Continuous Delivery at SAP: From dinosaur to spaceship

Darren Hague / SAP Global IT
November 1st, 2013
Agenda

The 6 stages of SAP IT’s journey to Continuous Delivery:

• Dinosaur Age
• Stone Age
• Agricultural Age
• Industrial Age
• Jet Age
• Space Age
The Dinosaur Age

Ageing technology, semi-waterfall processes
About SAP

World leader in enterprise applications

- Founded in 1972
- Vision: Help the world run better
- Innovation focus: Mobile, Cloud, In-memory

- Over 232,000 customers, 130+ countries
- Over 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
The way we were in SAP Global IT – early 2010

**Ageing platform from previous projects**
- Java 1.4 – over a year since End of Life in 2008
- Monolithic J2EE 1.3 application server
- Code deployed to physical hardware during downtime

**Semi-waterfall**
- Good:
  - Source control
  - Issue tracking
  - Build automation
  - Monthly releases
- Not so good:
  - Months-long lead time for new hardware
  - Labour-intensive & error-prone deployment
  - Labour-intensive QA cycle
  - Development, Ops & Infrastructure in different business units
The Challenge

2010 capacity
• 20000 PD of people available

2010 demand
• 60000 PD of project effort estimated
The Stone Age

A new project, a new platform
SAP ID Service project

Unified SAP web experience

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

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

Cross-functional

• Product Owner
• Scrum Master
• UI / UX designers
• Java developers & architects
• Infrastructure engineers
• QA specialists

Geographically distributed

• Germany
• Bulgaria
• UK
• Russia
• Israel
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)
- Up-to-date version of Tomcat app server
- OSGi platform for modular development
- Quick to install & restart

Same 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
Vision & Cultural Change

Chief Architect (my boss)
- Promoted concepts of Continuous Delivery
  - Automate everything, especially testing
  - Version control everything
- “Hey everyone: read the Humble & Farley book”
- SAP ID Service as pilot project

Director of Web & KM unit (his boss)
- Provided trust
  - 10% of unit’s effort for continuous delivery
  - Codename: “Monsoon”
- Provided cover
  - 10% “taxed” from project budgets
  - Not an explicit line item
Monsoon Phase 1: Virtualization & Chef

Virtualization
- Dev, QA & Production – all virtualized
- Private Dev server VM per developer
  - VMs allocated by infrastructure team member
  - VM requests serviced in hours, not months

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

DevOps skills needed
- Some (all?) team members need to learn Chef & Ruby
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
```
Chef server

Node 14.1-3-hap.wii.sap.corp

Environment: _default

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

Available Recipes
- java
- java:openjdk
- java:sun
- jenkins

Run List
- ids-idp-pb-setup
- mongodb
- ids-idp-node
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
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 automate all these actions.

Able to create a complex multi-server landscape with a handful of commands
Industrial age

Behaviour-driven testing with Cucumber
Cucumber: Behavior-driven Testing

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

```java
@UserStory("MOCPS-1522")
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

@UserStory("MOCPS-1522")
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

@When("^I login using my valid credentials$")
public void loginUsingValidCredentials() {

    String loginName = getTestUserProfile().get(USER_PROFILE_ID);
    String password = getTestUserProfile().get(USER_PROFILE_PASSWORD);
    ((LoginPage) getWebPage()).login(loginName, password);
}
Annotations drive reporting
Jet age

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

- Allocates VMs via Cloud API
- Manages Chef servers
  - One Chef server per project landscape
  - Central library of cookbooks
- Project self-service
  - Create an entire project (Dev, QA, Prod servers) in one config file
    - Developers create own servers on demand
- Web UI and REST API
- 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
chefservers:
  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]
chefservers:
  tags: f5manager
runlist:
  - recipe[monsoon]
  - recipe[f5::manager]
  - recipe[hyperic::setup_monitoring]
servers:
```
Private Bar
Speedy self-service for developers

With Monsoon Barkeeper’s Private Bar functionality a developer can quickly spawn a private development or try-out server.
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 Landscapes
- Deploy release to Green Pool in PROD Landscape

Install on PROD-Blue Landscape
- Deploy release to Blue Pool in PROD 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 over 700 scenarios and 9000 steps
  - Aim to keep each suite to < 3 minutes
  - If a suite exceeds this, split it
  - Multicore developer machine helps
    - 4 cores => 8 parallel threads for the test suite
Recap: Impact of Continuous Delivery

- **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? Lots of stress, 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**
  - Investment in an external Scrum trainer/Agile coach
  - 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
  - DevOps & Cucumber help remove 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 distributed pairing
    - If pair programming, almost zero review overhead
  - Pomodoro technique
    - Break work into 25-minute chunks
    - Lots of mini deadlines improve productivity
    - Alleviates intensity of pair programming
Latest improvement experiment – “ProdOps”

• Problem: to deploy to Production, Product Owner has to email Ops:

  Hi Ops guy,

  Could you deploy build #5345 to Production please?

  Thanks,
  Product Owner

• This is just another manual step that we can automate, creating a bit of fun as we do so…
Deploy Button
Thank you

Contact information:

Darren Hague  
Architect, Business Innovation & Application Services  
SAP (UK) Ltd  
Email d.hague@sap.com  Twitter @dhague

http://developer.sap.com