Jenkins in the Enterprise

Building resilient CI infrastructure

Kohsuke Kawaguchi
Harpreet Singh
CloudBees

#jenkinsconf

Thank You To Our Sponsors

Platinum

CloudBees

Gold

zend The PHP Company

MidVision™ Release the innovation

PRAQMA

XebiaLabs the deployment automation company

codecentric

Silver

SOASTA Test Faster. Release Sooner.

Corporate

cloudera®

Community

comSysto Lightweight Java User Group Munich

Java User Group Berlin Brandenburg

polish jug

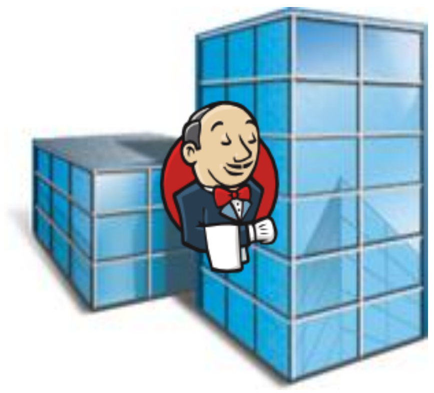
CloudBees Solutions for Jenkins

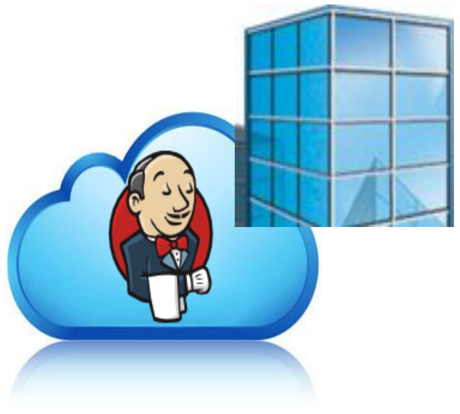
Jenkins Enterprise
by CloudBees

Jenkins Operations Center
by CloudBees

DEV@cloud

DEV@cloud
Hybrid





On Premise **In the Cloud** **Hybrid**

No matter how you use Jenkins

Agenda

- Running Jenkins at scale
- Workflow

Scaling Jenkins Vertically

Single effort that carries everyone

Success!

Let’s scale this baby up

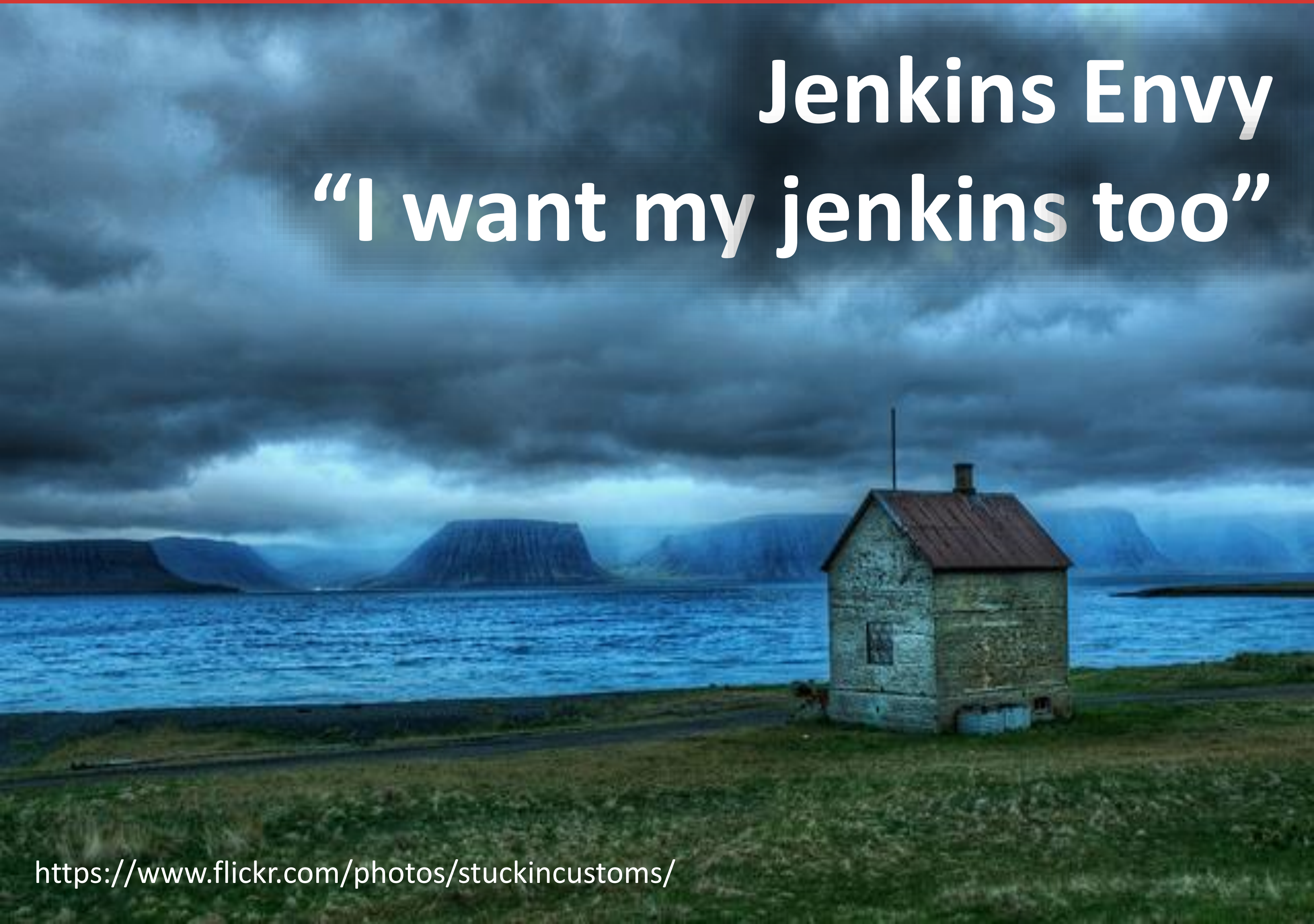
https://www.flickr.com/photos/tuis_imaging/

Scaled it - Success!

https://www.flickr.com/photos/grapesteinbeck/

We still have a problem

Scaling Jenkins Virally



Jenkins Envy “I want my jenkins too”

https://www.flickr.com/photos/stuckincustoms/

Jenkins everywhere

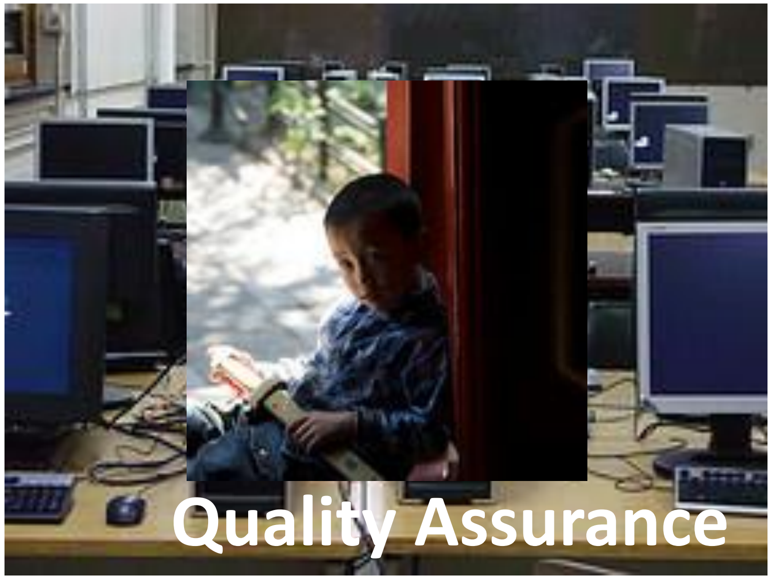
Operations: https://www.flickr.com/photos/dawdledotcom/

QA: https://www.flickr.com/photos/thearchigeek/

Production https://www.flickr.com/photos/treyguinn/

Bad for the bottom line…

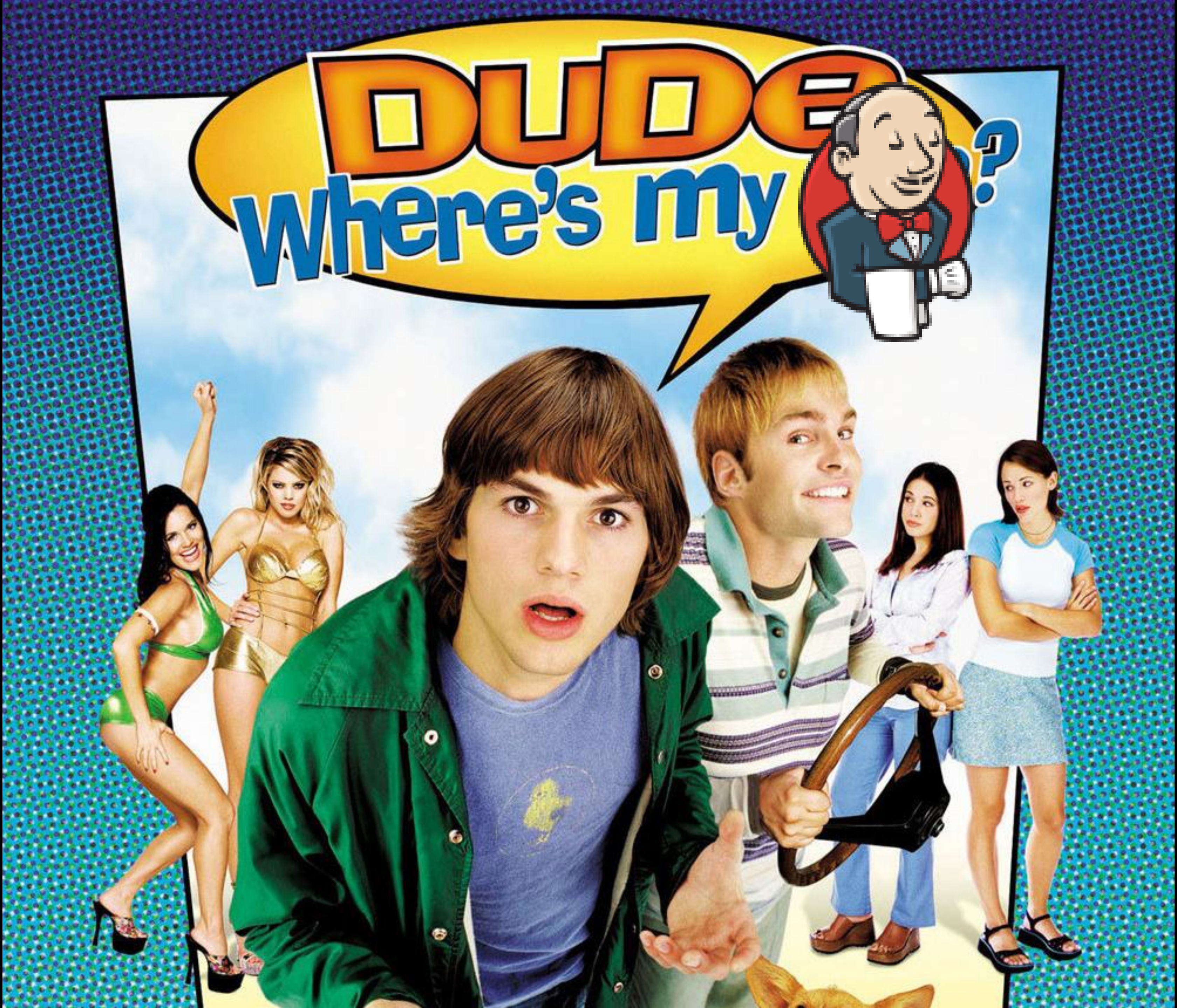
Security by intention



Operations: https://www.flickr.com/photos/dawdledotcom/

QA: https://www.flickr.com/photos/thearchigeek/

Production https://www.flickr.com/photos/treyguinn/

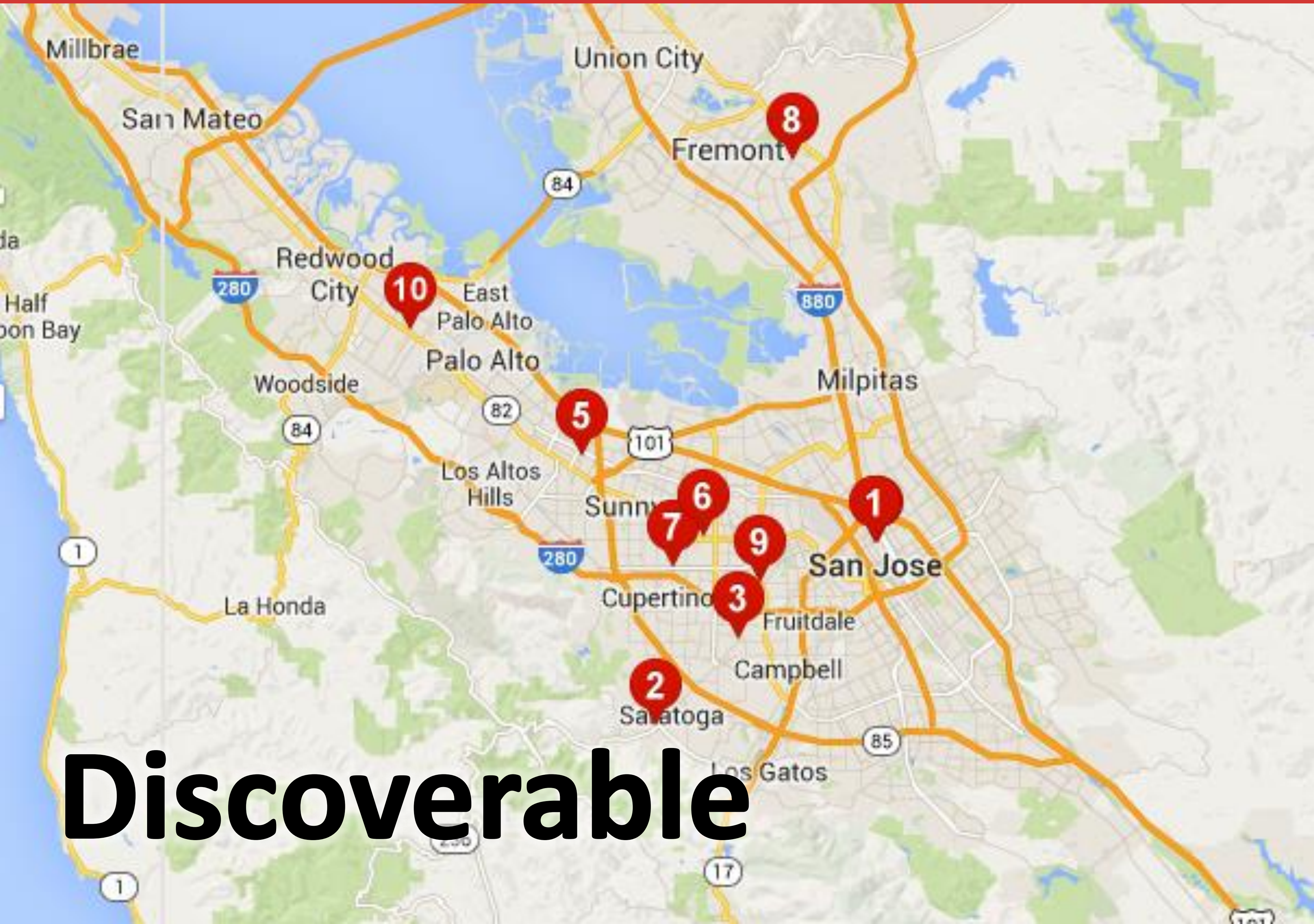

DUDE
Where's my ?

Exhausted Admins

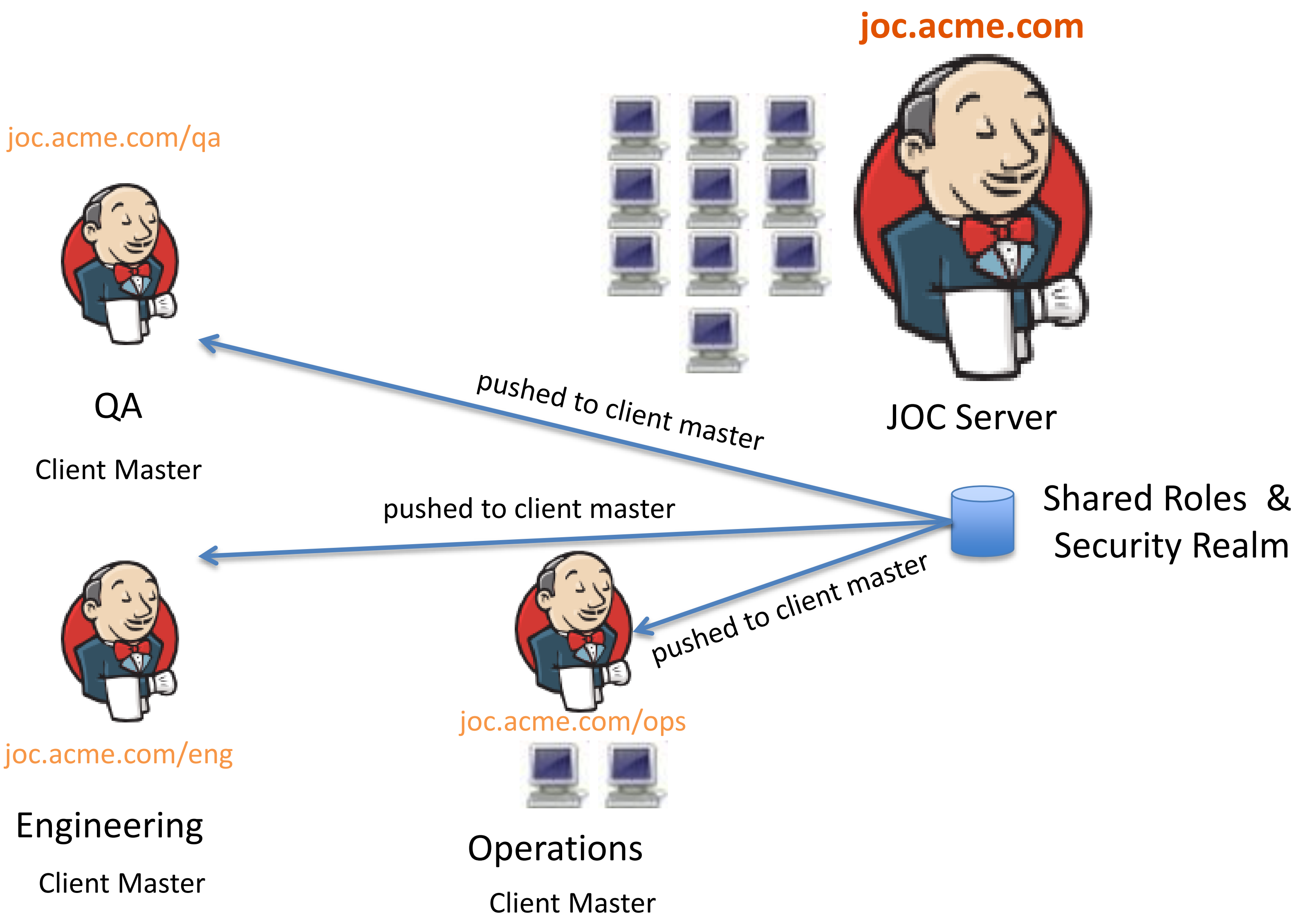
Jenkins Operations Center by CloudBees

Share resources

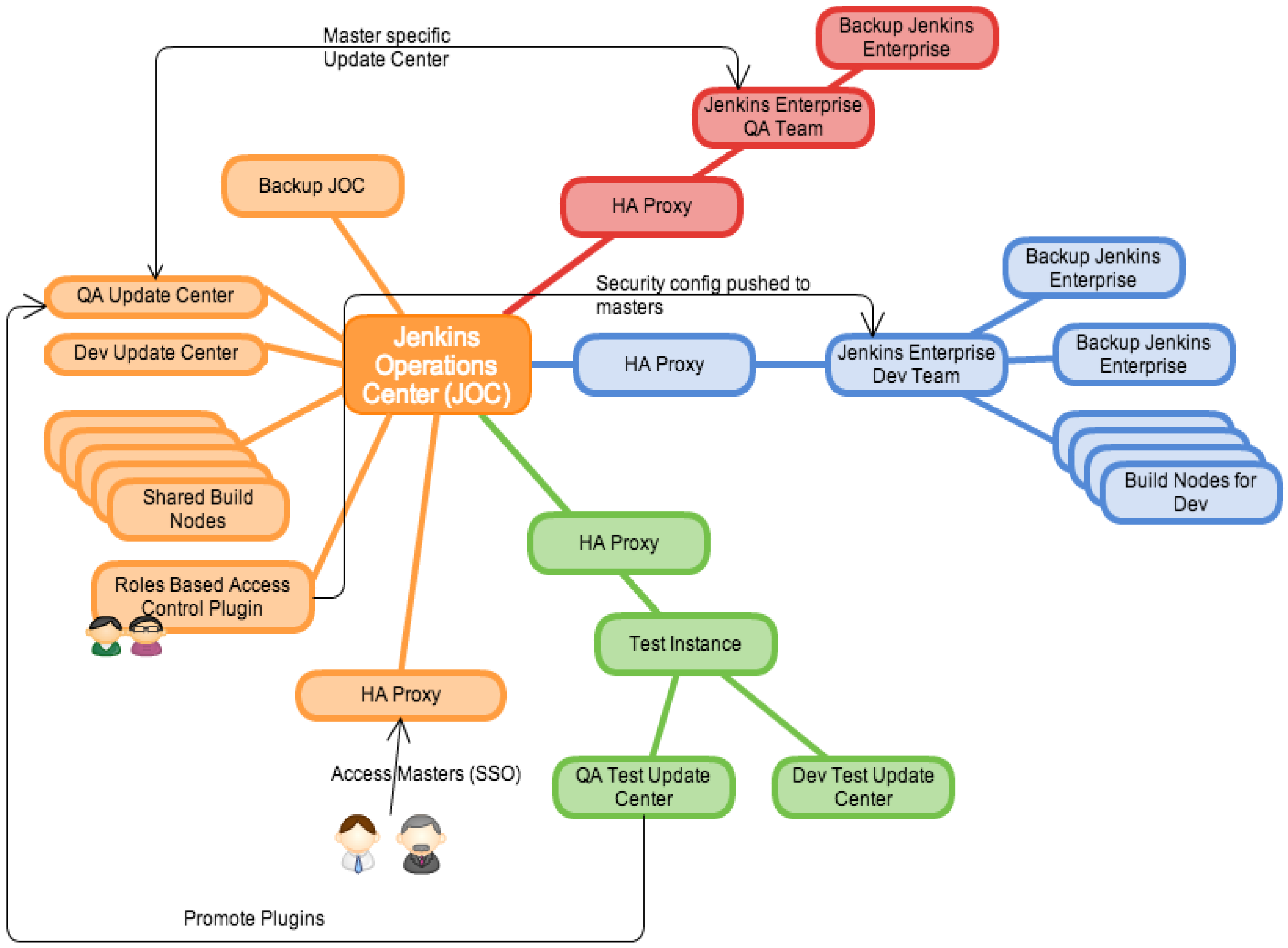
Centralized Security

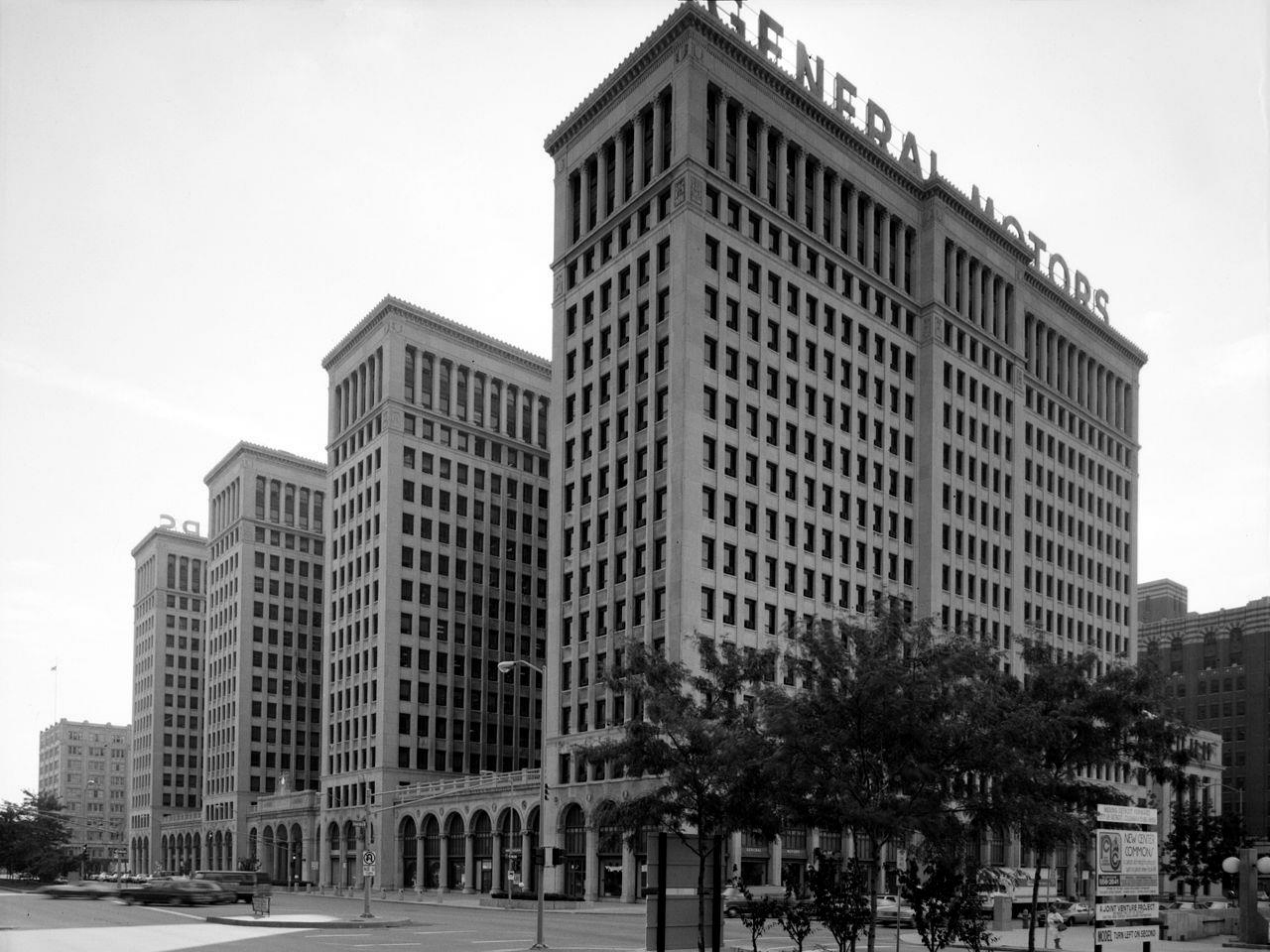


Discoverable


joc.acme.com
JOC Server
joc.acme.com/qa
QA
Client Master
pushed to client master
pushed to client master
pushed to client master
Shared Roles &
Security Realm
joc.acme.com/eng
Engineering
Client Master
joc.acme.com/ops
Operations
Client Master

Reference Architecture


Master specific Update Center
Backup Jenkins Enterprise
Jenkins Enterprise QA Team
Backup JOC
HA Proxy
QA Update Center
Security config pushed to masters
Backup Jenkins Enterprise
Dev Update Center
Jenkins Operations Center (JOC)
HA Proxy
Jenkins Enterprise Dev Team
Backup Jenkins Enterprise
Shared Build Nodes
Build Nodes for Dev
HA Proxy
Roles Based Access Control Plugin
Test Instance
HA Proxy
Access Masters (SSO)
QA Test Update Center
Dev Test Update Center
Promote Plugins


GENERAL MOTORS
NEW CENTER COMMONS
A JOINT VENTURE PROJECT
MODEL TURN LEFT ON SECOND

Demo

Jenkins Enterprise by CloudBees Plugins

High Availability

- High Availability
- Restart Aborted Builds
- **Long Running Builds***

Large Installation

- Backup Scheduling
- Consolidated Build View
- Custom Update Center
- Folders
- Folders Plus
- Plugin Usage
- **Monitoring***
- Nodes Plus
- Support plugin
- Templates
- Validated Merge

Security

- Role-based Access Control (RBAC)
- Secure Copy
- WikiText Descriptions

Optimized Utilization

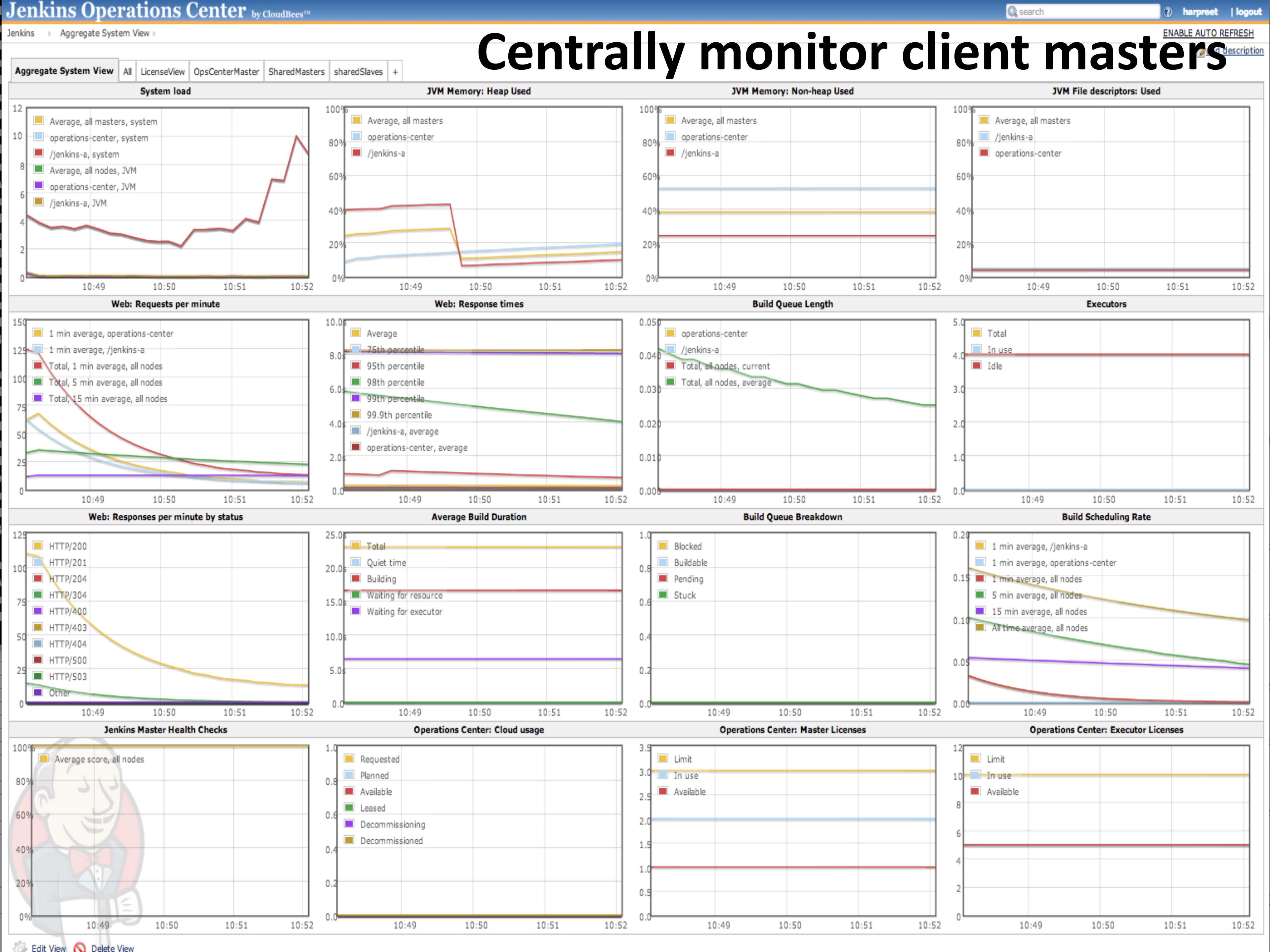
- Even Load Strategy
- Fast Archiver
- Label Throttled Build Execution
- **NIO SSH Slaves***
- Skip Next Build
- VMware ESXi/vSphere Auto-Scaling

*** Released May 2014**

JOC 1.1 release

- JNLP slave & JNLP cloud
- monitoring

Centrally monitor client masters



Workflow

Use Cases: orchestrated activities

- Multi-stage continuous deployment pipeline
- Run part of build with a temporary server
- Blue/green deployment with auto commit/abort
- Parallel tests with automatic sharding
- Automatic per-branch jobs (à la Literate plugin)

Characteristics

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

Workflow: the one-pager

```
with.node('linux') {
    git(url: 'git://server/myapp.git')
    sh('mvn clean package')
    archive('target/myapp.war')
    stage('Test')
    parallel({
        sh('mvn -Psometests test')
    }, {
        sh('mvn -Pothertests test')
    })
    input('OK to deploy?')
    stage(value: 'Deploy', concurrency: 1)
    sh('mvn deploy')
}
```

Key features (what I already covered)

- Entire flow is one concise Groovy script
  - for-loops, try-finally, fork-join, etc.
- Can restart Jenkins while flow is running
- Human input/approval integrated into flow

More key features

- Allocate slave nodes and workspaces
  - as many as you like, when you like
- Standard project concepts: SCM, artifacts, …

Crux

Resumption of Groovy flows

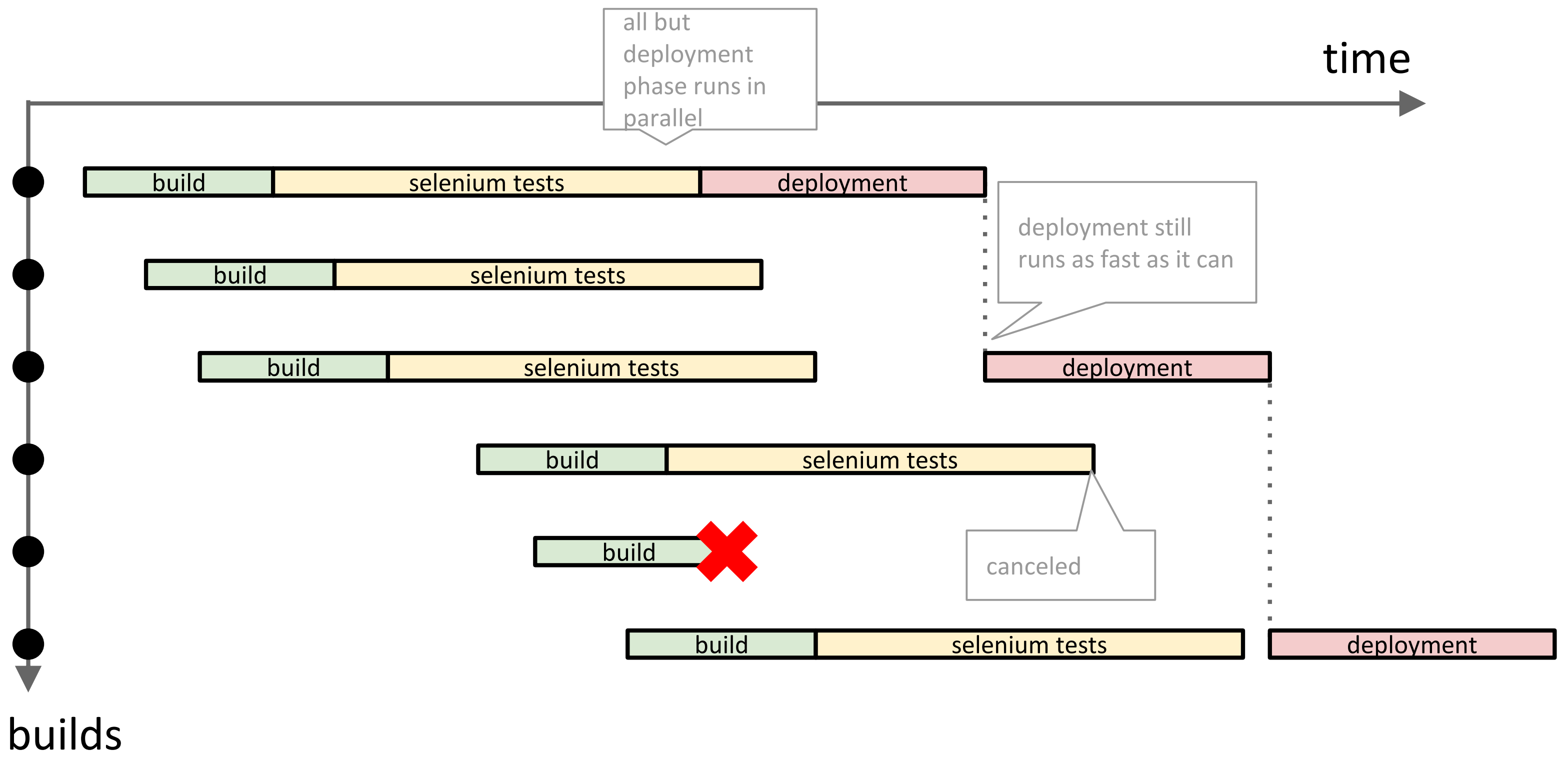
- Transformed to “continuation-passing style”
- Run on custom interpreter of Groovy
- Sate of program saved at each pause point
- Variables serialized and restored after restart
  - pickles: extensible object replacements

Resumed builds to the user

- It “just works”
- (Serializable) local variables restored too
- Shell-like steps survive restart
  - Reconnection of slave, too
- Jenkins Enterprise: resume from checkpoint
  - Can pick up artifacts from original build
  - No need to rerun earlier expensive steps

Stages (aka James Nord operator)

- Pipeline throttling primitive
- Special semaphore: only newest build may wait


all but deployment phase runs in parallel
time
build
selenium tests
deployment
deployment still runs as fast as it can
canceled
builds

Demo

Design: overall

- suite of Jenkins plugins
  - Jenkins Enterprise may add checkpoints, &c.
- pluggable flow definition & execution engine
  - Groovy CPS is recommended choice
  - STM (proof of concept)
  - Activiti or other BPMN should be possible

Design: flows

- persistent record of execution
- directed acyclic graph of nodes
- some nodes represent one step
- others indicate block start/stop structure
- nodes may have associated metadata
  - console log fragment contributes to main log
- pluggable visualizations for different views

Design: steps

- standalone API for asynchronous build steps
- *context* serves as an identifier & callback
  - also offers logger, build, workspace, &c.
- support for block-structured steps
  - invoke body 0+ times with revised context
- standard step for “durable” shell/batch scripts
- standard steps for SCMs (git, svn, hg)
  - >1 SCM per build possible

Design: interoperability

- run on existing Jenkins slaves
- SCM plugins supported with modest changes
- coming soon: existing build steps & publishers
- coming soon: trigger existing jobs
- standard build history, artifacts
- needs ongoing core changes (currently 1.568+)
  - features factored out of standard projects

Still to come

- Shared workflow code in VCS
- Imagine the demo becoming this:

```
acme_process {
  git = 'git://server/myapp.git`
}
```

Still to come

- more build steps
- workspace management
- Cancel button
- robustness, polished UI
- Groovy sandbox

Status

- Open for contribution
- github.com/jenkinsci/workflow-plugin
- 0.1-beta-1 binaries on experimental UC
- requires Jenkins 1.568+ today
- fundamentals all work now
- aiming for 1.0 this year
- considered strategic by CloudBees

Lots to get excited about!

- Enterprise?
  - JOC / JE
- OSS / early adapter?
  - Workflow
  - Traceability

Project setup

- one workflow is defined as a job
- single script for all steps
- build triggers & parameters like regular projects
- SCM, publishing, … are all part of script
- each build shown using regular Jenkins view
- Graphical visualizations of actual build possible
  - (not of job definition; could be too dynamic)