


Maintaining huge Jenkins clusters

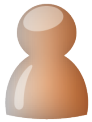
- Have we reached the limit of Jenkins?

Robert Sandell
Sony Mobile Communications
www.sonymobile.com
www.rsandell.com

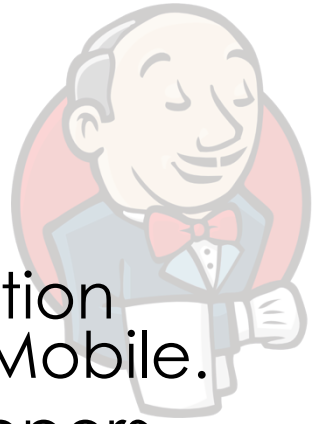
@jenkinsconf

TOC

- How We Work
- Jenkins & Gerrit topography setup
- What can our users do
- Automated maintenance
- Future plans



About Me



Robert "Bobby" Sandell a.k.a. rsandell

I work in the *Development Environment* section within the Software R&D division at Sony Mobile.

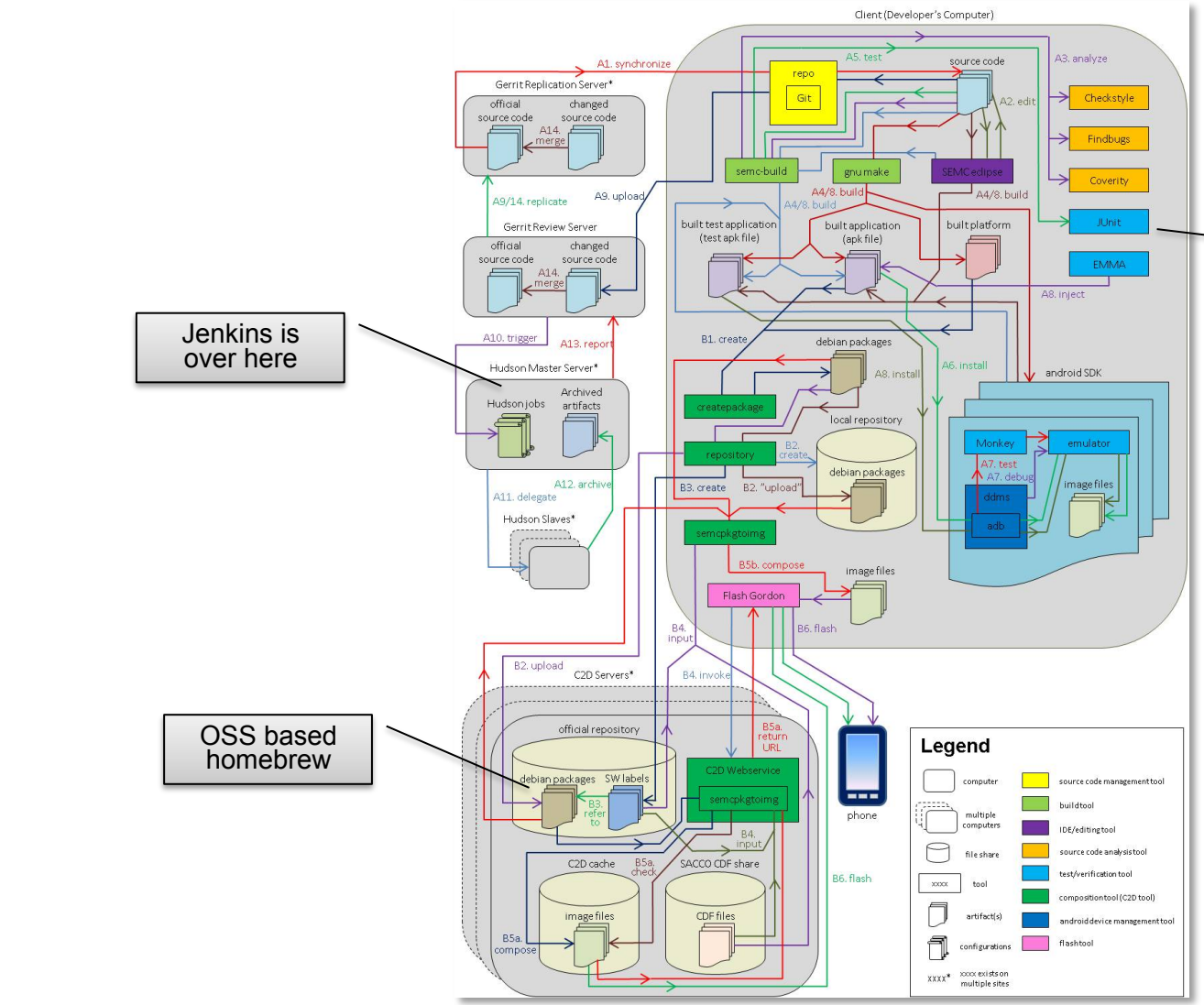
We are forging the tools used by the Developers e.g. SCM, IDE, emulator, compilers, build, CI and some verification tools.

My current role is Tool Manager** for Jenkins as a member of the *Build & CI* Team.

Maintainer of several Jenkins plugins e.g. Gerrit Trigger, Build Failure Analyzer and Multi Slave Config.

** TM is management speak for: Besides coding I also prioritize and manage the backlog and act as ambassador to the users.

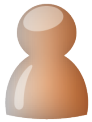
The Tool chain



Mostly OSS

Jenkins is over here

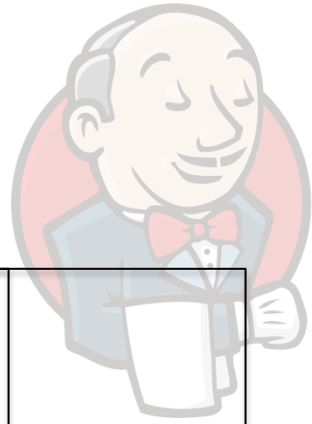
OSS based homebrew

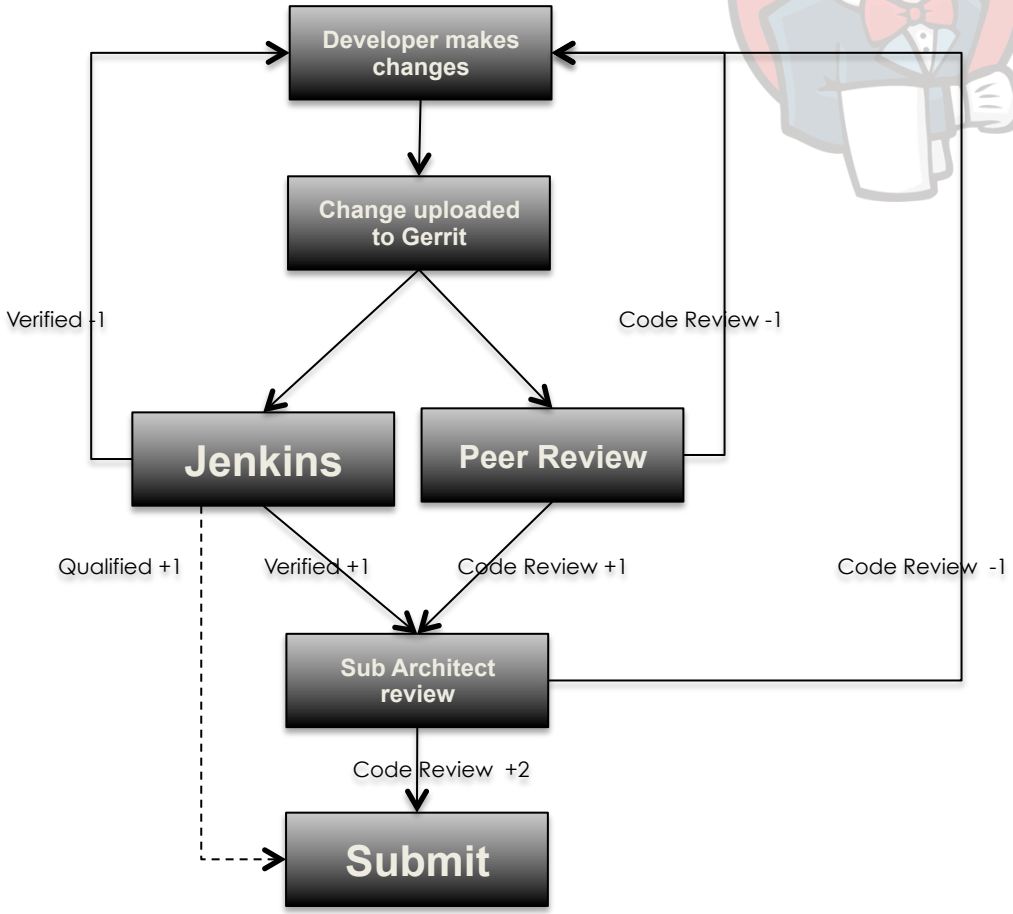


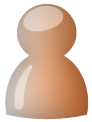
How We Work

The tale of adding a small feature

- A change needs to get votes before it can be merged; Verified +1 and Code Review +2.
- Jenkins is the "only user" in Gerrit allowed to set Verified +1
- Only after the change has been automatically built (on real devices or emulators), tested, analyzed and accepted, the change is Verified +1
- Pretty much every GIT needs to have a corresponding Jenkins build project
- To help us enable this workflow we created the Gerrit Trigger Plug-in for Jenkins



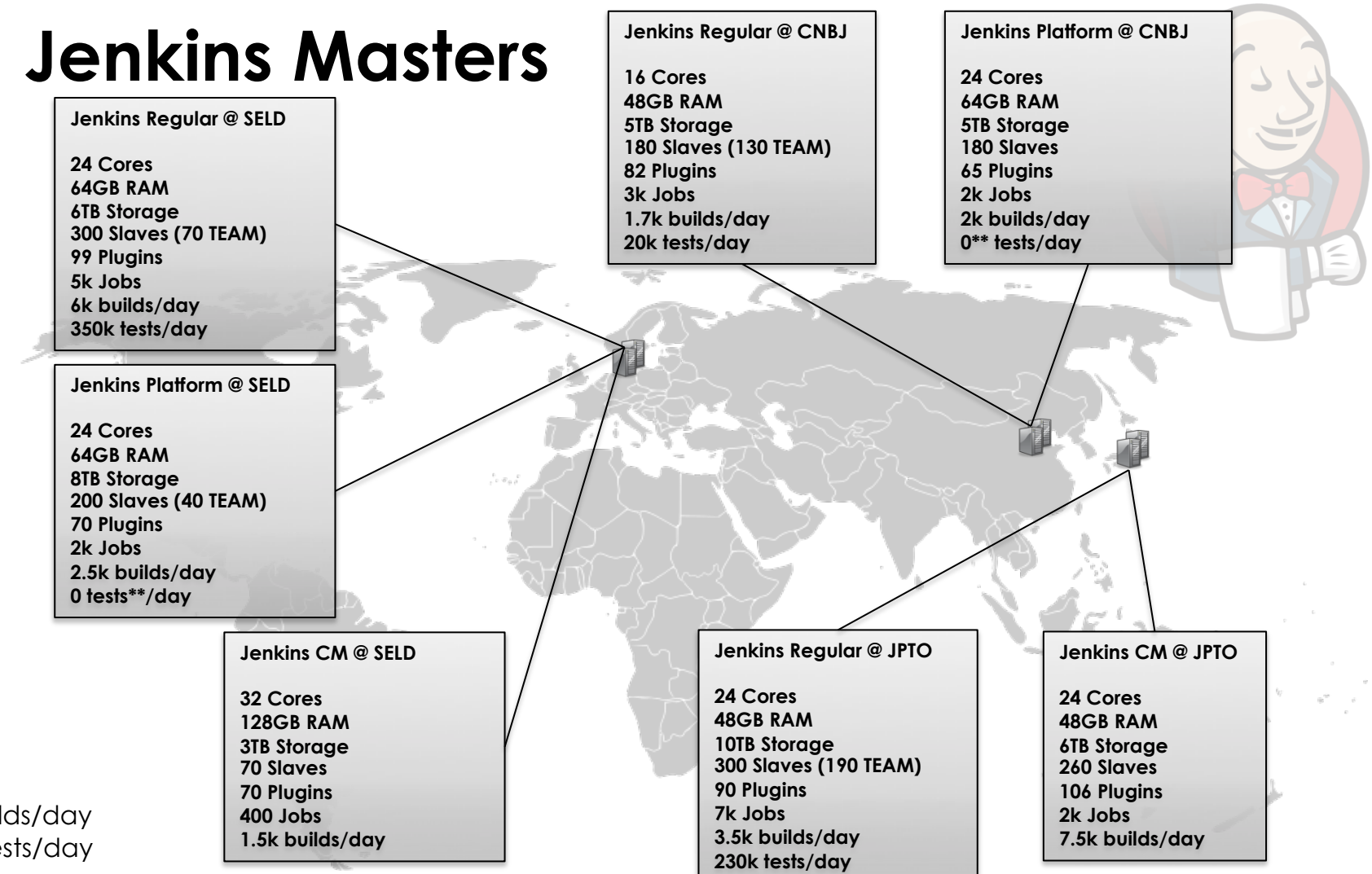




Some typical builds

- Besides our build farm most teams have one or more Jenkins slaves located near their desks, so called "Team Slaves." The slaves have physical phones connected to them via USB.
- The teams have Jenkins jobs configured for periodic builds, which
  - Flash the latest Android Platform onto the phone
  - Build the application from HEAD of the branch
  - Install it on the phone
  - Run several tests/static analysis on it
- When builds are triggered from Gerrit, teams can choose to run the build on their team slave(s) (testing using a real device) or on a build farm machine (testing using an emulator).
- The System team also do full Android platform builds on every commit for their own integration branches.
- We also do "Official builds"
  - Full build of the Android platform
    - Labeled and uploaded to the binary repository
  - Periodically triggered several times per day
  - Multi configuration project
    - All phone configurations and variants for current release branch
  - A successful official build will trigger a chain of other builds that perform automatic smoke tests in the verification lab.

Jenkins Masters



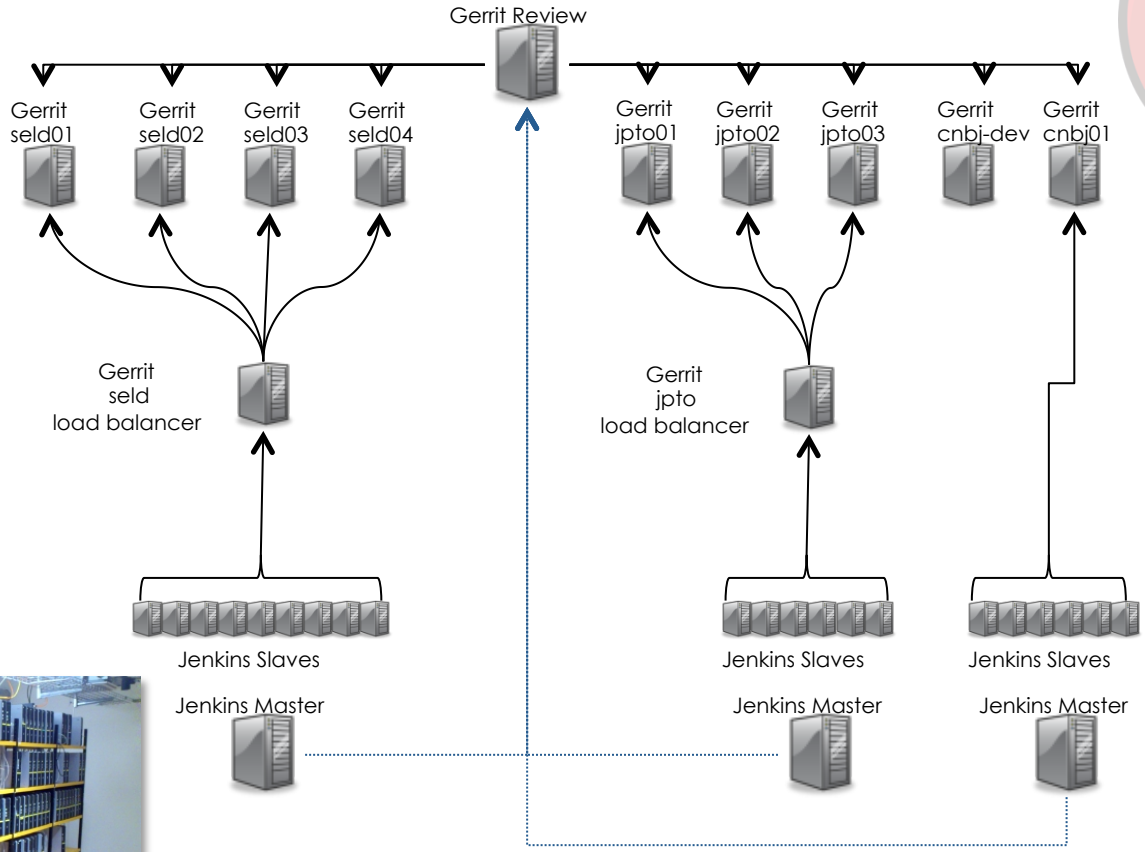
Totals:

16K Builds/day
600K Tests/day

1500 (Non virtual) Slaves – All of them acts local reference mirrors (updated and repacked daily)!
Around 25-30% of all builds are triggered by Gerrit

** There are more tests executed, but the count is based on recorded test results by Jenkins

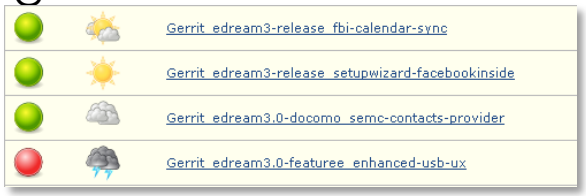
Jenkins/Gerrit Setup



Servers: Ubuntu 10.04 64 Server
Slaves: Ubuntu 12.04 64 Workstation

What can users do

- If they can login (and everyone can), they are allowed to:
  - Create and configure jobs
  - Delete any job or build
  - Run any build
  - Basically do anything except administer
  - But certain privileged/power users can also administer
- Mostly Freestyle or Matrix jobs
  - Lots of bash and python scripts.
  - Applications are mostly built using an in-house developed plug-in, wrapping the functionality our in-house application build system.
- We provide job templates and a user guide
  - Deviations occur quickly
  - Users usually end up copying a previous team job
  - But many teams have one or two power users that helps explaining and setting up builds
- We encourage users to keep to our project naming standard
  - This way we can sort various projects in different views depending on project, branch, etc.
  - Helps with the collection of Metrics
  - Is the build triggered by Gerrit? Periodically? Experimental?



Maintaining the clusters

Split responsibility between IT, application managers and our team

The IT guys takes care of everything up to the OS level on the slaves and servers.

- Keep them cool and connected
- Replace/upgrade hardware
- OS Patching
- Etc.

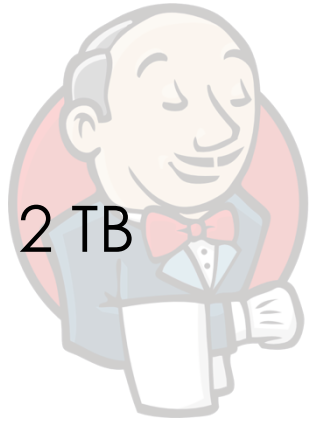
The AM team takes care of the every day Jenkins work and first line Jenkins/Git/Gerrit support

- User Support
- Manage Jenkins
- Analyze and fix transient build errors
- Etc.

We take second line support, larger development work and try to plan ahead.

- Plan Jenkins upgrades
- Manage/develop plug-ins
- Handle user requests
- Educate users
- Engage with the community
- Etc.

Automated Maintenance
Problem #1: Disk Space



- Disk space on the slaves is between 1 and 2 TB
- Some "pimped" slaves with 240GB SSD
- A full Android build is ~50GB
- The Android SDK is ~1.5GB
- Each slave has a git reference mirror of about 100GB

Cron script on Master
Running once a day

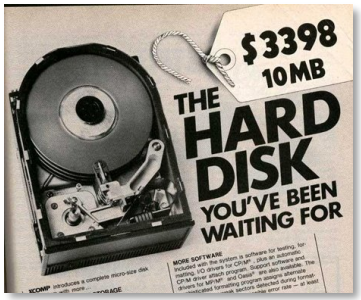
```
while free_disk_space < 20%
  removeOldestArtifact()
```

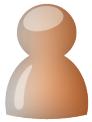
We can keep artifacts stored for about 2 weeks sometimes 2 days depending on load.

Periodic Jenkins job on Master
Running every hour or 30 min

```
foreach slave
  ssh-connect and check disk-space
  if < 80GB
    remove workspace for all non-building jobs
    if nothing is building
      rm -rf /tmp/*
```

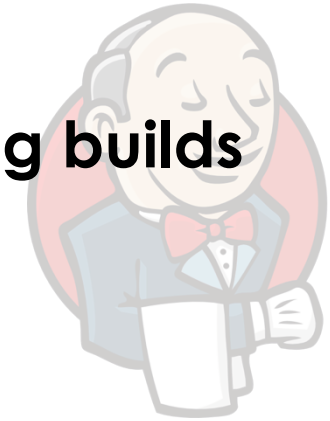
A workspace might be available for a few hours after the build completes.





Automated Maintenance

Keeping slaves up to date without disturbing builds



- CF Engine and Puppet can't handle the load.
- Small tools and maintenance scripts needs to be distributed.
- Don't want a lib updated in the middle of an official build.

Launching a slave:

- `rsynch the needful`
- `launch slave.jar`

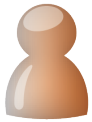
Update local git ref mirror every evening:

```
list = gerrit ls-projects | grep -f white-list
foreach pos in list
  git clone --bare --git-dir ~/.repo-mirror/$pos
```

Jobs can then clone with --reference $REPO_MIRROR

Update packages every morning:

- `Put the slave "temporary offline"`
- `Wait for builds to finish`
- `Run cf-agent and apt-get upgrade`
- `If reboot needed`
  - `Force fsck on next reboot`
  - `Reboot`
- `Mark the slave as online`



Automated Maintenance
Surveying and measuring

Slave sanity

----- SLAVE: Check that there is >= 50 GB free disk space on . -----

----- SLAVE: Check that there is >= 1 GB free disk space on /tmp -----

----- SLAVE: Check that there is 1 instance of slave.jar -----

----- MASTER: Check that there is 1 connection to each slave -----

----- SLAVE: Check that the .ssh directory is correct -----

----- SLAVE: Check that slave home dir doesn't contain a .repo directory

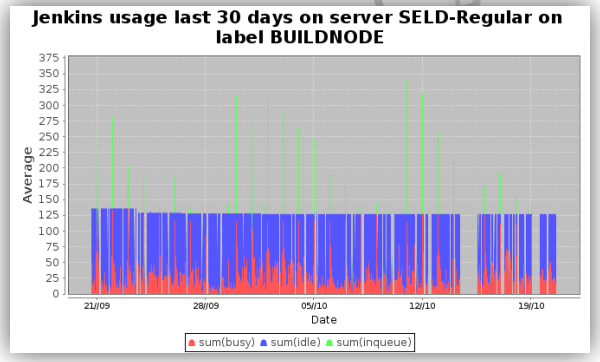
----- SLAVE: Check that the hosts are pingable -----

----- SLAVE: Check that /tmp isn't readonly -----

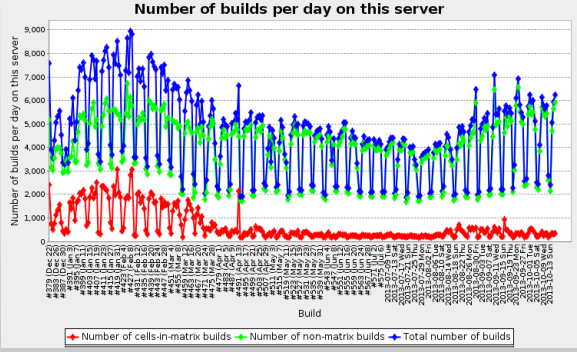
Package sanity

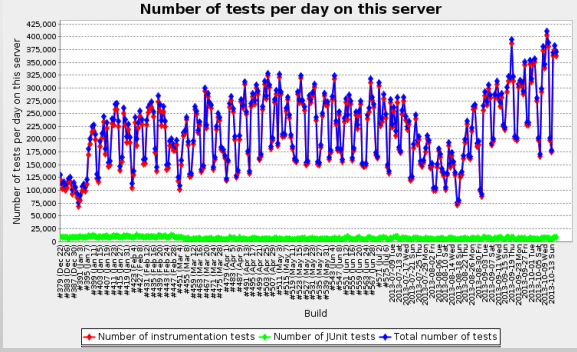
Check versions of vital packages like:

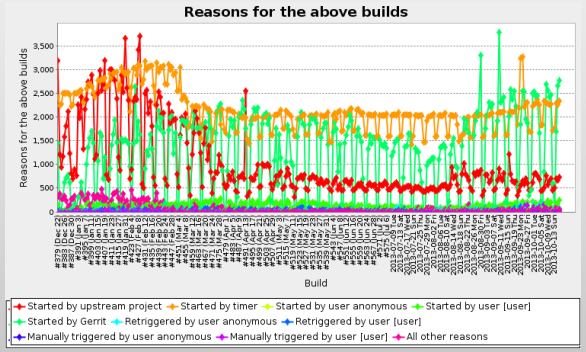
- Semc-build
- Python
- signtool

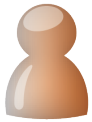


Periodic counting with the plot plugin







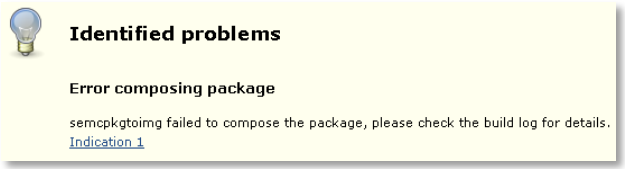


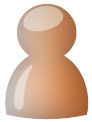
How to install a plugin

1. Check the source code
   - Any global hooks?
     - RunListener, ItemListener, QueueSorter, PageDecorator etc.
   - Any big iterations?
     - getItems(), getComputers(), getBuilds() etc.
   - General "feeling" about the code
     - Complexity, Unit tests, does it look like crap?
       i.e. how hard would it be if we need to fix a bug?
2. Play with it in a local instance
3. Install on test server(s)
4. Install on one of the production servers
5. Deploy world wide
6. Still keep a paranoid eye on it for another week or so.

Plugins worth mentioning for a maintainer

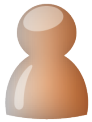
- Multi Slave Config Plugin
  - *For handling multiple slaves in one click*
- Sectioned View Plugin
  - *For news*
- Scriptler
  - *For debug and quick fixes*
- Job Config History
  - *To find someone to blame*
- Disk Usage Plugin
  - *To find someone else to blame*
- Build Failure Analyzer
  - *So we don't have to find the same issue again*



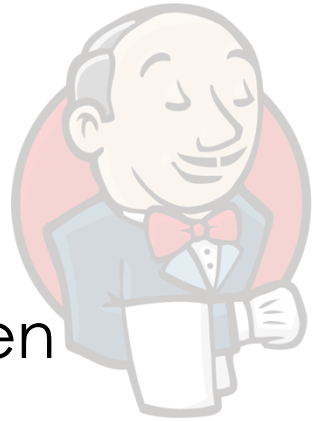


Bumps in the road

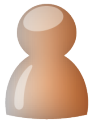
- Politics on Jenkins as integration blocker
- Hudson 1.377 to Jenkins 1.404
  - Remoting API
  - Sell-in of Jenkins "fork" to management
- Regressions in plug-ins
  - The CVS plugin is staying at version 1.6!
- Performance
  - NFS -> SAN 40min startup to 10min
  - Ext4 vs. XFS 40min startup to 20min
  - Spring cleaning 1h 40min startup to 1h 10min
  - Queue synchronization
    - Why is the UI so slow at lunch time?
  - 1.447.2 -> 1.480.2 1h 40min startup to 15min
    - Nothing in the change log about this except "Misc performance improvements" in 1.452 !?
  - How to test lazy loading penalty/gain in 1.509?



Future



- Metadata
  - How to find something on such an open master.
- Templates and pipelines
- Security
- Move to the Cloud?
  - Or just stick our heads up there every once in a while?
- Increasing the cluster size?
  - Do we need to?



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.