

Docker Swam *Advanced*

Docker Meetup Bochum - 09.2018

Dr. Halil-Cem Gürsoy
adesso AG

[@hgutwit](#)

Über mich...

- Principal Architect @ adesso AG
- Seit ~ 20 Jahren in der Java-Welt
 - davor im wissenschaftlichen Umfeld mit Pascal
- Verteilte Enterprise System
 - Build, Deployment & Persistenz
- Kein Linux-/System-Admin !!!
 - ... aber oft in "DevOps Teams" 🤔

Eine lange Docker-Reise

- Seit Docker 0.8 (2014) produktiv mit Docker unterwegs
- Build & Testumgebungen für große agile Teams
 - Java/JEE, Testcluster, Datenbanken (NoSQL, RDBMS)
- Continuous Delivery mit Docker & Docker Swarm
 - Jenkins CI, Gitlab CI, Atlassian Bamboo ...



SwarmKit

- SwarmKit ist ein Toolkit um verteilte Systeme auszubauen
- Standalone als auch als Bibliothek verwendbar (*golang*)
- Docker Swarm inkludiert SwarmKit

SwarmKit

- Inhärent ein verteilter K/V-Store auf Basis von RAFT
 - Daher kein externer Service wie *etcd* oder *Consul* notwendig
 - Content wird verschlüsselt
- Kommunikation der Nodes über TLS
 - Automatischer Austausch der Zertifikate (Backup? Ups...)

SwarmKit und Container

- Viele verschiedene Komponenten
 - Manager (Orchestrator, Allocator, Scheduler, Dispatcher)
 - Worker (Engine, Agent)
- *scheduler -> dispatcher -> agent*
- Verschiedene Objekte (Service, Task, Network, Volume,...)



Swarm Mode

- Seit Docker 1.12 ist *SwarmKit* Bestandteil der Docker Engine
- Es stehen alle SwarmKit-Features damit zur Verfügung
- Ein Knoten muss als Manager fungieren, um Swarm-Befehle auszuführen
- ...aber muss nicht der Leader sein
(-> anders im alten *Swarm Standalone* !)

Swarm Init

- Initialisierung des Clusters mit `docker swarm init`
 - darf nur auf einem Knoten ausgeführt werden!
- Vorsicht bei mehreren Interfaces!
 - `--listen-addr & --advertise-addr`
 - `--data-path-addr` für die Kommunikation der Container
- Interface-Name = erste IP-Adresse wird verwendet (!!)

Kommunikation

- Cluster-Kommunikation ist verschlüsselt (alle 12h Key-Rotation)
- Authentifizierung untereinander über TLS
 - Rotation alle 90 Tage - Vorsicht bei Backup's/Snapshots!
- Overlay-Netzwerke per Default **nicht** verschlüsselt
 - aber kann man ja einschalten

Manager

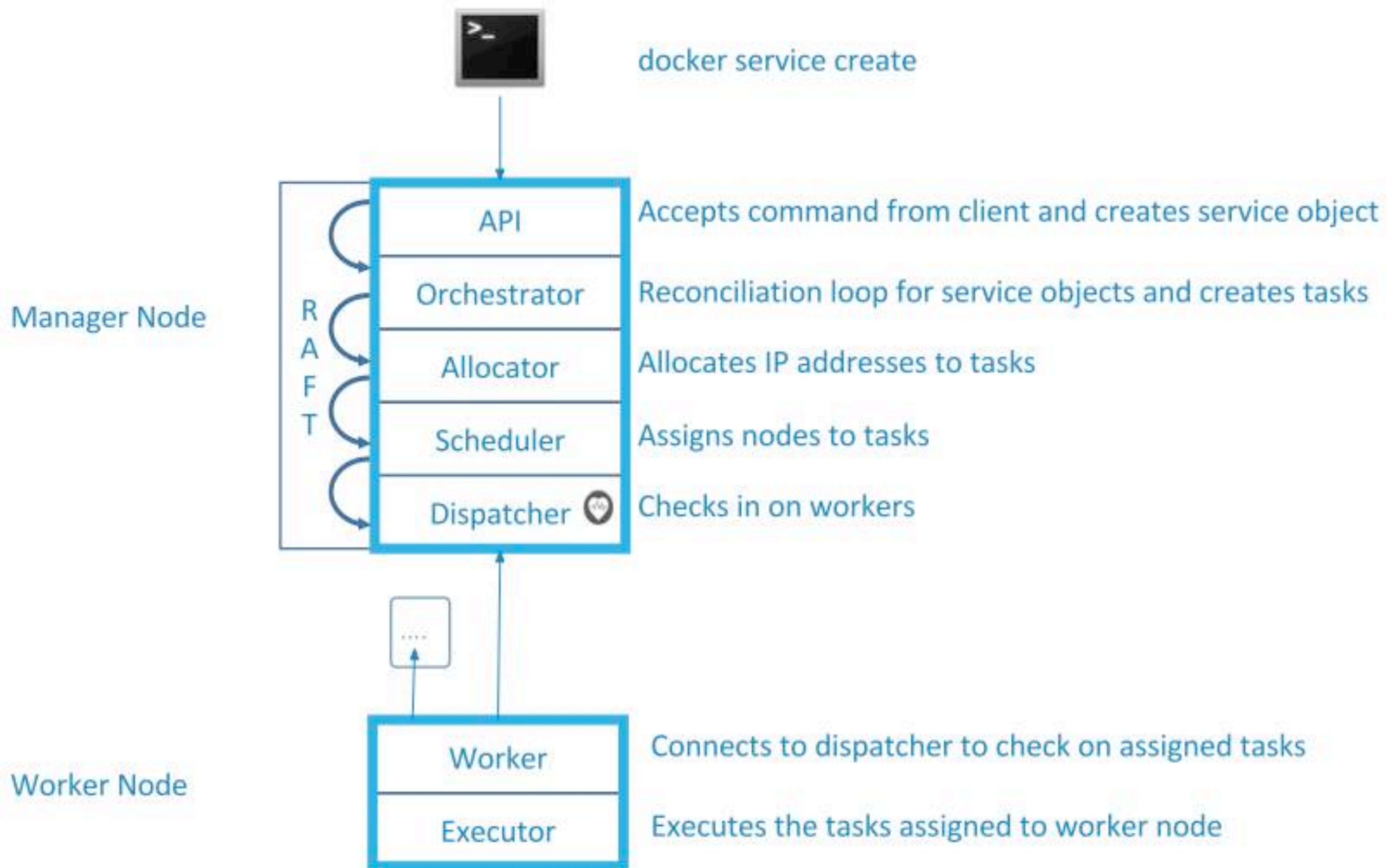
- $2N+1$ Topologie um Ausfallsicherheit zu erhalten
- Produktion: 3 oder max. 5 Manager
 - Vorsicht bei > 5 ; Konsensus schwieriger zu erreichen
- Dedizierte Server nur für Manager ohne weitere Workloads
 - Stack Deployments nur auf Worker
 - Manager können auch mal CPU/Speicher-Hungrig sein!

Manager - Scaling

- Je mehr Nodes und Stacks -> Mehr Hauptspeicher
- Je öfter Stack- & Service Deployments / Updates -> mehr CPU
- 2GB / 4 CPU auch für etwas größere Setups OK

Services & Stacks

- Services über `docker service create` deployen
- Oder als Service in einem Stack: `docker stack deploy`
- Definition des Stacks in einem Compose-File
 - Dokumentation inzwischen hinreichend wirr
- Mit aktueller Version Auswahl zwischen Swarm und K8s



Stack File

- Gut: Extension fields
- YAML-Merge

x-restart:

&default_restart_policy

condition: any

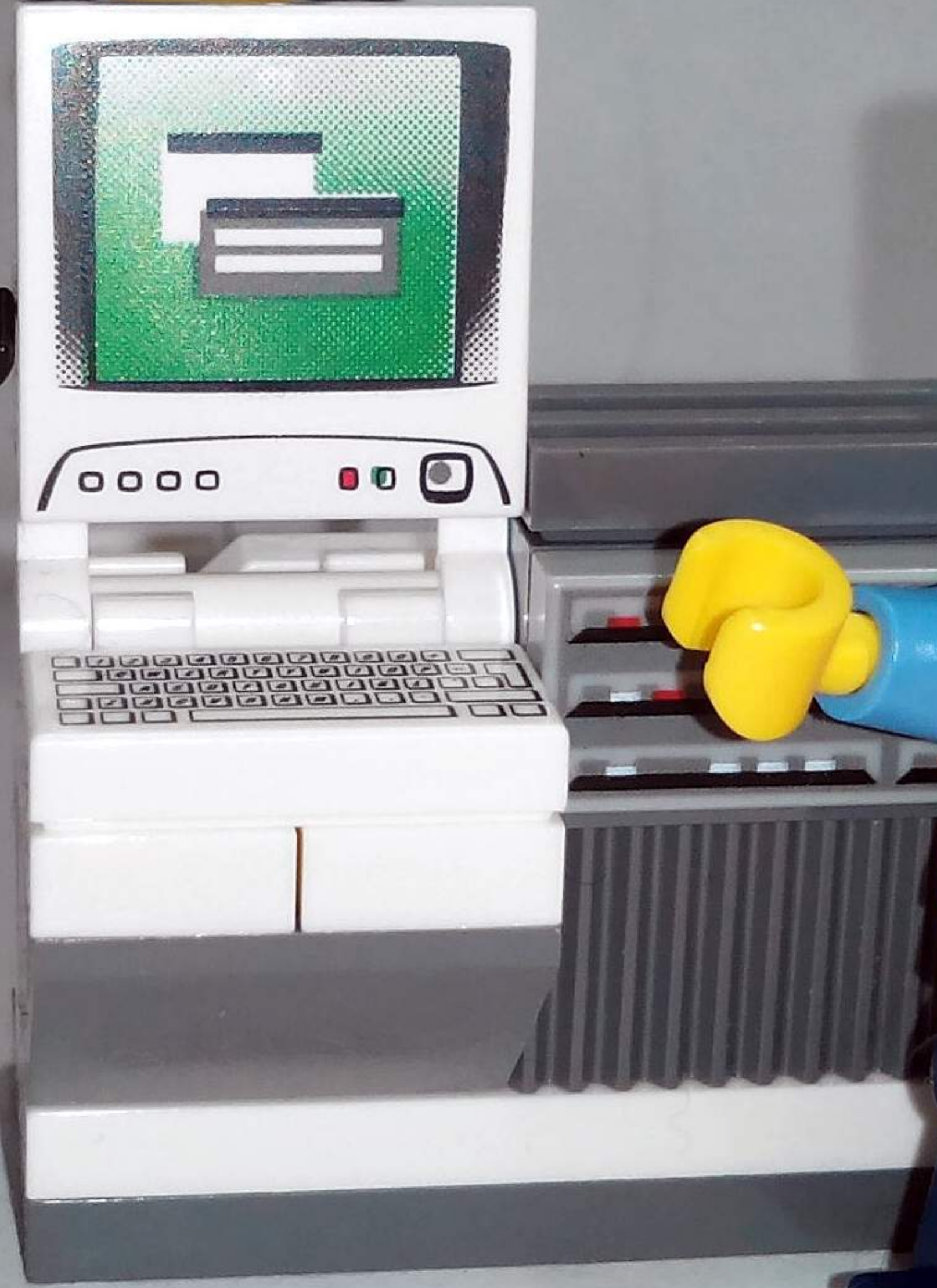
delay: 5s

max_attempts: 3

window: 190s

Stack File

```
services:
  user-db:
    image: user-db:12654-master-91f7a81
    << : *default_pqlstopping
    networks:
      - ${STACK_NAME}
    logging:
      driver: "gelf"
      options:
        << : *default_gelfoptions
        tag: ${STACK_NAME}_user-db
    volumes:
      - type: bind
        source: ${SHARED_STORAGE_DIR}/${STACK_NAME}/user-db
        target: /var/lib/postgresql/data
    deploy:
      restart_policy: *default_restart_policy
    resources:
      limits:
        << : *default_limits
```



Daily Work

- Verschiedene Enterprise-Kunden mit Docker Swarm
- Initial "nur" für Entwicklung, dann oft auch in Prod
- Oft ~ 10 - 20 Nodes mit dedizierten Managern
- Nodes Midsize-Server (64-128GB, 4-8 CPU)
- Meistens VM's unter ESX 🙄
 - leider oft kein DO, AWS, GCE oder Bare Metal



Stack Pitfalls

- "Eigentlich" sollen alle Optionen aus `docker service create` zur Verfügung stehen
- Inkonsistenzen zw. Dokumentation und Issues in den verschiedenen GitHub-Repositories
- Beispiel: `stop_signal` und `stop_grace_period` mit viel Verzögerung in der Doku gelandet



Volumes

- Was passiert, wenn ein Stateful Service (z.B. DB) auf einem anderen rescheduled wird?
- Oder mehrere Instanzen dieses Services in verschiedenen Stacks deployt sind?
- Wie findet ein Service "sein" Volume wieder? -> "Identität"

Volumes

- Docker Volume Plugins kommen und gehen, z.B. *Flocker*
- Herausforderung ist neben Performance und POSIX-Kompatibilität die "Identität" eines Volumes
- Einige Plugins lösen dies z.B. über transparent angelegte Unterverzeichnisse für Stacks
- Alternativ:
 - "self management" im Rahmen des Deployments
 - Bind mount eines *shared filesystem*
 - `source: ${SHARED_STORAGE_DIR}/${STACK_NAME}/user-db`








"Eventual Consistency"

- Overlay Netzwerke können im Stack-File definiert werden
- Werden mit `stack deploy` angelegt
- Anlegen des Netzwerkes ist kernelseitig relativ Aufwendig (VXLAN, IPVS und mehr)
- Anlegen des Containers auf dem Node "überholt" Netzwerk -> Fehler auf dem Node *"no such network"*
- Netzwerk vor `stack deploy` anlegen, ggfs. etwas warten



Infrastruktur

- Auf einem fragilen Fundament kann kein stabiles Haus stehen
- Bsp.: ESX wird regelmäßig vom Netzwerk getrennt

Beschreibung	Typ	Datum und Uhrzeit...	Ziel	Ereignistyp-ID
 Der Host ist verbunden.	 Informationen	06.09.2018, 03:08:22	 DE9899SZH-DB	vim.event.VmConnectedEvent
 Der Host ist nicht verbunden.	 Informationen	06.09.2018, 03:08:00	 DE9899SZH-DB	vim.event.VmDisconnectedEvent

100 Items

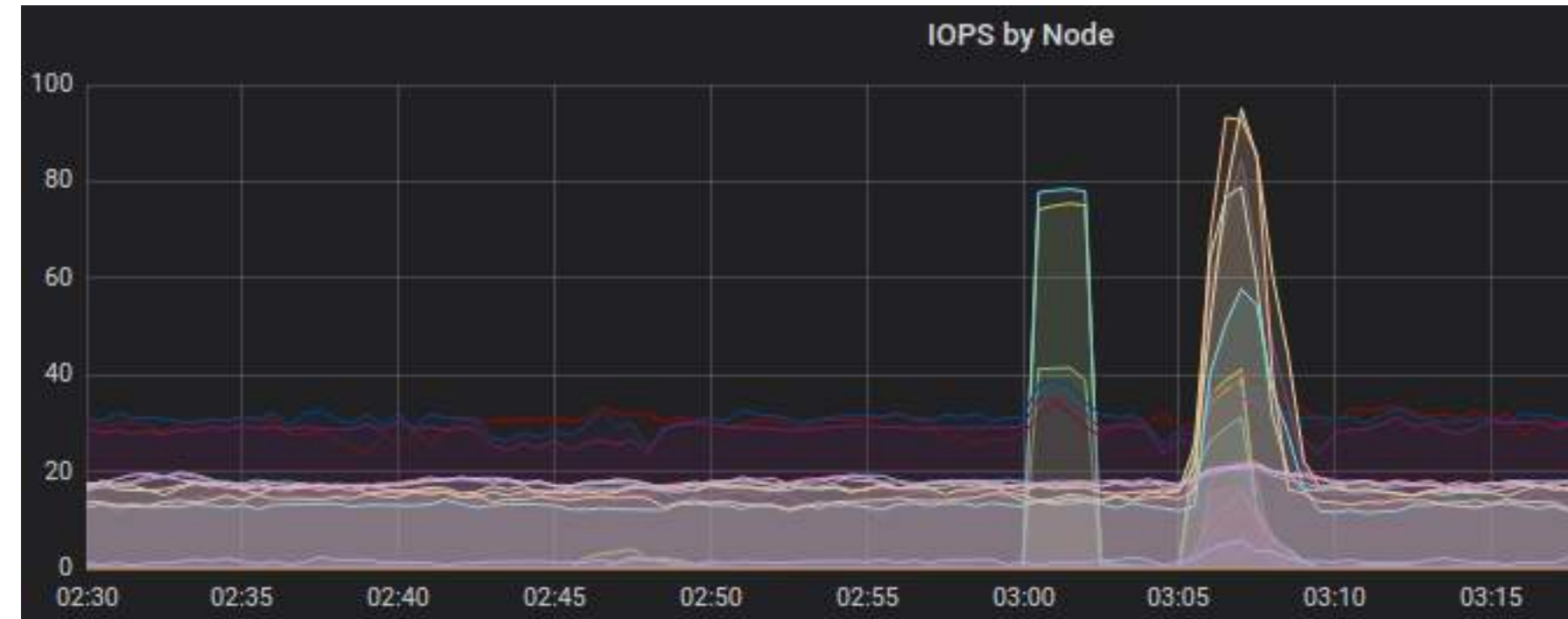
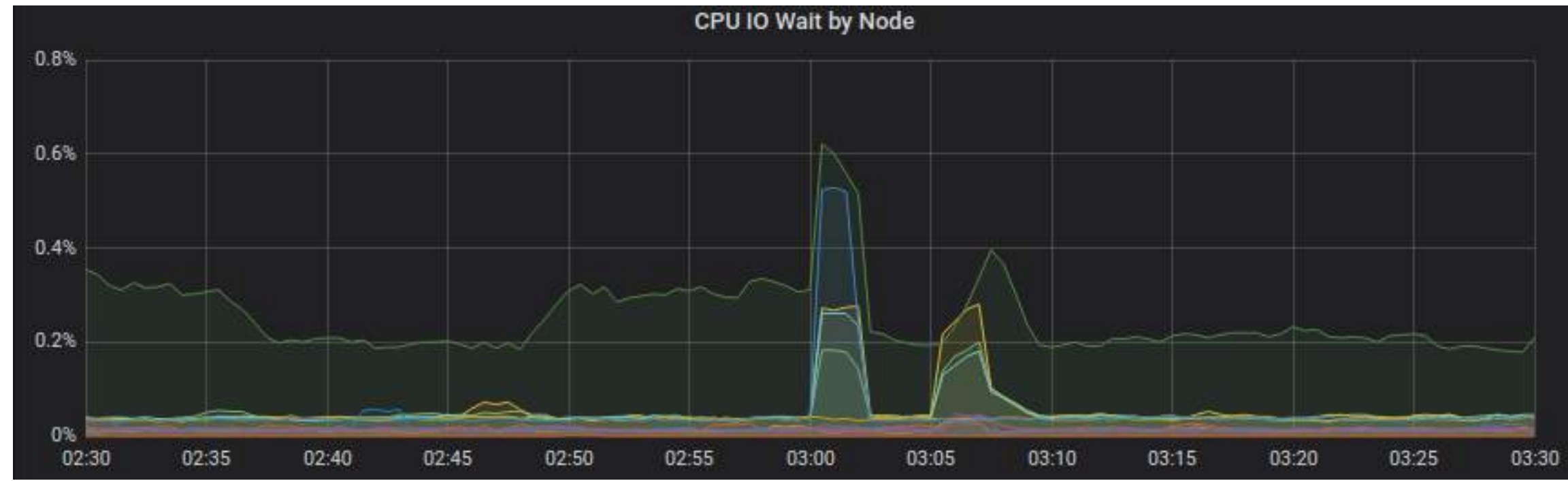
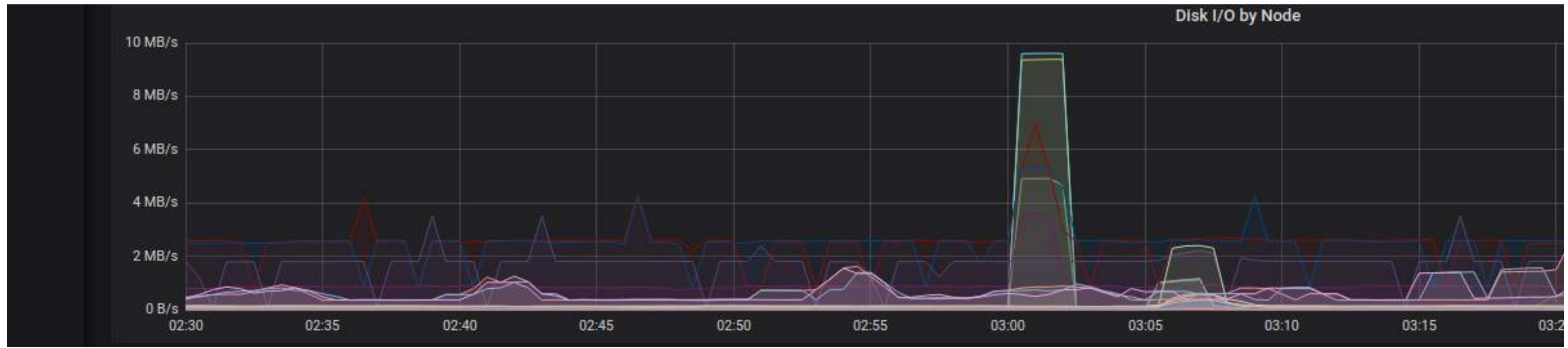
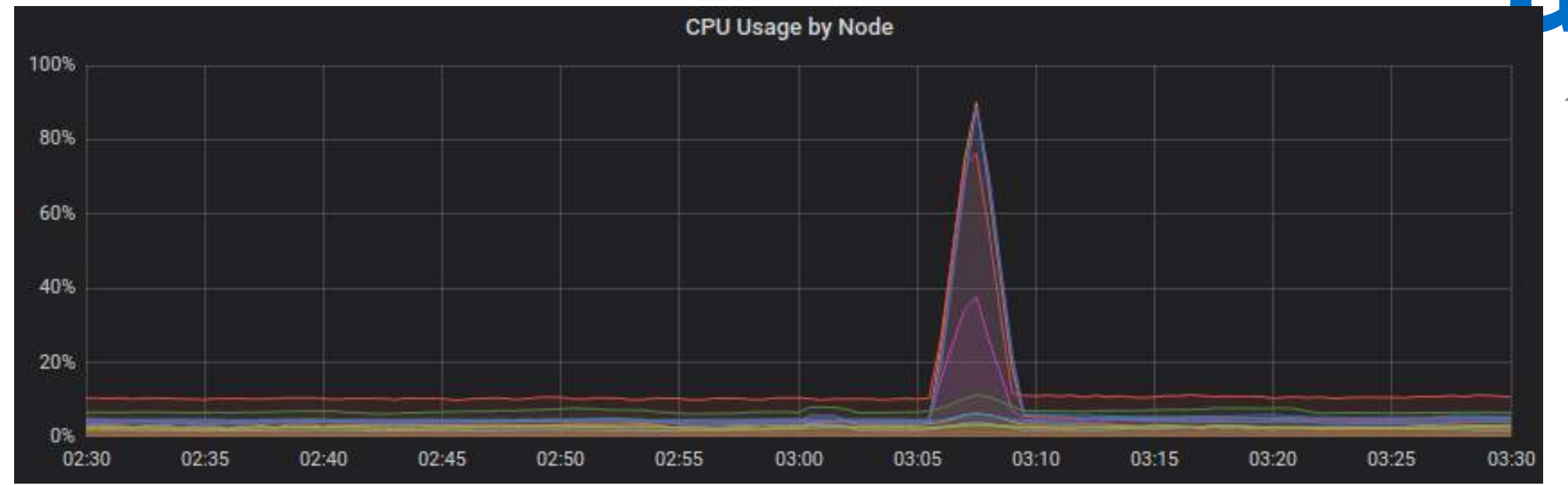
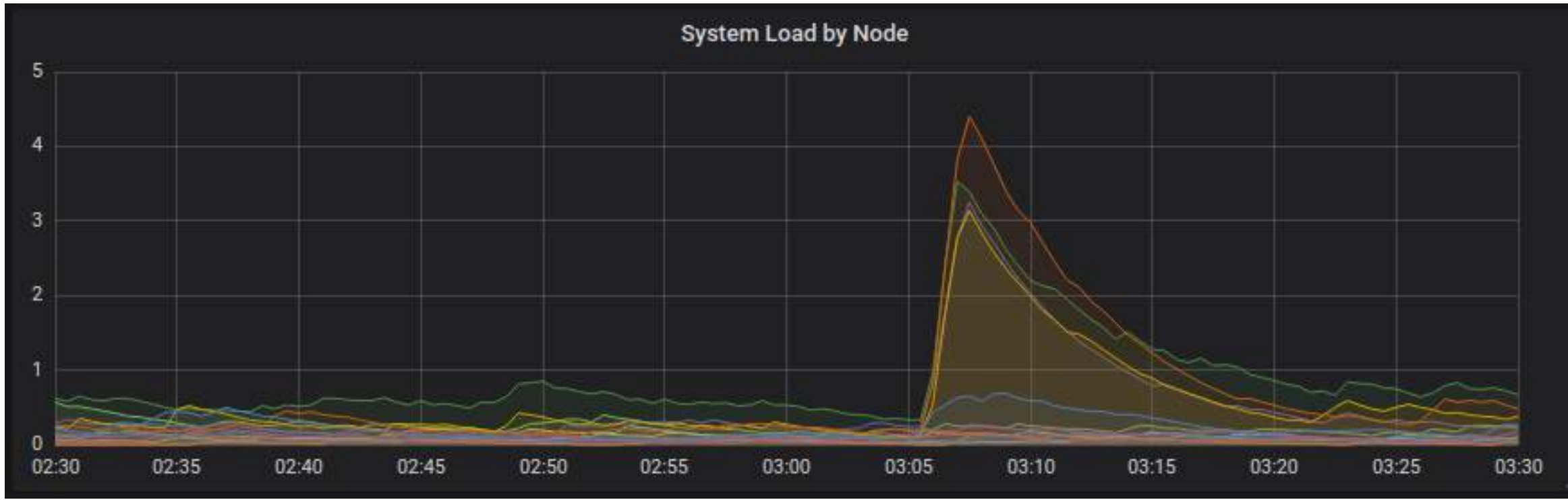
Datum und Uhrzeit: 06.09.2018, 03:08:00

Typ: Informationen

Ziel:  DE9899SZH-DB

Beschreibung:

 06.09.2018, 03:08:00 DE9899SZH-DB auf dem Host  ist nicht verbunden.



Infrastruktur

- Netzwerke regelmäßig Quelle großer Freuden
 - auch beliebt: UDP Package Lost / TCP retransmission
- Auf ESX VM-Migration besser ausschalten (im Docker Kontext)
- VM's auf ESX-Server pinnen
- Mit Node-Label und placement-pref für eine sinnvolle Verteilung über zugrunde liegende Infrastruktur sorgen



"Evil Container"

- Ganz aktuell: Offizielles PostgreSQL-Image
- SIGTERM triggert *smart shutdown*:
"It shuts down only after all of the sessions terminate."
- Nach 10 sec. (default) `kill -SIGKILL` auf den Hauptprozess
- Ergebnis: `postgres: autovacuum` Prozesse im Status **Ds**
- Replicas: **2/1**
- Hilfe: Entfernen des `docker-containerd-shim`-Prozesses



Stacks & Configs & Secrets

- Docker Secrets und Docker Configs können in Stack-Files definiert werden
- Problem: Re-Deployments eines Stacks beim Update ignoriert Änderungen an Secrets und Configs

Best Practice Stack File

- Nur Service-Definitionen in ein Stack-File packen
 - ... und Volume-Mounts
- Netzwerke "external" und mit *docker network* konfigurieren
- Analog *docker config* und *docker secret* nutzen
- Deployments und Updates in einem Script "wrappen" mit dem "drumherum":
 - Service stoppen, Config updaten, Service starten....

Stacks

- Update eines Stacks (z.B. geänderte Images):
`docker stack deploy`
 - Mit `--force` kann ein Rebalancing erzwungen werden
- Wenn nur ausgewählte Services geupdated werden sollen:
`docker service update`
- Die Update-Policies ziehen in beiden Fällen!
- Mit `docker service scale myservice=0/1` Stoppen und Restarten eines Services für einen Zeitraum erreichen



"Alles kaputt"

- Typische Anzeichen wenn im Cluster etwas "krumm" ist
 - Stack deployments erfolgen nicht vollständig
 - Globale Services laufen nicht auf allen Knoten
 - Container bleiben nach Service/Stack-Undeployment übrig
 - Manager haben Quorum verloren

Analyse

- `journalctl -u docker` für die aktuellen Logs
- `swarmctl` gibt bessere Übersicht über den aktuellen Status
- Im Notfall Analyse der Swarm-DB mit Hilfe von *swarm-rafttool*
 - Sehr low level, aber Einblick in die verschiedenen Objekte

Recovery procedure

- In `/var/lib/docker/swarm` werden alle Swarm-Infos abgelegt
- Manager stoppen, `/var/lib/docker/swarm` restoren
- `docker swarm init --force-new-cluster`
- danach wieder alle Worker und Manager joinen

Heute unbeachtet

- Monitoring (Prometheus) und Logging (z.B. Graylog)
- Health Checks
- Service Rollbacks und Rolling Updates
- Auto Scaling
- LinuxKit, InfraKit
- TLS im Detail
- Autorisierung und Auth Plugins
- Tiefergehende Isolierung über Namespaces
- Cluster Lock
- ...und vieles mehr!





"Orchestrator-Porn-War"



