Best Practices for a Mission-Critical Jenkins

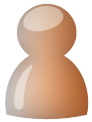
Mike Rooney
Jenkins Connoisseur

http://linkedin.com/in/mcrooney

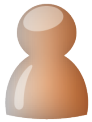
Jenkins Uses

- Genius.com
  - staging deployment, code reviews, automated branching and merging, monitors
- Canv.as
  - continuous deployment, scoring, monitoring, newsletter mailing
- Conductor
  - environment creation, staging / prod deployment, selenium monitoring

Hand-check: How critical is your Jenkins?



What problems have you faced?



Problems

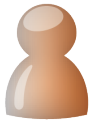
- disk failure / data loss
- hardware failure / downtime
- load / latency

Solution

- make Jenkins instance trivial to respin
  - ideally a one-liner that even handles DNS
  - “create.sh jenkins”

Persistence

- $JENKINS_HOME
  - plugins, users, jobs, builds, configuration

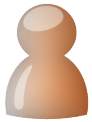


Persistence

- git / svn
  - make $JENKINS_HOME a checkout
  - have a Jenkins job that commits daily
  - examples: http://jenkins-ci.org/content/keeping-your-configuration-and-data-subversion

Persistence

- EBS on AWS
  - put $JENKINS_HOME on an EBS volume
  - snapshot nightly via a Jenkins job



Environment

- Jenkins is more than a .war
  - specific Jenkins version
  - startup options
  - dependent packages: git, ruby gems, pip
  - ssh keys, m2 settings
  - swap, tmpfs, system configuration

Environment

- configuration management:Puppet/Chef*

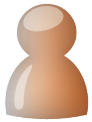
```
class jenkins () {
    package {"jenkins":
        ensure  => "installed",
        provider => "rpm",
        source => "http://pkg.jenkins-ci.org/redhat/jenkins-1.460-
    }

    package {["git", "rubygem-json"]:
        ensure  => "installed",
    }

    python::module {["robotframework", "robotframework-seleniumlib

    file {"/opt/tomcat7/.ssh/known_hosts":
        ensure  => "file",
        content => "puppet:///modules/jenkins/ssh_known_hosts",
    }
}
```

* https://wiki.jenkins-ci.org/display/JENKINS/Puppet



Environment

- standalone
  - puppet apply path/to/your/manifest.pp
- puppetmaster
  - set up /etc/puppet.conf, run puppet agent

Putting it Together

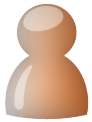
- have manifest handle $JENKINS_HOME
  - clone git repo, mount EBS volume, etc

Putting it Together…on AWS

- upload manifests to S3 on check-in
  - a Jenkins SCM job using S3 plugin
- use cloud-init to install puppet, download manifests, and run puppet
  - a custom AMI with an rc.local script also works
- when it dies: "create.sh jenkins"
  - ec2-launch-instance config user-data

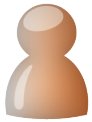
Monitoring

- … but how do you know when it's down?
- check out services like Pingdom
  - notifies you when a URL does give HTTP 200 OK



Going further: Elastic Beanstalk

- handles provisioning simply from a .war
- pros
  - just give it a war
  - automatically replaces unhealthy instances
  - behind a load-balancer (consistent URL)
  - normally hard AWS changes like AMI, Security Groups, or Key Pairs are now trivial to make
- cons
  - behind a load-balancer (cost overhead)
  - no UI option (yet) for controlling AZ
  - no great way to pass data to instances for puppet
  - locked in to Amazon Linux AMI (CentOS)



Going further: Elastic Beanstalk

- set min/max instances to 1
  - ignore scaling triggers, irrelevant in this case
- use beanstalk CLI to set desired AZ (if EBS)
  - https://forums.aws.amazon.com/thread.jspa?threadID=61409
- puppet
  - use a custom AMI that specifically runs Jenkins manifests
  - but this requires a specific AMI for each Beanstalk application.
  - let's get creative…

Going further: Elastic Beanstalk

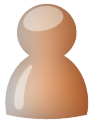
- passing data to instances

Environment Properties

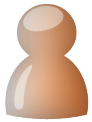
These properties are passed into the application as environment variables. Learn more >>

AWS_ACCESS_KEY_ID	
AWS_SECRET_KEY	
JDBC_CONNECTION_STRING	
	Note: Connection string to JDBC database (e.g. RDS) for application use.
PARAM1	puppet_role=beanstalk::jenkins

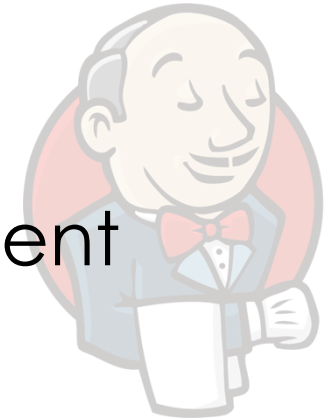
- PARAM1..5 meant as args to .war
- end up in /etc/sysconfig/tomcat7 JAVA_OPTS
- parse out and:
  - puppet apply –certname=$PARSED_ROLE



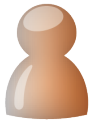
Questions?



High Availability Artifacts



- protect: artifacts, reports, userContent
- from:
  - planned downtime:
    Jenkins restarts/upgrades, server upgrades
  - unplanned downtime:
    software/hardware failure



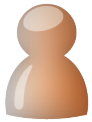
High(er) Availability Artifacts

- easy mode:
  - put Jenkins behind nginx/apache, shadow userContent and relevant directories
  - still available during Jenkins restarts, or very high Jenkins load/latency
  - not safe from server downtime

High Availability Artifacts

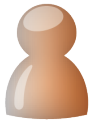
- advanced mode: S3
  - 99.99% availability, 99.999999999% durability*
    - if you store 10K objects, expect to lose one every 10 million years
  - use Jenkins S3 plugin to upload artifacts to S3

* http://aws.amazon.com/s3/faqs



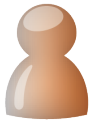
Fault-tolerant Jobs

- design with possible downtime in mind
  - SCM triggering is great, but keep polling too



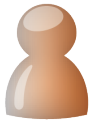
Fault-tolerant Jobs

- */15 * * *
  - BAD:
    update users where join_time < 15m ago
  - GOOD:
    update users where id > last_id_updated

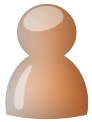


Error handling

- for non-critical jobs, use email / IM post-build notifiers
  - but be careful of creating too much noise, people will ignore or filter it out
- for critical jobs, integrate Jenkins with a service like PagerDuty
  - Jenkins emails myalert@pagerduty.com
  - PagerDuty texts / calls the people on-call until resolved
  - a failing build will wake you up at 4AM



Questions?



Security: Authentication

- read-only
- matrix-based
- HTTP basic auth

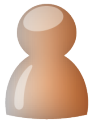
Security: Authentication

- but what about traffic sniffing?

Security: HTTPS

- throw nginx/apache in front of Jenkins

  #EXAMPLE

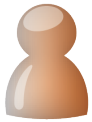


Security: Authorization

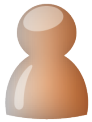
- use project-based matrix authentication
- give anonymous/authenticated readonly
- use it if you've got it:
  LDAP, Active Directory, UNIX
- Jenkin's own database also works fine
- ensure each user has their own account

Security: Authorization (AWS)

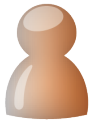
- when interfacing with AWS API/CLI, use IAM so Jenkins can only access what it needs



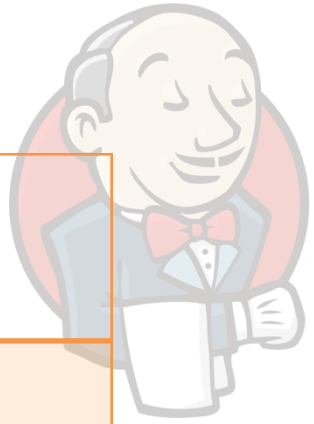
Security: Audit Trails



Questions?



Thank You To Our Sponsors



Platinum Sponsor	CloudBees
Gold Sponsors	JFrog, CLOUDANT
Silver Sponsors	LIFERAY, SendGrid Email Delivery. Simplified., CLOUDSMITH
Bronze Sponsors	WAN DISCO