Monitoring Kubernetes with Prometheus

Tobias Schmidt - ContainerDays NYC November 4, 2016

github.com/grobie - @dagrobie

github.com/grobie/prometheus-on-kubernetes

Bootkube or Minikube

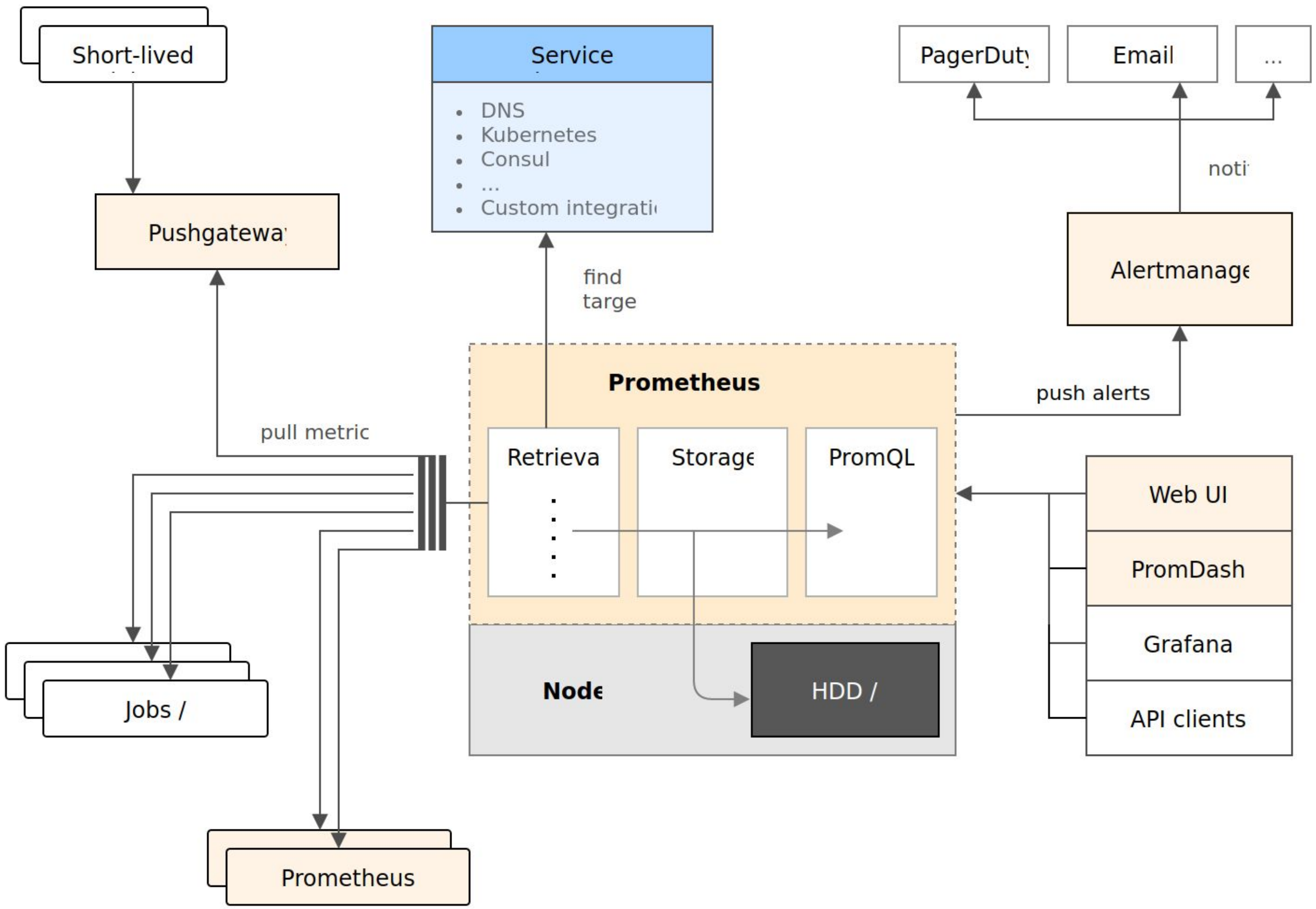
Prometheus

Monitoring system and time series database

Motivation

- Microservice architecture
  - Many more services than in traditional host-based monitoring
  - Short lifecycles
  - Heterogeneous workloads
- Insight
  - Detailed (instance/endpoint/version/... drilldown) and aggregated (across a service) of everything (hardware to service)
  - Trends (act before something becomes a problem)
- Alerting
  - Symptom vs. Cause
  - Grouping, flexible silencing

Overview



Short-lived
Pushgatewa
Service
DNS
Kubernetes
Consul
...
Custom integrati
find targe
PagerDuty
Email
...
noti
Alertmanage
push alerts
Prometheus
Retrieva
Storage
PromQL
pull metric
Node
HDD /
Web UI
PromDash
Grafana
API clients
Jobs /
Prometheus

Examples

```
# HELP etcd_store_writes_total Total number of writes seen ...
# TYPE etcd_store_writes_total counter
etcd_store_writes_total{action="compareAndDelete"} 2
etcd_store_writes_total{action="compareAndSwap"} 4016
etcd_store_writes_total{action="create"} 218
etcd_store_writes_total{action="set"} 5
```

```
count by(job)(up == 0) / count by(job)(up)
rate(etcd_store_writes_total{action="set"}[1m]))
sum without(action)(rate(etcd_store_writes_total[1m]))
```

Configuration

```
# prometheus.yaml prometheus.io/docs/operating/configuration/
global:
    # Settings applying to all jobs

scrape_configs:
    # Define different scrape jobs

Rules_files:
    # Load files specifying rules to pre-calculate expressions
    # as well as alerts.
```

Configuration

```
scrape_configs:
- job_name: etcd
  static_configs:
  - targets: ["172.17.4.51:2379"]

- job_name: kube-components
  kubernetes_sd_configs:
  - role: endpoints
  relabel_configs:
  - # Custom filtering and label mapping
```

Configuration

```
# continued
relabel_configs:
- action: keep
  source_labels: [__meta_kubernetes_service_name]
  regex: "kube-(.*)-prometheus-discovery"
- action: keep
  source_labels: [__meta_kubernetes_endpoint_port_name]
  regex: "prometheus"
- action: replace
  source_labels: [__meta_kubernetes_service_name]
  target_label: job
  regex: "(kube-.*)-prometheus-discovery"
```

Kubernetes

Container orchestration system

Domain objects

- Pod
  - Group of one or more containers, share context and namespaces
  - Co-located and co-scheduled (allows for side-cars)
- Service
  - Logical set of Pods, stable access points
- Deployment
  - Declaration of the desired state (what to run, how to get there)
- Daemon / Pet / Replica sets
  - Definition of groups of pods (each node / stateful / stateless)
- ConfigMap
  - Configuration data / files (can be mounted in containers)

Workshop

Monitoring Kubernetes with Prometheus

git checkout 1.setup

kubectl get nodes

Running Prometheus in Kubernetes

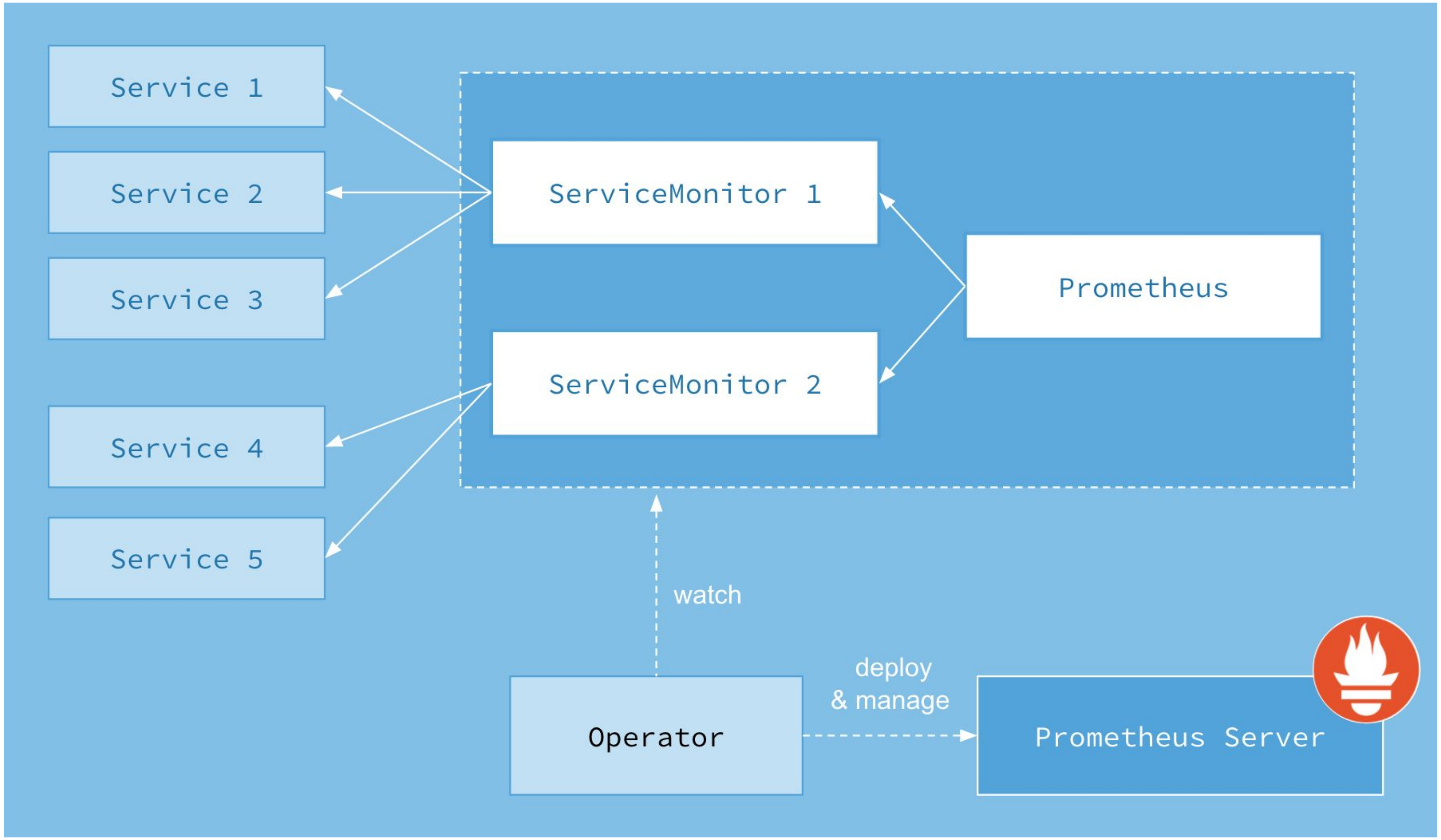
Installation and configuration

Running Prometheus

inside of Kubernetes

- What we need
    - Pod specification defining how to run Prometheus
    - Load and manage configuration
    - Service specification to access Prometheus on stable IP
- Options
    - Write own pod+service+petset+... manifests
    - Kubernetes Helm chart in the making https://github.com/kubernetes/charts/pull/151
    - CoreOS wrote an Operator managing Prometheus and its configuration: https://github.com/coreos/kube-prometheus https://coreos.com/blog/the-prometheus-operator.html

Prometheus Operator



© CoreOS https://coreos.com/blog/the-prometheus-operator.html

git checkout 2.install-prometheus

scripts/deploy

Monitoring Kubernetes infrastructure

Configuration and discovery

git checkout 3.monitor-nodes

scripts/deploy

git checkout 4.monitor-kubernetes

scripts/deploy

git checkout 5.install-grafana

scripts/deploy

Monitoring services in Kubernetes

Configuration and discovery

git checkout 6.monitor-example-app

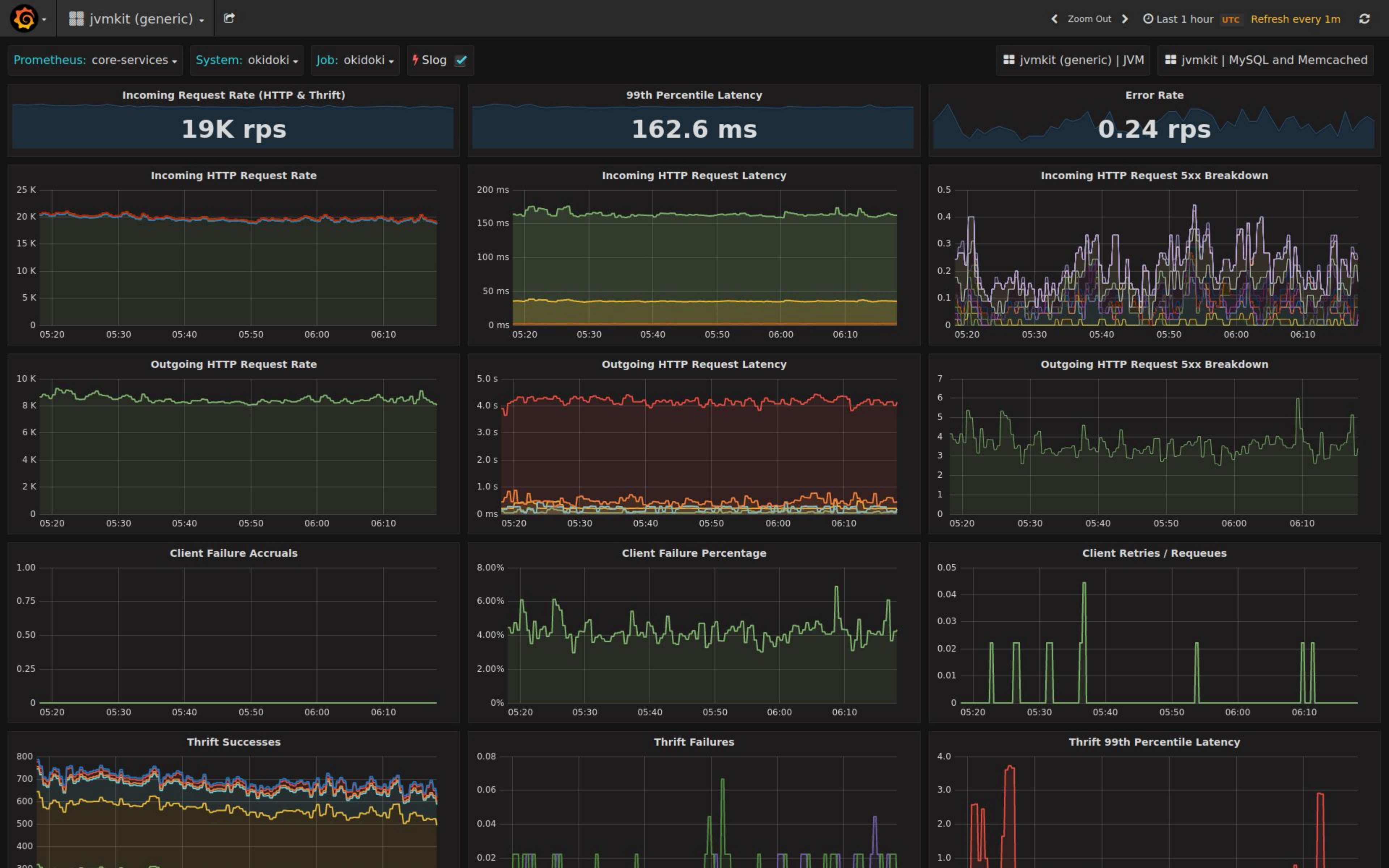
scripts/deploy

Practical examples

Queries and dashboards

git checkout 7.add-rules

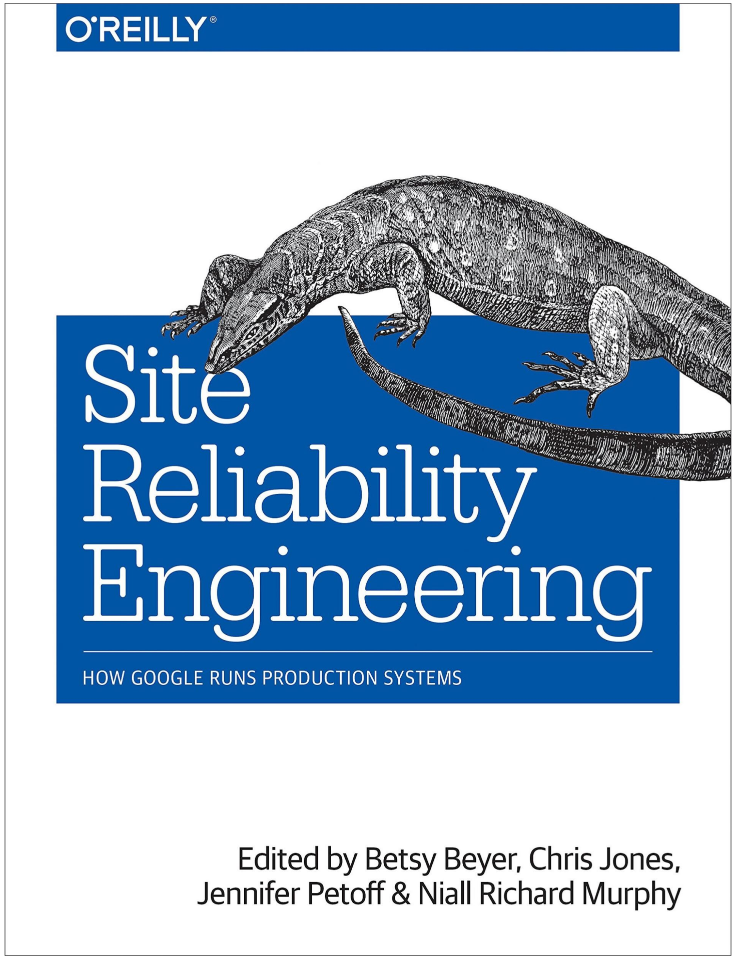
scripts/deploy


jvmkit (generic)
Zoom Out
Last 1 hour UTC
Refresh every 1m
Slog
jvmkit (generic) | JVM
jvmkit | MySQL and Memcached
Incoming Request Rate (HTTP & Thrift)
99th Percentile Latency
Error Rate
Incoming HTTP Request Rate
Incoming HTTP Request Latency
Incoming HTTP Request 5xx Breakdown
Outgoing HTTP Request Rate
Outgoing HTTP Request Latency
Outgoing HTTP Request 5xx Breakdown
Client Failure Accruals
Client Failure Percentage
Client Retries / Requeues
Thrift Successes
Thrift Failures
Thrift 99th Percentile Latency
05:20
05:30
05:40
05:50
06:00
06:10

Prometheus: core-services
System: okidoki
Job: okidoki

Incoming Request Rate (HTTP & Thrift): 19K rps
99th Percentile Latency: 162.6 ms
Error Rate: 0.24 rps

Further reading


O'REILLY

Site Reliability Engineering

HOW GOOGLE RUNS PRODUCTION SYSTEMS

Edited by Betsy Beyer, Chris Jones, Jennifer Petoff & Niall Richard Murphy

My Philosophy on Alerting

Rob Ewaschuk

https://docs.google.com/document/u/1/d/199PqyG3UsyXlwieHaqbGiWVa8eMWi8zzAn0YfcApr8Q/preview

Thank you

Tobias Schmidt - ContainerDays NYC November 4, 2016

github.com/grobie - @dagrobie

SOUNDCLOUD