Jenkins and Chef
Infrastructure CI and Application Deployment

Dan Stine
Copyright Clearance Center
www.copyright.com

June 18, 2014
About Me

- Software Architect
- Library & Framework Developer
- Infrastructure Lead & Product Owner
- Enemy of Inefficiency and Needless Inconsistency

dstine at copyright.com
sw at stinemail.com
github.com/dstine
About Copyright Clearance Center

Global licensing solutions that make © work for everyone
- Get, share, and manage content
- Rights broker for world’s most sought after materials
- Global company (US, Europe, Asia) – HQ in Danvers, MA

Industry-specific software systems
- Internal and external user base
- Applications, services, databases
- Organic growth over many years

In 2011, CCC adopted a Product Platform strategy for growing its software portfolio
Agenda

• Context
• Primer
• Deployment Process Design
• Cookbook Builds
• Application Deployment
• Wrap Up
CONTEXT
Standard Software Platform

• Started platform definition in 2011
  – Homogeneous by default

• Tools
  – Java, Spring, Tomcat, Postgres
  – Git/GitHub, Gradle, Jenkins, Artifactory, Liquibase

• Process
  – Standard development workflow
  – Standard application shape & operational profile
Initial Delivery Pipeline

Source Code (GitHub) → Build Artifacts (Artifactory) → "Don"

DEV → TEST → PROD
Initial Delivery Pipeline

• Automated build process
• Publish build artifacts to Artifactory
  – Application WARs
  – Liquibase JARs
• Manual deploys
  – (Many apps) x (many versions) x (multiple environments) = TIME & EFFORT
  – The more frequently a task is performed, the greater the return from improved efficiency
Improved Deployment Process

• Goals
  – Reduce effort
  – Improve speed, reliability, and frequency
  – Handle app deploys and db schema updates
  – Enable self-service

• Process Changes
  – Manual → Automated
  – Prose instructions → Infrastructure as code
Target Delivery Pipeline
PRIMER
Layers of System Management

• Orchestration
  – Processes collaborating in a distributed system

• Configuration
  – Install and configure packages and software

• Provisioning
  – Hypervisor (VMware, EC2)
Infrastructure as Code

• Develop and manage software infrastructure with practices similar to those used to develop software applications

• Examples
  – Source Code
  – Modularity
  – Abstraction
  – Testing
Configuration Management

“Process for establishing and maintaining consistency of a product’s performance, functional and physical attributes with its requirements, design and operational information throughout its life” (wikipedia)

• CM tools for managing software systems
  – CFEngine, Puppet, Chef, Salt, Ansible

• Embody Infrastructure as Code principles

• Define desired state of machine
  – Each run inspects state and makes necessary changes, if any
Chef

• Configuration management tool
• Exposes DSL hosted in Ruby
  – Express “what” not “how”
  – Clean, purposeful capture of intent
• Favor DSL when writing code
  – Ruby is available, if required
Chef Terminology (1)

- **Chef client** is installed on **nodes** (machines) which are registered with the **Chef server**
- Developers write code on **workstations** and use tools such as **knife** to interact with server
- Chef models node configuration as a set of **DSL resources** (e.g. package, service, directory) which are mapped to internal **providers** (actual code to execute)
  - Can define custom resources
Example Chef Code

This resource declaration

directory '/a/b/c' do
  owner 'admin'
  group 'admin'
  mode '0755'
  action :create
  recursive true
end

ensures that

$ ls -ld /a/b/c
drwxr-xr-x.  5 admin admin 4096 Feb 14 11:22 /a/b/c
Chef Terminology (2)

- A **recipe** declares a set of resources with desired configuration
- A **cookbook** contains a set of semantically-related code and is the fundamental unit of distribution for Chef code
  - Compare to JAR for Java code
- A **data bag** holds JSON information in one or more **data bag items** accessible from Chef code
- Chef **environments** model deployed environments
- Each node has a **run list** containing recipes
DEPLOYMENT PROCESS DESIGN
Basic Approach

- Deploy custom applications with Chef
- Execute schema updates with Liquibase
- Coordinate everything with: Jenkins
Jenkins as Coordinator

• General purpose job executor
  – Shell script steps for Chef API
  – Gradle steps for Liquibase updates
  – Arbitrary code at any point in lifecycle

• UI
  – Smooth integration with Active Directory
  – Authentication and authorization

• Administration
  – Familiar with Jenkins from application build jobs
CCC Application Group

- Set of deployable units that are versioned and released together
- For example, a group might have
  - UI
  - REST service
  - Message consumer
  - DB
- Build with a single command
- Deploy with a single command
Technical Design Goals

• Provide clean API
  – Specify only essential differences between apps
  – Custom Chef resource is the interface
  – Codify & enforce standards

• Balance consistency & flexibility
  – Code in semantically-versioned cookbooks
  – Configuration in data bags

• Controlled cookbook promotion
  – Chef environment specifies cookbook version constraint
Cookbook Types

• Library Cookbooks
  – Encapsulate common re-usable logic
  – Define custom resource to install an app
    • And the implementing provider

• Application Cookbooks
  – Depend on library cookbooks
  – One cookbook per application group
    • One recipe per application
    • Recipes use custom resource
  – Lightweight
Data Bags

• Contain application configuration
  – Service endpoints, JAVA_OPTS, etc.

• One data bag per application group
  – One data bag item per environment

• “Live” reflection of deployed configuration
  – Edit → push to Chef server → deploy
  – Master always matches state of Chef server
Custom Resource Usage

Ensure my-app-ui WAR is deployed:

```ruby
ccc_webapp "my-app-ui" do
  provider :ccc_webapp
  artifact_group
  artifact 'com.copyright.myapp'
  artifact 'my-app-ui'
  container
  http_port 'MY-APP-UI'
  shutdown_port '8080'
  shutdown_port '8900'
  properties_template
  app_group_data_bag
end
```
Custom Resource Actions

• Retrieves Java, Tomcat & WAR from Artifactory
• Installs Java and Tomcat in standard locations
• Creates and configures Tomcat container
• Installs WAR in the container
• Opens port in host firewall
• Generates application properties file
• Starts container

(Each action taken only if necessary)
Data Bag Structure

data_bags/my_app/DEV.json (data bag item)

"version": "1.4.9",
"runtime": {
  "my-app-ui": {
    "java_opts": "-Xmx2G -XX:MaxPermSize=1024m"
  }
},
"app_config": {
  "db.url": "jdbc:postgresql://devdb:5432/myapp",
  "svc.foo.url": "http://devsvc:9000/foo"
}

data_bags/my_app/TEST.json
...
data_bags/my_app/PROD.json
Cookbook Data Bag Code *

ccc/providers/webapp.rb  (library cookbook)

    app_group_data = data_bag_item(app_group_data_bag, node.chef_environment)

    java_opts = app_group_data['runtime'][artifact]['java_opts']
    // pass java_opts to Tomcat container

    app_config = app_group_data['app_config']
    // pass app_config to template resource declaration

my_app/templates/ccc.properties.erb  (application cookbook)

    db.url=<%= @app_config['db.url'] %>
    svc.foo.url=<%= @app_config['svc.foo.url'] %>

* Included for future reference
Roles

• Deployers
  – Update data bags & environment files
  – Initiate deployments

• Tech leads
  – Maintain application cookbooks

• Framework developers
  – Maintain library cookbooks
  – Maintain framework
  – Process improvement
COOKBOOK BUILDS
Cookbook Build Process
Jenkins Build Server

• For each application group
  – Cookbook CI job
  – Cookbook release job
  – Same master as application build jobs

• New class of slaves
  – Ruby with required gems
  – Chef with credentials for Chef server
  – EC2 credentials to create test nodes
Cookbook CI Job

• Triggered when new Chef code is merged

• Static analysis
  – JSON syntax (json gem)
  – Ruby syntax and style (Tailor)
  – Chef syntax (Knife)
  – Chef style and correctness (Foodcritic)

• Integration testing
  – Test Kitchen with kitchen-ec2 plugin
Integration Testing Lifecycle

• Spin up EC2 instance(s) to mimic actual deployment topology of application group

• Run Chef on each instance (node)

• Execute asserts – pass or fail

• Dispose of instance(s)
Integration Testing Details

• Instances created from AMI
  – Preconfigured with Ruby and Chef
• Using Chef Solo
  – Avoid adding ephemeral nodes to Chef server
• Faux Chef environment “CHEFDEV”
  – JSON for real environments is reserved
• Tag EC2 instances for traceability
• Troubleshoot by running Test Kitchen from workstation
Cookbook Release Job

• Triggered manually
• Runs same tests as CI job
• Uploads new cookbook version to Chef server
• Tags Git repo
APPLICATION DEPLOYMENT
Application Deploy Process

1. Workstation
2. Chef Repo (GitHub)
3. Master
4. Deploy Job
5. Deploy Slaves
6. Deploy Server
7. Chef Server
   - Cookbooks
   - Data Bags
   - Environments
8. EC2
   - Deployed Instances
9. VMware
   - Deployed Instances
10. Build Artifacts (Artifactory)
Jenkins Deploy Server

- Separate master for deploys

- Slaves
  - Ruby with required gems
  - Chef with credentials for Chef server
  - SSH keys for nodes
Deploy Job Types

• Each app group has two deploy jobs
  – DEV deploy for Development
  – Non-DEV deploy for Operations
  – Will provide more flavors over time

• Job parameters
  – Environment (non-DEV jobs only)
  – Application group version
What Does a Deployer Do?

• Makes configuration changes
  – Edits application data bag item
  – Edits environment file (if necessary)
  – Merges code

• Executes job in Jenkins
Example Deploy Job Run

• Deployer enters parameters
  – Application version = 1.4.9
  – Environment = TEST

• Then automation takes over
  – Confirms my_app data bag has TEST version 1.4.9
  – Uploads TEST environment file and my_app data bag item for TEST to Chef server
  – Finds all nodes in TEST environment with run list containing my_app recipes
  – Runs Chef client on each found node
  – Sends email notification
Push-Button Deploys

**Project rs.DEV1**

This build requires parameters:

- **APP_GROUP_VERSION**
  Application group version to deploy. Must match databag value.

[Build]

**Project rs**

This build requires parameters:

- **ENVIRONMENT**
  Environment to which the application group should be deployed.
- **APP_GROUP_VERSION**
  Application group version to deploy. Must match databag value.

[Build]
WRAP UP
Most Important Advice

• Beware of overly prescriptive “lessons learned” and “best practices”

• Synthesize a solution for your context

• That said...
Principles & Guidelines (1)

• Standardize wherever possible
  – Technology, design, process
  – Achieve economies of scale
  – Exceptions are permissible but rare

• Every tool must have an API
  – Avoid “hitting the wall” down the road
  – Tradeoff some out-of-the-box capabilities
Principles & Guidelines (2)

• Use multiple communication paths
  – All-hands presentations
  – Kickoff meetings with each team
  – Developer walkthroughs
  – Documentation

• Be opportunistic
  – Find and nurture your early adopters
Principles & Guidelines (3)

• Balance process and progress
  – Must provide tangible results
  – And also build foundation for future
  – Just like with application development!

• Start with a big pain point
  – Providing relief builds credibility going forward
  – Hopefully recoups bandwidth to reinvest
“When Will You Be Done?”

• DONE is a dangerous word
  – Business won’t stop evolving
  – Neither will the supporting applications
  – Nor should the supporting infrastructure

• X is a journey, not a destination
  – For many values of X
    • Deployment automation
    • Continuous delivery
    • DevOps
Thank You

• Copyright Clearance Center Engineering Team

• Jenkins User Conference Organizers and Sponsors
Resources

• “Infrastructure in the Cloud Era”
  Adam Jacob and Ezra Zygmuntowicz
  http://www.slideshare.net/adamhjk/infrastructure-in-the-cloud-era
  http://www.youtube.com/watch?v=HaABapTwQ2c

• Chef cookbook versioning policy
  http://chef-community.github.io/cvp/
Thank You To Our Sponsors

<table>
<thead>
<tr>
<th>Platinum</th>
<th>Gold</th>
<th>Silver</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="CloudBees" /></td>
<td><img src="image2" alt="black diamond" /> <img src="image3" alt="MidVision" /> <img src="image4" alt="choose digital" /> <img src="image5" alt="JFrog" /> <img src="image6" alt="XebiaLabs" /></td>
<td><img src="image7" alt="SOASTA" /> <img src="image8" alt="Eliassen Group" /> <img src="image9" alt="Kensho" /> <img src="image10" alt="SERENA" /> <img src="image11" alt="assembila" /></td>
</tr>
</tbody>
</table>