glu

deployment automation platform

July 2011

Yan Pujante
http://www.linkedin.com/in/yan
blog: http://pongasoft.com/blog/yan
@yanpujante

deployment automation platform

A little bit about me...

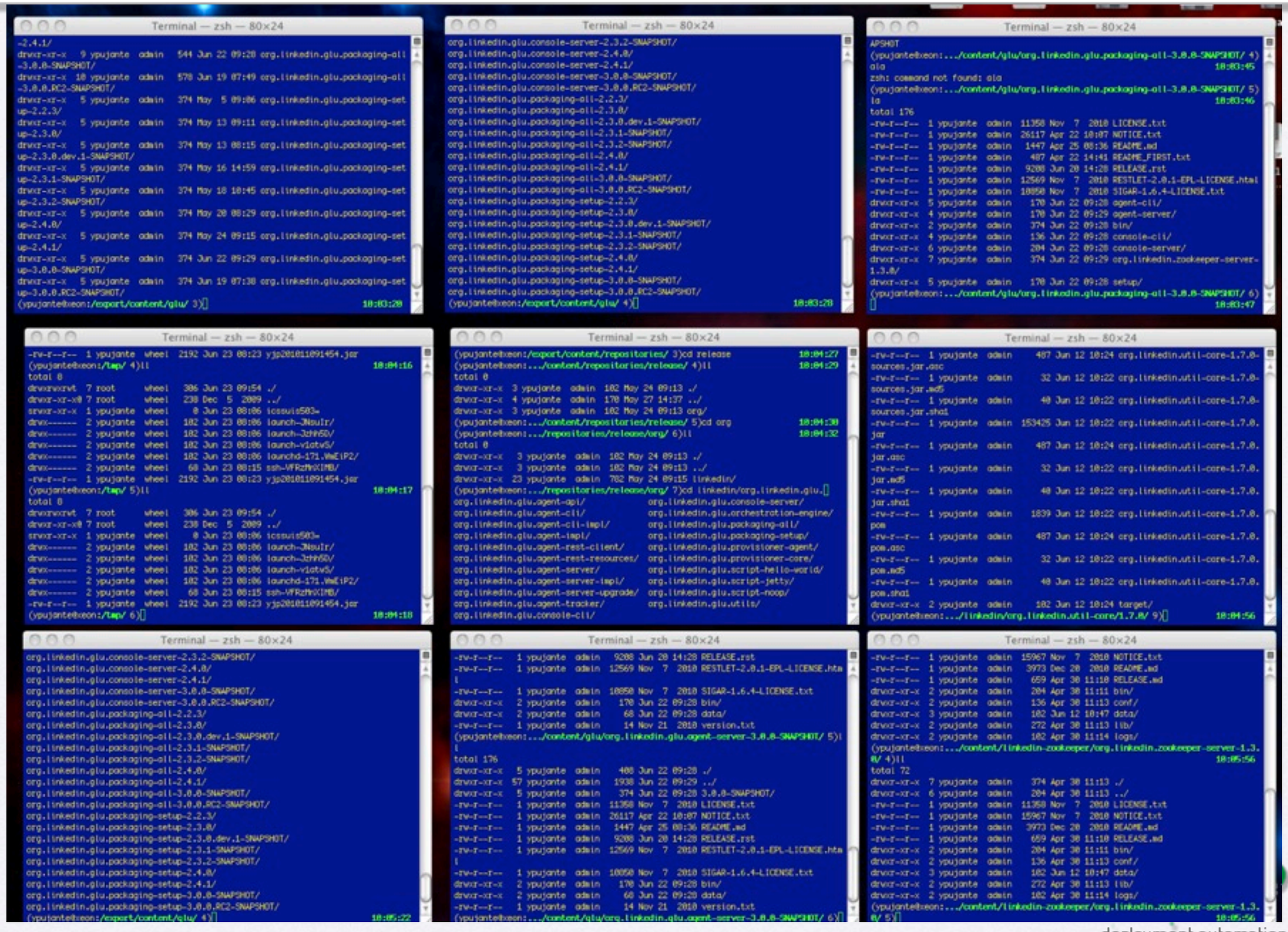
- Software engineer (16 years experience)
- Software is my passion (28 years! TI-99/4A)
- Currently *not* working... for a boss... :)
  - glu, kiwidoc (www.kiwidoc.com)
- Worked @ LinkedIn for 8 years (founding team!)
  - Worked on a lot of infrastructure projects and early features (security, payment, graph, etc...)
  - Last (big) project was glu (main author/contributor/maintainer)

deployment automation platform

Why glu ?

deployment automation platform

Before glu.... :'(



deployment automation platform

Before glu...

- Operations performs manual deployment:
  - ssh, rcp, etc...
  - non shared manually edited scripts

➡ extremely time-consuming

➡ error prone

deployment automation platform

glu project

- Address operations pain points
- Deploy (and monitor) applications to an arbitrary large set of nodes:
  - efficiently
  - with minimum/no human interaction
  - securely
  - in a reproducible manner
- ensure consistency over time (prevent drifting)
- detect and troubleshoot quickly when problems arise

deployment automation platform

After...

deployment automation platform

After... :)

glu-dev-1 | All [product]

Dashboard | Plans | System | Model | Admin | admin | Help

Recent **Deploy - Fabric [glu-dev-1] - PARALLEL**

Show Errors Only: ☑ Auto Refresh: ☑

Refresh Pause Abort

0/12 - 0%

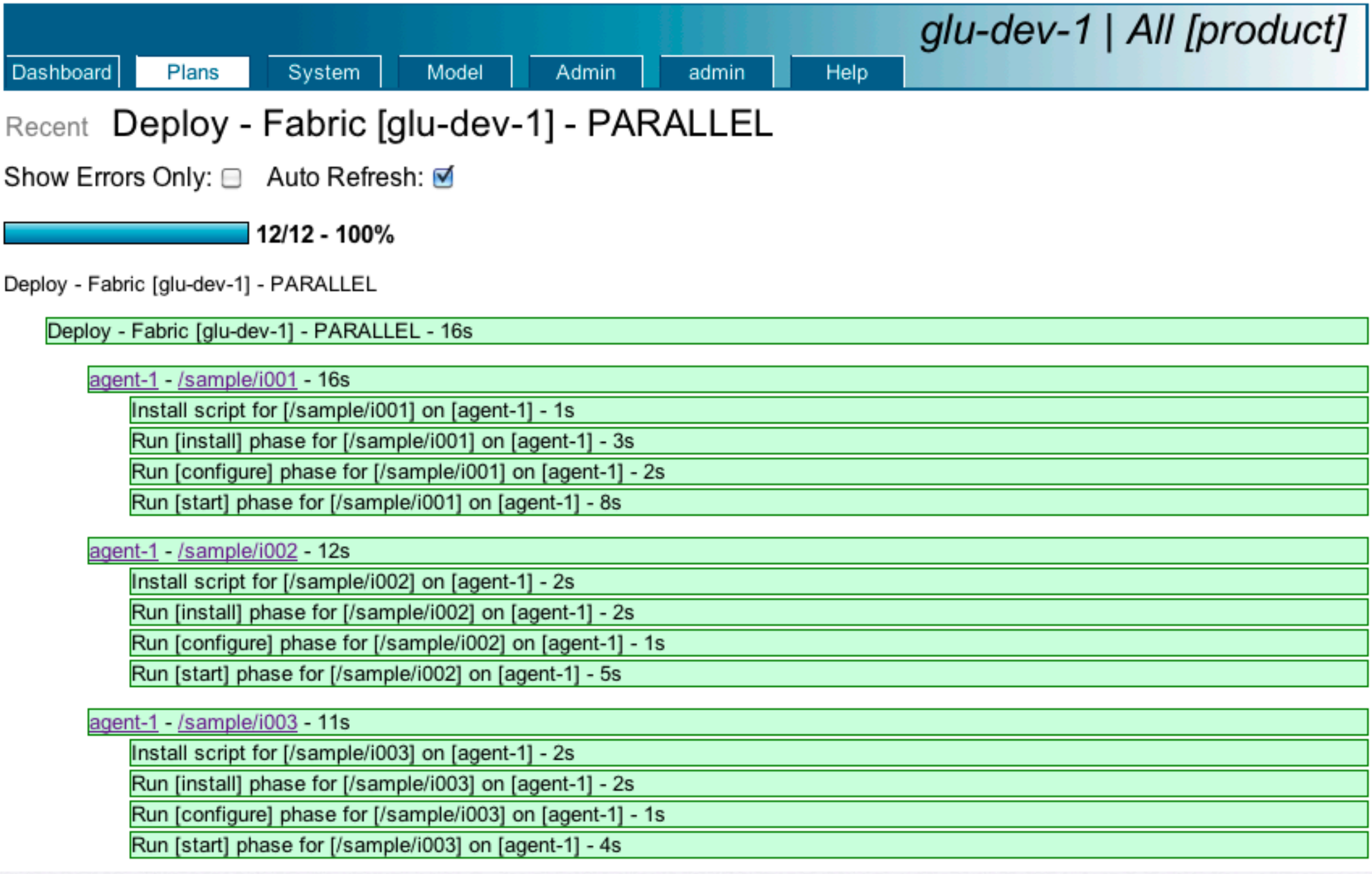
Nothing to do here...
Sit back and enjoy!

Deploy - Fabric [glu-dev-1] - PARALLEL

- Deploy - Fabric [glu-dev-1] - PARALLEL - running [1s]
  - agent-1 - /sample/i001 - running [1s]
    - Install script for [/sample/i001] on [agent-1] - running [1s]- [Cancel]
    - Run [install] phase for [/sample/i001] on [agent-1] - not started
    - Run [configure] phase for [/sample/i001] on [agent-1] - not started
    - Run [start] phase for [/sample/i001] on [agent-1] - not started
  - agent-1 - /sample/i002 - running [1s]
    - Install script for [/sample/i002] on [agent-1] - running [1s]- [Cancel]
    - Run [install] phase for [/sample/i002] on [agent-1] - not started
    - Run [configure] phase for [/sample/i002] on [agent-1] - not started
    - Run [start] phase for [/sample/i002] on [agent-1] - not started
  - agent-1 - /sample/i003 - running [1s]
    - Install script for [/sample/i003] on [agent-1] - running [1s]- [Cancel]
    - Run [install] phase for [/sample/i003] on [agent-1] - not started
    - Run [configure] phase for [/sample/i003] on [agent-1] - not started
    - Run [start] phase for [/sample/i003] on [agent-1] - not started

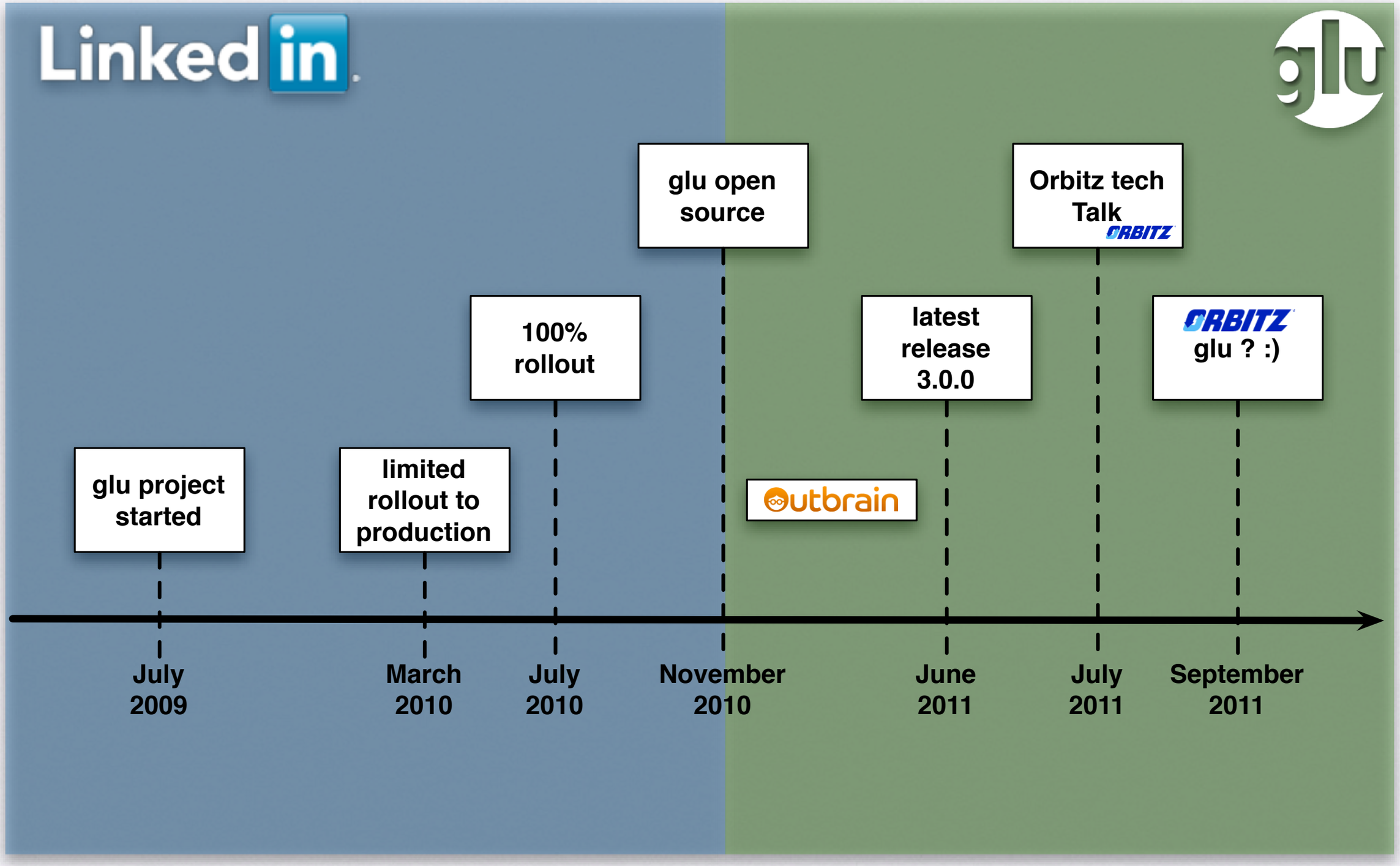
deployment automation platform

After... :D



deployment automation platform

History of glu



deployment automation platform

Rollout to production

- glu project started in July 2009
- Initial rollout to LinkedIn production in March 2010
- Gradual until full rollout in July 2010
- As of June 2011 LinkedIn glu numbers:
  - 5 different ‘fabrics’ (2 prod + 2 stg + 1 int. lab)
  - ~2650 nodes, ~9000 instances, ~300 services
- LinkedIn working on ‘glu on the desktop’ (dev)

deployment automation platform

glu open source

- Before I left LinkedIn, open sourced glu (~3 months effort)
  - 1.0.0 released in November 2010
  - 2.0.0 released in February 2011 (tagging)
  - 3.0.0 released in June 2011 (parent/child)
  - (~ 20 releases total... smaller releases)

deployment automation platform

glu interest

- since 11/2010, glu has generated a lot of interest
  - oubrain.com is using glu (integrated in CI!)
  - companies interested in glu: Orbitz, Netflix, GigaSpaces, Rearden Commerce, etc...
- some academic use (Budapest university)
- a lot of 'followers' on github
- lots of downloads

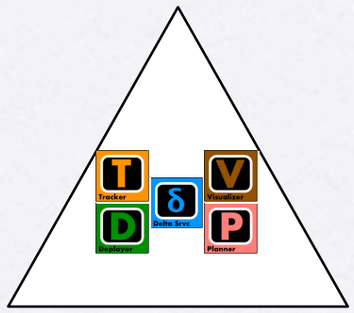
deployment automation platform

Architecture

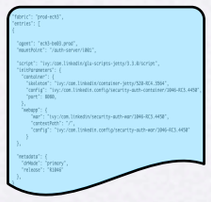
deployment automation platform

Components/Concepts

- 3 physical components



- 3 concepts



deployment automation platform

ZooKeeper

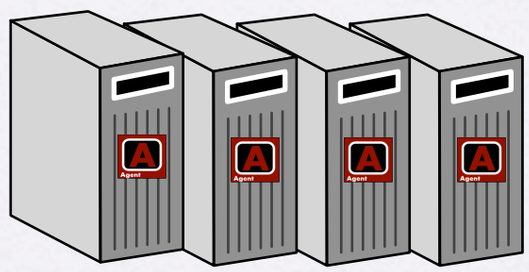
- 1 ZooKeeper cluster (3 or 5 instances enough)
  - ZooKeeper is an Apache project
  - similar to a (networked) filesystem (think nfs)
    - + 'directories' can also contain data
    - + ephemeral nodes
    - + powerful watcher concept => notifications
- ZooKeeper is used to maintain the state of the system

deployment automation platform

glu Agent

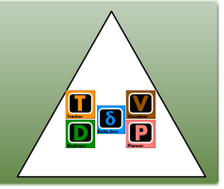
- 1 agent per node => as many agents as there are nodes
- agent is active process (groovy)
- (secure) REST API

- Reports its state to ZooKeeper



deployment automation platform

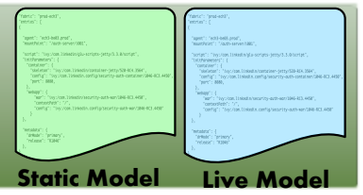
glu orchestration engine



- 1 orchestration engine
- runs inside a webapp
  - offers both browser and REST interface
- Listens to ZooKeeper events (to compute ‘live state’)
- Talks to the agents

deployment automation platform

Static/Live Model



- model is a json document which describes
  - where to deploy
  - what and how to deploy
- “Static” is what you want
- “Live” is what is actually deployed/running

deployment automation platform

Static Model: Where ?

```
{
  "fabric": "prod-chicago",
  "entries": [{
   "agent": "node01.prod",
   "mountPoint": "/search/i001",

   "script": "http://repository.prod/scripts/webapp-deploy-1.0.0.groovy",
   "initParameters": {
      "container": {
        "skeleton": "http://repository.prod/tgzs/jetty-7.2.2.v20101205.tgz",
        "port": 8080,
      },
      "webapp": {
        "war": "http://repository.prod/wars/search-2.1.0.war",
        "contextPath": "/"
}}}]}
```

- “agent” => node which runs this agent
- “mountPoint” => unique key
  - can deploy more than 1 ‘thing’ per agent

Static Model: What / How ?

```
{
  "fabric": "prod-chicago",
  "entries": [{
   "agent": "node01.prod",
   "mountPoint": "/search/i001",

   "script": "http://repository.prod/scripts/webapp-deploy-1.0.0.groovy",
   "initParameters": {
      "container": {
        "skeleton": "http://repository.prod/tgzs/jetty-7.2.2.v20101205.tgz",
        "port": 8080,
      },
      "webapp": {
        "war": "http://repository.prod/wars/search-2.1.0.war",
        "contextPath": "/"
}}}]}
```

- “script” => instructions about what ‘deploy’ means
- “initParameters” => parameters provided to the script

deployment automation platform

glu Script

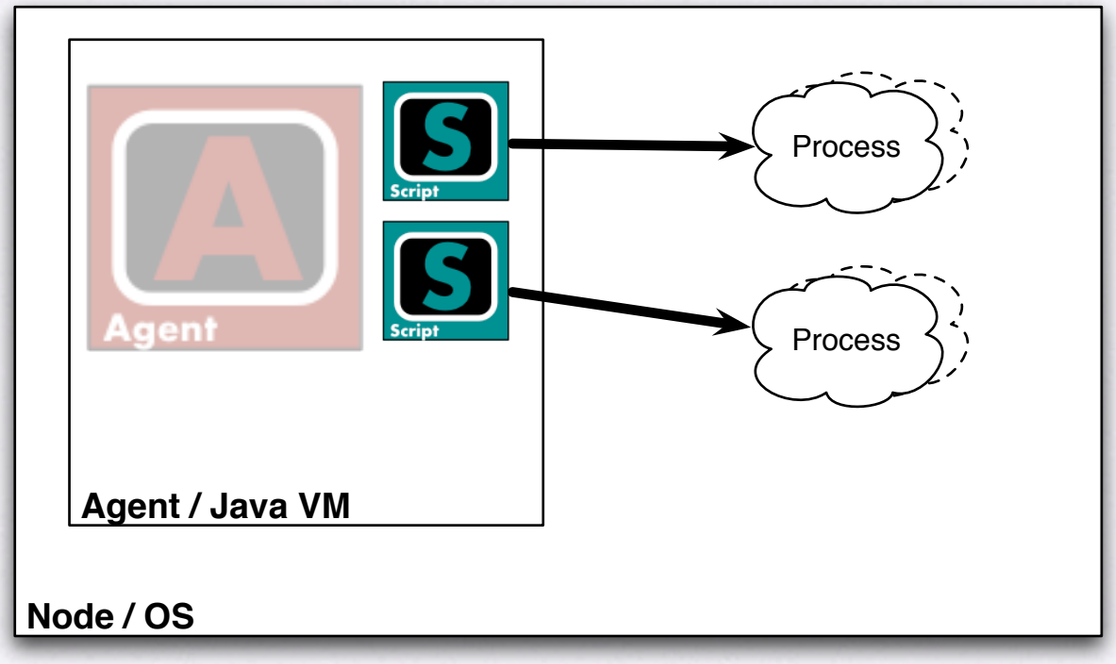
```
class MyGluScript
{
  def port
  def pid

  def install     = { /* install code */ }
  def configure   = { /* configure code */ }
  def start       = { /* start code */ }
  def stop        = { /* stop code */ }
  def unconfigure = { /* unconfigure code */ }
  def uninstall   = { /* uninstall code */ }
}
```

- groovy class which defines
  - a set of ‘phases’ (install, start, etc...) backed by a state machine
  - properties (exported to ZooKeeper)
- glu does not dictate what goes in each ‘phase’

deployment automation platform

glu Script runtime



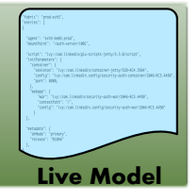
- glu Script code runs inside the (java) VM of the agent
- in general, a glu Script will spawn external processes (ex: webapp container, memcached, etc...) but it is not a requirement!

deployment automation platform

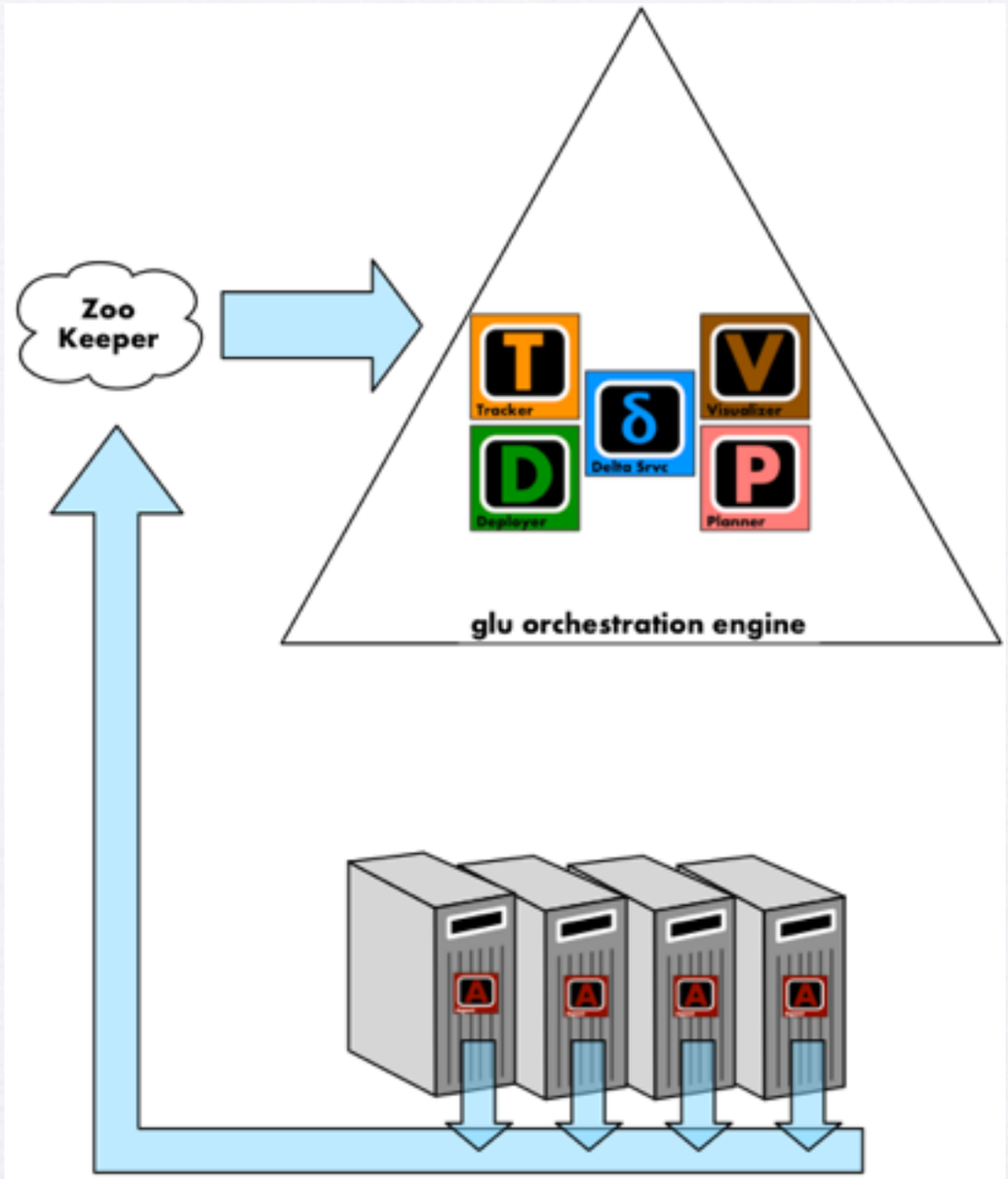
How does it all work ?

deployment automation platform

Live Model

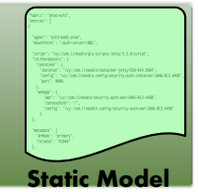


- each agent reports its state to ZooKeeper
- the orchestration engine listens to ZooKeeper and builds the ‘live’ model

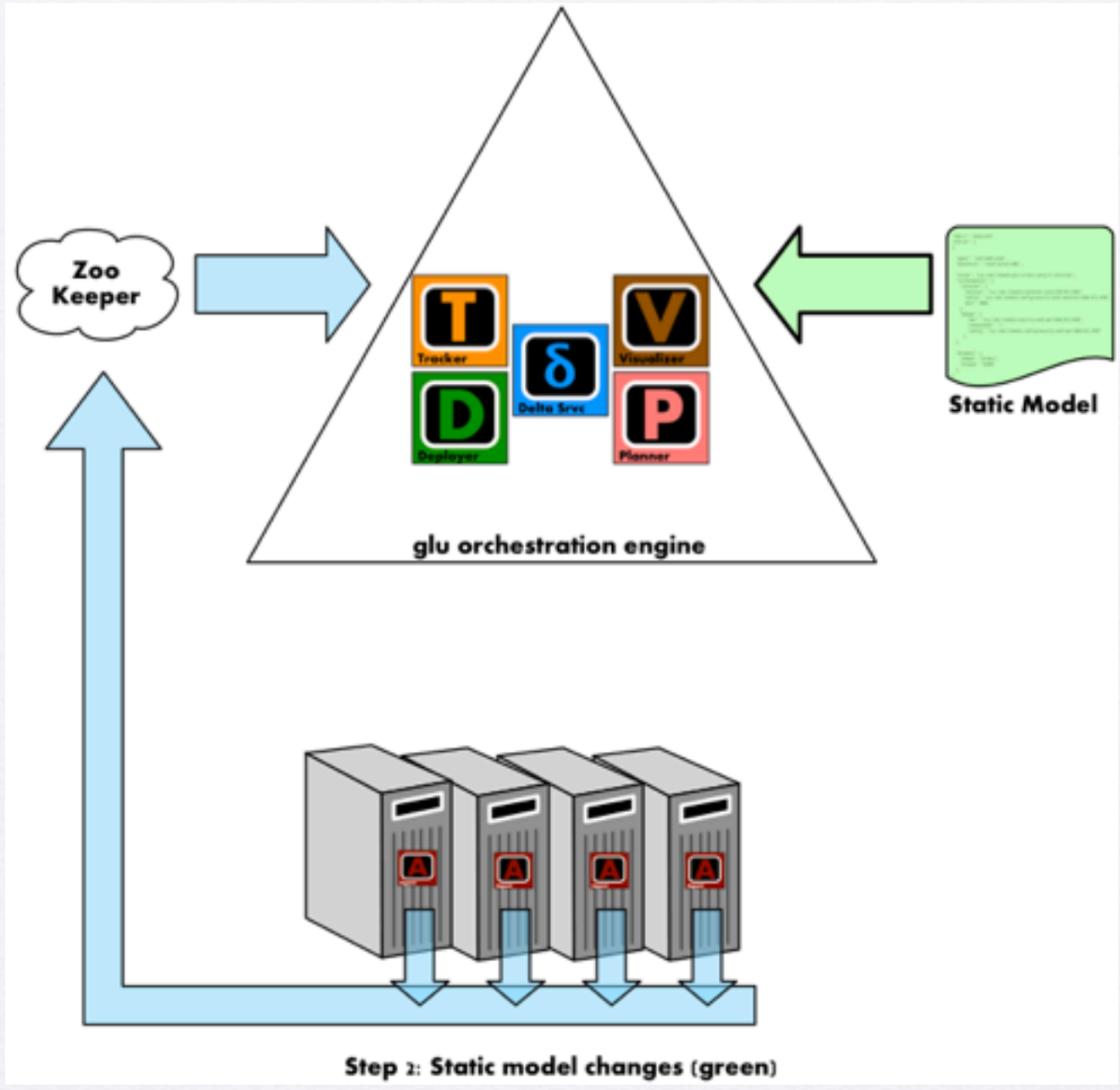


deployment automation platform

Static Model



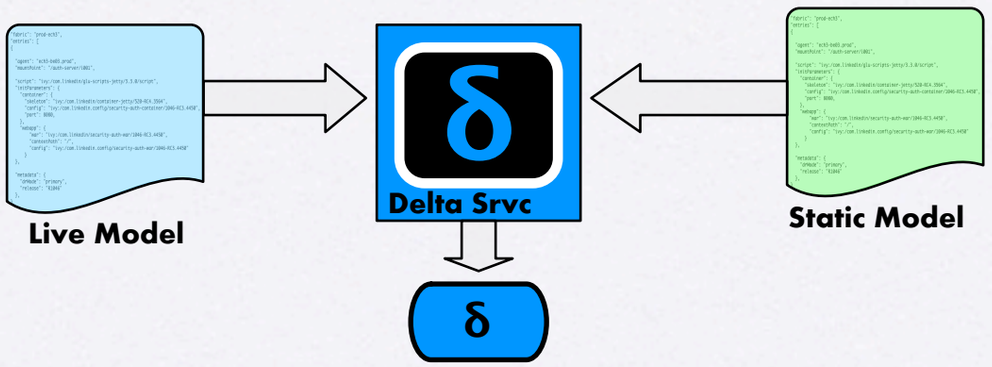
- the ‘static’ model is loaded in the orchestration engine



Step 2: Static model changes (green)

deployment automation platform

Delta Computation

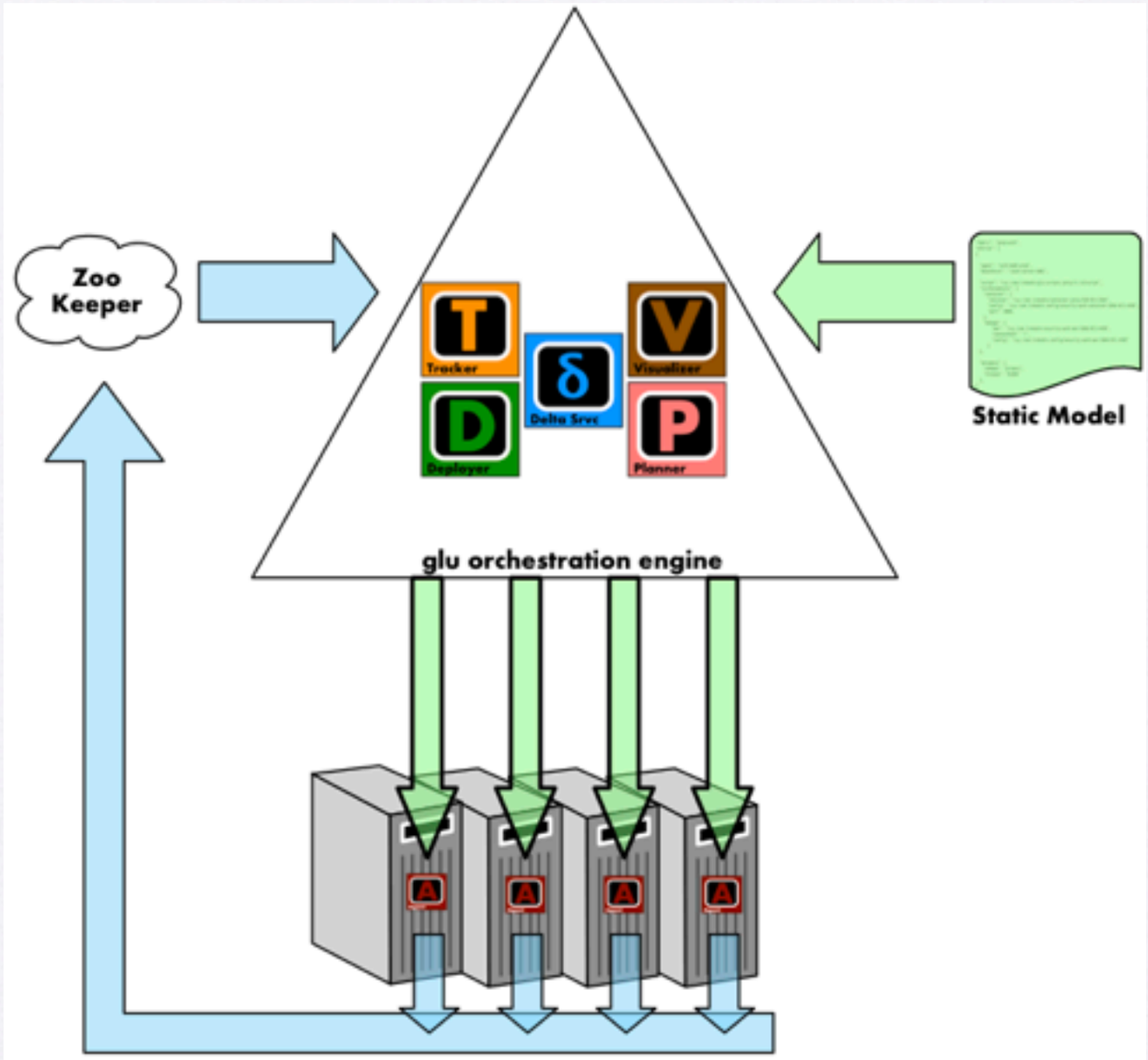


- orchestration engine computes a delta by comparing the static model and the live model
  - “desired” state vs “current” state

deployment automation platform

deployment plan

- delta is used to compute a deployment plan
- orchestration engine sends commands (REST) to the appropriate agents

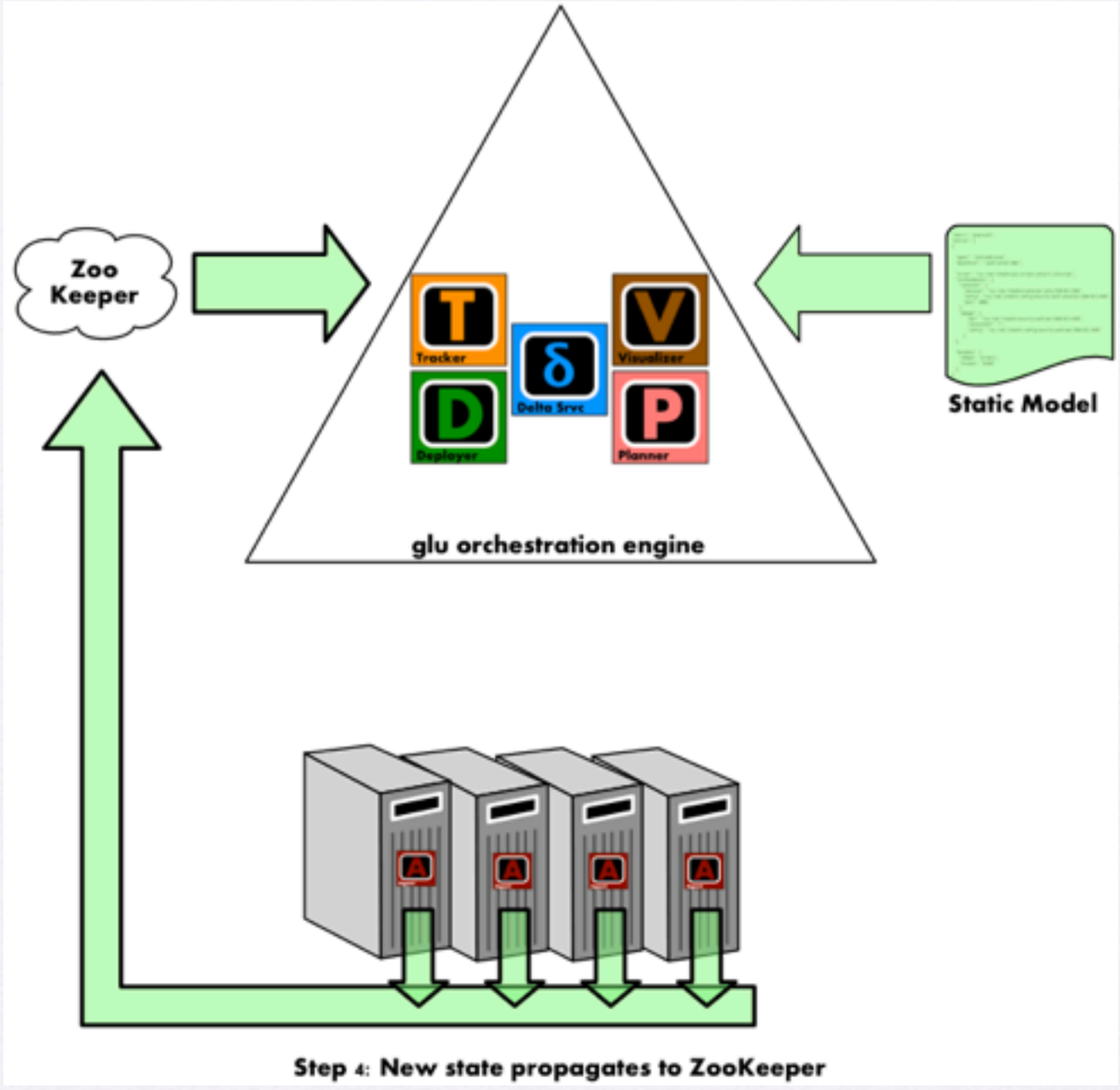


Step 3: glu executes a deployment plan (parallel) in response to the change

deployment automation platform

Live Model updated

- as the agents run the commands they update their state in ZooKeeper

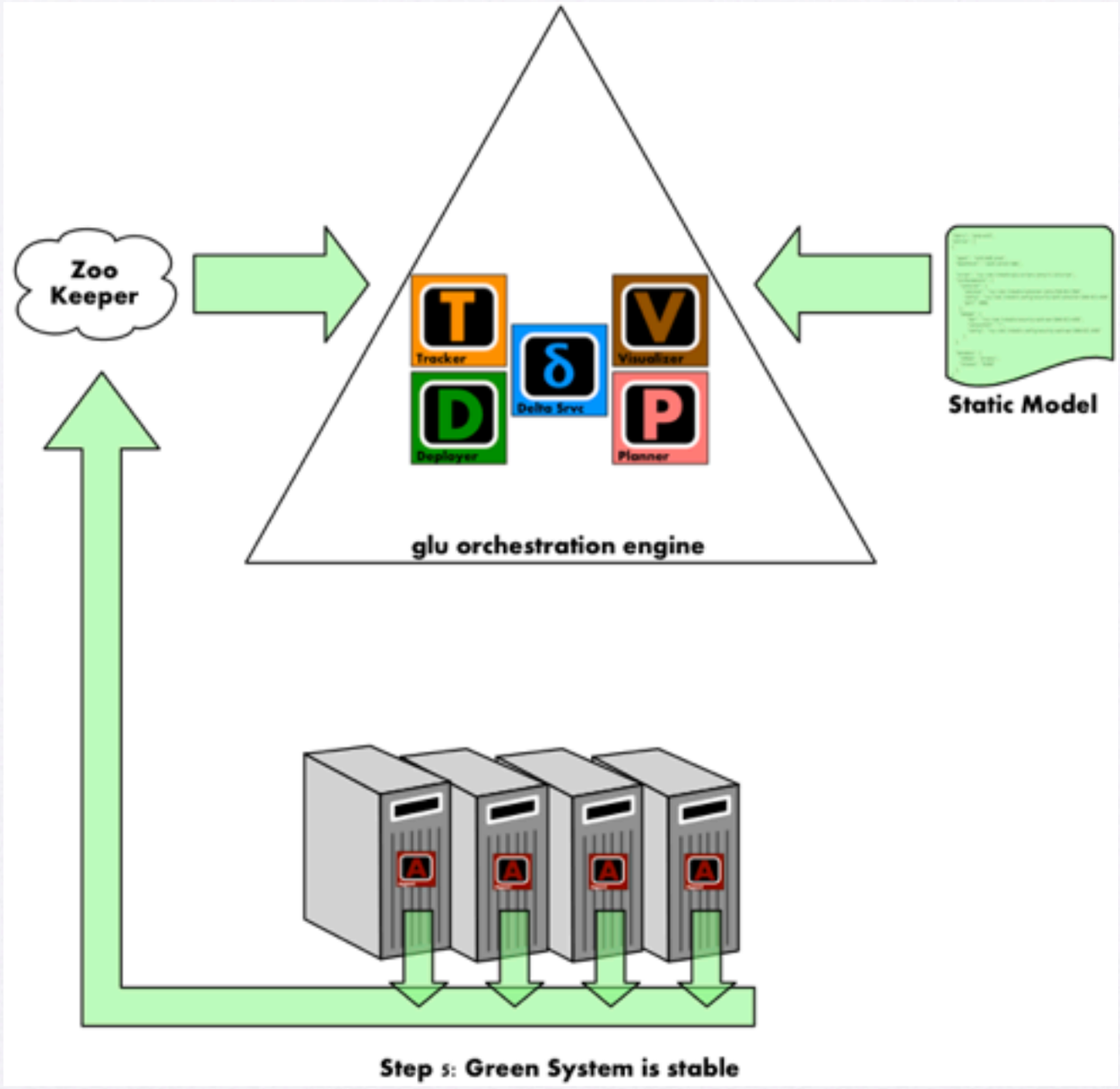


Step 4: New state propagates to ZooKeeper

deployment automation platform

System Stable

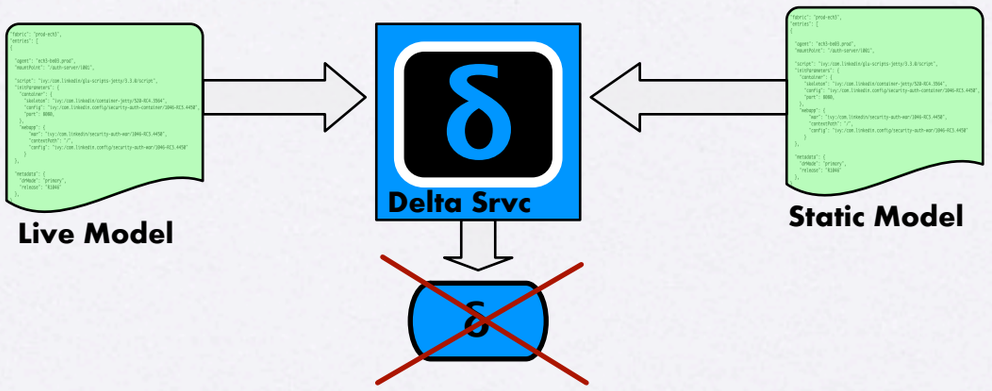
- The live model and the static model match
- => no more delta



deployment automation platform

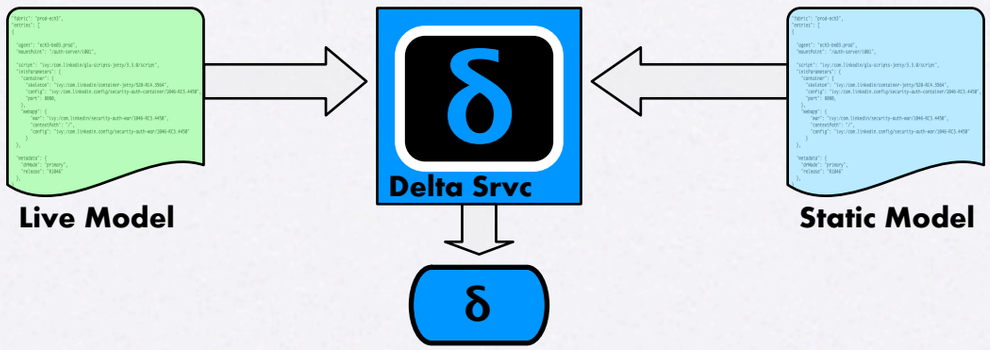
System Stable (no delta)

- remains stable until:
  - static model changes (ex: new version of software)
  - live model changes (ex: hardware crash)



deployment automation platform

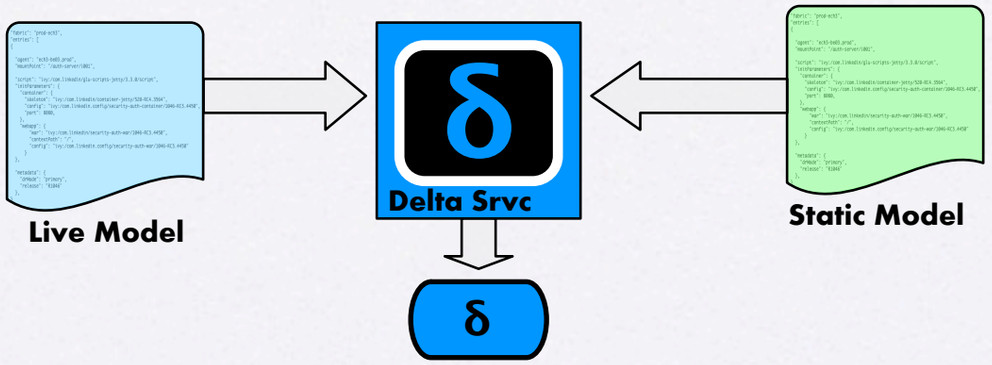
Static Model Changes



- Static model changes
  - ex: new version of software, new node, etc...
  - => delta => deploy/upgrade software, provision new nodes

deployment automation platform

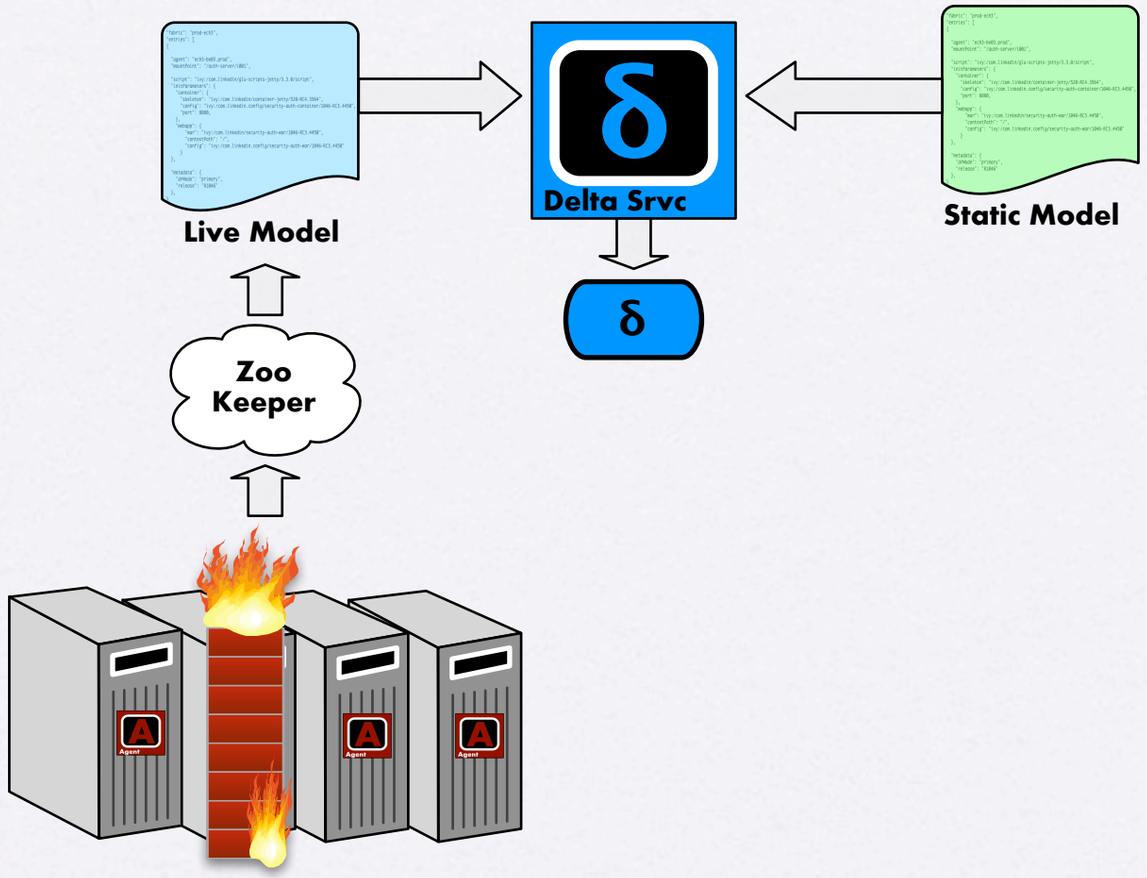
Live Model Changes



- Live Model changes
  - ex: hardware crash, bad behavior, high load, etc...
  - => delta => monitoring!

deployment automation platform

Monitoring: built-in

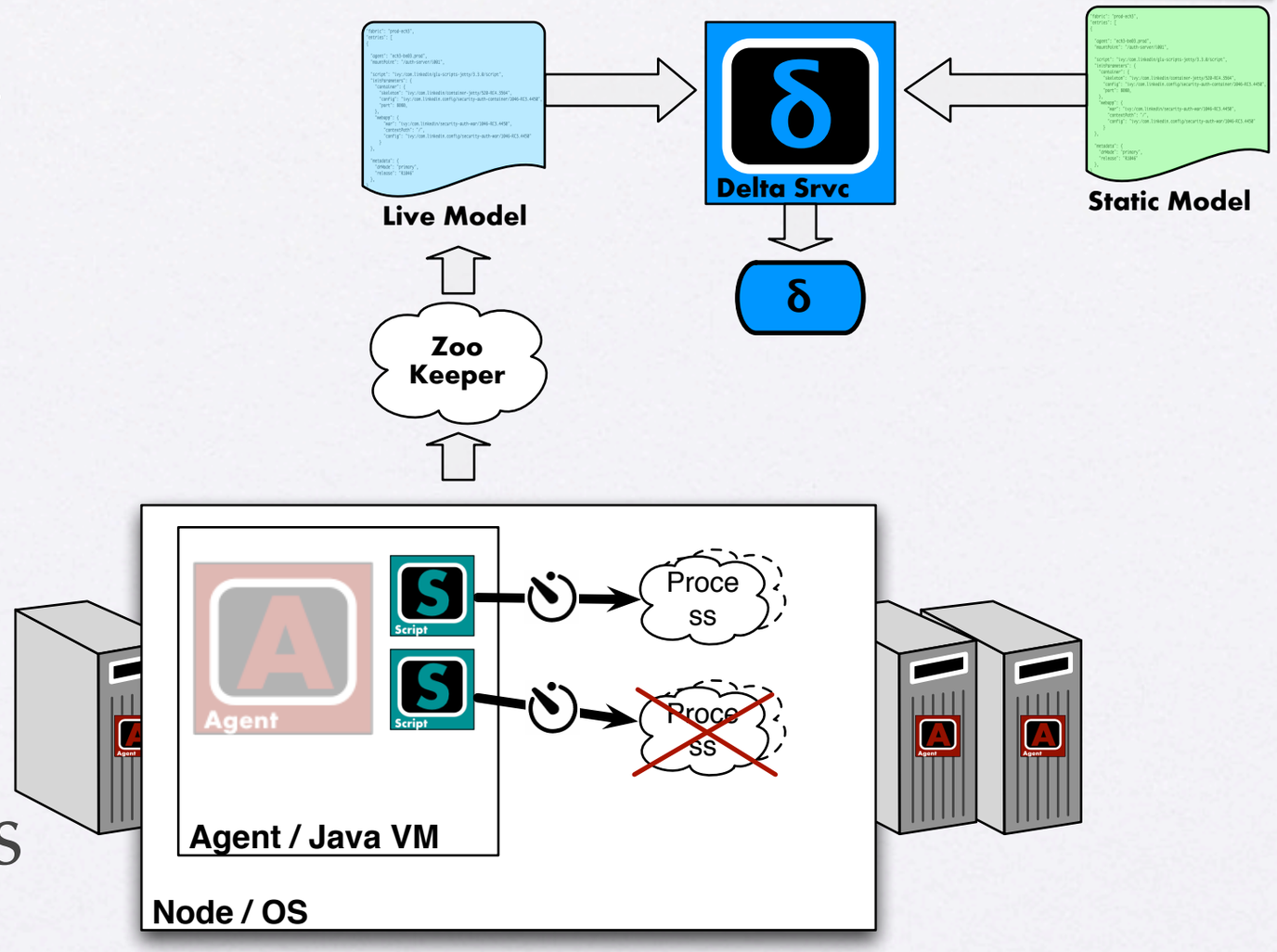


- agent registers a ZooKeeper ephemeral node
  - => when agent disappears, state changes!

deployment automation platform

Monitoring: add-on

- script runs in **“active”** agent
- agent has **“timer”** capability
- =>script can also monitor what it starts and change state when failure detected



deployment automation platform

Monitoring: advanced

- You can even build a full monitoring solution on top of glu
- Not enough time/space here :)
- Check out my blog (source examples included!) @ http://www.pongasoft.com/blog/yan/categories/glu/

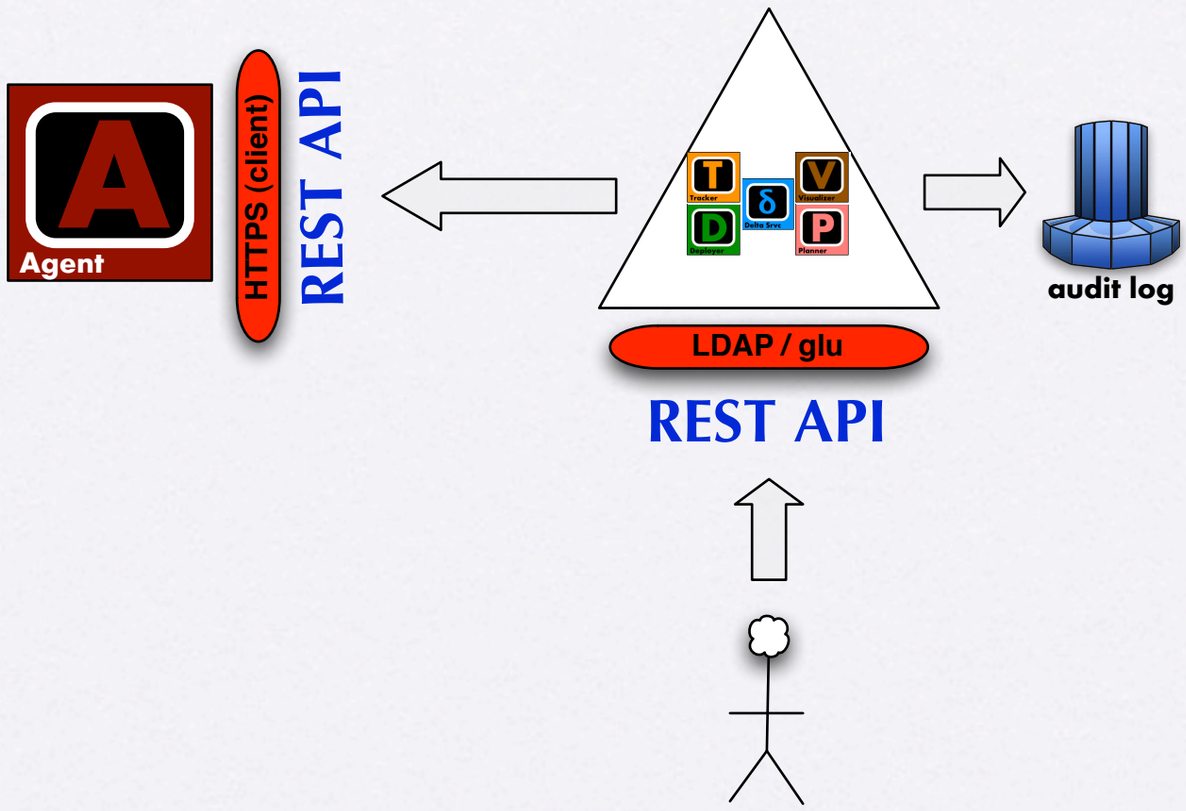
mountPoint:4	I:4	E:1	agent:1	status:2
/monitor	1	1	agent-1	ERROR High load detected...
/sample/i001	1	0	agent-1	running
/sample/i002	1	0	agent-1	running
/sample/i003	1	0	agent-1	running

deployment automation platform

What about security ?

deployment automation platform

Security



- User must authenticate (LDAP and/or glu)
- Agent REST API is ‘protected’ behind HTTP**S** with client auth
- Every ‘change’ is audited in the audit log

deployment automation platform

Live Demo...

deployment automation platform

glu as a **platform**

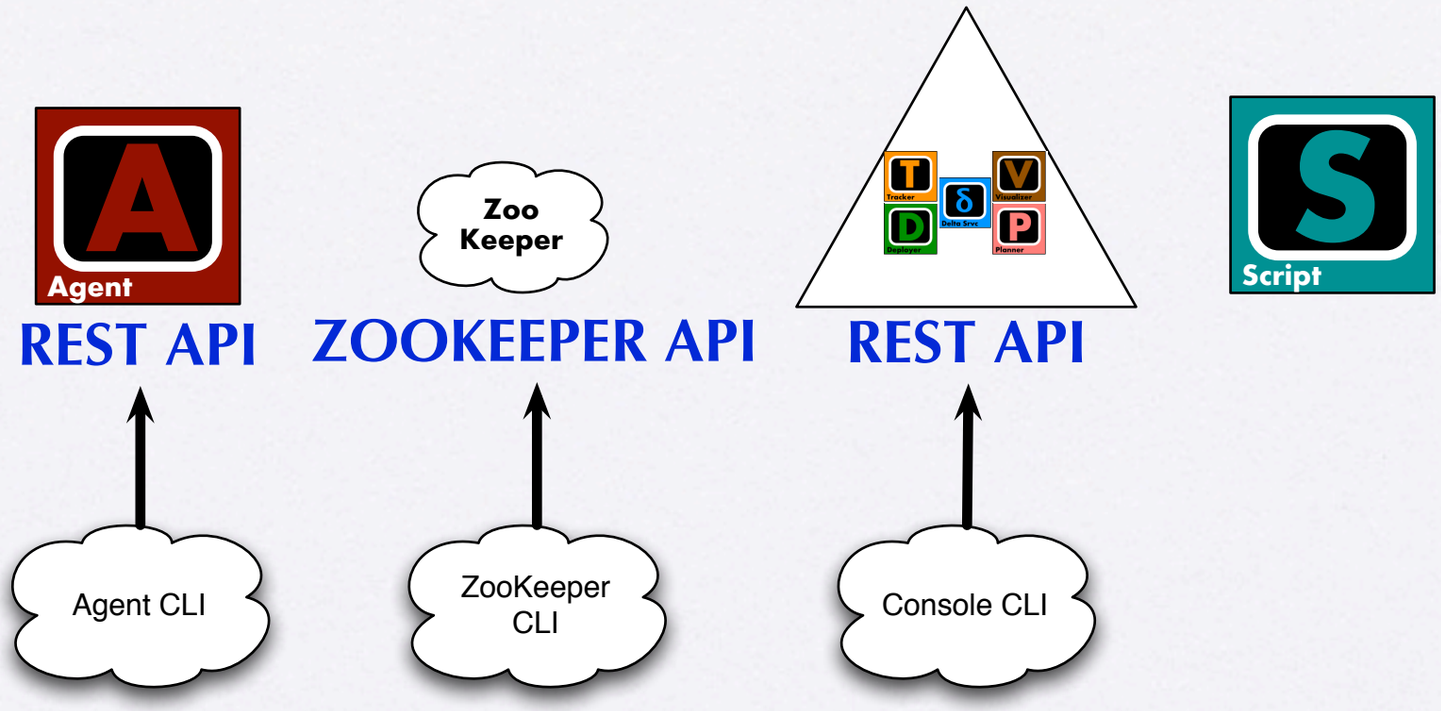
deployment automation platform

glu is more than a tool

- glu is a tool with a lot of customization points
- it is also a **platform** on top of which you can build your **own** deployment (and optionally monitoring) solution

deployment automation platform

APIs



- Agent CLI and Console CLI are mostly wrappers/ examples around the REST API
- => you can use the REST API directly or use the CLI

deployment automation platform

glu Script

- A glu script is any code you want (groovy/java) made easier by agent capabilities (but you don't have to use them!)

- **shell.exec** capability allow you to write your script in any language you want (will be 'promoted' native soon...)

```
class RubyGluScript {
  def install = {
    shell.exec("./ruby/install.rb")
  }
  def start = {
    shell.exec("./ruby/start.rb")
  }
}
```

deployment automation platform

Agent

REST API

- One way to look at the agent: script engine remotely accessible through a (secure) REST API
  - => can also be used on its own (no ZooKeeper or orchestration engine)

deployment automation platform

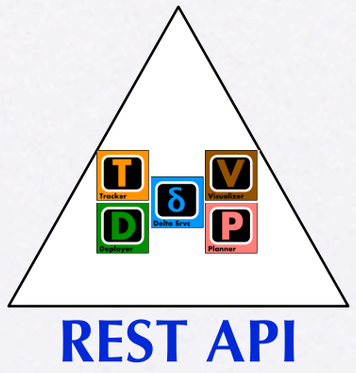
ZooKeeper

ZOOKEEPER API

- ZooKeeper is independently accessible
- => can build your own listeners/watchers directly
- => use AgentsTracker library which comes with glu (check the blog for more details)
- Ex: build a monitoring solution

deployment automation platform

Orchestration Engine



- For example, you can integrate your CI directly with glu by using the orchestration engine REST api (ex: outbrain.com)

- Although very customizable, you can also build your own UI if you do not like the one that comes with glu

deployment automation platform

Much more...

- Powerful tagging/filtering feature allow to create concepts that glu does not know about (ex: webapp, frontend, cluster, etc...)

- Query language allows you to slice & dice the models
  - => build higher level constructs (like dynamic node assignment)

deployment automation platform

glu vs puppet

- Disclaimer: I have spent 2 years with glu (I wrote it :-)) and 1 day with puppet...

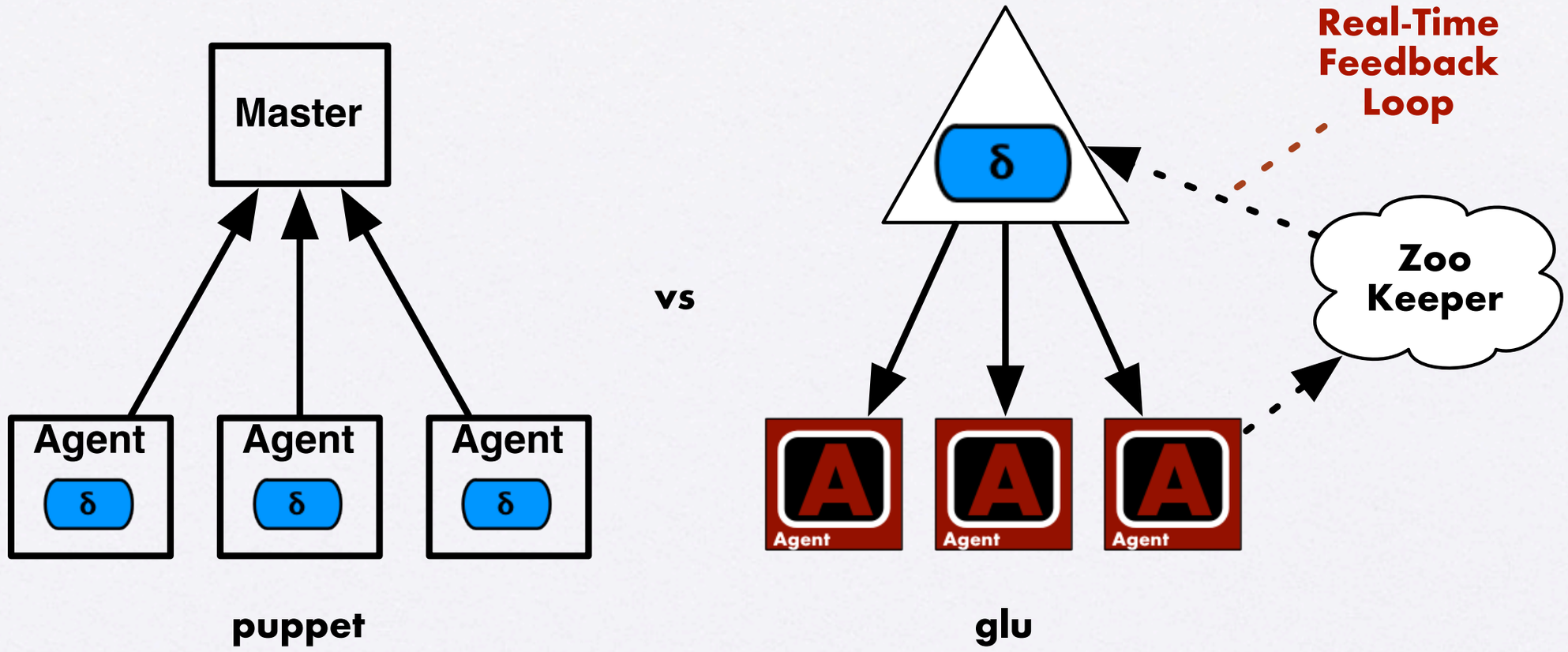
deployment automation platform

glu vs puppet

- Great news: intrinsically similar concepts
  - ‘desired’ vs ‘current’!
  - declarative approach
- Minor difference:
  - puppet is ruby vs glu is groovy/java

deployment automation platform

glu vs puppet: orchestration



- delta computation / orchestration takes place at a different level
  - => glu can orchestrate across nodes
  - => glu delta is system wide (and real-time)

deployment automation platform

glu vs puppet: conclusion

- puppet is very good at ***configuring*** the infrastructure of a machine (users, groups, packages, etc...)
  - => static/stable does not change often
- glu is very good at ***provisioning*** dynamic applications on an ensemble of machines (the system)
  - => changes often, real-time failure detection (monitoring), “bounce”, etc...

deployment automation platform

glu can use puppet :)

```
class PuppetGluScript {
  def puppetManifest

  def install = {
    // download manifest
    puppetManifest = shell.fetch(params.puppetManifestURI)
  }

  def start = {
    // execute manifest
    shell.exec("puppet apply ${puppetManifest}")
  }
}
```

deployment automation platform

References

deployment automation platform

References

- glu source: github.com/linkedin/glu (links to all you need)
- blog: www.pongasoft.com/blog/yan
- twitter: @glutweets

deployment automation platform