Continuous Delivery: From dinosaur to spaceship in 2 years

Darren Hague / SAP Global IT
February 5, 2013

Public
Agenda

• **Background**: About SAP, SAP Global IT and the SAP ID Service project

• **Dinosaur Age**: Ageing technology, semi-waterfall processes

• **Stone Age**: A new platform

• **Agricultural Age**: DevOps Tools: Monsoon, Selenium Chef, Cocktail

• **Industrial Age**: Distributed version control; Behaviour-driven testing

• **Jet Age**: Evolving Continuous Delivery with Barkeeper and Bamboo

• **Space Age**: Transforming the team: To Boldly Go…
Background
About SAP

World leader in enterprise applications

- Founded in 1972
- Vision: **Help the world run better**
- Innovation focus: In-memory, Mobile & Cloud
  - E.g. 250Tb RAM, 4000-core database server
  - More than 232,000 customers in 130+ countries
  - More than 65,000 employees in 50+ countries
  - Suite database schema: 30,000+ tables

SAP customers produce 70% of the world’s chocolate & 72% of the world’s beer
SAP Global IT

Global IT

Responsible for corporate hardware & software

- Run by Oliver Bussmann - @sapcio
- Just like any company’s IT division
- **SAP runs SAP**: we drink our own champagne
- Use & contribute to Open Source projects
- Gradually adopting Lean/Agile techniques
- **Generates** revenue for the company
SAP ID Service project

Goal 1: Unified SAP web experience

- One single account for SAP web users
- Seamless sign-on to all SAP sites
- Identity Provider for SAP’s Cloud customers
- Social sign-on and integration with 3rd party apps

Goal 2: Scale & reliability

- Over 4 million users today
- 20+ million coming from recent acquisitions
- Target of 1 billion users by 2015
SAP ID Service Project Team

Geographically distributed
  • Germany
  • UK
  • Russia
  • Israel

Cross-functional
  • Java developers & architects
  • Infrastructure engineers
  • UI / UX designer
  • QA specialist
  • Product Owner
  • Scrum Master
The Dinosaur Age

Ageing technology, semi-waterfall processes
The way we were – early 2010

Ageing technology from previous projects
- Java 1.4 – over a year since End of Life in 2008
- Monolithic J2EE 1.3 application server
- SAP had J2EE 5 in 2006, but we couldn’t migrate
- Code released to physical hardware during downtime

Semi-waterfall processes
- Good parts
  - Source control, issue tracking, build automation, monthly releases
- Not so good parts
  - 3-6 month lead time for new hardware
  - Labour-intensive deployment process took several days
  - Mostly manual, week-long QA cycle
  - Development, Ops & Infrastructure in different business units
  - SAP ID Service: 6 months discussion before first code was written
The Stone Age

A new platform
Tools & platform for the new project

New Platform: SAP Lean Java Server
- Same foundation as SAP NetWeaver Cloud
- Runs on SAP JVM 6 (server-optimised JavaSE 1.6)
- Equinox-based OSGi container
- Tomcat 7 & Spring 3 embedded
- SAP-optimised SAML library
- Persistence: MongoDB & Novell eDirectory

Toolkit:
- JIRA for issue tracking
- Bamboo for continuous integration
- Perforce for version control
- Eclipse for IDE
- Ant & Ivy for build / dependency management
Agricultural age

DevOps Tools: Monsoon, Chef, Selenium, Cocktail
What is Monsoon?

Monsoon: SAP IT’s implementation of Continuous Delivery

• Provides structured approach to “Infrastructure as a Service”
• Works with SAP’s private cloud
• Uses open-source tools like Chef to make VM provisioning easy
• Promotes best practices for continuous delivery
  • Version control everything
  • Automate everything
• Inspired by the DevOps movement

Read the book: http://continuousdelivery.com/
Monsoon Phase 1: Virtualization & Chef

Virtualization
- Dev, QA & Production – all virtualized
- VMs allocated manually
- Developer VM request: delivered in hours

Chef
- Install Chef client on VM
- Central Chef server for all projects & landscapes
- Just run “chef-client” to install & configure apps

Sample Chef recipes
- Install Java, app server & reverse proxy
- Configure app server & SAML trusts
- Install JARs & WARs from the latest build
- Create configuration files with correct parameters
Chef server

Chef Server Environment: None

Node: Chef-3-3ap.wm.sap.corp

Environment: _default

The node's environment

Available Roles
- jvm_apache
- memcached
- mongodb
- mongodb-backup
- splunk-app-ids
- splunk-db-ids

Run List
- ids-idp-pb-setup
- mongodb
- ids-idp-node

Available Recipes
- java
- java:openjdk
- java:sun
- jenkins
Sample of a Chef recipe

```ruby
default.rb
#
# check if OS version is supported and install required packages
#
if platform?("redhat")
  case node['platform_version']
  when /^6/;
    package "compat-expat1" do
      action :install
    end
  end
  # let's set to 2.2.22 for new RedHat 6 template
  node.default[:apache_httpd][:version] = "2.2.22"
end
log("==> Your platform is supported by this cookbook.")
else
  log("==> Sorry your platform is not supported by this cookbook. Take care!") { level :warn }
end
#
# check if path to installation tmp exist, create if not

directory "#{node[:apache_httpd][:install_tmp]}" do
  mode "0777"
  owner "root"
  group "root"
  action :create
  recursive true
end
#
# check if path to installation root exist, create if not
```
Automated Testing: Originally, not much

When we started:

- No culture of developer-created tests
- Some automated regression tests from QA team
- Tests run once a month after QA deployment
- Developers fix bugs for previous cycle when they should be working on next

Slow progress, waiting for the release train
Developer frustration
Stakeholder frustration
Selenium: Browser-scripted Testing

- Developers & QA work together
- Record simple scripts in the browser
- Develop more complex scripts in Java
- Tests can be run from JUnit
- Run during the build by Bamboo
- Developer gets feedback in minutes

Better quality scripts by working together
No waiting for the release train
Monthly QA cycle much shorter
No nasty surprises
Selenium IDE

Test can be run with the command:

```java
java -jar selenium-server.jar
   -htmlSuite "*firefox"
   "http://test.server"
   TestSuite.html
   results.html
```

- This can be added to an automated build
Cocktail: automated test & deployment

To get SAP ID Service running:

- Create virtual machines
- Register each VM with Chef server
- Execute chef-client
- Validate the installation (ping ports, etc)
- Test functionality via Selenium scripts

An internal tool called Cocktail was developed to execute all these actions.

Able to create a complex multi-server landscape in 4/5 commands.
Recap: Phase 1 of Continuous Delivery in Global IT

**DevOps** concept: entire infrastructure is built automatically

- **Atlassian Bamboo** used to build, unit test & orchestrate infrastructure deployment / integration testing
- **Cocktail** controls provisioning, deployment & integration testing
- **VMware vSphere** provisions & boots VMs
- **Opscode Chef** used to deploy & configure nodes
- Automated functional testing of landscape using **Selenium**
Industrial age

Distributed version control with Git

Behaviour-driven testing with Cucumber
Distributed Version Control

From Perforce to Git

- DVCS gives safety & freedom
  - Local branches, lots of commits as save-points
  - Combine commits before push to origin/master
- Origin/master repo on GitHub Enterprise
  - Single linear history in central repo
  - Avoid branches by fetch/rebase before pushing
- Feature branches on local repo
  - Easy to switch between stories
  - Also, feature toggles for incomplete stories
  - Fetch & rebase to keep branch in sync
  - Push feature branch commits onto origin/master
Cucumber: Behaviour-driven Testing

- Product owner works with team
- User stories transformed into Gherkin:

```
Scenario: Log on success for SAP Store user

Given I am using a SAP Store active test user

When I try to access protected content of the SAP Store
Then I should see the “SAP Store" login overlay

When I login using my valid credentials
Then I am logged in
And the main SAP Store page is displayed
```

- Gherkin steps pattern-match to Java methods
- Feature files mapped to JUnit stub classes
- “Definition of done” includes Cucumber creation
- Product owner gets fast feedback
Gherkin lines pattern-match to Java methods

Scenario: Log on success for SAP Store user

Given I am using a SAP Store active test user
When I try to access protected content of the SAP Store
Then I should see the “SAP Store” login overlay

When I login using my valid credentials
Then I am logged in
And the main SAP Store page is displayed

@Given("^I am using a SAP Store active test user$")
public void loginUsingValidCredentials() {
    // use the login page to log in with the test user's credentials
    String loginName = getTestUserProfile().get(TestSPUser.USER_PROFILE_ID);
    if (loginName == null) {
        loginName = getTestUserProfile().get(TestUser.UID);
    }

    String password = getTestUserProfile().get(TestSPUser.USER_PROFILE_PASSWORD);
    if (password == null) {
        password = getTestUserProfile().get(TestUser.PASSWORD);
    }

    ((LoginPage) getWebPage()).login(loginName, password);
}
Jet age

Evolving Continuous Delivery with Barkeeper and Bamboo
Monsoon Phase 2: Barkeeper

- Controls the infrastructure, replacing cocktail
- Allocate VMs via Cloud API as well as managing them
- Manages Chef servers
  - One Chef server per project landscape,
  - Central library of cookbooks
- Project self-service
  - Create an entire project (Dev, QA, Prod servers) via one YAML file
  - “Private Bar” concept
    - Developers creates own servers on demand
- Web UI and REST API with command line tool (knife plugins)
- Everything under version control
Project landscape definition

description: SAP ID Service

chefrepo: git@github.wdf.sap.corp:ids/chef-repo.git

cloudprovider: sap-id-service

template: RedHat.5.WDF.internal.general.V2.1

network: BSS General Monsoon

bootstrap:
  - recipe[monsoon]

runlist:
  - recipe[monsoon]

landscapes:

- name: test
description: SAP ID Service Test Landscape
chef sync control: PIPELINE
chefserver:
  runlist:
    - recipe[monsoon]
    - recipe[f5::manager]
    - recipe[hyperic::setup_monitoring]
servers:
  - name: idp
description: Identity Provider
tags: appserver
runlist:

- name: prod
description: SAP ID Service Production Landscape
chef sync control: PIPELINE
template: RedHat.5.WDF.allnet.V2.1
network: BSS SCN IDMZ Monsoon
bootstrap:
  - recipe[monsoon]
cheffserver:
  tags: f5manager
runlist:
  - recipe[monsoon]
  - recipe[f5::manager]
  - recipe[hyperic::setup_monitoring]
servers:
Barkeeper
Control and manage systems, landscapes and servers

The Barkeeper Tool automatically creates and manages the project servers (virtual and physical servers) as specified in the project definition file. Barkeeper provides central access to all servers in the landscapes and shows their status.
Private Bar
Speed for developers (system & app)

With Monsoon Barkeeper’s **Private Bar** functionality a developer can quickly spawn his private development or try-out server.
Cookbook Shelf
Sharing of installers

Cookbooks are the fundamental units of distribution in Chef. You can create your own cookbooks, or you can download other people's shared cookbooks.

Cookbook Community
As Monsoon becomes more popular, its users need a central place to share the cookbooks that already exist, and the community needs to discover new cookbooks.

This is a central place for organizations to find the cookbooks they need, regardless of where the cookbook came from. As you explore the site, you’ll see that cookbooks can be listed, commented on, and followed.

Getting Help
Sometimes you just need a little help. Let us give you some pointers to diagnose problems.

© 2013 SAP AG or an SAP affiliate company. All rights reserved.

The Cookbook Shelf is the central directory for sharing cookbooks that are customized to our infrastructure and can be re-used by other projects.
Build pipeline for Continuous Delivery

SAP ID Service Pipeline

Build
- Build Release
- Build Snapshot

Install on DEV Landscape
- Deploy release to DEV Landscape

Testing of DEV Landscape
- Company Test Suite
- IdP Test Suite
- Legacy Basic Rest Test (DEV)
- Legacy Consolidation Test (DEV)
- Legacy Full Rest Test (DEV)
- Legacy Rebind Rest Test
- Service Provider Test Suite (DEV)
- SP-User REST API Test Suite (DEV)
- SSO IdP Test Suite
- SSO SP Test Suite
- UI Test Suite (DEV)
- Unit tests

OK to "git push" if this and everything above is Green
- UI Test Suite part 2 (DEV)

Prepare QA and TEST Chef Repositories
- Prepare Chef in TEST Landscape
- Prepare Chef in QA Landscape
- Prepare Chef in STAG Landscape

Install on Test and QA-Green Landscapes
- Deploy release to Green Pool in QA Landscape
- Deploy release to Green Pool in STAG Landscape
- Deploy release to TEST Landscape

Testing of TEST and QA-Green Landscapes
- QA Test Suite (Standby Pool)

Install on QA-Blue Landscapes
- Deploy release to Blue Pool in QA Landscape
- Deploy release to Blue Pool in STAG Landscape

Prepare PROD Chef Repository
- Prepare Chef in PROD Landscape

Install on PROD-Green Landscape
- Deploy release to Green Pool in PROD Landscape

Install on PROD-Blue Landscape
- Deploy release to Blue Pool in PROD Landscape
# Build stage 1: Build & deploy to DEV

## Build

<table>
<thead>
<tr>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build Release</td>
</tr>
<tr>
<td>Build Snapshot</td>
</tr>
<tr>
<td>Chef Repo Sync</td>
</tr>
</tbody>
</table>

## Install on DEV Landscape

<table>
<thead>
<tr>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy release to DEV Landscape</td>
</tr>
</tbody>
</table>
Build stage 2: Run all test suites

Testing of DEV Landscape

- Admin Console Test Suite
- Admin Console Test Suite 1
- Company Test Suite
- IdP Test Suite
- Integration Tests (No Cucumber)
- Legacy Basic Rest Tests (DEV)
- Legacy Consolidation Tests (DEV)
- Legacy Full Rest Tests (DEV)
- Legacy Rebind Rest Test
- Password Policy Test Suite
- Service Provider Test Suite (DEV)
- Service Provider Test Suite (DEV)
- SP-User REST API Tests (DEV)
- SSO IdP Test Suite
- SSO SP Test Suite
- UI Test Suite (DEV)
- UI Test Suite part 2 (DEV)
- UI Test Suite part 3
- UI Test Suite part 4
- UI Test Suite part 5
- Unit tests
Build stage 3: deploy & test inactive “Green” systems

Install on Test and QA-Green Landscapes

- Deploy release to Green Pool in QA Landscape
- Deploy release to Green Pool in STAG Landscape
- Deploy release to TEST Landscape

Testing of TEST and QA-Green Landscapes

- QA Test Suite (Standby Pool)
Build stage 4: Activate Green & Deploy to Blue systems

**Install on QA-Blue Landscapes**

- Deploy release to Blue Pool in QA Landscape
- Deploy release to Blue Pool in STAG Landscape
Cycle time is critical

- Minimise the time from commit to green build
- Continuously monitor & improve build performance
- < 10 minutes for developer build, deploy & test
- < 30 minutes for central build & deploy to QA
- Parallelisation is key, especially for tests
  - We have nearly 600 scenarios and 6000 steps
  - Aim to keep each suite to < 3 minutes
  - If a suite exceeds this, split it
- Multicore developer machine helps
  - Currently 8 parallel threads for the test suite
Recap: Impact of Continuous Delivery in SAP Global IT

- **Before**: Production releases ~monthly
- **Now**: Production release ~twice a week
- **Before**: Pre-release QA cycle 1-2 weeks
- **Now**: QA cycle < 1 day
- **Before**: Error in Prod? Shitstorm & late night
- **Now**: Switch to Blue in <1 minute, fix next day
- **Before**: Project idea to go-live in 6-12 months
- **Now**: New project can be in Production in 1 week
- **Before**: Business stakeholders frustrated
- **Now**: Business stakeholders happy

Technology supports all this, but the team still has to deliver working code.
Space age

Transforming the team, To Boldly Go…
Attempting to Transform the Team

2010-11: Waterfall with monthly iterations
- Developers each with own competence & codebase
- Everyone commits code “when it’s ready”
  - Typically on the deadline day before QA begins
- Very little communication
  - Communication when integration problems occur
  - Lots of blaming

2011 – early 2012: Team adopts Scrum(-ish)
- Everyone thinks they know Scrum
- Scrum = daily call, not much else
- Slightly better communication
- Daily calls often taken over by single “big issues”
- Otherwise, not much difference
Really Transforming the Team

- **May 2012 – Scrum Training**
  - Business invests in external Scrum trainer
  - Entire team together for 1 week in Berlin
    - Except 1 team member in London
  - Deep learning about Lean principles
    - Lots of games, colours & Post-its®
    - Focus on continuous team self-improvement

- **Results**
  - Pair programming, shared ownership
  - Use of DevOps & Cucumber removes silo thinking
  - Product Owner orders backlog & shields team
  - Scrum Master runs Daily Scrum, Sprint Planning, Sprint Review & Sprint Retrospective
  - **Radical difference in team productivity**
Culture of Continuous Improvement

- **Team is always working to improve itself**
- **Retrospective at the end of each sprint**
  - Several improvement suggestions each time
  - Vote on top 3-5 to implement in next sprint
  - Focus on team behaviours, not product scope
- **Evaluating new tools & techniques:**
  - Gerrit for code review
    - Initially for regulatory “4 eyes” control
    - Extremely useful for design communication
    - If pair programming, almost zero overhead
  - Pomodoro technique
    - Break work into 25-minute chunks
    - Lots of mini deadlines improve productivity
    - Alleviates intensity of pair programming
Thank you

Contact information:

Darren Hague
Engineering Architect
SAP (UK) Ltd
Email d.hague@sap.com  Twitter @dhague

http://developer.sap.com