DevOps in der Praxis

Matthias Klein

About InnoGames

- Betreibt und entwickelt Browser- und Mobile Games seit 2007 für über 150 Millionen Spieler
- Mehr als 400 Mitarbeiter aus über 30 Nationen sind an den Standorten Hamburg und Düsseldorf tätig

About Me

- Seit 2009 als Administrator in der Gaming Industrie tätig
- Zuständig für das Payment System
- Ansprechpartner für MySQL, Mail, ELK

1. Hilfe, mein Chef will DevOps!
2. Refactoring, aber wie?
3. Vom Fail zum Win
4. Utopia

Was der Chef sagt

„Ich habe da was über dieses DevOps gehört/gelesen, das klingt ja alles super. Das machen wir jetzt auch!“

Was bei vielen Mitarbeitern ankommt

„Da wird gerade mal wieder eine neue Sau durchs Dorf getrieben, da müssen wir für die da oben mitmachen. Jetzt muss ich auch noch den ganzen Dev-Krempel lernen und die Typen dürfen in meinen Systemen rumpfuschen. Und um die ganzen Hipster-Tools muß ich mich dann auch noch kümmern.“

Was DevOps NICHT ist

- Entwickler und Admins können alle Aufgaben erledigen
- Benutzen von „DevOps“-Methoden
- Benutzen von „DevOps“-Tools
- Einfach implementierbar

Was DevOps wirklich ist

Eine **Kultur** im Unternehmen/Projekt/Team, die auf gegenseitigem **Respekt** und **Vertrauen** basiert. Alle haben das Ziel, das **Projekt** ständig zu verbessern. **Erfolge und Misserfolge** werden dem **Team** zugeschrieben.

Was verändert DevOps für mich?

- Kommunikation mit allen anderen
- Besseres Verständnis für die Bedürfnisse anderer
- Die beste Lösung für das Projekt wird gewählt
- Continuos Delivery ist ein Segen, ehrlich

Wir denken, wir machen DevOps

- Gute Kultur, wir arbeiten weiter daran
- Alle relevanten Personen sind in den Meetings dabei
- Automatisierung mit Puppet, Jenkins, .deb
- Monitoring mit Naemon, Graphite, ELK

Wir sollten da mal was tun...

- Starke Fragmentierung in den Puppet Manifests, Duplikate
- Entwickler können zu wenig selbst tun
- Verbesserungen in der Netzwerkinfrastruktur können nicht genutzt werden

Ein unerwarteter Helfer...

- Refactorings kosten viel Zeit, der Nutzen kann oft nicht beziffert werden
- Da wir Kreditkartendaten verarbeiten, verlangt ein Partner eine Zertifizierung von uns. Dies bedeutet umfangreiche Umstrukturierungen.
- Hat eigentlich mal jemand den Failover getestet?

Die Applikation soll das Rechenzentrum wechseln

- Wir können mittels BGP die Public IP auf das andere Rechenzentrum schalten
- Alle Dienste nutzen die Datenbank im eigenen Rechenzentrum
- Kritisch: Datenbankrepliaktion

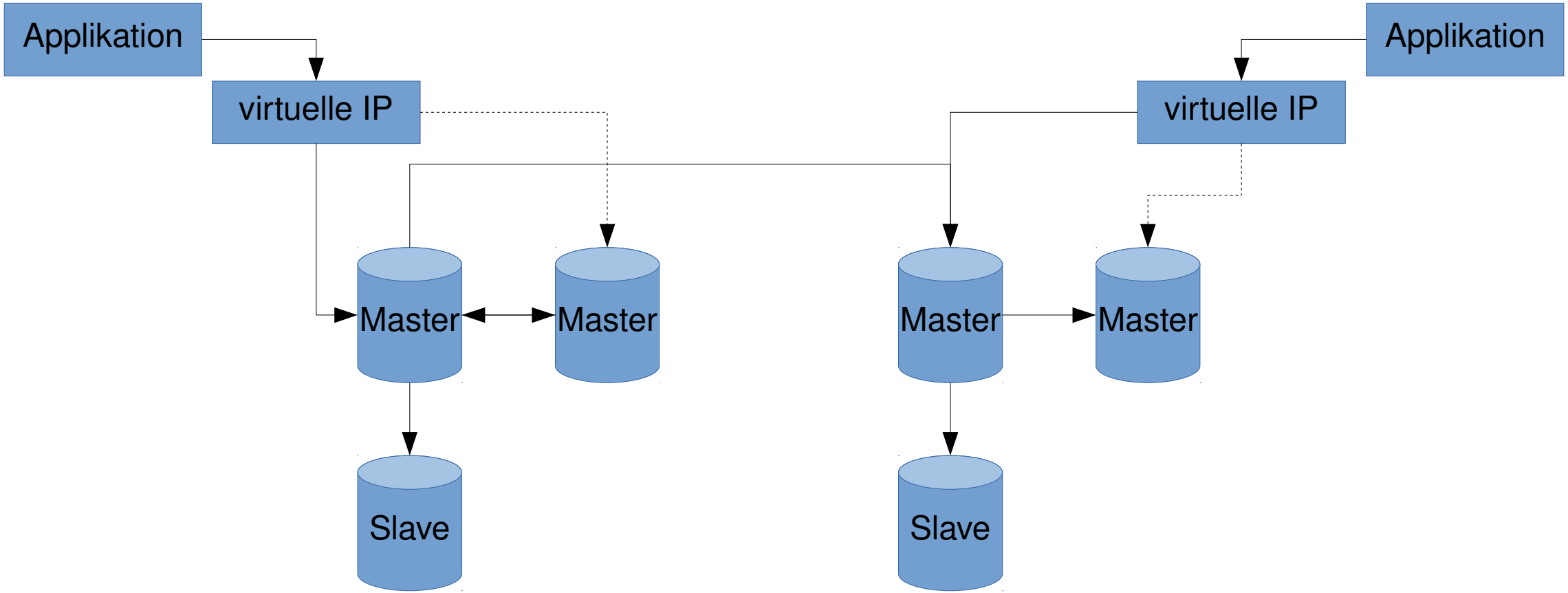
Exkurs - synchrone Replikation

- Relativ neue Technologie
- Teils sehr spezielle Anforderungen an Hard- und Software
- Längere Zeit vom Commit zum OK
- Sehr robust mit Selbstheilungskräften
- Automatischer Failover durch die Applikation oder Drittsoftware

Exkurs – asynchrone Replikation

- Gereifte Technologie (seit 2001 verfügbar)
- Slave Lags möglich
- Inkonsistenzen möglich
- Automatischer Failover nur durch Drittsoftware oder Applikation möglich

Unsere Lösung



Dynamisches DevOps

- Je nach Anforderung wird die Toolchain und Methodik gewählt
  - kleines Helferscript wird geändert: git push in den master
  - kleinere Änderungen, die Kernaufgaben der anderen betreffen: branch + merge request + ggfs. Tests
  - Strukturelle oder potenziell gefährliche Änderungen: vorbereitende Meetings, gemeinsame Durchführung, Nachbesprechung

Wir erstellen einen Plan

- Herunterfahren aller daemons → Sind für die asynchrone Abarbeitung zuständig, kein Problem
- Herunterfahren der Webserver → Die Downtime beginnt
- Im bis dato passiven Rechenzentrum wird Master – Master konfiguriert
- Hochfahren der Public IP → Ende der Downtime (Plan: 10 Minuten)
- Aufräumarbeiten und Beobachtung
- Wir wechseln zurück, um die andere Richtung zu testen

Wir führen den Plan durch

- Früh am Morgen treffen sich
  - LeadDev + BackendDev
  - SysAdmin + seine Vertretung
  - TeamLead Netzwerk
- Alles funktioniert wie geplant, die Downtime bleibt innerhalb der Vorhersage
- Plötzlich rotes Nagios: das nun passive Rechenzentrum kann nicht mehr replizieren → Mit dem Wechsel zurück wird das nun wohl nichts

Fehlersuche

- Risiko- / Schadensabschätzung mit allen Anwesenden
- Abstimmung der weiteren Vorgehensweise, nur unbedingt benötigte Personen verbleiben, andere „auf Bereitschaft“
- Fehlerbehebung
- Nachbesprechung
- Maßnahmen für die Zukunft

Wir führen den letzten Teil des Plans durch

- Früh am Morgen
  - Sitzt der zuständige Sysadmin zu Hause am Rechner
  - Der LeadDev ist per Telefon erreichbar
- Alles funktioniert

Ich hätt da mal ne Frage...

„… die Umsatzzahlen für letzte Woche waren gestern anders als heute. Habt Ihr irgendwas gemacht?“

- Die „Task Force“ findet heraus: Der daemon zur Berechnung der Gebühren lief nach dem ersten Switch nicht und hat nach dem Zweiten seine Arbeit für die Vergangenheit getan.
- Schaden: Vorhersagen und Analysen, die auf Grund falscher Daten entstanden, verbrannte Arbeitszeit

Auf der Suche nach dem Schuldigen

- Ops haben keinen Check implementiert, ob der Daemon läuft
- Devs haben den Ops nicht gesagt, daß es den Daemon gibt
- Es existiert kein Check, der die Validität der Daten prüft

- Wir haben einen Zustand, der so nicht sein soll, das müssen wir beheben. Es gibt keinen Schuldigen.

Post Mortem

- Durch die Fragmentierung von Puppet dauert die Vorbereitung für einen Switch viel zu lang, im Ernstfall kann man das nicht vorbereiten
- Wir benötigen eine generische Lösung zur Überprüfung, ob ein Daemon läuft und macht, was er soll
- Die Namen der Daemons machen ein einfaches an-/abschalten unmöglich,
- Die Namen der Daemons machen eine generische Generierung von systemd services und dazugehörigen checks unnötig schwer

Das 2 Rechenzentren Problem

- Im aktuellen Setup muß nach einem Failover die Replikation von Hand angepasst werden, aber
  - Multi Source Replication verursacht mehr Traffic
  - Synchrone Replikationen benötigen eine ungerade Anzahl von Rechenzentren
  - Mischformen benötigen Anpassungen in der Applikation

Was tun?

- Alle Beteiligten von der Notwendigkeit einer Änderung überzeugen
- Gemeinsam die beste Lösung auswählen und implementieren

Zu guter Letzt

- Die reine Anwendung von DevOps Tools und Methoden bedeutet nicht, daß man DevOps macht
- Die Kultur und der Umgang miteinander sind ausschlaggebend
- DevOps Tools und Methoden unterstützen auf dem Weg zu einer ständigen Verbesserung
- Einen vernünftig Umgang untereinander mit dem Ziel, das Produkt zu verbessern, kann man DevOps nennen. Man muß es aber nicht.

NOCH FRAGEN?

VIELEN DANK FÜR IHRE AUFMERKSAMKEIT!

INNOGAMES @GITHUB:

WWW.GITHUB.COM/INNOGAMES

INNOGAMES IS HIRING:

WWW.INNOGAMES.COM

 @innogames  /innogames  /innogames

 matthias.klein@innogames.com