DevOps & 8000 Server

Matthias Klein

About Me

- Seit 2009 in der Administration von Browsergames tätig
- TeamLead Backend Operations bei InnoGames
- Früher Sysadmin bei „Die Stämme“
- Aktuell zuständig für: Payment, Conversion, MySQL, Mail ...

1. Theorie
2. Vergangenheit
3. Gegenwart
4. Zukunft

Was ist eigentlich … DevOps?

- Kultur (Verhalten untereinander, Verantwortung gegenüber dem Projekt, gegenseitiges Vertrauen)
- Praktiken (Regeln, Rollen, Prozesse, Metriken)
- Tools (Nutzung gemeinsamer Technologien, Erstellung von Tools füreinander)

Was ist eigentlich … Kultur?

- Alle Beteiligten bilden ein Team (auch der SysAdmin)
- Das Produkt steht im Vordergrund, nicht die Einzelleistung
- Es gibt kein „Fingerpointing“, im Fehlerfall werden Möglichkeiten zur Verbesserung erarbeitet und implementiert
- Teamentscheidungen werden vom Team getragen

Was ist eigentlich … Praktiken?

- Vor Projektstart werden Zuständigkeiten zugewiesen
- Der Workflow wird einheitlich für das Projekt abgesprochen (auch über mehrere Komponenten)
- Definition der Prozesse von der Anfrage bis zur Auslieferung
- Woran wird der Erfolg gemessen und wie

Was ist eigentlich … Tools?

- Festlegung der Grundtechnologien (z.B. git oder svn, php oder java, python oder bash)
- „Helferlein“ werden allen zur Verfügung gestellt (z.B. scripte)
- Erstellung von Tools, die man selbst zwar nicht braucht, die aber anderen die Arbeit erleichtern

Meine persönlichen Erfahrungen (I)

- Starke Trennung von Teams (Dev / Ops / QA)
- Starre Prozesse (z.B. Abzeichnen vor Deployments)
- Devs dürfen nicht auf Produktion zugreifen → Deployment nur durch Ops
- Devs erstellen ihre Testumgebung selbständig
- Ops verwalten die Produktion selbständig → Devs müssen nachfragen, wenn sie Konfigurationen von dort wollen
- Bei Problemen mußte immer ein Schuldiger gefunden werden

Meine persönlichen Erfahrungen (I)

- Der Rollout von Bugfixes dauert unnötig lang
- Nervige Anfragen nach logfiles, configs, etc.
- Unterschiedliche Konfigurationen von Produktion, Test und lokalen Umgebungen
- Keine gemeinsamen Ziele, individuelle Ziele widersprechen sich häufig
- Sehr wenig Automatisierung (z.B. simples Kopieren von Konfigurationen)

Meine persönlichen Erfahrungen (II)

- Devs haben /root auf der Produktion
- Devs deployen selbständig und ohne Ankündigung an Ops
- Devs installieren und konfigurieren benötigte Software selbständig in der Produktion
- Nur Standardsoftware wird von Puppet verwaltet
- Wenig Kommunikation zwischen Devs und Ops

Meine persönlichen Erfahrungen (II)

- Devs zerstören die Produktionsumgebung, weil sie für die Installation einer Software die Anweisungen vom ersten Google-Treffer befolgen
- Die Server laufen plötzlich Amok, nach längerer Fehlersuche findet man das letzte Release als Ursache
- Server lassen sich nicht neu aufsetzen, weil benötigte Software nicht von Puppet verwaltet wird
- Die Anwendung funktioniert nicht mehr, weil ein „unnötiges“ Paket entfernt wurde

Mit DevOps wird alles besser ;-)

- Gemeinsames Analysieren von Problemen/Anforderungen und Erarbeitung von Strategien zur Verbesserung
- „Jammern auf hohem Niveau“ ist explizit erwünscht
- Jeder muß die DevOps-Kultur (vor)leben
  - Wir sind ein Team, jeder ist für den Erfolg verantwortlich
  - Wenn Fehler passieren, ist nicht das Individuum, sondern das Team schuld – erstmal ;-)

Wie wird DevOps zum Erfolg?

- Jeder muß sich selbst zurücknehmen und die DevOps Kultur verinnerlichen
- Devs machen so wenig Ops wie möglich
  - Weil sie kein Interesse daran haben
  - Weil es unnötig ist
    - Durch bereitgestellte Informationen
    - Durch bereitgestellte Tools

DevOps und Entwicklerteams (persönliche Erfahrungen)

- Je größer das Team, desto schlechter die Aussicht auf Erfolg
- Ideal sind bis zu 6 Entwickler, kann je nach Team auch mit mehr funktionieren
- Bei großen Teams sollten einige aus verschiedenen Bereichen ein DevOps Team bilden und dann in den jeweiligen Bereich weitertragen

Unsere Maßnahmen

- Jeder SysAdmin arbeitet mindestens 1 Tag/Woche im Büro des Entwicklerteams und ist dann nur für deren Belange da
- Teilnahme an StandUp, Retro und Planungsmeetings
- Entwickler sorgen dafür, daß ihre Software redundant lauffähig ist
- Admins sorgen dafür, daß Server automatisiert (neu) aufgesetzt werden können
- Alle sorgen für ein reproduzierbares Deployment

Puppet

- Jede Maschine muß allein mit puppet zur Produktionsreife kommen (funktioniert nicht, wenn backups eingespielt werden müssen)
- Jede Entwicklungsstufe verwendet die selben Manifeste und Templates, Variablen werden über hiera verwaltet
- Puppet Environments und git branches ermöglichen das Testen von Änderungen
- External Nodes und ein Admintool sorgen für die Zuordnung

Paketmanagement

INSTALL.SH

```
#!/bin/bash

pip install "$1" &
easy_install "$1" &
brew install "$1" &
npm install "$1" &
yum install "$1" & dnf install "$1" &
docker run "$1" &
pkg install "$1" &
apt-get install "$1" &
sudo apt-get install "$1" &
steamcmd +app_update "$1" validate &
git clone https://github.com/"$1"/"$1" &
cd "$1";./configure;make;make install &
curl "$1" | bash &
```

Paketmanagement

- Verhindert in Kombination mit Nagios checks, daß unbekannte/unerwünschte Pakete installiert sind
- Stellt einen einheitlichen Patch-Stand über alle Systeme sicher
- Mittels apt-mirror, apt-ftparchive und einem selbst entwickelten Repomanager werden Pakete zur Verfügung gestellt

Jenkins

- Devs haben die Testabdeckung und -qualität auf nahezu 100% erhöht
- Jenkins wird als Jobstarter benutzt, um Probleme mit unterschiedlichen Versionsanforderungen zu umgehen
- Jobs werden in VM‘s ausgeführt, nicht in … weil
  - Vagrant: sehr schlechte Performance, undefiniertes Verhalten bei mehreren gleichzeitig startenden boxen
  - Docker: bildet die Produktion nicht korrekt genug ab, soll nicht als VM mißbraucht werden
- Erstellen von .deb Paketen

Vagrant

- Einfache Konfigurationshilfe: https://github.com/innogames/invade
- Bildet die Produktionsumgebung so gut wie möglich auf dem Entwicklerrechner ab
- Provisionierung via Puppet


Live Demo

ToDo / WiP

- Entwickler erhalten die Möglichkeit, eigene Änderungen in Puppet vorzunhemen
- Refactoring und Veröffentlichung der Puppetmodule auf GitHub
- Erstellen und Provisionieren von Maschinen nach Bedarf


noch
Fragen?