PUZZLE ITC

changing IT for the better

Daniel Tschan
CTO

Continuous Delivery mit OpenShift, 2nd Edition

OpenShift Tech Lab, Bern, 27.10.2014

Grundlagen

Kleine Geschichte

«Integration Hell» und «Works on My Machine»

1991: Object Oriented Design: With Applications

1999: Extreme Programming Explained

2001: CruiseControl

2005: Hudson

2010: Continuous Delivery

Voraussetzungen

Agiler Entwicklungsprozess

Continuous Integration

Continuous Integration

Versionsverwaltung für Quelltext

Automatisiertes kompilieren jeder Codeversion

Tests automatisch ausführen, auf prod. Stack

Häufiges einchecken/integrieren

Automatisches Reporting

Automatisches Deployment auf Integration

Warum Continuous Delivery?

Continuous Integration weiterentwickeln

Fehlerquellen eliminieren

Effizienz steigern

Flexibilität erhöhen

Feedbackzyklen verkürzen

Zusammenarbeit verstärken

Continuous Delivery

Möglichst identische Umgebungen

Ein Artifakt für alle Umgebungen

Eindeutige Versionsnummer

Vollautomatisiertes Deployment auf Knopfdruck

Environment Detection

DB Management Tool/Migrationsskripte

OpenShift

Platform as a Service (PaaS) Produkt von Red Hat

Public oder private cloud

Softwarestacks auf Knopfdruck

Java, Ruby, JavaScript, Python, PHP, .NET, ...

3 Applikationen gratis hosten auf openshift.com

Liquibase

Versioniert die Datenbank

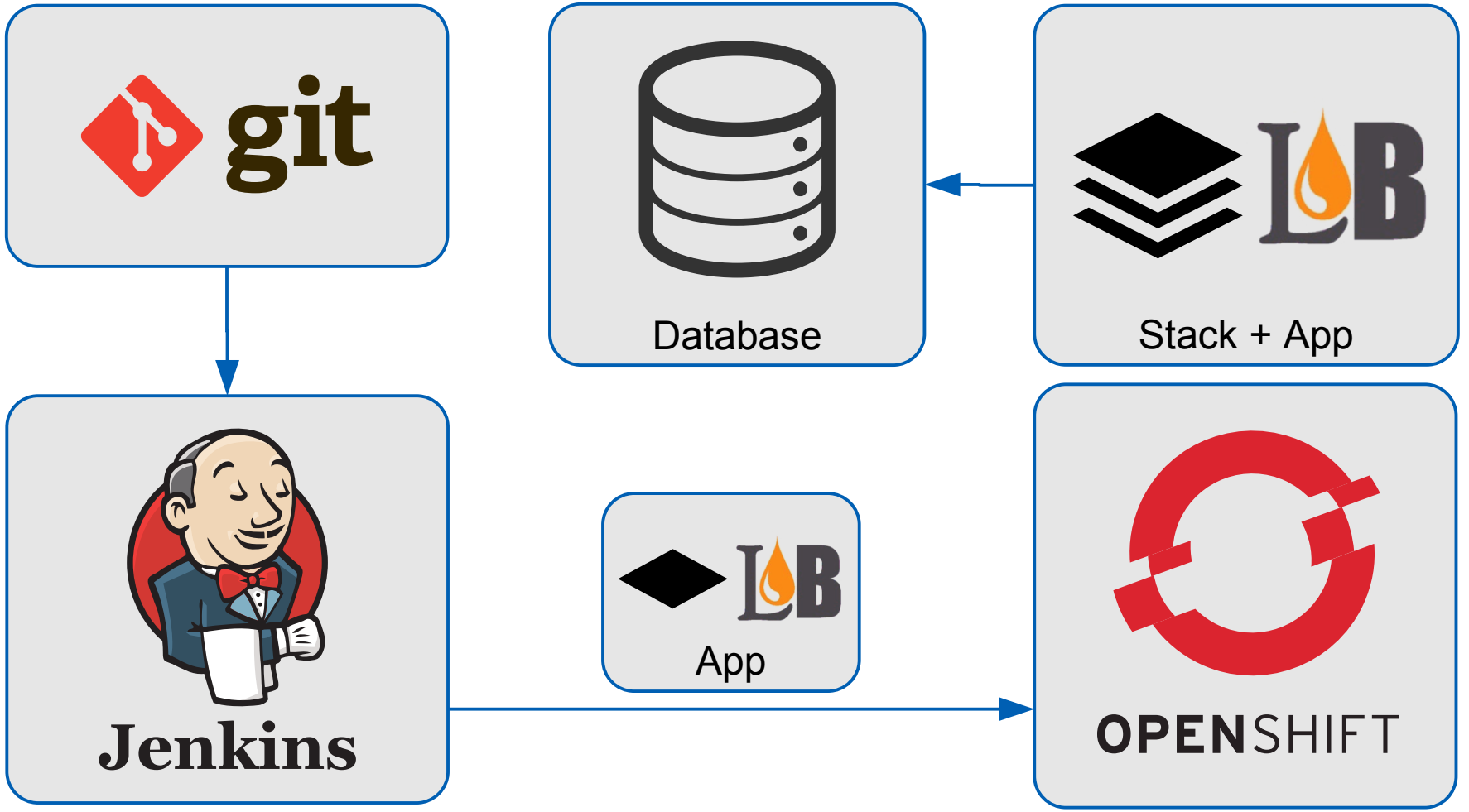
Bestimmt die auszuführenden Migrationsskripte

Führt diese in korrekter Reihenfolge aus

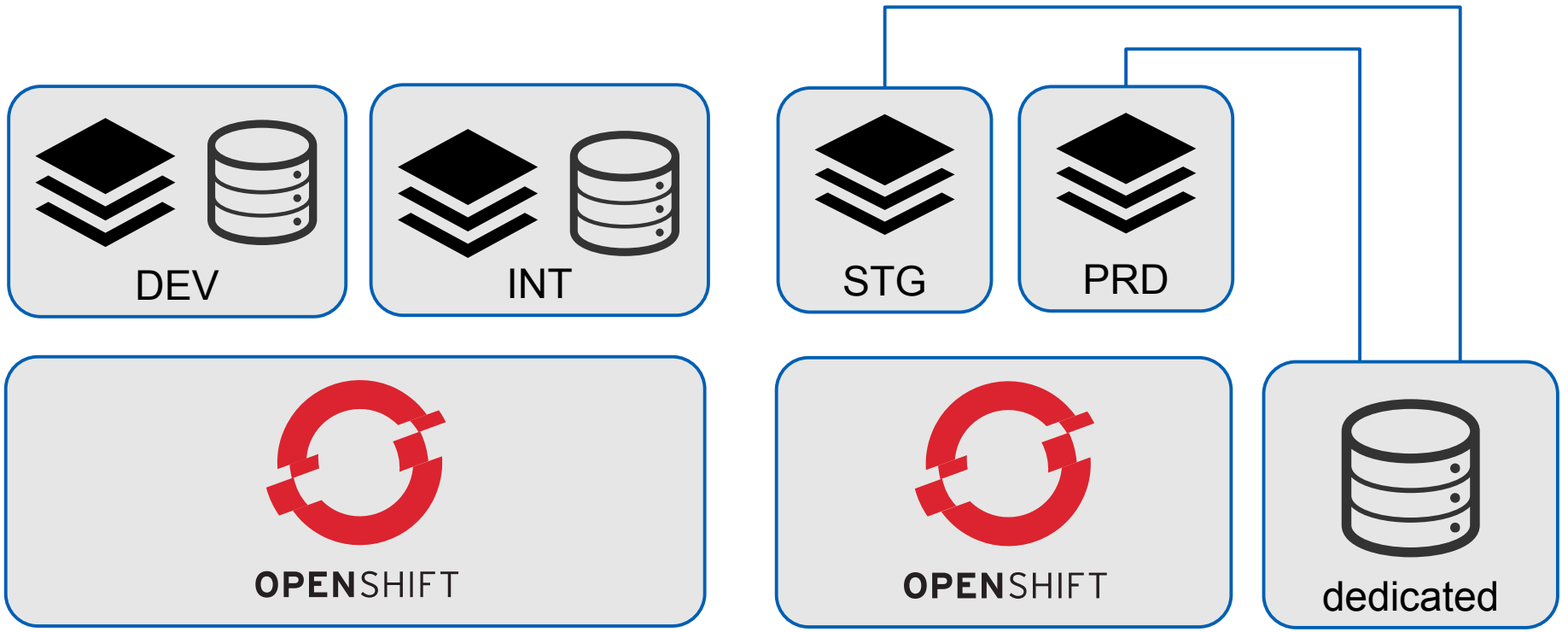
2

Umsetzung bei Puzzle

Umsetzungsbeispiel mit Java EE



Identische Umgebungen



Jenkins CD Support bisher

Promoted Builds Plugin

Auf Knopfdruck Build X auf Umgebung Y deployen

Parametrized Trigger Plugin

X und Y an Deploy Job übergeben

Copy Artifact Plugin

Deployment Artifakt von Build nach Deploy Job kopieren

Freestyle Jobs

Konfiguration und Kombination der Plugins

Jenkins Workflow Support

DSL basierend auf Groovy

Entwickelt für Continuous Delivery Pipelines

Entwickelt von CloudBees (Jenkins Enterprise)

Resume nach Restart/Crash von Jenkins

Jenkins Workflow Support

Wiederverwenden von DSL Scripts über integrierten Git Server: http://jenkins.example/workflowLibs.git

Bleeding Edge: JENKINS-26481

Benötigt Jenkins >= 1.580.1

Installation: Plugin «Workflow: Aggregator»

Ein Artifakt

Workflow Build Job auf Puzzle Jenkins Server

```
node {
  git url:
    "https://github.com/puzzle/quickstart-javaee6-liquibase.git"
  String mvnHome = tool("maven3")
  sh "${mvnHome}/bin/mvn -B verify"
  sh "mv target/javaee6-liquibase.war \
        target/ROOT.war"

  def openShift = new com.puzzleitc.OpenShift()
  openShift.tar(".", "target/ROOT.war")
}
```

Automatisiertes Deployment

Workflow Deploy Job auf Puzzle Jenkins Server

This build is parametrized:

Run Parameter: BUILD
Project: *name of build job*

Choice Parameter: TIER
Choices: dev, int, stg, prd

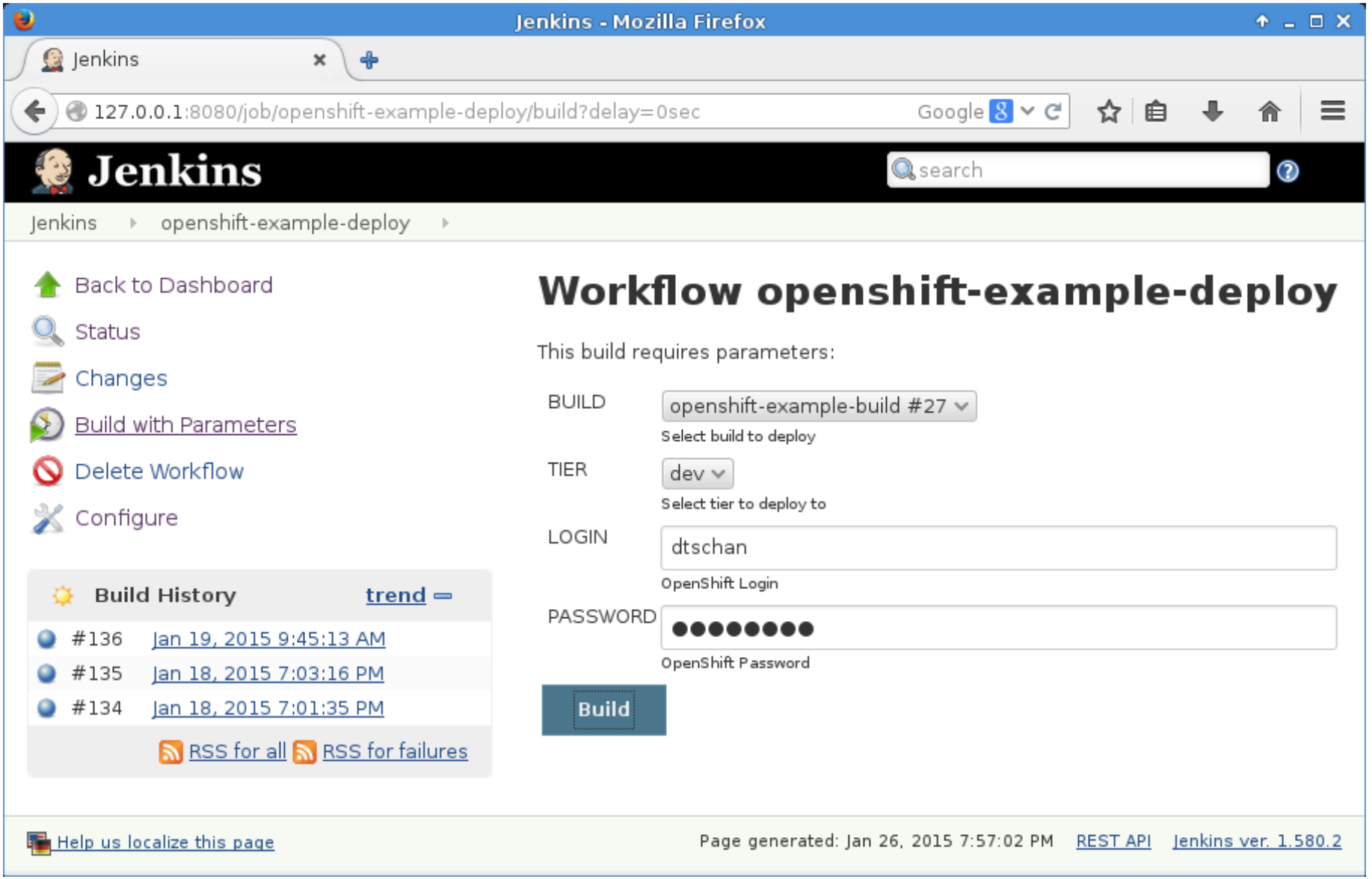
String Parameter: LOGIN
Password Parameter: PASSWORD

Automatisiertes Deployment

Groovy CPS DSL:

```
node {
  def openShift = new com.puzzleitc.OpenShift()
  openShift.deploy(
    build: BUILD,
    tier: TIER,
    user: LOGIN,
    password: PASSWORD,
    serverDev: "broker.openshift-dev.puzzle.ch",
    serverPrd: "broker.openshift.puzzle.ch",
    appDev: "techlabdev",
    appPrd: "techlab",
    options: "--hot-deploy")
}
```

Deployen auf Knopfdruck



Environment Detection

Umgebungsspezifische Konfiguration über Umgebungsvariabeln:

- DNS Name
- Datenbankverbindungsdaten
- Daten-/Logverzeichnisse

Environment Detection

Umgebungsvariabeln setzen:

Automatisch durch OpenShift

`rhc env set` (automatisieren!)

DB Change Management

Liquibase ist Teil des Deployments

Alle Liquibase Change Sets sind Teil des Deployments

Applikation führt bei Start Liquibase aus

Liquibase führt Change Sets aufgrund des Datenbankstandes aus

Liquibase Integration

```
public class LiquibaseProducer {

  @Resource(lookup = "java:ourDataSource")
  DataSource dataSource;

  @Produces @LiquibaseType
  public CDILiquibaseConfig createConfig() {
    CDILiquibaseConfig config = new CDILiquibaseConfig();
    config.setChangeLog("/db.changelog.xml");
    return config;
  }
```

Liquibase Integration

```
  @Produces @LiquibaseType
  public DataSource createDataSource()
                          throws SQLException {
    public dataSource;
  }

  @Produces @LiquibaseType
  public ResourceAccessor create() {
    public public ClassLoaderResourceAccessor(
      getClass().getClassLoader());
  }
}
```

Warum Liquibase in der App?

Applikationsartefakt von Umgebung unabhängig

Applikation und DB sind immer kompatibel

Datenbankverbindung gegeben

Unterstützt Cluster und Cloud Deployments

Warum nicht OpenShift Jenkins?

Verletzt ein «Artefakt für alle Umgebungen»

Schwer zu warten: custom build script in jedem Job

Kein Workflow Support

Ressourcen beschränkt

Maven Version hardcoded

Nächste Schritte

OpenShift Umgebungsvariabeln zentral verwalten

Integration von Artifactory in Prozess

Weiterentwickeln Prozess für OpenShift Scalable Apps

Referenzen

Continuous Delivery

OpenShift User Guide

Liquibase Documentation

Jenkins Workflow Plugin

Q & A

15 Jahre Puzzle ITC
Jubiläums Tech Talk
22. Oktober 2015

Save The Date

DDL vs DML

Aus fachlicher Sicht identisch:

```
DROP TABLE table
```

```
DELETE FROM table
```