

2.– 5. September 2013
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Volle Fahrt voraus!

Continuous Delivery praktisch

Dr. Halil-Cem Gürsoy

adesso AG

Über den Referenten...

- 15 Jahre Entwicklung und Beratung
- „Software Architect“ bei adesso AG, Dortmund
- Schwerpunkte
 - Enterprise Java (Spring)
 - „Cloud“ / PaaS und Storage in der „Cloud“
 - Buildmanagement
 - Continuous Delivery & Deployment

Vorab einige Punkte ...

... sorry, ich habe nur 70 Minuten Zeit!



Einige Fragen ...

- Was ist überhaupt...
 - Vagrant
 - Puppet
 - Jenkins
- Und was ist eigentlich konkret...
 - Continuous Delivery
 - ... Und was ist anders zu Continuous XYZ
- Wie passt alles zusammen



Infrastructure as Code DevOps

Vagrant

Provisionierung von Virtuellen Maschinen oder Cloud Ressourcen

Vagrant

- System zum Erstellen von Virtuellen Maschinen
- Initial nur Support für VirtualBox
- Support für VMWare
- Und über Plugins für diverse Cloud-Plattformen
- Unterstützt diverse „Provisioner“
 - z.B. Puppet & Chef



Vagrant DSL – Pure Ruby

```
config.vm.define :vm1 do |vm1_config|

  vm1_config.vm.hostname = 'simplebox1'
  vm1_config.vm.network :private_network, ip: '192.168.56.11'
  vm1_config.hostmanager.aliases = %w(simplebox1.localdomain)

  vm1_config.vm.provider :virtualbox do |vb|
    vb.gui = true
    vb.name = "simplebox1"
    vb.customize ["modifyvm", :id, "--memory", "1024"]
    vb.customize ["modifyvm", :id, "--cpus", 1]
  end #vb

  vm1_config.vm.provision :puppet do |puppet|
    puppet.manifests_path = "manifests"
    puppet.module_path = "modules"
    puppet.manifest_file = "simple-shard-1.pp"
    puppet.options = "--verbose"
  end #puppet
end # vm1
```



Vagrant in Action

```
$ vagrant up vm1
Bringing machine 'vm1' up with 'virtualbox' provider...
[vm1] Importing base box 'ppt-prcs64'...
[vm1] Matching MAC address for NAT networking...
[vm1] Setting the name of the VM...
[vm1] Clearing any previously set forwarded ports...
[vm1] Creating shared folders metadata...
[vm1] Clearing any previously set network interfaces...
[vm1] Preparing network interfaces based on configuration...
[vm1] Forwarding ports...
[vm1] -- 22 => 2222 (adapter 1)
[vm1] Running any VM customizations...
[vm1] Booting VM...
[vm1] Waiting for VM to boot. This can take a few minutes.
[vm1] VM booted and ready for use!
[vm1] Setting hostname...
[vm1] Configuring and enabling network interfaces...
[vm1] Mounting shared folders...
[vm1] -- /vagrant
[vm1] -- /tmp/mongodbfiles
[vm1] -- /tmp/vagrant-cache
[vm1] -- /tmp/vagrant-puppet/modules-0
[vm1] -- /tmp/vagrant-puppet/manifests
[vm1] Configuring cache buckets...
[vm1] Running provisioner: hostmanager...
[vm1] Running provisioner: puppet...
Running Puppet with simple-shard-1.pp...
```

Puppet

Definition des Sollzustandes eines Systems

Puppet Know How in 2 Minuten...

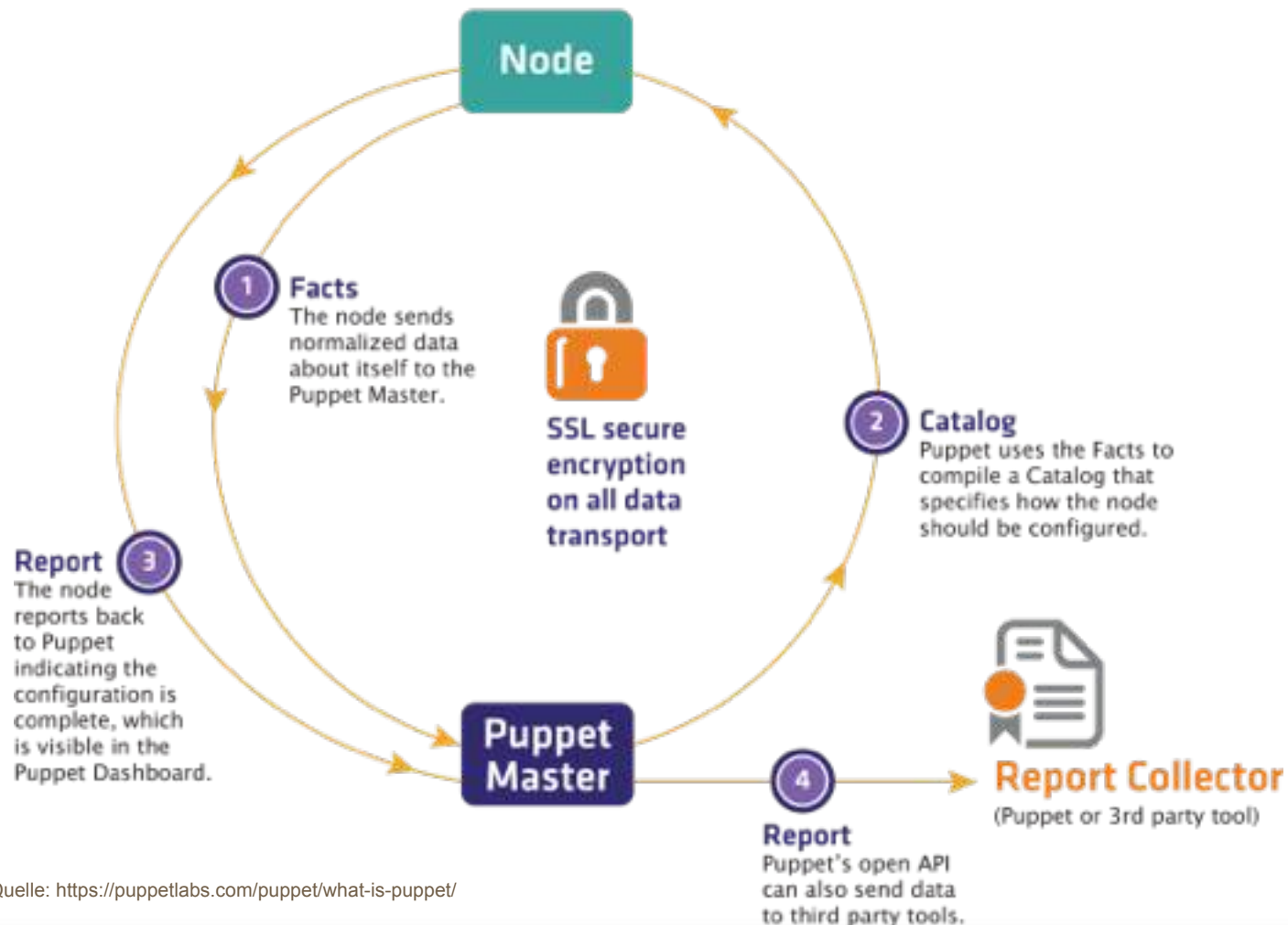
- Puppet besteht aus ...
 - Klassen
 - Manifesten
 - Modulen
- Puppet ist in Ruby implementiert
- Abstrahiert OS-Spezifika
 - Paket-Handling
 - Service-Handling (upstart vs. init.d vs. ...)



Mehr Puppet

- Alle Aktionen **idempotent**
- Client / Server oder Standalone
- „Enterprise“-Version
 - Bundle, Support
- Plugins in Ruby
- „Chef“ als Alternative
- „Maven vs. Gradle“

How Puppet works



Quelle: <https://puppetlabs.com/puppet/what-is-puppet/>

Puppet Know How ...

- Eigene DSL

```
service { $servicename :  
    name          => $servicename,  
    ensure        => running,  
    enable        => true,  
    require       => [File["/etc/init/  
                    $servicename.conf"]],  
    subscribe     => File[$config_file],  
}
```

Jenkins

*„An extendable open source
continuous integration
server“*



Jenkins

- Continuous Integration Server
- Fork aus dem Hudson-Projekt
- Erweiterbarkeit durch eine Unzahl von Plugins
- Platzhirsch unter den Open Source

Continuous Integration Servern

- Diverse Installationsmöglichkeiten
- Diverse SaaS-Angebote, z.B. *CloudBees*

Unser Szenario

- Web-Applikation muss gebaut und deployt werden
- Regelmäßige Builds und Auslieferungen!
- Jeder Build wird potentiell produktiv gesetzt
- Keine manuellen Eingriffe
- Vielleicht benötigen wir aber „Freigaben“

Agile

*Our highest priority is to satisfy the customer through early and **continuous delivery** of valuable software.*

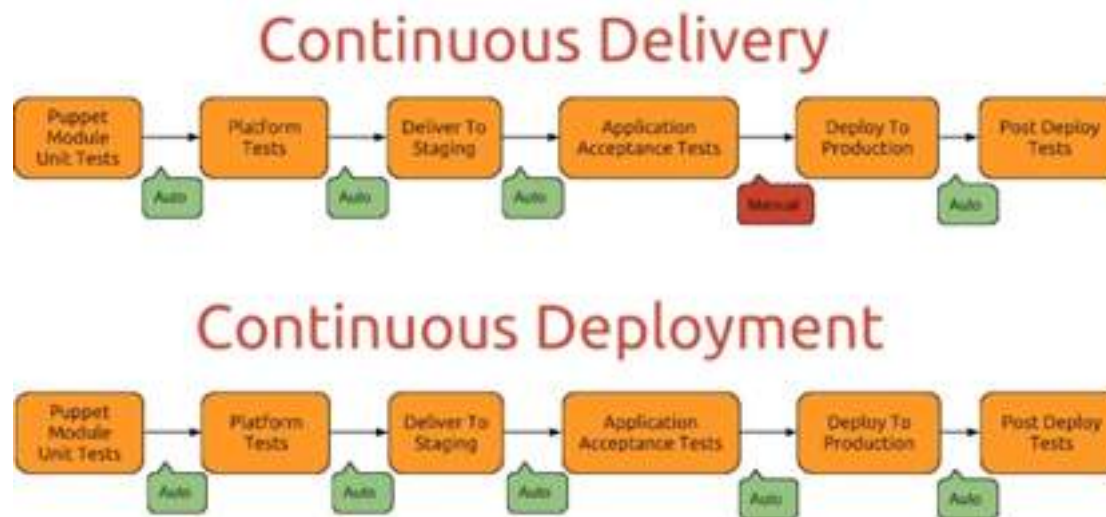
– [Agile Manifesto Principles](#)

Continuous Delivery

- Es werden kontinuierlich Builds erstellt
- Diese werden vollständig durchgetestet
- Bei Erfolg sind diese Produktionsreif
- ... und werden ausgeliefert
- vs. Continuous Integration
 - Beschränkt sich auf die Integration von Modulen
 - Vielleicht auch Integrationstests
 - Fokus ist Entwicklung, keine Produktivsetzung

Continuous Deployment

- Keine manuellen Eingriffe bei Produktivsetzung
- Auch keine manuellen Freigaben
- Deployments vollautomatisch in Produktion



Quelle: [Puppetlabs](#)

Unsere Build-Pipeline





Build

Snapshots
are
Evil!



„Maven-release-plugin is your dead“



Der Build

- Hier mit Maven
- Es werden immer „Releases“ erzeugt
- Auslieferung auf Repository Server
- „Golden Binary“
 - „Binary staging“
 - Support in Enterprise Repository Servern
- Diverse Maven-Plugins im Einsatz

Der Build - Spezialitäten

- „build-helper-maven-plugin“
 - Parsen der Version ohne Snapshot-Anteil
- „versions-maven-plugin“
 - Setzen von neuen Versionen
 - z.B. Git-Commithash, SVN-Revision, Jenkins-Buildnummer
z.B. *1.0.0-599*
- Jenkins „Parameterized Trigger Plugin“
 - Übergeben von Parametern an nachfolgende Schritte

Der Build – Ausführung in Jenkins

Maven Goals aufrufen

Maven Version

Goals

Build

Stamm-POM

Goals und Optionen

Der Build – Ausführung in Jenkins

- Maven Deploy aus Jenkins heraus

Bringe Artefakte in Maven-Repository aus (deploy)

Repository-URL	http://localhost:8081/nexus/content/repositories/releases/
Repository-ID	releases
	☒ Snapshots eindeutige Versionen zuordnen
Release environment variable	
	☐ Ausbringen (deploy), auch wenn der Build instabil ist.

Und der nächste Schritt

parameterized build on other projects

gers

Projects to build	example-step2,
Trigger when build is	Stable
Trigger build without parameters	☐

Predefined parameters

Parameters	gitrev=\$GIT_COMMIT
bnr=\$BUILD_NUMBER
------------	---

Parameters from properties file

Use properties from file	angularjs-springmvc/target/projectversion.properties
Don't trigger if any files are missing	☐



Provision Server

VM Provisionierung

- Hier lokal mit Vagrant über Shell-Befehle aus Jenkins
- VirtualBox-VM wird konfiguriert und gestartet
- In VM Java und Tomcat vorbereitet
- Jenkins-Plugins für diverse Cloud-Provider
 - z.B. OpenStack, AWS
- ... oder über Tools wie Puppet, Chef, Foreman usw.

Aufruf aus Jenkins

Branch Specifier (blank for default):

Shell ausführen

Befehl

```
export BUILD_ID=dontKillMe
cd angularjs-springmvc
vagrant up --no-provision
vagrant provision
vagrant status
```

[Liste der verfügbaren Umgebungsvariablen](#)

VM Innenleben

- Java und Tomcat müssen installiert werden
- Tomcat muss konfiguriert werden
- User innerhalb von Tomcat müssen eingerichtet werden
- Und das alles mit Puppet
 - Wiederholbar
 - Nachvollziehbar
 - Keine manuellen Konfigurationsschritte!

Java und Tomcat mit Puppet - Auszug

```
package { 'tomcat7':  
  require => [  
    Package['openjdk-7-jdk'],  
    Package['openjdk-7-jre'],  
    Package['openjdk-7-jre-headless'],  
    Package['openjdk-7-jre-lib'],  
    Package['authbind'],  
    Package['libtcnative-1'],  
  ],  
}  
  
file { '/etc/tomcat7/tomcat-users.xml':  
  owner => 'root',  
  require => Package['tomcat7'],  
  notify => Service['tomcat7'],  
  content => template('tc7/tomcat-users.xml.erb'),  
}
```



Deploy
App

Deploment der Applikation

- Verschiedene Wege nach Rom
 - Über Puppet, Maven, Shell-Skripte, dedizierte Werkzeuge ...
- Hier mit Jenkins „Deployment Plugin“
 - Basiert auf Cargo
 - Unterstützt diverse Container
 - Tomcat-Deployment über Management-Applikation
 - OK für Test
 - Produktion ggfs. mit anderem Werkzeug

Deployment mit Jenkins -Artefakte

- Vorher wird aber das Artefakt benötigt
- Unzureichender, komplizierter Support in Maven
- Einfacher:
 - über REST (ggfs. eigenes Plugin) falls angeboten
 - einfach HTTP / WGET
 - oder ein Jenkins-Plugin – wie in diesem Beispiel

Jetzt aber: Deployment

Deploy war/ear to a container

WAR/EAR files

Context path

Container

Manager user name

Manager password

Tomcat URL

Deploy on failure ☐



Test App

Testen

- Mehrere Testgruppen nach fachlich / technischen Kriterien
- Mehrere Jobs parallel ausführen
- Auf mehrere Testserver parallel deployen und testen

Testen

- Möglichst automatisieren
- Diversere Testwerkzeuge wie Selenium und JMeter
- Hohe Abdeckung erreichen
- „Happy Path“ ist Minimum
- In unserem Beispiel nur ein WGET ;-)



Und jetzt?

Build Pipeline



Vorteile

- Fallback - Deployment älterer Version möglich
- Aufsplitten in granulare Schritte
 - Schneller Feedback an Entwickler
- Hochautomatisiert
 - Keine Gefahr manueller Fehler
- Aufbrechen der Grenzen zwischen Entwicklung und Betrieb

Was fehlt im Beispiel? (Auszug)

- Datenbanken
 - DB-Versionierungswerkzeuge wie FlyWay oder LiquiBase
 - NoSQL
- Freigabe-Prozess (wenn notwendig)
 - „Build Promotion Plugin“ in Jenkins
- Komplexe Infrastrukturaspekte wie Firewall
- Konfigurationsdatenbank
- Canary Deployments, Blue/Green Deployment etc.



2.– 5. September 2013
in Nürnberg



Herbstcampus

Wissenstransfer
par excellence

Vielen Dank!

Dr. Halil-Cem Gürsoy

halil-cem.guersoy@adesso.de

Twitter: @hgutwit

G+ <http://goo.gl/hljRS>



Wir suchen Sie als

- Software-Architekt (m/w)
- Projektleiter (m/w)
- Senior Software Engineer (m/w)

jobs@adesso.de

www.AAAjobs.de

