wegweisend digital
T-SYSTEMS MULTIMEDIA SOLUTIONS

CONTINOUS DELIVERY

Aber sicher?!

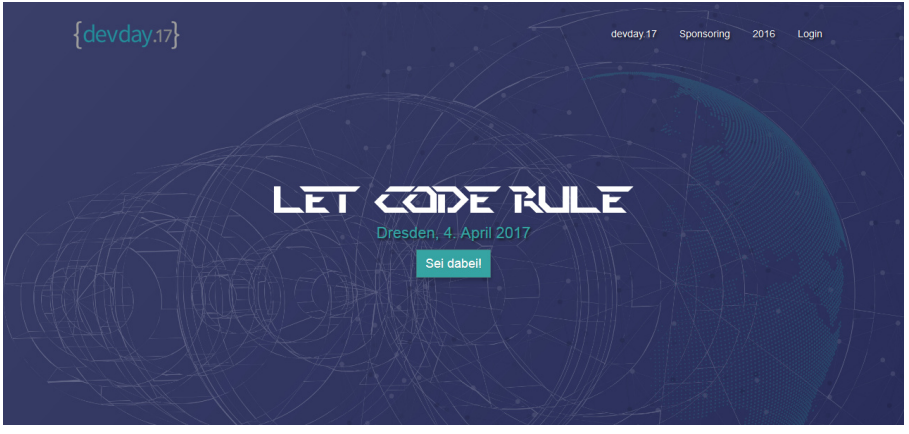
Softwareentwicklung in der industriellen Praxis – Dresden, 23.01.2017

T · · Systems ·

Jan Dittberner

- Softwarearchitekt
- Themen: Linux, PKI, Automatisierung, ...
- Debian Developer
- Teamlead Infrastructure CAcert.org

DevDay.17 - https://www.devday.de/



SECO
<Software Engineering Community>

Continous Delivery

Entwicklungsumgebungen

Nachvollziehbare Deployments

Security-Aspekte

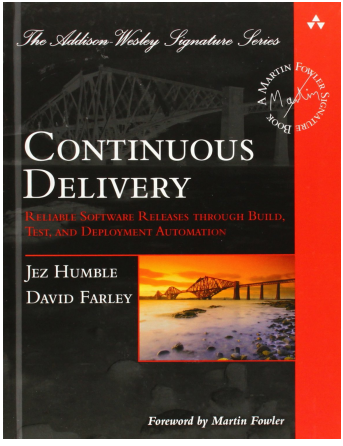
Fazit

CONTINOUS DELIVERY
URSPRÜNGE

2001: Principles behind the Agile Manifesto

`http://agilemanifesto.org/principles.html`

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



2010: Jezz Humble, David Farley, “Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation” (Addison-Wesley Signature)

CONTINOUS DELIVERY
AUSLIEFERUNG VON SOFTWARE - ENTWICKLUNG

Works on my machine

- Entwicklung lokal
- Software manuell oder mit IDE zusammengebaut
- manuell Entwicklergetestet
- Austausch per Fileshare, E-Mail, ...

Quelle: blog.codinghorror.com

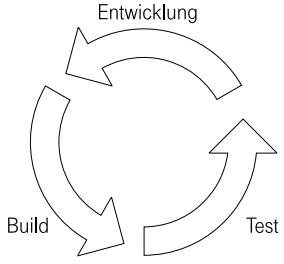
Grundlegendes Bewusstsein für Qualität

- Entwicklung lokal
- Einsatz von Versionskontrolle
- Softwarebuild automatisiert mit Skripten oder Tools wie gradle, Maven, ant, make, ...
- Unittests
- ...

CONTINOUS DELIVERY
AUSLIEFERUNG VON SOFTWARE - ENTWICKLUNG

Continuous Integration

- Entwicklung lokal
- Commit/Push in gemeinsames Sourcecode-Repository
- automatischer Build auf Continuous Integration Server (z.B. Jenkins)
- automatische (Unit-)Tests auf Continous Integration Server
- Feedback an Entwickler
- Ablage von Ergebnissen in Repositories



CONTINOUS DELIVERY
AUSLIEFERUNG VON SOFTWARE - ENTWICKLUNG

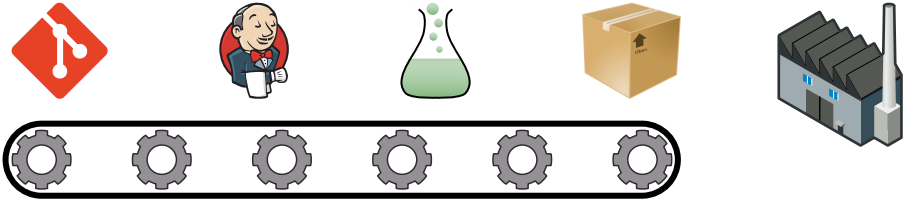
Continuous Delivery

- Erweiterung von Continous Integration
- automatisches Deployment der gebauten Software
- automatische Konfiguration von Umgebungen/Software
- umgebungsspezifische Tests
- Feedback an Entwickler/Tester/Betriebsteam/Kunden

CONTINOUS DELIVERY
AUSLIEFERUNG VON SOFTWARE - ENTWICKLUNG

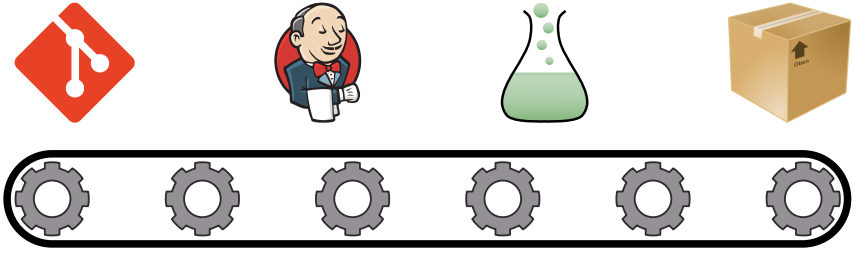
Continuous Delivery

- Steuerung durch Continous Delivery Pipeline



CONTINOUS DELIVERY
WARUM?

Continuous Delivery



- Features schneller beim Anwender
- schnelles Feedback
- Qualitätssteigerung
- mehr "'Denkarbeit"' statt langweiliger Routine

CONTINOUS DELIVERY
AUTOMATISIERUNG

- Softwarebuild
- Unit-, Acceptance-, Integration-, Performance-, ...-Test
- Umgebungsbereitstellung (Provisionierung)
- Deployment
- Lieferprozesse
- Dokumentation
- Monitoring/Reporting
- ...alles was sonst noch Routine sein sollte

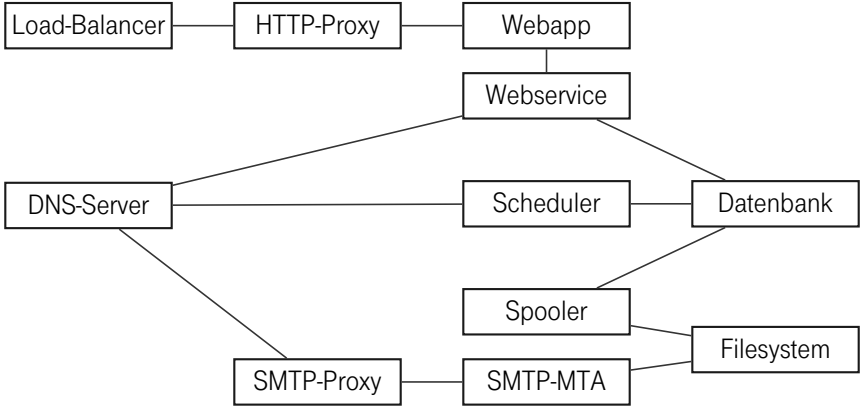


Figure: vereinfachtes Architekturbild

ENTWICKLUNGSUMGEBUNGEN
PUPPET

https://puppetlabs.com

- Tool für Configuration Management
- Infrastruktur wird als Code beschrieben (Puppet DSL, YAML, Ruby)
- neue Maschinen können innerhalb weniger Minuten nachvollziehbar eingerichtet werden
- kann auch mit anderen Tools realisiert werden (Beispiel cfengine, Saltstack, Ansible, ...)

ENTWICKLUNGSUMGEBUNGEN
PACKER UND VAGRANT

https://www.packer.io
https://www.vagrantup.com

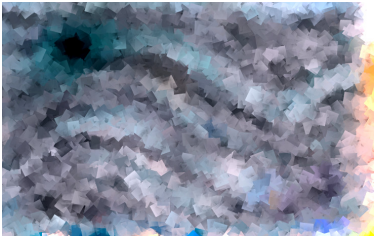
- Mit Packer kann man sauber definierte Basisimages erstellen
- Vagrant steuert (lokale) VMs über eine textuelle Beschreibung
- dadurch können Entwickler mit einem produktionsnahen OS arbeiten
- gleicher Puppet-Code für lokale und andere Umgebungen
- lokaler Test von Deployments möglich

Auch Sicherheitsfunktionalität sollte produktionsnah umgesetzt sein:

möglichst gleiche Einstellungen, Schlüssellängen, Algorithmen, ...auch in Entwicklungs- und Testumgebungen

so nicht
Quelle: starecat.com

so auch nicht



...sondern so

NACHVOLLZIEHBARE DEPLOYMENTS
DEPLOYMENTS

- ... müssen nachvollziehbar sein und zur Software passen
- ... müssen getestet werden
- auch Änderungen an der Persistenzschicht (z.B. Datenbankänderungen) gehören dazu
- ... und natürlich Konfigurationsdaten

NACHVOLLZIEHBARE DEPLOYMENTS
JENKINS

- Software übersetzen, paketieren, automatisch testen mit Maven oder anderem Build-Tool
- Konfiguration der Software vor dem Deployment
- Ausführen von Datenbankmigrationen (bei uns mit Liquibase)
- Deployment auf Entwicklungs- und Testumgebungen

NACHVOLLZIEHBARE DEPLOYMENTS
METADATEN

Information welcher Softwarestand,
in welcher Konfiguration, wo deployt ist

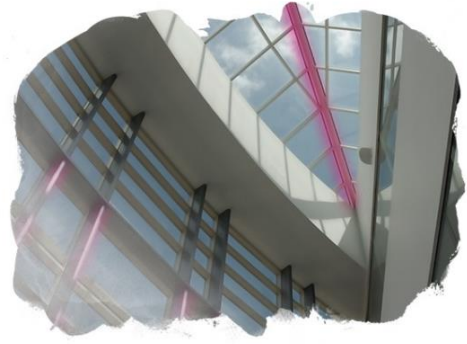
- benötigt für Testaussagen
- benötigt für Monitoring
- beim Build werden Versionen in definierte Dateien geschrieben
- beim Deployment werden daraus Custom Facts für Puppet und Wiki-Seiten
- von Puppet in PuppetDB übertragen

NACHVOLLZIEHBARE DEPLOYMENTS
MONITORING

- Auch für Entwicklungs- und Testsysteme nützlich
- Vermeiden von "'Blindflug"', Früherkennung von Problemen
- Üben für den Wirkbetrieb
- um manuelle Einrichtung zu vermeiden registrieren sich unsere Maschinen beim Puppetlauf selbst fürs Monitoring durch Icinga2

SECURITY-ASPEKTE
VERSCHLÜSSELUNG VON CREDENTIALS

- betriebliche Anforderung: keine Klartextpasswörter on Disk
- Lösungen:
  - Jasypt
  - Wildfly-Vault (Applikationsserver-spezifischer Mechanismus)
  - Encrypted Tablespaces in Datenbank
  - Hashicorp Vault

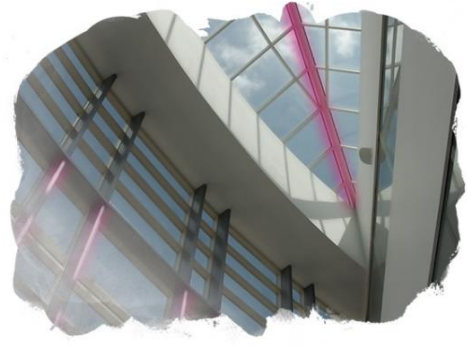


SECURITY-ASPEKTE
VERSCHLÜSSELUNG VON CREDENTIALS

Jasypt

http://www.jasypt.org/

- Verschlüsselung von Properties, Texten oder ganzen Dateien mit Standard-Algorithmen
- einfach integrierbar mit Spring, Hibernate und anderen Java-Frameworks
- Verschlüsselung passiert bei uns während der Deploymentkonfiguration
- Entschlüsselung beim Applikationsstart/ zur Laufzeit

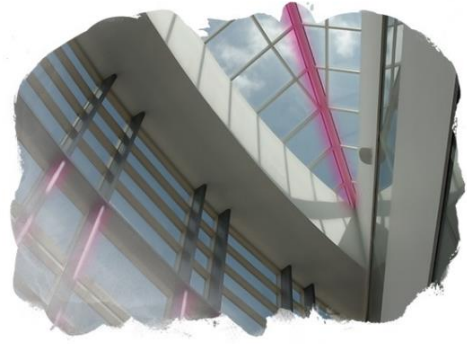


SECURITY-ASPEKTE
VERSCHLÜSSELUNG VON CREDENTIALS

Alternative: Hashicorp Vault

https://www.vaultproject.io/

> **Vault** secures, stores, and tightly controls access to tokens, passwords, certificates, API keys, and other secrets in modern computing. Vault handles leasing, key revocation, key rolling, and auditing. Through a unified API, users can access an encrypted Key/Value store and network encryption-as-a-service, or generate AWS IAM/STS credentials, SQL/NoSQL databases, X.509 certificates, SSH credentials, and more.



SECURITY-ASPEKTE
ZERTIFIKATE

- Anforderung: **Alle** Netzwerkverbindungen müssen TLS-verschlüsselt sein
- Herausforderung: mehrere hundert gültige Zertifikate in unterschiedlichsten Keystore-Formaten benötigt
- Lösung:
  - Projekt-CA inkl. OCSP-Responder für Entwicklung und Test (aktuell auf Basis von OpenSSL)
  - Maven-Plugins für Erzeugung von Signing-Request und Keystore-Formaten
  - Jenkins-Jobs für Steuerung der CA, Erzeugung von umgebungsspezifischen Keystores und Checks auf Zertifikatsablauf

SECURITY-ASPEKTE
ZUGRIFF AUF MASCHINEN

- dedizierte Applikationsnutzer
- zentrale Verwaltung von SSH-Keys (SKM)
- Verteilung des SKM-Master-Keys per Puppet
- kein Zugriff auf Produktionsmaschinen für Entwicklungs-/Testteam

SECURITY-ASPEKTE
ABSICHERUNG DER (ORACLE-)DATENBANK

- Encrypted Tablespaces
- Database Vault
  - ermöglicht Vier-Augen-Prinzip für Datenbankänderungen
  - verhindert Datenzugriff auch für DBAs
- zu beachten bei Datenbankmigrationen

SECURITY-ASPEKTE
ABSICHERUNG VON DEPLOYMENTS

- unkonfigurierte Lieferartefakte
- Konfigurationsdateien nur als Templates in der Lieferung
- Konfigurationsdaten liegen in getrennten Repositories
- Konfiguration findet auf separaten Maschinen statt
- Deployment per Key-Based SSH

FAZIT

- Continuos Delivery ist sinnvoll – auch für sicherheitsrelevante Projekte
- Entwicklungs- und Testumgebungen so wirkbetriebsnah wie möglich
- **Alles** automatisieren – auch Security-Aspekte

KONTAKT

wegweisend digital
T-SYSTEMS MULTIMEDIA SOLUTIONS

Jan Dittberner

T-Systems Multimedia Solutions GmbH

Mail: jan.dittberner@t-systems.com
Twitter: @jandd

https://coffeestats.org/