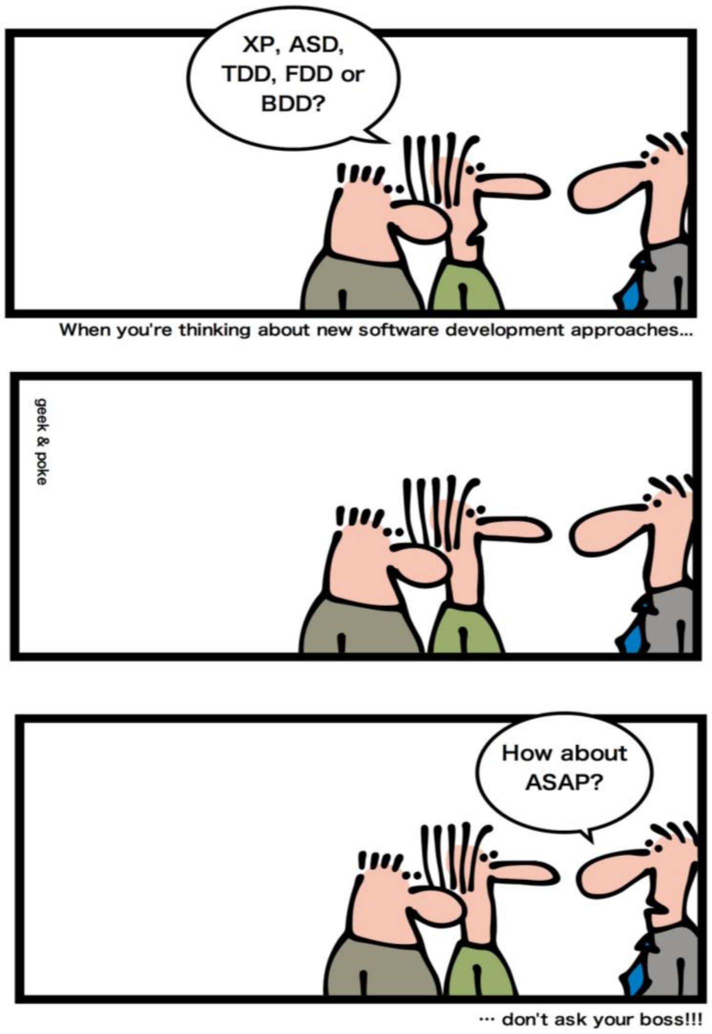

XP, ASD, TDD, FDD or BDD?
When you're thinking about new software development approaches...
geek & poke
How about ASAP?
... don't ask your boss!!!

image © (CC by 3.0) sourc http://geek-and-poke.com/geekandpoke/2013/6/6/dont-ask-your-boss

Demo

http://192.168.71.20/vmwarePool/

http://192.168.71.20/job/AAT_Demo1/job/AAT1/

AATs commonality

- Get some build artifacts (job, release or stable)
- Install them (RPMs after all…)
- Execute some commands on the remote hosts
  - setup config
  - run tests
  - Start DB instance
- Retrieve files from remote server
  - Test results
  - application logs

Templates Demo

http://192.168.71.20/template/

http://192.168.71.20/job/AAT_Demo1/job/AAT1/

Build Flow plugin

- Create pipelines with a DSL syntax.
- Allows plugins to register extensions.

Build Flow (2)

```
def ext = extension.‘my-build-flow-extensions'

def sutRun = ext.getLastStableRun(‘../myproj/commit’)
def testsRun = ext.getLastStableRun(‘../myproj-subsystem-tests/commit’)

def myparams = new HashMap(params)
// Run Parameters are of the form projectname#buildnum
myparams[‘sutRun '] = sutRun.project.fullName + "#" + sutRun.number
myparams[‘testsRun '] = testsRun.project.fullName + "#" + testsRun.number

out.println 'Subsystem test Properties:'
myparams.sort().each { out.println "   $it.key -> $it.value" }

build(myparams, “myproj-aat_master-LEVEL-1”)

build(myparams , "myproj-aat_master-LEVEL-2")

parallel (
  { build(myparams, " myproj-aat_master-LEVEL-3") },
  { build(myparams, " myproj-aat_master-CUST_FOOBAR_SPECIFIC") }
)
```

Basic Flow

http://192.168.71.20/job/Build_Flow/

Build Flow (3)

- We want feedback quickly

  Allow concurrent runs of the flow and downstream jobs
- But the jobs are parameterized
  - So no job coallescing
  - Overloads Jenkins
- Solution?
  - Locks & Latches
  - Throttle current builds
  - Constantly buy more hardware

Concurrent extensions

```
def ext = extension.‘concurrent-extensions'

def sutRun = ext.getLastStableRun(‘../myproj/commit’)
def testsRun = ext.getLastStableRun(‘../myproj-subsystem-tests/commit’)

def myparams = new HashMap(params)
// Run Parameters are of the form projectname#buildnum
myparams[‘sutRun '] = sutRun.project.fullName + "#" + sutRun.number
myparams[‘testsRun '] = testsRun.project.fullName + "#" + testsRun.number

out.println 'Subsystem test Properties:'
myparams.sort().each { out.println "   $it.key -> $it.value" }

build(myparams, " myproj-aat_master-LEVEL-1 ")

ext.block("level2 ") {
  build(myparams , "myproj-aat_master-LEVEL-2")
}

ext.block(“level3“) {
   parallel (
      { build(myparams, " myproj-aat_master-LEVEL-3") },
      { build(myparams, " myproj-aat_master-CUST_FOOBAR_SPECIFIC") }
   )
}
```

Demo

http://192.168.71.20/job/Build_Flow/

Links

- BuildFlow extension

  https://github.com/jenkinsci/buildflow-extensions-plugin

  Requires snapshot build of buildflow
- Cucumber plugin

  https://github.com/jenkinsci/cucumber-testresult-plugin

  Requires custom gherkin build (due to upstream bugs)
- Jenkins enterprise

  http://www.cloudbees.com/jenkins-enterprise-by-cloudbees-available-plugins.cb

Thank You To Our Sponsors

Platinum

CloudBees

Gold

ZEROTURNAROUND

appvance

black diamond SOFTWARE

New Relic.

JenkinsMobi Mobile ALM

XebiaLabs the deployment automation company

BrightRoll

JFrog

Silver

AppDynamics

Confreaks

LIFERAY.

SOASTA Test Faster. Release Sooner.

Questions

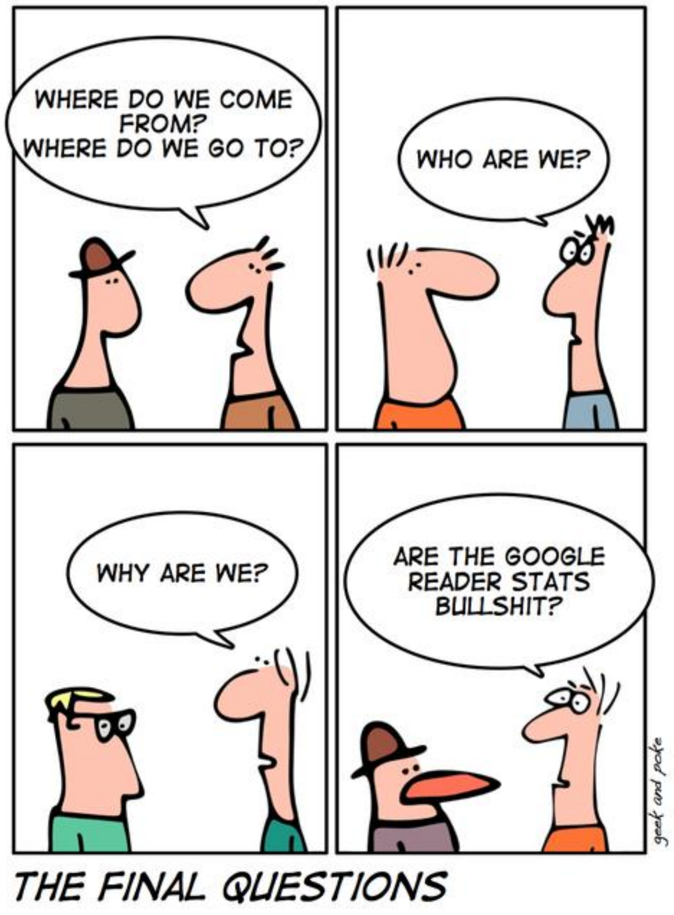

WHERE DO WE COME FROM? WHERE DO WE GO TO?
WHO ARE WE?
WHY ARE WE?
ARE THE GOOGLE READER STATS BULLSHIT?
geek and poke
THE FINAL QUESTIONS

image © ([CC by 3.0]) source [http://geek-and-poke.com/geekandpoke/2007/10/16/the-final-questions.html]

Things to keep an eye out for

- *Aggregate results of triggered jobs in Build Flow.*
- *Prevent code review (Gerrit) builds recording Fingerprints for Maven2/3 builds*
- *Custom "pluggable artifact storage (JENKINS-17236)" implementation*
- *Builds in RAM disk with custom workspace archiver (conditional on build state!)*