java –jar hudson-3.0.0.war
Plugins
- descriptors

Projects (Jobs)
- config.xml

Slaves
Labels

config.xml

Hudson
Master
Master/Slave/Views

Plugin Config

Job config

Build archival

**config.xml**

```
/jobs/${name}
${descriptor}.xml
```

```
/builds/${number}
```

**build.xml**
Early efforts

• Single master scalability
• “I need to develop a new plugin for generic uptake across existing projects”
• “I need to develop a new plugin for my projects”
• “I have a new project coming online (no consumers) and I would like a little privacy while I set up the job”
• When is it okay to reboot a Hudson master?
1 butler? Butler of butlers?
Carson’s Resource Model

- /carson/v1/organizations
  - GET
  - POST

- /carson/v1/organizations/:id
  - GET
  - PUT
  - DELETE

- /carson/v1/projects
  - GET
  - POST

- /carson/v1/projects/:id
  - GET
  - PUT
Carson’s Resource Model

- GET POST
  /carson/v1/plugins

- GET PUT DELETE
  /carson/v1/plugins/:id

- GET POST
  /carson/v1/slaves

- GET PUT DELETE
  /carson/v1/slaves/:id
Config Management (git repo)

- Job config
- Plugin config per org
- Slave config per org

Shared Resources

Master/Slave nodes
Build Archival (Storage)
JFrog Artifactory for Artifact Management

IT

Hudson Plugin Manager

Artifactory For Internal Plugin promotion

Http Gateway

Idap
Adding new orgs or dev instances

POST /carson/v1/organization

RouterService

Http Gateway

http://host/hudson/crm
http://host/hudson/hcm

sync

hcm
crm

IAAS
Job Promotion

PUT /carson/v1/project/:id

prod

move

dev
Hudson REST interface

/rest/admin/config
/rest/admin/quiet-down
/rest/admin/restart
/rest/queue/clear
/rest/queue/nodes
/rest/queue/{nodeName}
Hudson REST interface

/rest/projects

/rest/projects/{projectName}
  • POST GET DELETE

/rest/projects/{projectName}/builds

/rest/projects/{projectName}/config

/rest/projects/{projectName}/enable
  • boolean query param
Plugin Stages of Maturity

- Maturity not determined by version
- Maturity determined by virtual repository (the artifactory way)
- Organization decides whether it will use plugins of lower maturity
Hudson plugin containers

- Teams did not mind restarting dev masters
- Dev Managers care very much about restarting production masters
  - Easier if a restart doesn’t cancel any jobs or block any from getting on the queue

- Hudson 3.0.0 core
  - @Extension container
  - Smoothie Container (jsr-330)

- Experimental
  - OSGI (jsr-330)

- PluginStrategy
- ExtensionFinder
Embracing restarts

PUT /carson/v1/plugin/:id

Http Gateway

Update route

enable prod
disable dev
copy
## Node Management Extension Points

<table>
<thead>
<tr>
<th>Extension Point</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XLCSlave</td>
<td>Integrate Linux XLC containers for cheap virtualization</td>
</tr>
<tr>
<td>KerberosNodeMonitor</td>
<td>(map #(test %) -&gt; (filter #(labels? :ade %) nodes) :tickets))</td>
</tr>
<tr>
<td>NFSNodeMonitor</td>
<td>(map #(test %) -&gt; (filter #(labels? :nfs %) nodes) :mountPoints))</td>
</tr>
<tr>
<td>CacheManagerLifecycle</td>
<td>Implement ivy/gradle/maven cache usage/cleaning policies</td>
</tr>
</tbody>
</table>
New Extension Points added via Plugins

Example: “release” versioning
Provisioning Extension Points

<table>
<thead>
<tr>
<th>@ToolInstallation</th>
<th>We made the rule that new builders with software requirements must provide this ExtensionPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>@ToolInstaller</td>
<td>Plug in chef/puppet scripts that pull installations from artifactory</td>
</tr>
<tr>
<td>@ToolProperty</td>
<td>Useful to formalize the contract for how different jobs might need to install build tools</td>
</tr>
</tbody>
</table>
## View Extension Points

<table>
<thead>
<tr>
<th>MatrixView</th>
<th>We made the rule that new builders with software requirements must provide this ExtensionPoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>ArtifactoryQuotaView</td>
<td>Plug in chef/puppet scripts that pull installations from artifactory</td>
</tr>
<tr>
<td>CramItBadgePlugin</td>
<td>Useful to formalize the contract for how different jobs might need to install build tools</td>
</tr>
</tbody>
</table>

- We wrote a big SCM plugin and quite a few custom @Trigger extensions
- Also created a brand new Hudson ExtensionPoint to work with JFrog
CI hierarchy of needs


Reliance: requests for new reports, new features. Presence is assumed

Self-Actualization: System and CI process evolve together
Dev Org
“pre-merge”
Spider Sense

Choose feature based on “max information increase” (Shannon Information)

Update predictive model

Identify “high-impact” changes
Application of CI in SOA env

Composability
- new modules
- new dependencies
- new endpoints

Compile/package/test

Refactoring
- cohesion, coupling
- patchability
- idempotency
Components in FusionApps

• ~ 10407 ADF libs
  – “Model” ADF libraries contain Entity definitions and Java Service interfaces
  – “UI” ADF libraries contain the task flows, page defs, jsff, etc. (resources)

• ~ 470 SCA composites

• 1:1 ratio between module/component and Project
Continuous Release

Feed Forward:
- Package (plan deployments)
  - Connection plans
  - SOA Config plans
  - Credential plans
  - Assemble Security Policies
  - Http Access config
  - BIP Catalogs
  - Translation
Assembling Service Network

- Build
  - Collect all abstract portType information
  - scac phase runs against contracts in MDS
  - scac compile phase unordered
  - Assemble portType to logical provider map

- Provision
  - Assembling Service Network
  - Collect all abstract portType information
  - scac phase runs against contracts in MDS
  - scac compile phase unordered
  - Assemble portType to logical provider map

- Package
  - Enumerate every unbound port
  - generate SOA config plans
  - integrate with Deployment Admin

- Publish

- Deploy(Concrete)
  - bind concrete endpoint details to generated plan
  - Inject environment
  - Deploy composites

- Plan(Abstract)
Assembling Service Network

- Build is just scac validation IF we avoid dependency on endpoint urls (runtimes)
- Introduction of a “publish” phase and strict separation between “abstract” interface and concrete endpoints
- Endpoints are introduced only at deploy time (post scac compile) using config plans generated by analysis of the service network (“bottom-up assembly”)
- Deployments can be done UN-ordered (ie in parallel)
- ConfigPlans are also readable as endpoint requirement specs
- Config Plans are never “versioned” files.
- No need to have developers “tokenize” anything.
CI “events” in SOA lifecycle

- new providers
- new consumers
  - new consumers with new topology requirements
  - new “wires” within existing topology
  - How many emulations/tests added?
- call out any schema types not uniquely defined
- updates to third-party contracts
Connection Management

- Types
  - ServiceFactory
  - WebService
  - Http
  - UCM
  - Forum
  - WSRP
  - BI JDBC
  - BI SOAP
  - EXTAPP

- Same basic Principle:
  - Publish (Discover)
  - Plan (Determine Edges)
  - Deploy (Bind “logical” to “concrete” and apply pre-ear): topology written into MDS

- Credential Plans
  - “plan” phase is replaced by an “erase” phase: continuous amnesia?
Security

Resource Types
• Permission Owners
• Framework-driven

Merged Jazn-data policy

App Roles / Privileges
• Top-Down centrally managed

Grants
• bottom-up resource driven
Maven Default Lifecycle

validate -> compile -> test -> package -> integration-test -> verify -> install -> deploy

gal

maven-ant-plugin

gal

weblogic-maven-plugin

gal

Custom-hudson-plugin
Full Parallelized Builds

- Extract("discover")
- Create Edges of Dependency Graph
- Generate Build "plan"
- Send Build "Events" to slave machines

Full Parallelized Builds
Incremental Builds

- Dep Graph
- Map “Transaction Delta” to Nodes
- Incremental Rebuild/Repackage plan
- Create Patch
Impact Analysis

- Runtime A
  - Component A.1
  - Component A.2
  - Impacted

- Runtime B
  - Component B.1
  - impacted

Bug fix
Hudson Maintenance BuildWrapper

Hudson SCM plugin

Impact Analysis

Plan/Execute Build

Build Events

Change Events

Policy Pipeline

Deployment Manager

Environment Definitions

Translation Module

Versioning Policy

App Restart Policy
Change “Events” tracked
Internal Hackfest results

• Build avoidance
• Bulk rebuild trigger
• LogBeautifiers
• Dependency graph views (GraphViz processing.js)
• OSN integration