Best Practices in der IT-Administration, Version 2018

Michael Prokop,
am 20.03.2018

SynPro SOLUTIONS

INFRA LOVERS

% whoami
mika

- @mikagrml
- Grml.org Erfinder + Projektleiter
- Debian Entwickler
- Grml Solutions
- SynPro Solutions

Roadmap

Die Rolle des Admins
Damals vs. Heute
Tooling & Best Practices
Stolperfallen & Schmerzen
Ausblick

Die Rolle des Admins

Das Admin-Dilemma

gute IT-Infrastruktur / IT-Administration spürt man nicht und wird als selbstverständlich empfunden

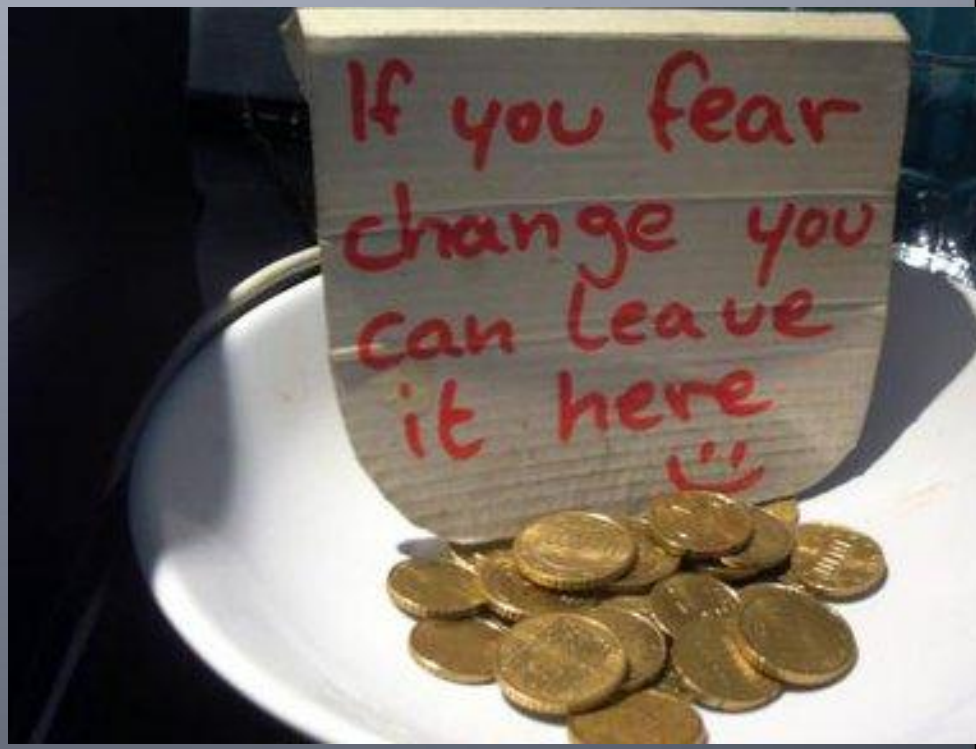
Das Admin-Dilemma

gute IT-Infrastruktur / IT-Administration spürt man nicht und wird als selbstverständlich empfunden

… aber wehe es geht etwas nicht

Der moderne™ Admin

"Hierzulande musst du so schnell rennen, wie du kannst, wenn du am gleichen Fleck bleiben willst" – spricht die rote Königin zu Alice (aus: Alice hinter den Spiegeln)



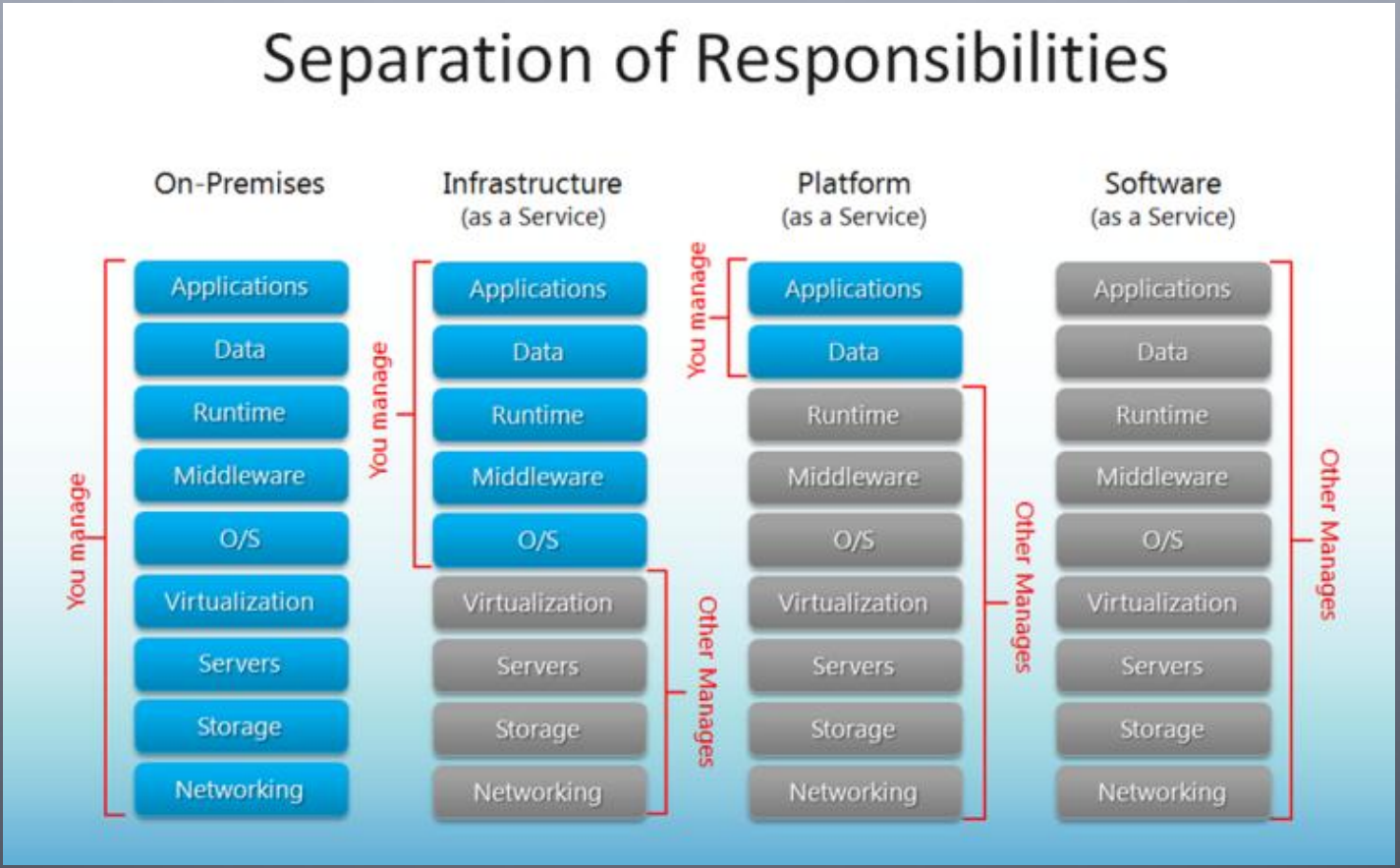
Quelle: https://twitter.com/gabrealness/status/723434940734431237

Know your tools

"If I had eight hours to chop down a tree, I'd spend six hours sharpening my ax." – Abraham Lincoln

"We learn geology the morning after the earthquake." – Ralph Waldo Emerson

… as a Service



Quelle: http://robertgreiner.com/2014/03/windows-azure-iaas-paas-saas-overview/

Damals™
vs.
Heute

Damals™ vs Heute

- Fette Server vs. VMs+Container, Microservices, ServerLess,…
- Eigene Hardware vs. Cloud
- Statische Infrastruktur vs. Service Discovery + Autoscaling
- Neues (manuelles) Deployment alle X Monate/Jahre vs. unzählige Male pro Woche/Tag/Stunde/Minute (inkl. Auto-Deployments)

Damals™ vs Heute

- Vertikales vs. Horizontales Skalieren
- Open Source ist böse vs. Open Source ist überall („it‘s a corporate thing“)
- “Mein Server hat schon >2 Jahre Uptime” (security updates?!) vs. Service Availability (Live Migration, Load Balancers, Service-Verfügbarkeit,…)

Damals™ vs Heute

- $Skript vs. Konfigurationsmanagement
- Prozedural/imperativ (Perl, Ansible, Chef) vs. deklarativ (Puppet, SaltStack, Terraform)
- IPv4 vs. IPv4 + IPv6
- Teure SSL-Zertifikate (oder kein SSL/TLS) vs. Let’s Encrypt

Damals™ vs Heute

- Sourceforge vs. Github
- RRD vs. Time Series Databases
- Shell/Perl vs. Go/Python/Ruby(/Shell/Perl)
- Mailreports vs. Dashboards

Damals™ vs Heute

- IT läuft da hinten im Kammerl™ vs. Produktionskritisch/IoT/…
- Vor-Ort-Admins vs. Remote(-Only) Admins
- Admin ↔ Developer vs. DevOps/SRE
- Elfenbeinturm IT vs. Infrastructure as Code

Neue Ansätze, Trends,…

- Chatops
- BigData
- NoSQL
- Künstliche Intelligenz mit Machine Learning, Neuronale Netze, Deep Learning, Natural Language Processing, Data Mining,…
- … und vieles, vieles mehr!

Tooling & Best Practices

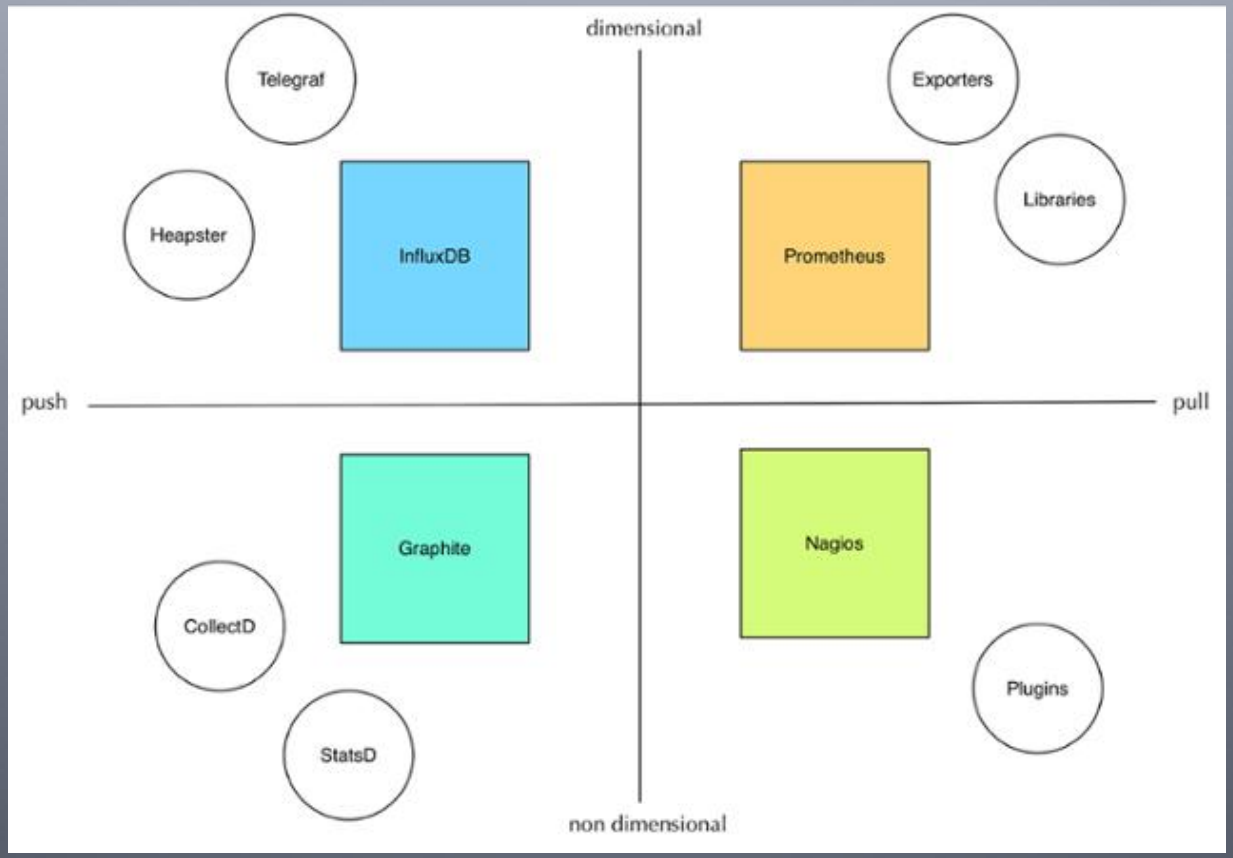
Monitoring

- Was nicht im Monitoring ist, kann nicht wichtig sein
- Monitoring aus Geschäftssicht (nicht alles was überwacht werden kann ist auch sinnvoll)
- Actionable(!) Notifications
- Dynamisch (Service Discovery)
- Elastisch + Adaptiv (während Backup-Lauf darf Load höher sein)

Was brauchen wir in der Praxis?

- History mit einbeziehen
  - wir pendeln zw. 79% + 81% Disk Usage? uninteressant
  - Interessanter: wann muß ich reagieren, wenn sich die Disk weiter *so* füllt?
- Beliebige Dimensionen miteinander kombinieren (und zwar auch erst im Nachhinein!)

Verschiedene Ansätze 1/2



Quelle: https://blog.outlyer.com/metrics-nagios-graphite-prometheus-influxdb

Verschiedene Ansätze 2/2

- “Nagios”-Ansatz: kleine, statische Umgebungen, mit Blackbox-Ansatz
- “Prometheus”-Ansatz: dynamische oder cloud-basierte Umgebungen, mit Whitebox-Ansatz
- Viele Lösungen bewegen sich irgendwo dazwischen (Icinga, check-mk,…)

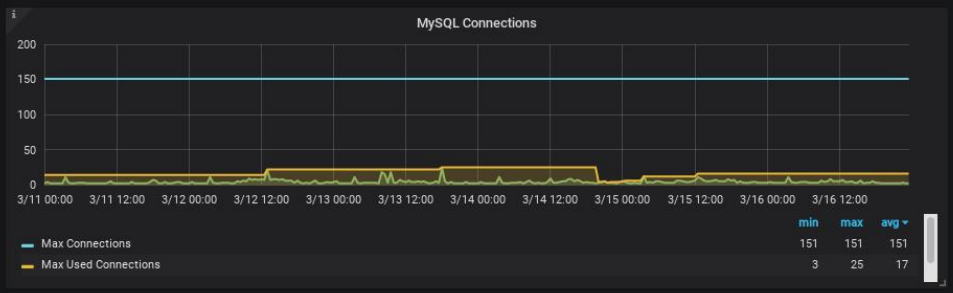
2.1 days

174.86

4 GiB

41%

Connections



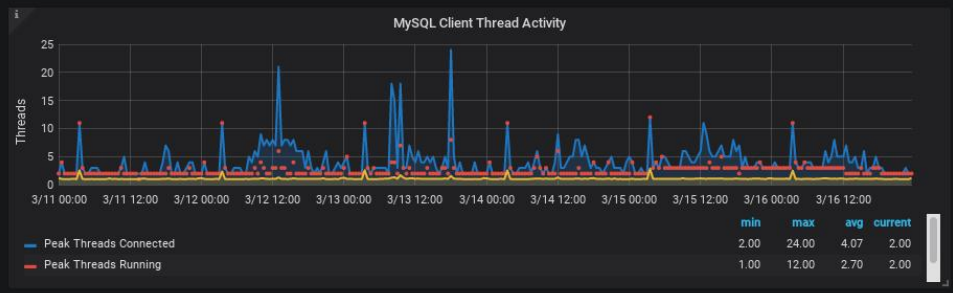
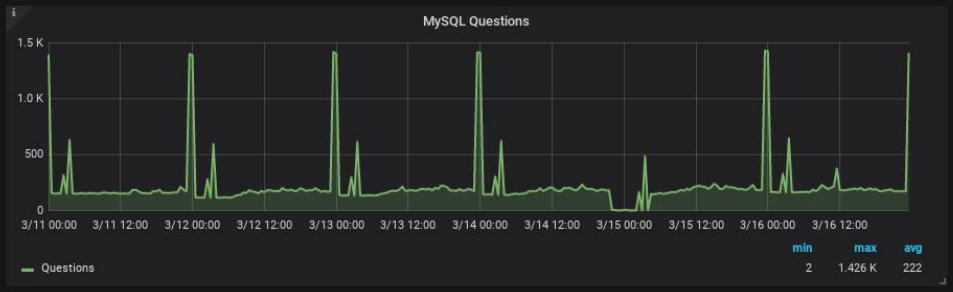
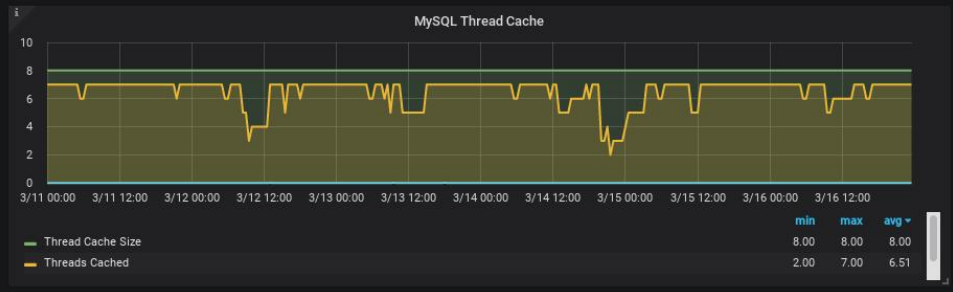


Table Locks





Status

HEALTHY

Monitors In Quorum

3

Pools

1

Cluster Capacity

45.8 TiB

Used Capacity

19.12 TiB

Available Capacity

OSDs IN

27

OSDs OUT

0

OSDs UP

27

OSDs DOWN

0

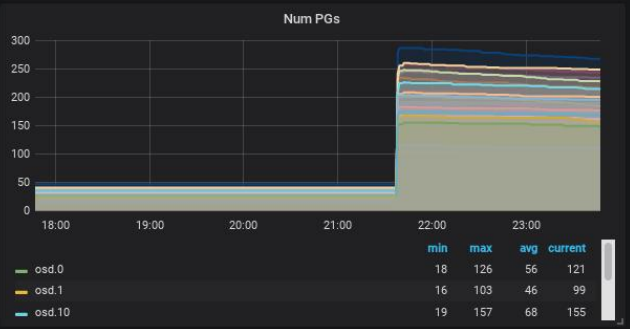
Agerage OSD Apply Latency

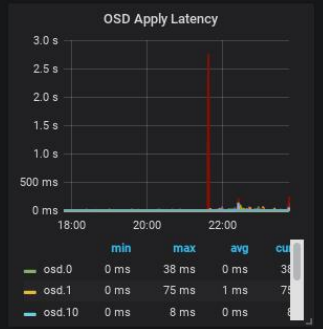
Agerage OSD Commit Latency

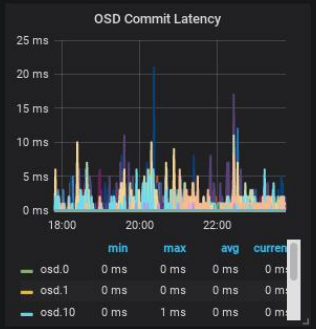
188 µs

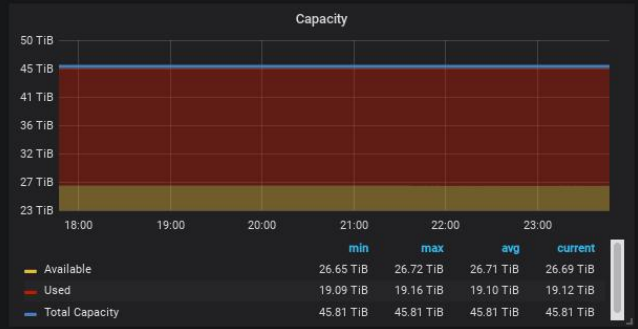
Agerage PGs per OSD

180

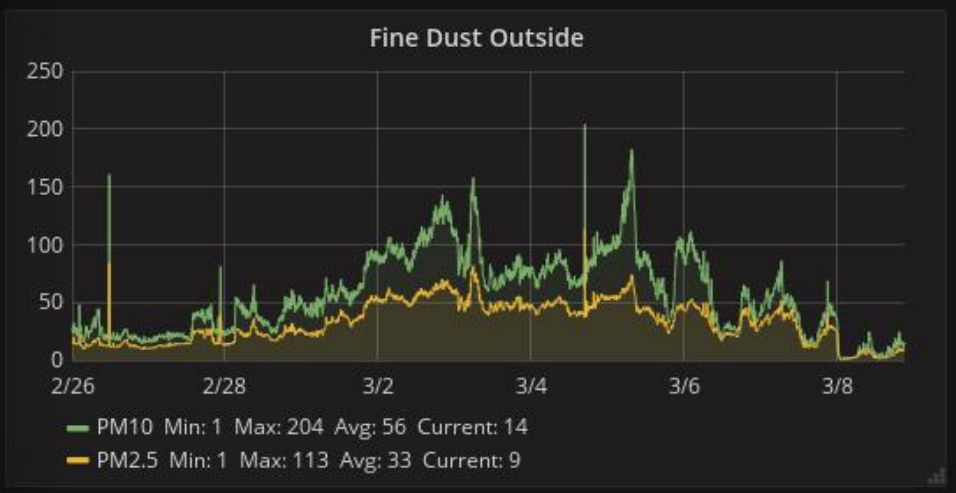


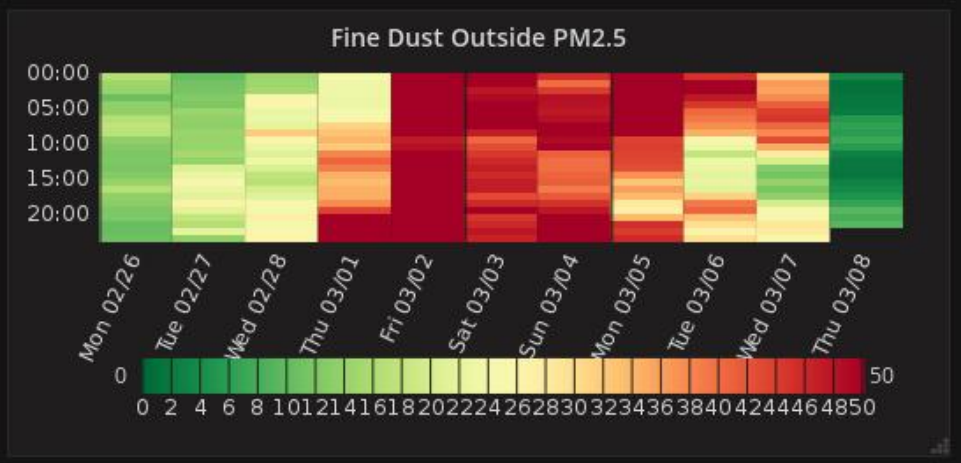


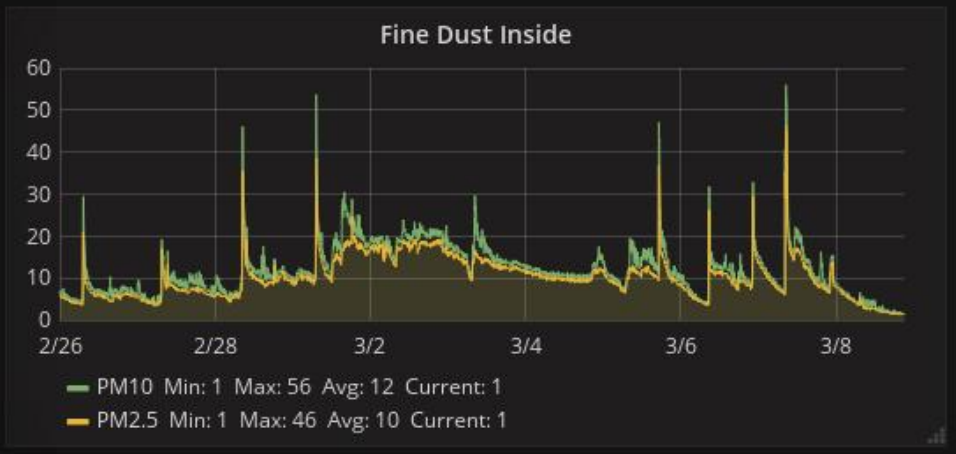


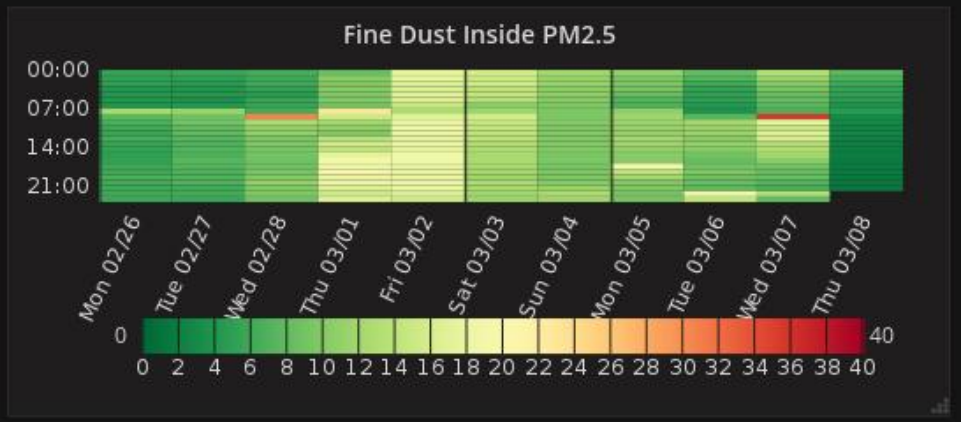


Kleiner Exkurs des Admins@HOME



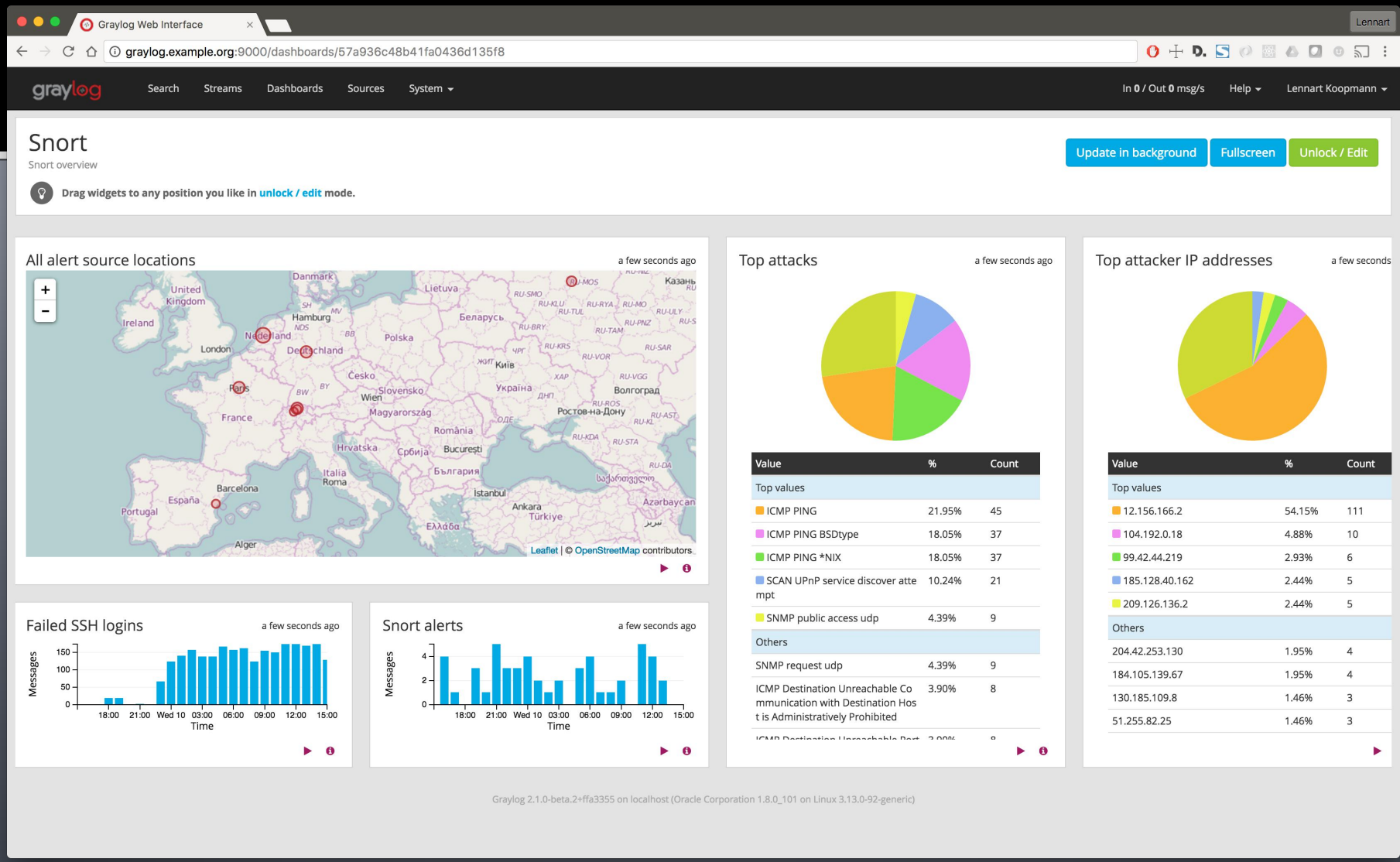






Zentrales Logging

- Vereinfacht Troubleshooting (“auf welchem System muß ich eigentlich suchen?”)
- Ermöglicht Alerting/Monitoring auf bestimmte Events
- Kann volle Festplatten wegen großen Logfiles (die eh niemand anschaut) vermeiden
- Vermeidet SSH-Logins auf Produktivsystemen (Antipattern!)
- Beispiele: ELK, Graylog,...



Quelle: https://github.com/Graylog2/graylog-guide-snort/blob/master/README.md

Software-Auswahl

- Wichtige Tools auf allen Systemen mitliefern → einheitlich + Erwartungshaltung erfüllen
- Software-Auswahl kann je nach Umgebung/Admin(s)/Anforderungen natürlich unterschiedlich aussehen
- dstat/iostat/htop/tmux/$SHELL/…
- Know your Tools! (Shell & Editor!)

Versionskontrolle

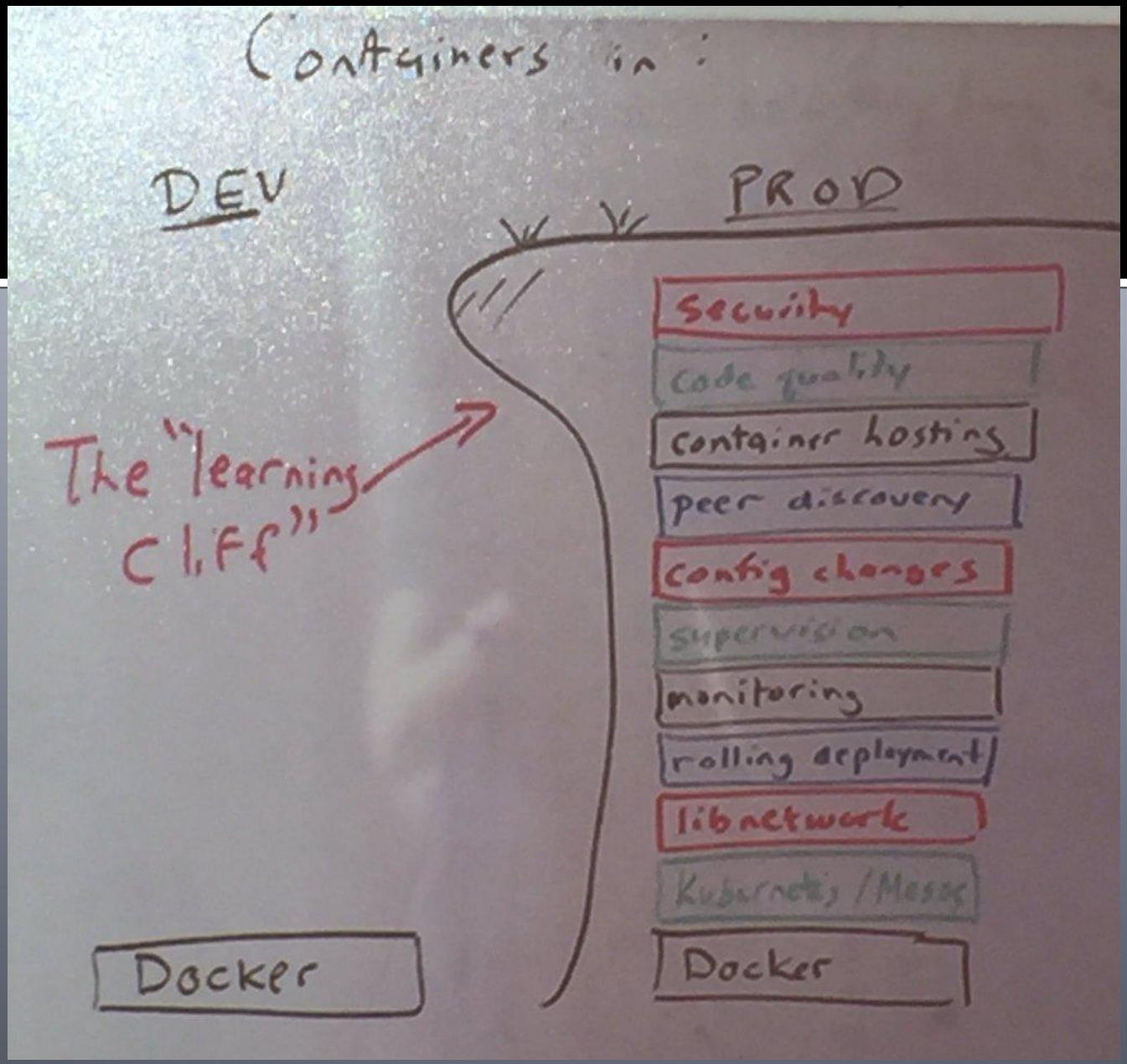
- **Alles** unter Versionskontrolle stellen
  - Code
  - Konfiguration
  - Dokumentation
- Git

Systemd

- CPU/Memory/…. Limiting + Accounting
- Service overwrites
- `systemd-analyze [blame]`
- `journalctl -u $SERVICE`
- `systemctl daemon-reload`
- “Deal with it”

Die obligatorische Docker-Folie

Quelle: https://twitter.com/sadserver/status/718455853540487168



Quelle: https://twitter.com/mfdii/status/697532387240996864

Konfigurationsmanagement

- Ansible
- Chef
- Puppet (+ Mcollective)
- Salt

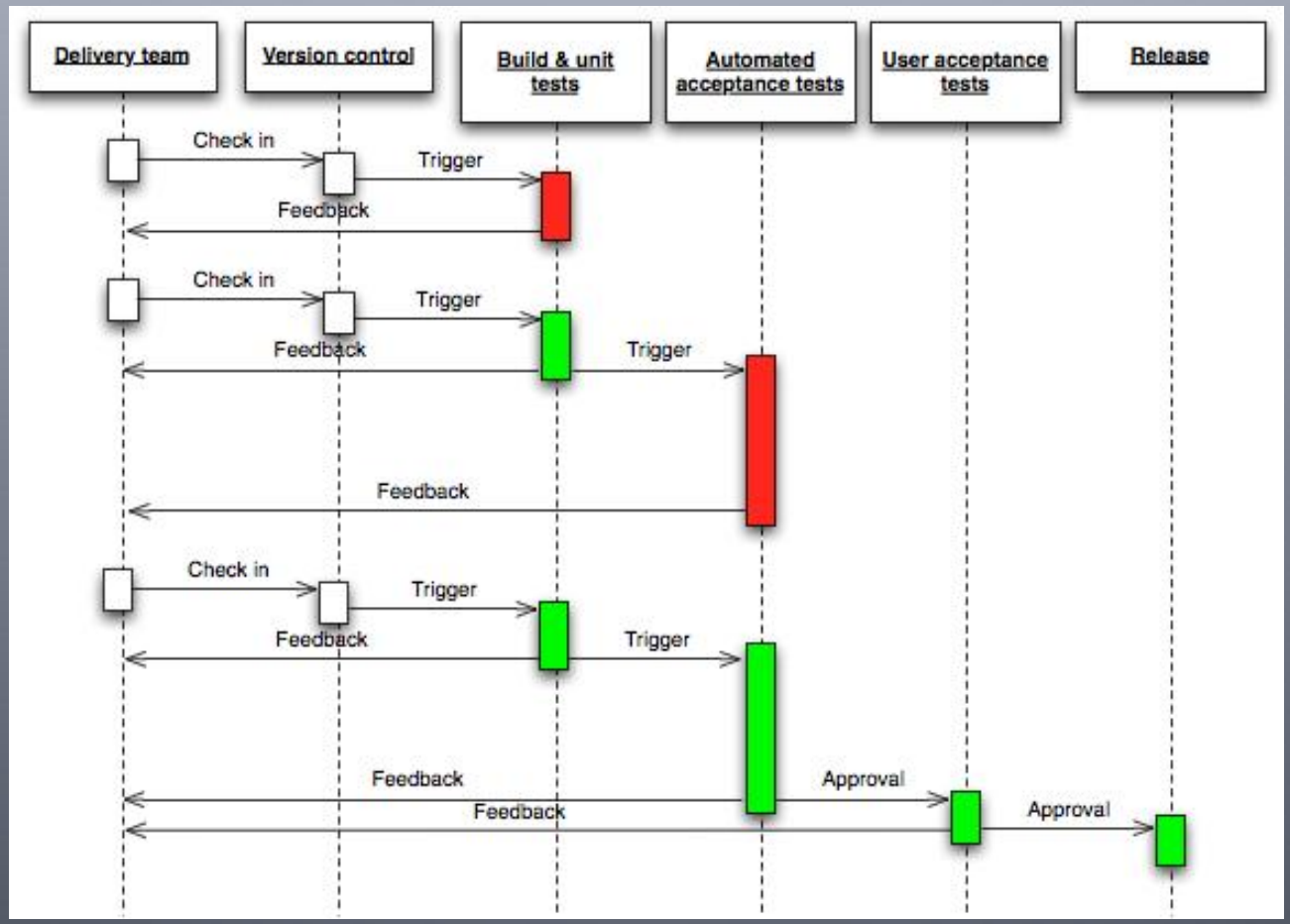
- Fabric (Python)
- Capistrano (Ruby)
- Rundeck
- …

→ Deployment, Provisionierung + Orchestrierung

Puppet related

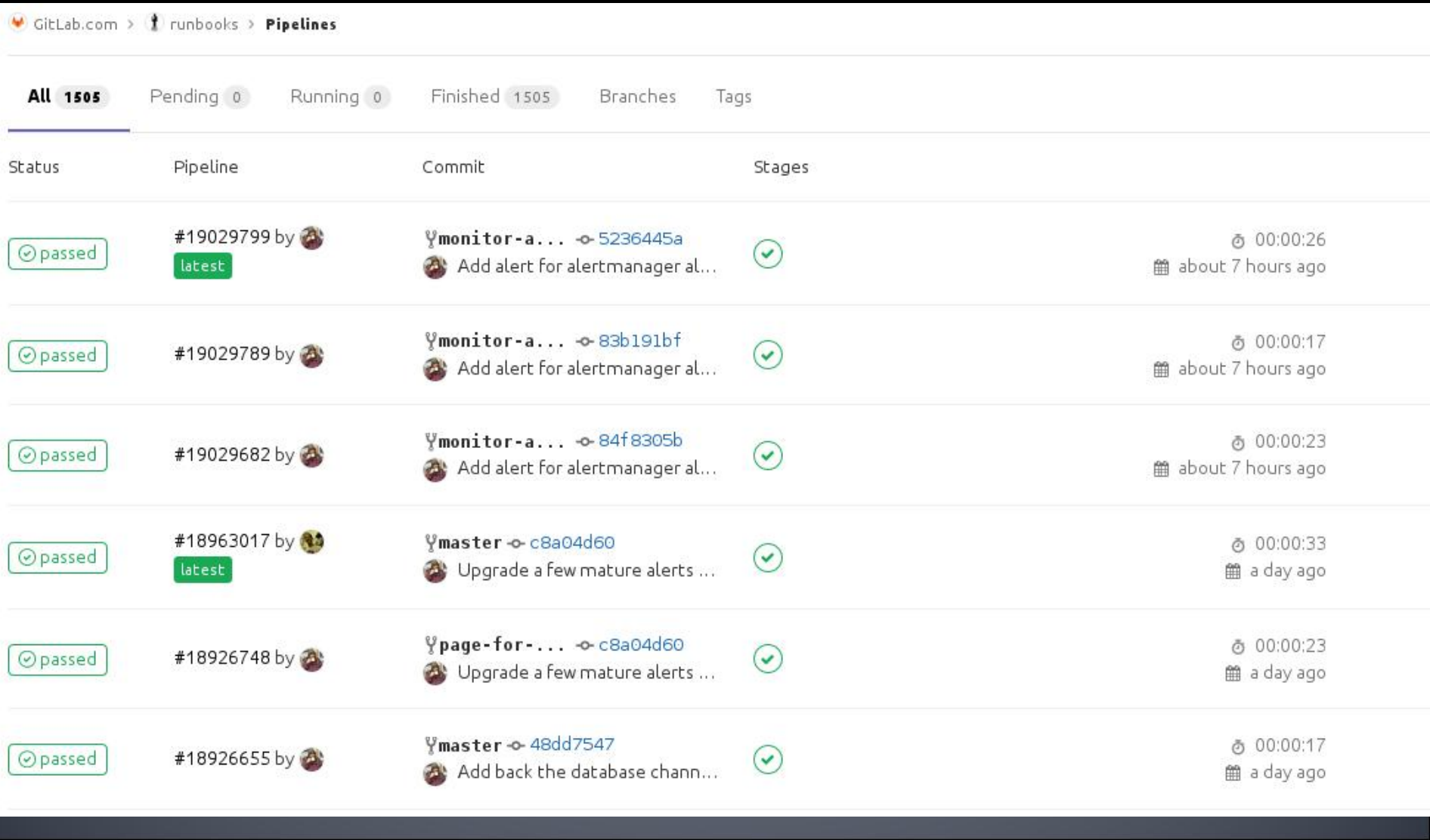
- Puppet environments + r10k
- Hiera
- Puppetdb
- Mcollective / Choria
- Octocatalog
- Puppet Dashboard → puppetboard
- Community: Puppet Forge

Deployment Pipeline

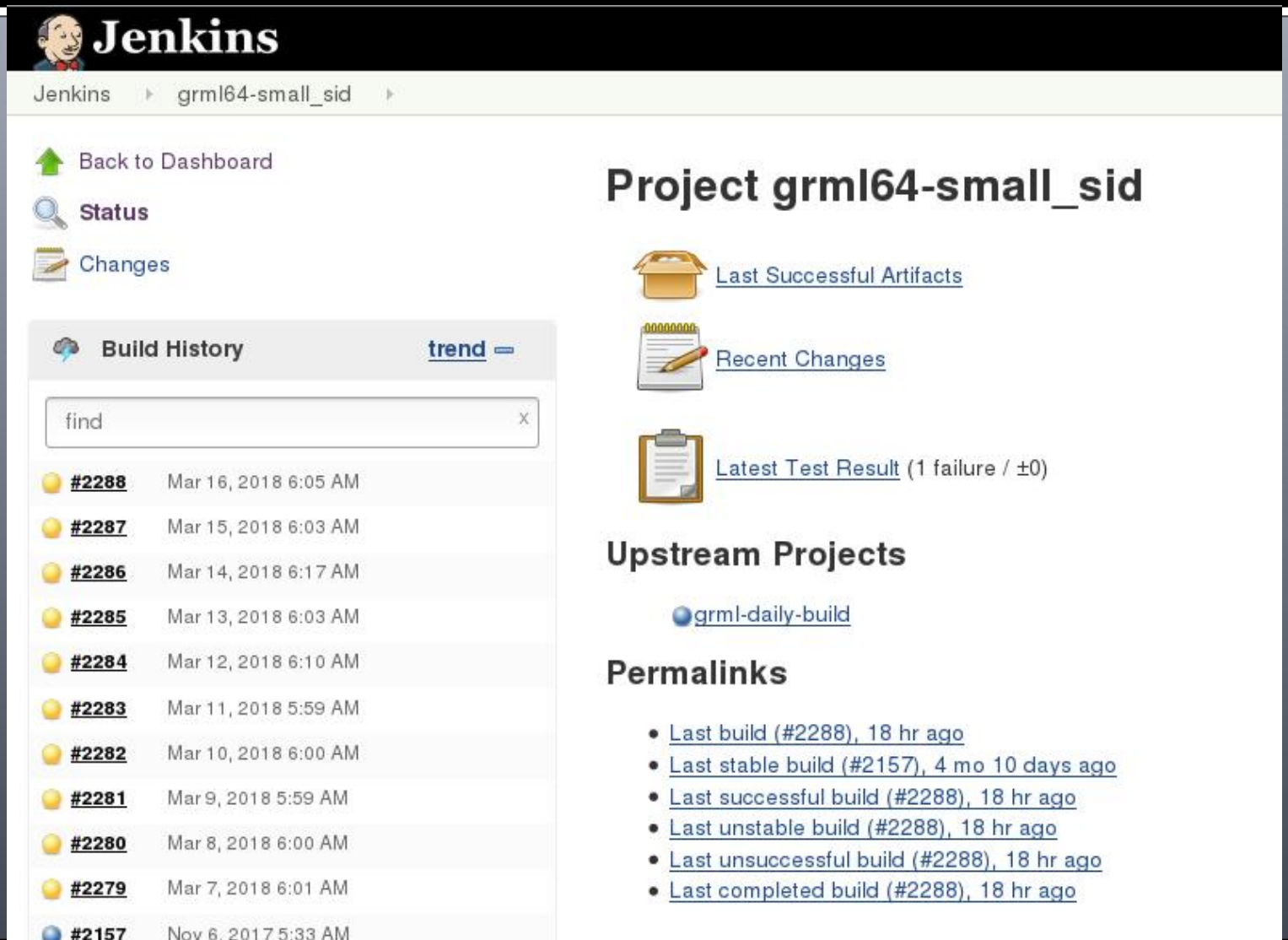


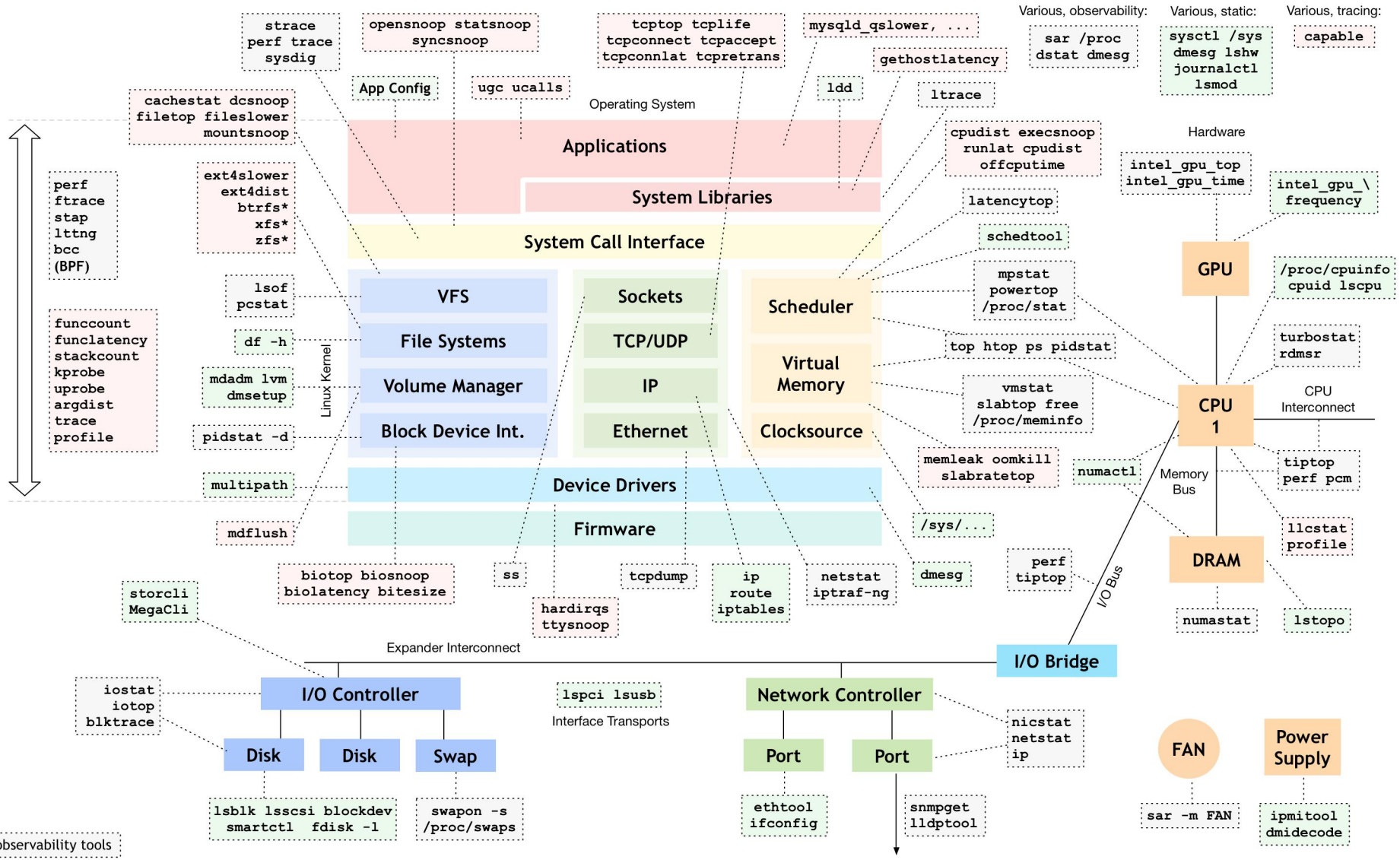
Quelle: http://continuousdelivery.com/2010/02/continuous-delivery/

Continuous Delivery - Gitlab



Continuous Delivery - Jenkins





Quelle: http://www.brendangregg.com/linuxperf.html

Diverses

- Dokumentation
  - Maintenance
  - Inventory
  - Journal
  - …
- Dashboards
- Ticket-System/Issue-Tracker

Gitlab
Gerrit
Phabricator
Review Board
Github
...

Quelle: https://memegenerator.net/instance/53083760

Warum Code-Reviews?

Wissen teilen

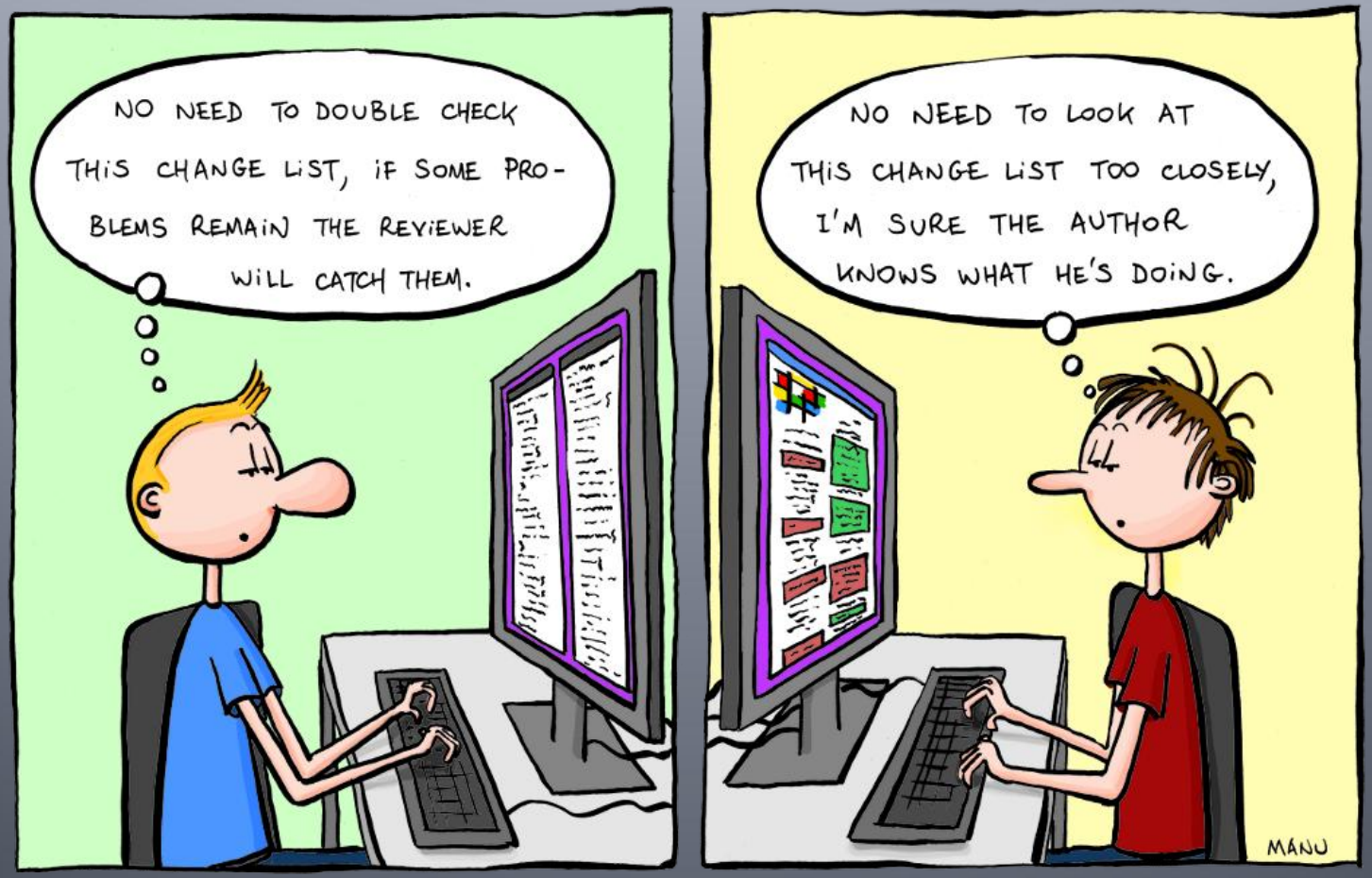
Wartbarkeit verbessern

Besserer **Code**

Broadcast **progress**

Communal **ownership**

Hust



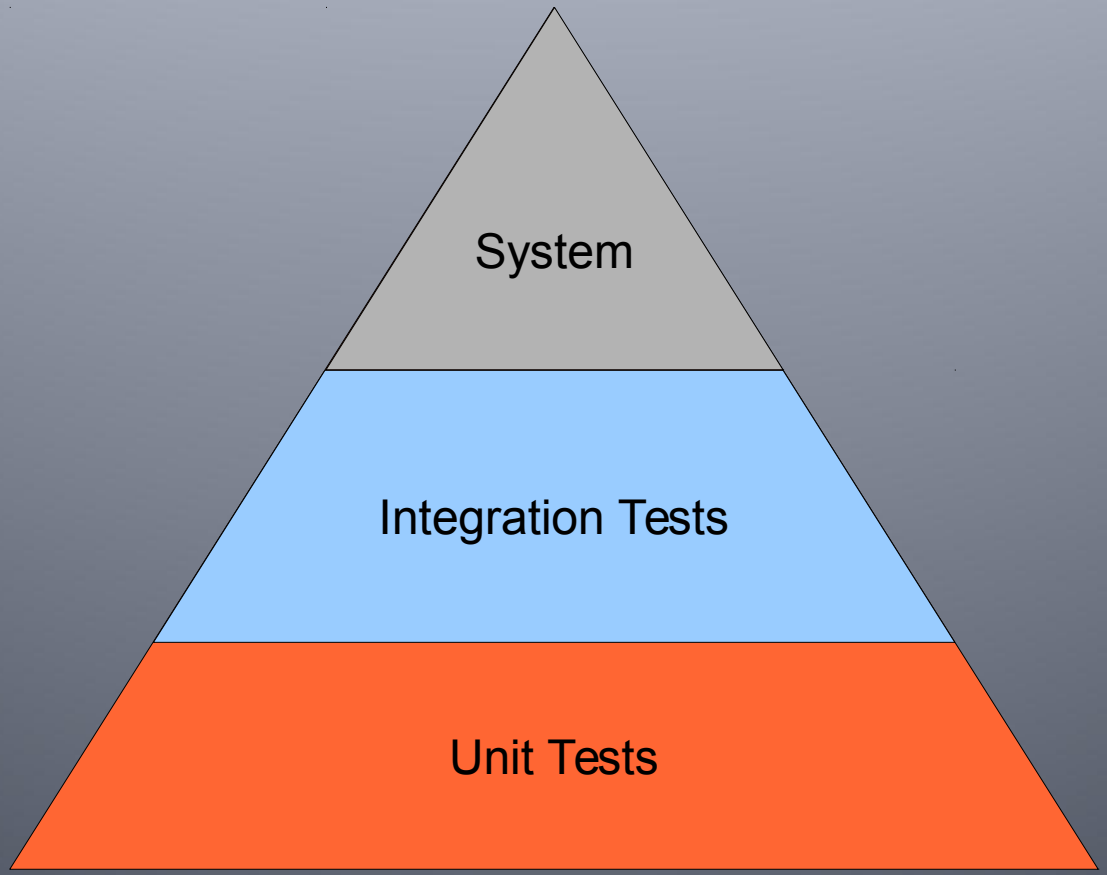
Quelle: http://www.bonkersworld.net/code-reviews/

Test-Umgebungen

"*Everybody has a testing environment. Some people are lucky enough to have a totally separate environment to run production in.*"

– Michael Stahnke [@stahnma]

Test-Pyramide



Goss

- https://github.com/aelsabbahy/goss/
- Serverspec-ähnliches Tool fürs Validieren
- Ausgabe in RSpecish, Nagios, JSON, JUnit + TAP
- Geschrieben in Go (statisches Binary)
- Linux-only (aktuell)

Puppet Testing

- puppet-lint (linting/code style)
- puppet parser validate (syntax check)
- rspec-puppet (unit tests)
- beaker (acceptance tests)
- octocatalog-diff / catalog preview (diff)

Stolperfallen & Schmerzen

Gute™ Wartung

- Nur regelmäßig angegriffene Systeme sind gewartet, der Rest ist Zufall
- Systeme automatisch beim Deployen/Provisionieren ins Monitoring mit aufnehmen + konfigurieren, beim Runterfahren auch wieder entfernen (APIs!)
- Automatische Upgrades (unattended-upgrades → Blacklist/Whitelist für kritische/schwer kontrollierbare Software)

Hyper Hyper!

- Early Adopter? Neue Paradigmen/Software → wer auf altem Zeug sitzt wird dann irgendwann von der (langsameren) Konkurrenz überholt
- Bequemlichkeit: Software die auf Public-IP lauscht (Memcache DDoS), Default-Passwörter (Hint: fail2ban, Firewalls + Monitoring-Checks)

Naming Things

„There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.“ – Leon Bambrick

- Wie wird man altes wieder los, beim Übergang zu neuem?
- crm.example.org vs crm01.example.org vs sugarcrm.example.org vs ...

Gefahren von Konfigurationsmanagement

- Ansammeln von Änderungen (Resultat: frische Installation geht gar nicht) → Mutable vs. Immutable (nicht modifizieren sondern neumachen) [Tipp: Terraform]
- Tools falsch angewandt, Beispiel Ansible Bootstrapping/Deployment vs. Konfigurationsmanagement [Blog-Artikel]
- Kontinuierliches Ausführen + Provisioning/Deployen ist wichtig!

Monitoring zu dynamisch

- Beispiel check-mk mit Service Checks: Tool fehlt (smartctl, megacli,…) → Check wird nicht aktiviert → potentielles Problem wird nie gemeldet
- Abhilfe? Tests!

Unprovisioning Things™

- Unverwendete Software auch wieder loswerden (braucht Security-Updates, schluckt Platz in Backups, Fehlalarme im Monitoring, Aufwand beim Refactoring von cfgmgmt-Code,…)
- Aus Monitoring, Firewall, DHCP, DNS & CO nehmen → Automatisierung (API)

Neue Dienste

- Installation + Setup sind **billig**
- Gute Integration (Monitoring, Backup,…) + Wartung (Security-Updates, Upgrades, Troubleshooting,…) sind **teuer**

Abhängigkeiten

- *„Sorry, wir können gerade nicht Deployen, weil Github down ist”*
- *“Upps, wir nutzen ja left-pad!”*
- *“Oh, DNS ist kaputt? Moment, ich schau im Wiki nach. Oh, das Wiki ist auch down.”*
- *“Woher das Binary kommt? Keine Ahnung. Der Admin der das hinterlassen hat arbeitet hier nicht mehr.”*
- *“Das Security-Update gibt es nur von $Vendor. Für den Download braucht es aber einen aktiven Account/der an die Lizenz gebunden ist/den nur $Kollege hat/…. [beliebig fortsetzbar]”*

Credentials + Passwörter

- Passwort-Handling ($Kollege verlässt Firma, Default-Passwörter die man nicht mehr loswird)
- AccessKeys im Cfgmgmt-Repos und öffentlich auf Github gepusht
- Private/Secret Keys unverschlüsselt auf USB-Stick/-Platte, $CLOUD-Storage,...

Regionale Kost

- einheitliche Zeitzonen (GMT)
- Systemzeit (NTP)
- Englische Locales („kein Weltraum links vom Gerät")

Packaging

- Vendored dependencies (Java, Javascript/Node, Rust, Go, Python, Ruby,…)
- Unser Stack wird immer größer + komplexer, ebenso wie unsere Abhängigkeiten
- s.a. https://apebox.org/wordpress/linux/1229 + anschließende Diskussion auf debian-devel-Mailingliste [URL]

Backups

- Niemand will Backup, alle wollen Restore
- Ob das Backup funktioniert, weiß man erst beim Restore (Testen + Dokumentieren!)
- Tipp: restic.net

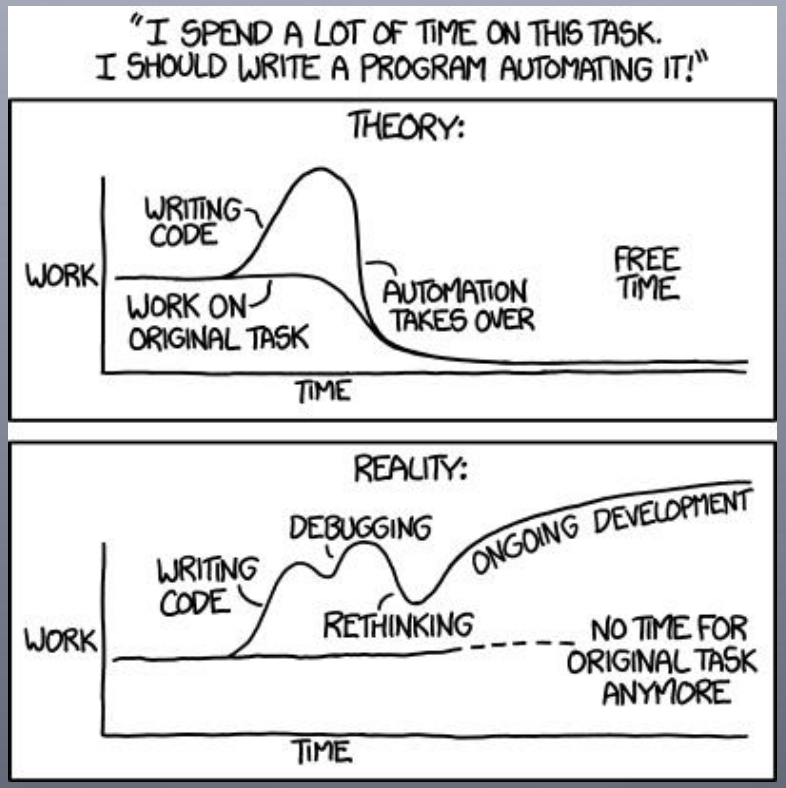
Enterprise™

- Restriktive Umgebungen (Firewall → Hotspot am Handy, Versand über private Mailaccounts, Dropbox,…)
- Bälle zwischen Abteilungen hin- und herwerfen (Passierschein A38!)

Schneeflocken & Tiere

- Pets (gesundpflegen) vs. Cattle (erschießen, neues holen)
- Neuinstallation vs. Upgrade (HW-Upgrade, Philosophie/Kultur)
- Snowflake-Effekt (sehen alle gleich aus, sind aber trotzdem anders)

Wann automatisieren?



Quelle: https://xkcd.com/1319/

Kultur-Probleme

- Yak Shaving (simpler Fix ist unmöglich, weil 42 Dinge einen beim Weg dorthin aufhalten)
- Broken Window (im Monitoring ist eh immer alles rot!)
- Not My Job (nicht links + rechts schauen)

Kultur-Probleme

- Verspieltheit (neue Tools ausprobieren, alte Baustellen bleiben offen)
- Technische Schuld bzw. nur Feuerwehreinsätze
- „Das war schon immer so!” (alternativ: „Das haben wir immer schon so gemacht!”)

Fortbildung

- „Ich habe keine Zeit mein Messer zu schleifen!”
- Schlechter Mitarbeiter kann mehr Schaden verursachen, als die Richtigen in der gleichen Zeit fixen können
- Post-Mortem-Analysen
- Regelmäßige Reviews (Dokumentation, Konfigurationsmanagement,…) inkl. Hinterfragen + Selbstreflektion

Ausblick

DSGVO, AKA General Data Protection Regulation (GDPR)

- Inkrafttreten: 24.05.2016
- Anzuwenden: 24.05.2018
- Datenminimierung
- Recht auf Vergessenwerden
- Recht auf Datenübertragbarkeit
- … und vieles mehr

- Achtung: IP-Adresse gilt als personenbezogenes Datum

The Evolution of Distributed Systems Management

- Type 0: Manually Deployed and Configured (Cisco IOS, Classic Redis, Zookeeper)
- Type 1: Host-Centric Configuration Management (Ansible, Chef, Puppet, Salt)
- Type 2A: Infrastructure-scoped Orchestration Tooling (Terraform, CloudFormation, Bosh,…)
- Type 2B: Application-specific Orchestration Tooling (Vitess/MySQL, Orchestrator/MySQL, Redis-Sentinel,…)
- Type 3: Compute-Platform-Native Application-Specific Frameworks (Mesos, Kubernetes)
- Type 3A: Mesos Frameworks (Titus, Peleton,…)
- Type 3B: Kubernetes (k8s) Operators (etcd operator, elasticsearch operator,…)
- Type 4: Hybrid (of Type 3 + Type 2A)?

Quelle: Kyle Anderson in http://www.xkyle.com/the-evolution-of-distributed-systems-management/

Take-aways

TL;DR – Admin in 2018

- Automatisierung
- Konfigurationsmanagement
- Versionskontrolle
- Deployment/Provisionierung/ Orchestrierung
- Monitoring
- Metriken/Graphing
- Logging

TL;DR – Admin in 2018

- Tests + Testing
- Continuous Integration/Deployment/ Delivery
- API Driven (Bsp: Kubernetes)
- Infrastructure as Code (inkl. Code Review)

TL;DR – Admin in 2018

- Entwickler + Admins zusammenbringen
- Kommunikation + Networking
- Dokumentation
- Selbstreflektion
- Fortbildung

Fragen?

@mikagrml
https://michael-prokop.at/blog/
michael.prokop (at) synpro.solutions