PUZZLE ITC

changing IT for the better

Daniel Tschan
Technischer Leiter

Continuous Delivery mit OpenShift

Puzzle Tech Talk, Bern, 23.10.2014

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

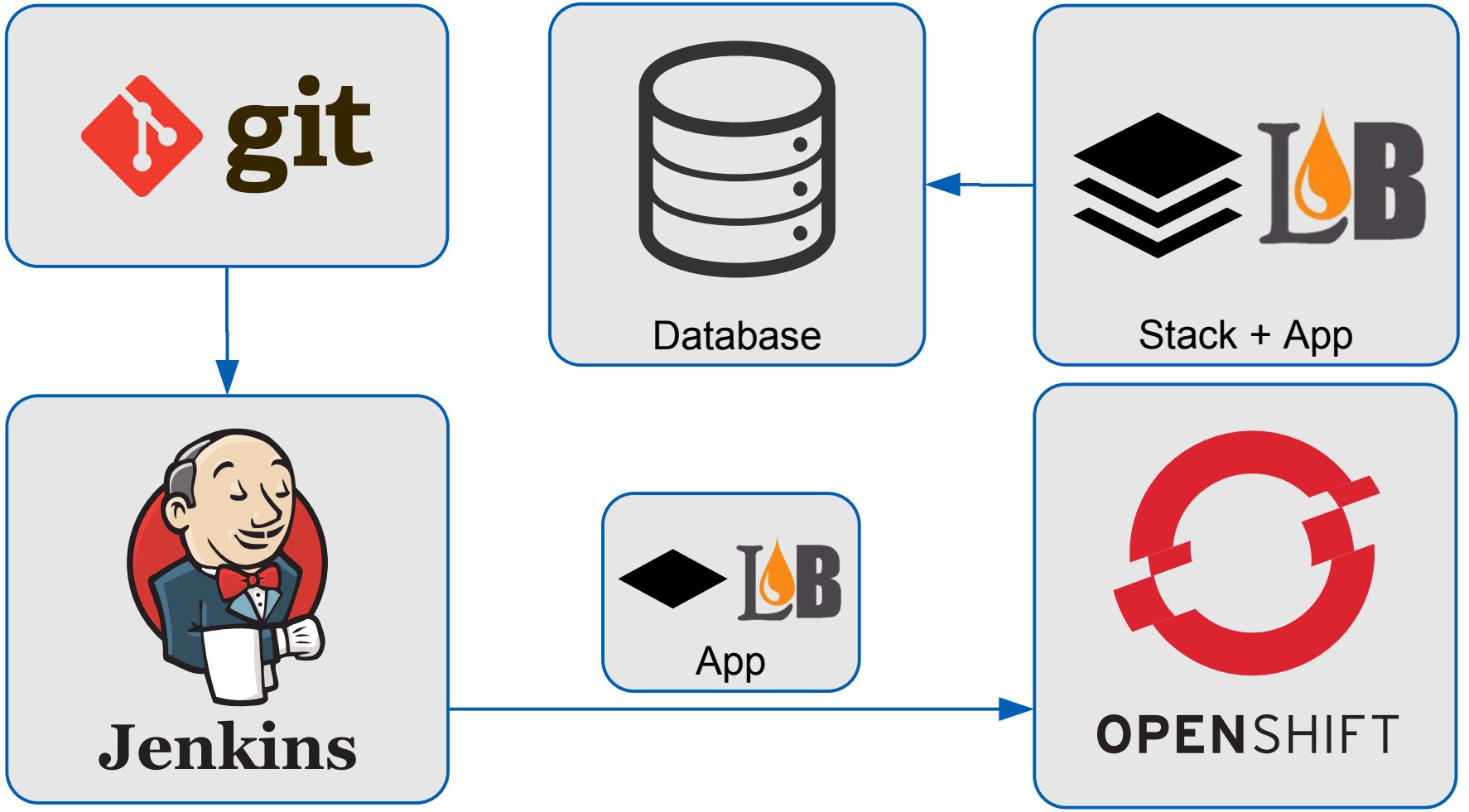
Bestimmt die auszuführenden Migrationsskripte

Führt diese in korrekter Reihenfolge aus

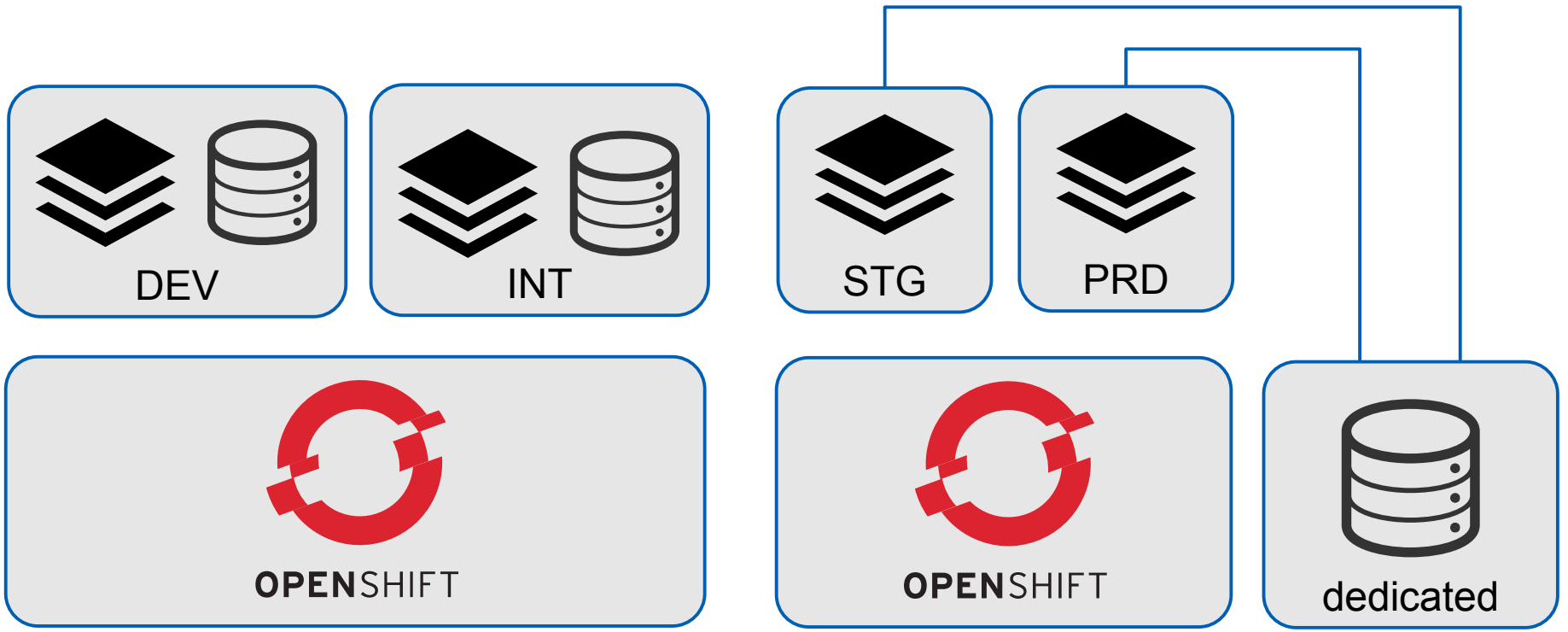
2

Umsetzung bei Puzzle

Umsetzungsbeispiel mit Java EE



Identische Umgebungen



Ein Artifakt

Gewöhnlicher Build Job auf Puzzle Jenkins Server

Post Steps: Execute Shell

```
pot tar kitchensink kitchensink/target/*.war
```

Post-build Actions:

archive the artifactes: `deployment.tar.gz`

pot = Puzzle OpenShift Tool

Eindeutige Versionsnummer

Jenkins Buildnummer: $BUILD_NUMBER

Maven Build Number Plugin

git rev-list HEAD | wc -l

Automatisiertes Deployment

Free-Style Deploy Job auf Puzzle Jenkins Server

This build is parametrized:

- Choice Parameter: env
- Build Selector for Copy Artifact: BUILD_SELECTOR

Build: Copy artifacts from another project

- Project name: Name des Build Jobs
- Which build: Specified by a build parameter: BUILD_SELECTOR
- Artifacts to copy: `deployment.tar.gz`

Automatisiertes Deployment

Build: Execute Shell

```
export POT_NAMESPACE_dev=dtschan
export POT_APP_dev=kitchensinkd

export POT_NAMESPACE_int=dtschan
export POT_APP_int=kitchensinki
export POT_ENV_int="JVM_PERMGEN_RATIO=0.3"

pot deploy $env
```

Deployen auf Knopfdruck

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

Deployen auf Knopfdruck

Promotion Plugin im Build Job konfigurieren

Actions: Trigger/call builds on other projects

Projects to build: Name des Deploy Jobs

Predefined parameters:

```
env=dev (oder int/stg/prd)
BUILD_SELECTOR=
  <SpecificBuildSelector>
    <buildNumber>$PROMOTED_NUMBER</buildNumber>
  </SpecificBuildSelector>
```

Promotions

Deploy to DEV

Promotion History

#2 (Wed Sep 17 17:12:07 CEST 2014) by SYSTEM

Qualification **(promoted 3 min 36 sec ago — 3 ms after build)**

Automatically promoted immediately after the build

Status

Successfully promoted (log)

Deploy to INT

This promotion has not happened.

Met Qualification

Unmet Qualification

Manual Approval

Approve

Deploy to PRD

This promotion has not happened.

Met Qualification

Unmet Qualification

Manual Approval

Approve

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

Ressourcen beschränkt

Maven Version hardcoded

Nächste Schritte

OpenShift Umgebungsvariabeln zentral verwalten

Integration von Artifactory in Prozess

OpenShift auf green.ch OpenStack FlexCloud

Weiterentwickeln Prozess für OpenShift Scalable Apps

Referenzen

Continuous Delivery

OpenShift User Guide

Puzzle OpenShift Tool

Liquibase Documentation

StackOverflow: Deploy Specific Build With Jenkins

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
```

Was macht pot tar?

```
rm -rf ${work}

mkdir -p ${work}/build_dependencies
   ${work}/dependencies/jbosseap/deployments
   ${work}/repo/.openshift

cp "$@" ${work}/dependencies/jbosseap/deployments

rsync -avC "${openshift_config_dir}"/.openshift/
   ${work}/repo/.openshift/

( cd ${work} &&
   GZIP=-9 tar cvfz ../deployment.tar.gz . )

rm -rf ${work}
```

Was macht pot deploy?

```
eval ns=\$POT_NAMESPACE_$1
eval app=\$POT_APP_$1
eval env=\$POT_ENV_$1

if [ "$env" ]; then
  rhc -n$ns -a$app env set $env
fi

rhc -n$ns -a$app app configure
  --deployment-type binary

rhc -n$ns -a$app app deploy deployment.tar.gz
```