Automation strategies for deploying Grails from Dev to Prod

Eric Helgeson
@nulleric

$ whoami

- Works @ Agile Orbit
- Full stack Engineer
- @nulleric

July 31st National Sysadmin Day

http://sysadminday.com/

Agenda

- Everything as Code
- Architecting for Production
- Infrastructure Testing
- Operating in Production

Everything as code

Why?

- Reproducible
- Reliable/Testable
- Shareable
- Communicate changes
- Deliver software more quickly

Automate our automation - Jenkins

Pains

- When Jenkins goes down development stops
- Updates (Jenkins version, Plugins, Jobs)
- Change job definitions

Jenkins as Code

- Jenkins completely described as code
- Using Chef and Jenkins Job DSL
- Ability to test changes locally (or have Jenkins build itself) in CI
  - https://erichelgeson.github.io/blog/2014/05/10/automating-your-automation-federated-jenkins-with-chef/

Plugins

```
plugins = {
  'greenballs'                  => '1.14',
  'credentials'                 => '1.22',
  'ssh-credentials'             => '1.11',
  'scm-api'                     => '0.2',
  'git-client'                  => '1.17.1',
  'git'                         => '2.3.5',
  ...
}

plugins.each_with_index do |(plugin_name, plugin_version), index|
  jenkins_plugin plugin_name do
    version plugin_version
    action  :install
  end
end
```

Auth

```
jenkins_script 'gh_authentication' do
  command <<-EOH.gsub(/^ {4}/, '')
    import jenkins.model.*
    import hudson.security.*
    import org.jenkinsci.plugins.*
    def instance = Jenkins.getInstance()
    def githubRealm = new GithubSecurityRealm(
      '#{node['jenkins-chef']['github']['site']}',
      '#{node['jenkins-chef']['github']['api_endpoint']}',
      '#{node['jenkins-chef']['github']['API_KEY']}',
      '#{node['jenkins-chef']['github']['API_SECRET']}'
    )
    instance.setSecurityRealm(githubRealm)
    def strategy = new #{node['jenkins-chef']
['AuthorizationStrategy']}()
    instance.setAuthorizationStrategy(strategy)
    instance.save()
  EOH
end
```

JDKs

```
java_ark 'oracle-jdk7_71-x86_64' do
 app_home '/opt/jdks/oracle-jdk7-x86_64'
  owner 'jenkins'
  group 'jenkins'
end

java_ark 'oracle-jdk8_45-x86_64' do
 app_home '/opt/jdks/oracle-jdk8-x86_64'
  action :install
  owner 'jenkins'
  group 'jenkins'
end
```

Jobs (Grails2)

```
class GrailsCiJobBuilder {

    String name
    String description
    String gitUrl

    Job build(DslFactory dslFactory) {
        dslFactory.job(name) {
            it.description this.description
            logRotator(-1, 5, -1, -1)
            scm {
                git this.gitUrl, 'master'
            }
            triggers {
                scm '*/5 * * * *'
            }
            steps {
                shell 'chmod a+x ./grailsw'
                grails {
                    target 'test-app'
                    target 'war'
                    grailsWorkDir './grails-work'
                    projectWorkDir './project-work'
                    projectBaseDir ''
                    serverPort ''
                    useWrapper true
                }
            }
            publishers {
                archiveArtifacts 'target/**/*.war'
                archiveJunit 'target/test-reports/
**/*Spec.xml'
            }
        }
    }
}
```

```
[
  [name: "Example Grails", gitUrl: "erichelgeson/grails2"]
  [name: "Another Example", gitUrl: "erichelgeson/grails2-demo"]
  ...
].each {
  new GrailsCiJobBuilder(
      name: "$basePath/$it.name",
      description: 'An example using a job builder',
      gitUrl: it.gitUrl
      emails: developers,
  ).build(this)
}
```

https://github.com/sheehan/job-dsl-gradle-example/

Jobs (Grails3/gradle)

```
class GradleCiJobBuilder {
    String name
    String description
    String gitUrl
    String gitBranch = 'master'
    String pollScmSchedule = '@daily'
    String tasks
    String switches
    Boolean useWrapper = true
    String junitResults = '**/build/test-results/*.xml'
    String artifacts = '**/build/libs/*.jar'
    List mailerRecipients = []

    Job build(DslFactory dslFactory) {
        dslFactory.job(name) {
            it.description this.description
            logRotator(-1, 5, -1, -1)
            scm {
                git this.gitUrl, gitBranch
            }
            triggers {
                scm pollScmSchedule
            }
            steps {
                gradle(tasks, switches, useWrapper)
            }
            publishers {
                archiveArtifacts artifacts
                archiveJunit junitResults
                mailer(mailerRecipients.join(' '), false, true)
            }
        }
    }
}
```

https://github.com/sheehan/job-dsl-gradle-example/

Demo

Everything as code

- Jobs are Groovy Classes
- Changes reviewed via Pull Request
- More -
  - https://github.com/sheehan/job-dsl-gradle-example
  - https://github.com/jenkinsci/job-dsl-plugin

Development Environments

Pains

- Ensuring everyone has the same dependancies
  - JDK/Database/Redis/etc
- Onboard new developers
- Differences between production and development
  - “Works on my machine”

Configuration Management

- Define environments as code
  - Chef, Docker, Ansible, etc
- Write (unit/integration) tests for infrastructure
- Dev boxes are the same as Production boxes
- Keep infrastructure code in the same repo as app

Vagrant

- Pulls everything together
  - App + Dependencies + Configuration + Virtualization as code
- Each developer has own instance
  - Locally or in Cloud

Vagrant

```
Vagrant.configure(2) do |config|
  config.vm.box = "boxcutter/centos71"

  config.vm.network "private_network", ip: “192.168.50.2”

  config.vm.provider "virtualbox" do |vb|
    vb.memory = "4096"
  end

  config.berkshelf.berksfile_path = “chef-repo/cookbooks/grails-app/Berksfile"
  config.vm.provision "chef_zero" do |chef|
    chef.install = true
    chef.version = "12.4.1"
    chef.cookbooks_path = "chef-repo/cookbooks"
    chef.add_recipe “grails-app"
    chef.json = {
      “grails-app" => {
        "user" => "vagrant",
        "jar" => “/vagrant/build/libs/grails-app-0.1.jar"
      }
    }
  end
end
```

Vagrantfile is Ruby

```
# Grails 3
File.open(“build.gradle") do |f|
  f.each_line do |line|
    if line =~ /^version/
      puts line.split(" ")[1]
    end
  end
end
```

```
# Grails 2
File.open(“application.properties") do |f|
  f.each_line do |line|
    if line =~ /^app.version/
      puts line.split("=")[1]
    end
  end
end
```

grails-app-server

- Opinionated linux environment for a Grails2/3/boot app
- References architecture for you to take and modify
- https://github.com/erichelgeson/grails-app-server

Demo

Building for production

Containers

- Jars are our containers
- $ grails3 package
- $ grails2 build-standalone

Why?

- Isolate JVM (class loader, etc)
- Single artifact
- Bundle different servlet containers for different parts of your app (tomcat for web, jetty for solr)
- Deployments == (s)cp
  - Hot deploys never work

Config

- Environment specifics to the app
  - Render template .groovy file or from your CM
  - Get from KV store such as Consul, etcd, zookeeper
  - Inject as ENV vars (Grails 3 switch to application.groovy)

```
environments {
  production {
      dataSource {
          dbCreate = "none"
          driverClassName = "org.postgresql.Driver"
          dialect = "org.hibernate.dialect.PostgreSQL9Dialect"
          url = "jdbc:postgresql://${System.getenv('DB_HOST')}:$
  {System.getenv('DB_PORT')}/${System.getenv('DB_NAME')}"
          username = "${System.getenv('DB_USER')}"
          password = “${System.getenv(‘DB_PASS')}"
          ...
      }
    }
}
```

Secrets as code

- https://square.github.io/keywhiz/
  - Java, familiar, uses FuseFS to present secrets
- https://vaultproject.io/
  - Go, API driven, ties in with other Hashicorp tools
- Chef-Vault || Encrypted Databags
- Amazon KMS/IAM Profiles/etc

Sessions

- Avoid sticky sessions/session migration
- Move sessions redis/memcached/Cookie
- Memcached Session Manager (Grails 2/3)
  https://github.com/erichelgeson/grails3-memcached-session
  https://github.com/erichelgeson/grails-standalone-tomcat8-memcached
- Spring-session-redis (Grails 3)
  https://github.com/erichelgeson/grails3-redis-session
- Cookie Session Plugin (Grails 2)

Runtime changes

- Log Reloading
  - Log4j - LogManager.resetConfiguration()
  - logback - scan('30 seconds')

Database Changes

- Get comfortable with writing schema and data migrations
- Backwards compatible schema
  - GORM getters/setters to the rescue

```
String getPhoneNumber() {
  return phoneNumbers.first()
}
```

Infrastructure testing

You wouldn't put a application change in without testing it.

Lint & Unit

- Fail your CI pipelines early before you spin up a VM
- ShellCheck && BATS (if you still love bash)
- Foodcritic/Rubocop && ChefSpec

Integration
(You're already doing it)

```
$ ps aux | grep grails-app
```

vs

```
describe service('grails-app') do
  it { should be_running }
end
```

serverspec

- Ruby DSL
  - Run directly inside of Chef - or -
  - Can be run over ssh (if you're not using CM)
- http://serverspec.org/resource_types.html

Examples

```
describe service('nginx') do
  it { should be_enabled }
  it { should be_running }
end

describe port(80) do
  it { should be_listening }
end

describe command("/usr/bin/nginx -t") do
  it "nginx syntax ok" do
    expect(subject.exit_status).to eq 0
  end
end

describe file('/opt/grailsapp/config.groovy') do
  it { should be_file }
  it { should be_owned_by 'grailsapp' }
end
```

Operating in production

Centralized Logging

- Centralized view of what is going on
- Lots of tools (Splunk, ELK, paper trail, etc)
- https://github.com/erichelgeson/simple-elk-docker

Centralized Stats

- Allow developers and operations to add metrics easily (statsd mentality)
- Measure everything, graph many, alert actionable

Influxdb + Grafana + collectd

- statsd/graphite/jmx/* adapters
- Add/remove stats in code easily and graph them
- As Code
  - https://github.com/JonathanTron/chef-grafana
  - https://github.com/SimpleFinance/chef-influxdb

Sentry Exception Logging

- Log all log.error and exceptions, params
- Send them to Slack, create GH issues
- Ties into Grails with `grails-raven` plugin (Grails3 soon)
- Plugins for everything
  - (log4j, logback, js frontend, ruby, python, …)

Demo

Logs vs Metrics

- Logs have lots of detail but must pull stats out
- Metrics reduce noise, but less detail
- Have both - cut down logs where metrics suffice

Daemonizing

- Grails 3.0.4/spring-boot 1.2.5 jar *is* an init script
  springBoot { executable = true }
  ln -s /path/boot.jar /etc/init.d/myapp
  service myapp start

- Systemd (Centos7/etc)

```
# cat /etc/systemd/system/myapp.service
[Unit]
Description=myapp
After=syslog.target

[Service]
ExecStart=/var/myapp/myapp.jar

[Install]
WantedBy=multi-user.target
```

Summary

- Everything is code (lint-able, testable, shareable) @ github.com/erichelgeson
- Be more confident in changes to app and infra
- Deliver software more quickly

Thanks