SRE: A day in the life ...

by Marcel Birkner
Bio

Marcel Birkner works as a Staff Reliability Engineer at Instana, an Application Performance Monitoring (APM) solution. He has long experience in software engineering and software automation. Currently he focuses on migrating the existing stack to Kubernetes and reducing overall system complexity.
Abstract

What does a typical day as an SRE look like? In this presentation I will discuss the challenges we face while running a SaaS platform that is used 24 / 7 / 365 around the globe. In doing so, we have embraced the core principles described in the Google SRE handbook. While we are not Google by any means, most of the principles apply to our daily work one way or another. Having a fully distributed team running a distributed system can be quite challenging. In this talk I’ll be covering:

- Core SRE principles
- How Instana has applied them to our daily work
- Lessons learned along the way
Who We Are
SRE Team

1 Team
3 Time zones
24 / 7 / 365 support
On-call rotation
Members have operations and software engineering background
Stats

280 TB / Month Ingress

8 PB / Month Cross AZ Traffic

30K+ ECU

8 different datastore clusters / region

4K+ Containers Running in SaaS
SaaS Regions

Multi Cloud Strategy

2 x AWS regions

2 x GCP regions

HashiCorp
Nomad/Consul

Kubernetes
How We Do It
## SRE by the book

<table>
<thead>
<tr>
<th>Automation</th>
<th>Error Budgets</th>
<th>Capacity Planing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software</td>
<td>SLI / SLO / SLA</td>
<td>Maintenance</td>
</tr>
<tr>
<td>Development</td>
<td>RCA</td>
<td>Network Ops</td>
</tr>
<tr>
<td>Cost Planning</td>
<td>OnCall</td>
<td>Platform Eng / Ops</td>
</tr>
<tr>
<td>Database Ops</td>
<td>Post Mortems</td>
<td></td>
</tr>
<tr>
<td>Dev Support</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...
Planned Day

- 30 min  Handoff Team AU
- 50%     Tickets/QoS
- 50%     Project work
- 30 min  Handoff Team US
Learn to say “No”

Actual Day

- Handoff Team AU
- Alerts
- Ping by Engineering
- Ping by SE / PM
- Ping by CS
- less Project work than planned
- Handoff Team US
Communication is vital

"Something is broken"

Engineering:
"Okay, will have a look"

Sales / CS:
"OMG" => Escalation to CEO => Escalates to VP Eng.

Private Slack Channels
  tech-*

Avoid Panic
SRE Team Priorities

- Quality of Service of SaaS platform
- Platform Security
  - regularly security scans
- Project Work
  - Multi Cloud (AWS & GCP)
  - Cost Management
  - Migrate platform to Kubernetes
  - Upgrade Elasticsearch clusters
  - Integrating new datastore (BeeInstant)
- Support On-Premises
- Developer Support
- Packaging and Delivery
Embracing Risk

- Redundancy / HA / failover
  - datastore clusters across AZ
  - horizontal scaleout of components
- Costs
  - Cost per monitored host
  - K8s / Nomad Orchestration bin-packing
  - Managing TU resources
- Beta Phase for new features
  - Test using internal units
  - Beta customers
- Coming soon: Error Budgets
Service Level Indicators / Objectives

- Custom SLOs for all components in SaaS platform
  - SLO configuration stored and versioned with backend code
  - Updated via REST API after each release
  - Identical across all regions
- Managed by Engineering and SRE
### Eliminating Toil

"The moment you have to do something twice, think about automating it"

<table>
<thead>
<tr>
<th>Task</th>
<th>Tools</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spin up new VM</td>
<td>Jenkins + Terraform</td>
</tr>
<tr>
<td>Setup / Expand datastore cluster</td>
<td>Chef recipes</td>
</tr>
<tr>
<td>Deploy / Update components</td>
<td>Jenkins + instanactl</td>
</tr>
<tr>
<td>Run migrations</td>
<td>Jenkins + instanactl</td>
</tr>
<tr>
<td>Configure Jenkins Job</td>
<td>Jenkins Job DSL (all jobs are generated)</td>
</tr>
<tr>
<td>Configure DNS</td>
<td>instanactl / external-dns (a few DNS entries are manually configured)</td>
</tr>
<tr>
<td>Setup GKE cluster</td>
<td>gcloud</td>
</tr>
<tr>
<td>Setup EKS cluster</td>
<td>eksctl</td>
</tr>
</tbody>
</table>
Monitoring Distributed Systems

We use Instana to monitor Instana

- Datastores (Cassandra, ClickHouse, CockroachDB, Elasticsearch, Kafka, ZooKeeper, ...)
- Infrastructure Monitoring
- Java DropWizard
- NodeJS
- Automatic Distributed tracing
- Automatic End-User-Monitoring
- Built-in alerting

Feedback Loop with PM & Engineering
Release Engineering

- Bi-Weekly Major Releases (Consistency)
- Continuous Release of Beta Features & Improvements & Hotfixes (24 / 7)
- Rotating Release Engineer
  - Knowledge Sharing / Release Engineer Playbook
- Rollout for new K8s environments fully automated
  - instanactl <environment> upgrade
    - check preconditions
    - run migrations
    - upgrade shared and tenant unit containers
    - check postconditions
- Post Mortem after each release / incident
  - improve / automate / refactor processes
Simplicity, Simplicity, Simplicity, Simplicity, ...
Infrastructure automatically becomes more complex over time due to growth and other external factors.
System architecture automatically becomes more complex when new features are added over time.
Plan your infrastructure and network design for growth and simplicity. Keep the overall system as simple possible and only as complex as really needed. This will make your life a lot easier during your typical work day. In times of crisis (i.e. outages) a simple system is easier to understand for all engineers involved to resolve the issue at hand.
Common Codebase (SaaS / On-Premises)

up to 2019

Each datastore its migration tool

- Cassandra (cassandra-migrator)
- ClickHouse (golang-migrate)
- Elasticsearch (http-client)
- Kafka (kafka-cli)
- MongoDB (mongo migrator)
  - replaced by CockroachDB
- PostgreSQL (flyway db)
  - replaced by CockroachDB

Runtimes: Ruby/Python/Java

2020

instanactl

- GoLang CLI
  - cobra library
  - golang-migrate library
- used by SaaS and On-Premises
- single place for migration scripts

Runtimes: Single GoLang Binary
Common Codebase (SaaS / On-Premises)

up to 2019

- separate configuration
- separate packaging (Docker / Packages)
  - SaaS: Docker
  - OnPrem: RPM / DEB
- separate delivery (Ansible / Chef)

Runtimes: Python / Ruby

2020

- same configuration
- same Docker images
- same migration tool
  - instanactl

Runtimes: GoLang Binary & Docker

Supported Operating Systems

- Ubuntu 16.04, 18.04
- Debian 8.x, 9.x, 10.x
- RedHat 7.2+
- CentOS 7.x
- Amazon Linux 2.x
Lessons Learned
Learn to say “No”  
Reduce Complexity  
Keep Tooling to a Bare Minimum  
Communicate Across Teams  
Know Your Tools  
Focus and Prioritize Work  
Share Knowledge (SRE runbook, screen recordings, blogs)