Cameron Tuckerman-Lee / DevOpsDays Shanghai / 2017-08-18

SRE at Airbnb

Cameron Tuckerman-Lee

Airbnb Site Reliability Engineer

SRE at Airbnb

How do you combine the culture and spirit of DevOps with an operations team?

DevOps & SRE

SRE Organization

Future of Ops

SRE at Airbnb

How is SRE at Airbnb organized? Cloud Infra and Reliability deep-dive.

DevOps & SRE

SRE Organization

Future of Ops

SRE at Airbnb

Operators should grow, learn, and be recognized for on-call work, while maintaining pager-life balance.

DevOps & SRE

SRE Organization

Future of Ops

DevOps & SRE

Centralized Ops

Centralized Operations Organization

- Positives
  - Reliability can be easily prioritized
  - Specialization of roles
- Negatives
  - Operators unfamiliar with code base
  - Tension between operations and development

Centralized Ops

Distributed Ops

Distributed Operations

- Positives
  - Agility can be easily prioritized
  - Developers are incentivized to build systems that are easy to operate (since they are the operators!)
- Negatives
  - Lack of specialization --- devs are forced to relearn difficult lessons over-and-over
  - Teams speak different uptime/reliability languages to each other

Centralized Ops

Distributed Ops

Hybrid Approach

Hybrid Approach: Two Pizza Teams + SRE Team

- Able to 'tune' a balance between reliability and agility
- Developers are still expected to run normal operations for their services == build operable services
- Centralized operations organization can build reusable tools to make operations / incident response easier.
- Specialization of roles without tension between operations and development teams.
- Organization that understand and recognizes the value in automating away their job.

> Fundamentally, it's what happens when you ask a software engineer to design an operations function...

Ben Treynor
VP Engineering, Google

SRE Organization

What makes up SRE at Airbnb?

Site Reliability Engineering is made up of three components:

Cloud Infrastructure

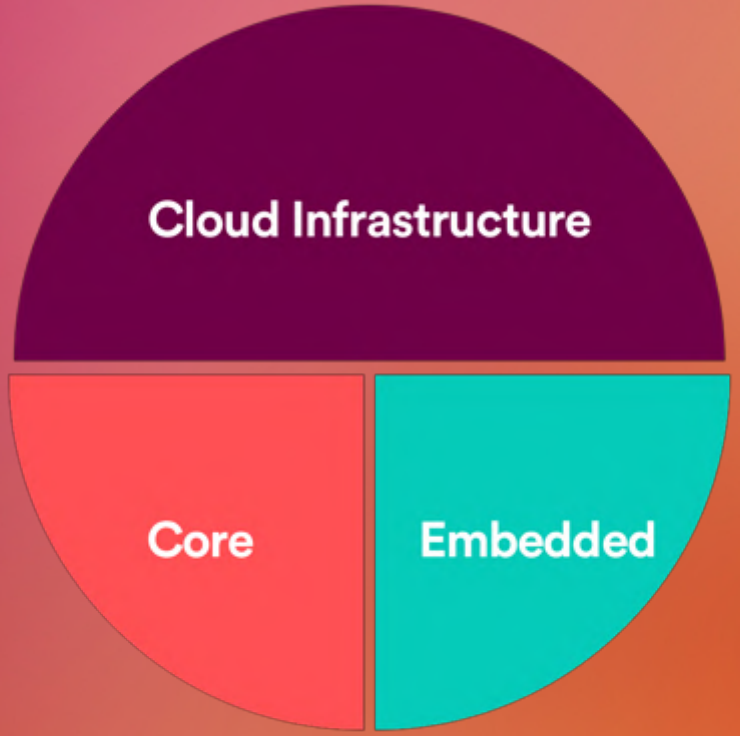
Manages our touch points with AWS and other cloud partners

Core Reliability

Develops tools and processes to improve operations, reliability, and incident response for all teams

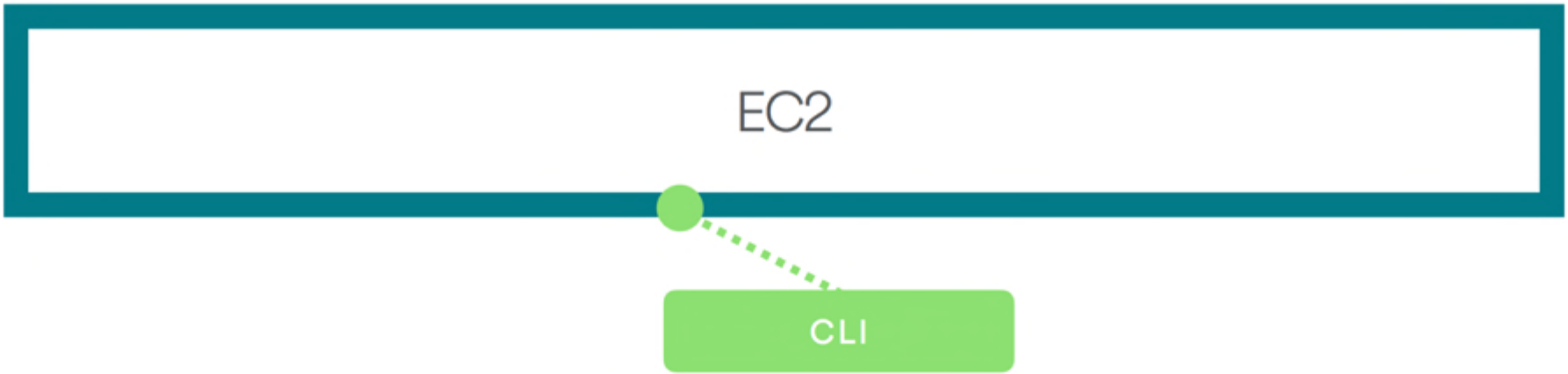
Embedded Reliability

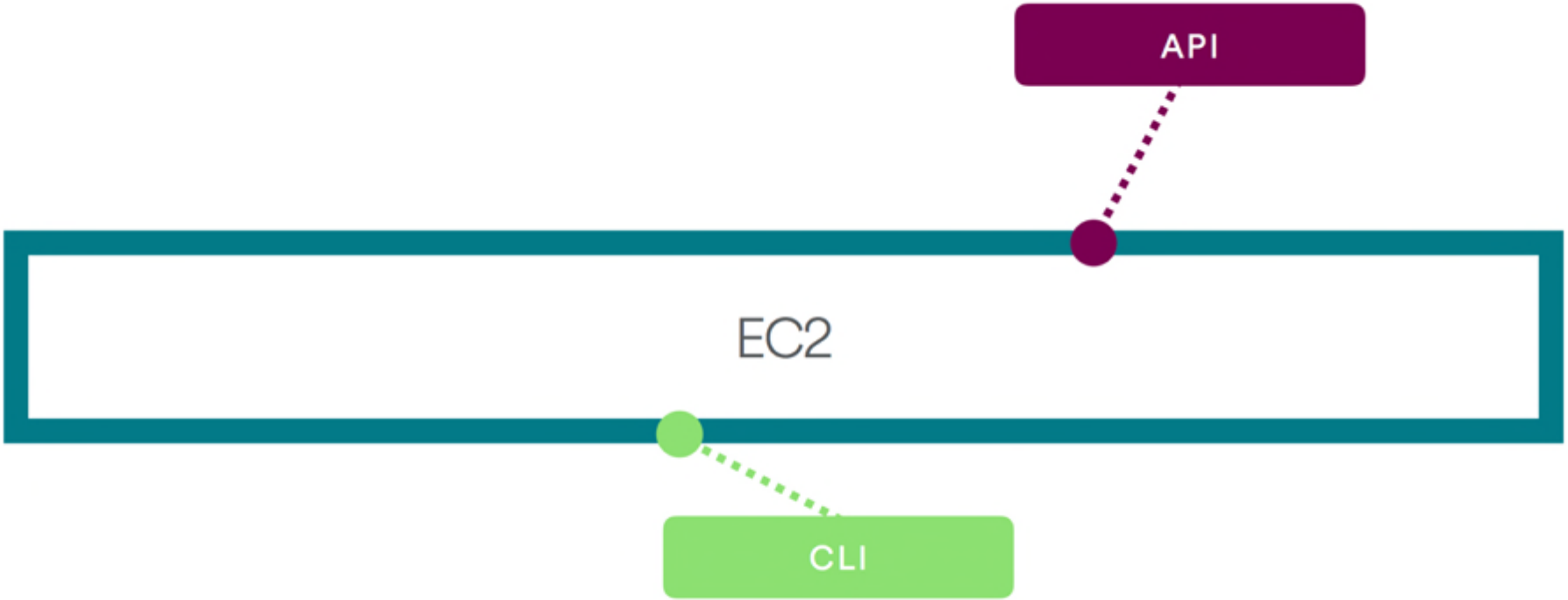
Temporary embedding of SREs in product teams to work on specific reliability or availability focused projects

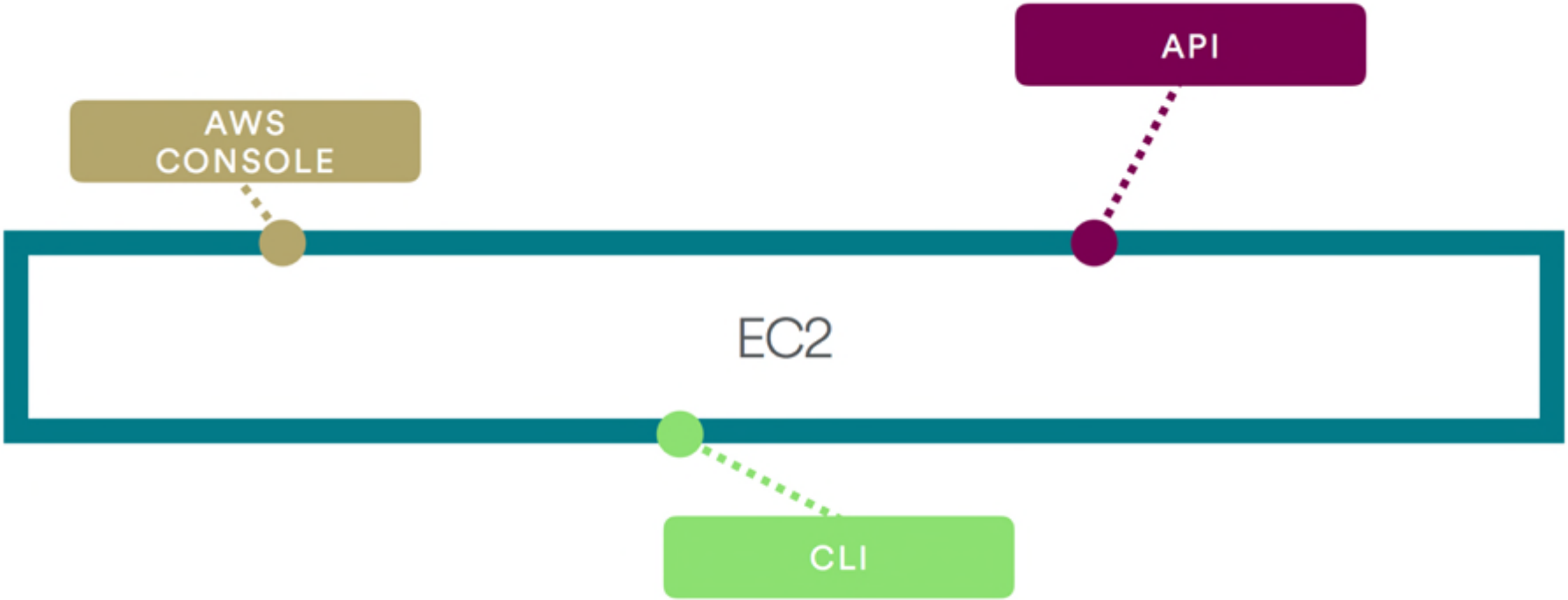


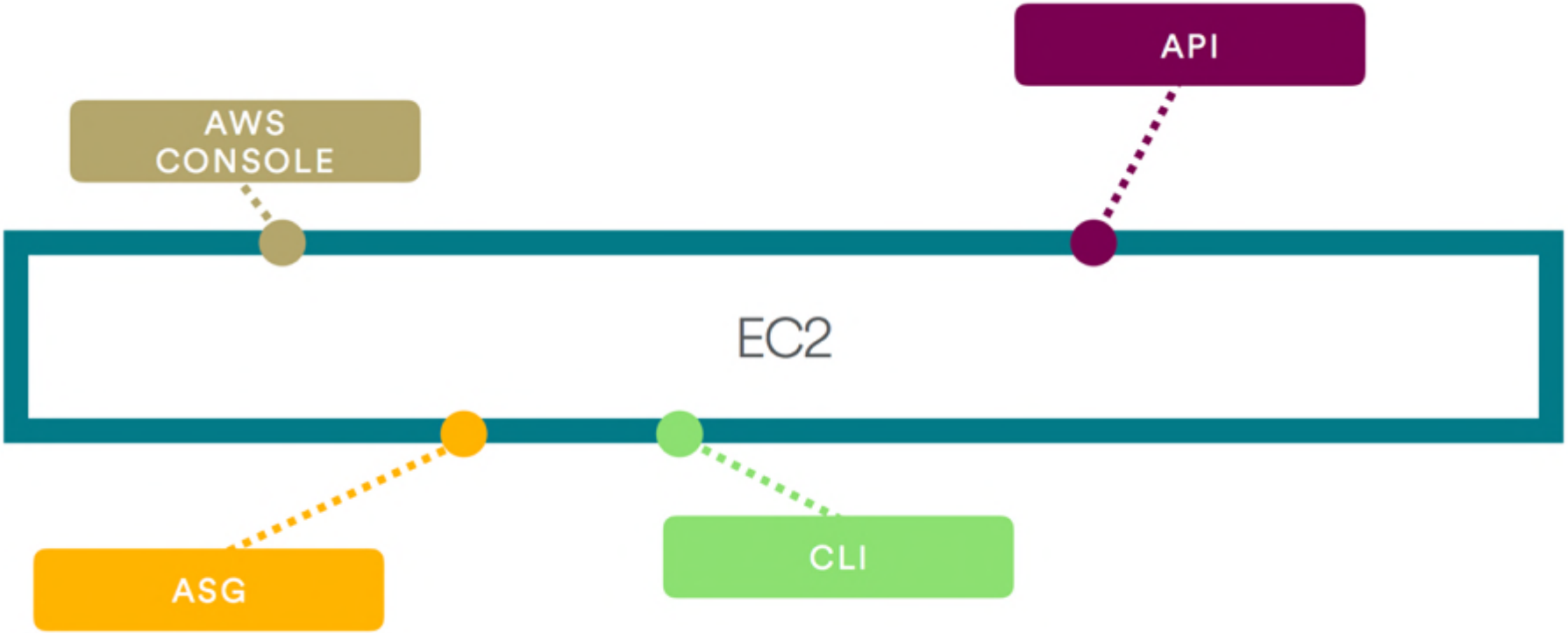
Cloud Infrastructure

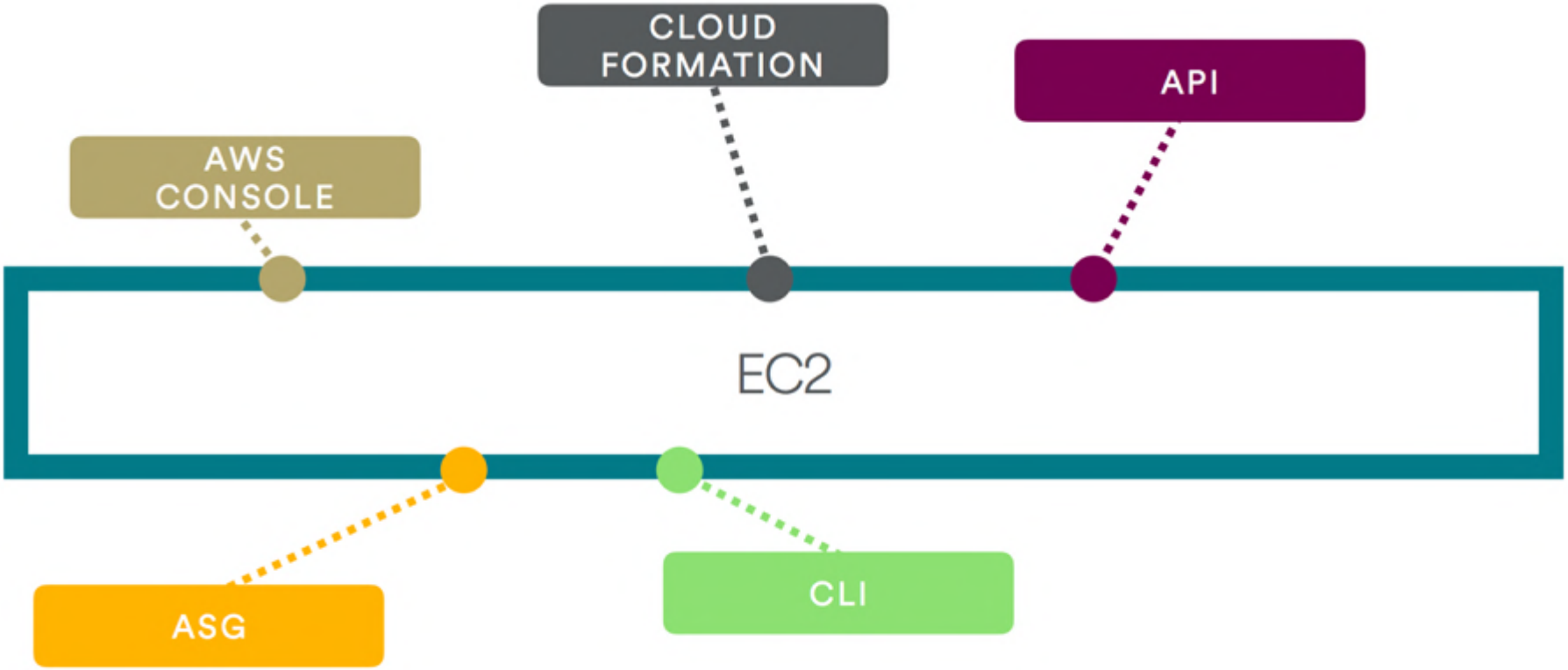

EC2

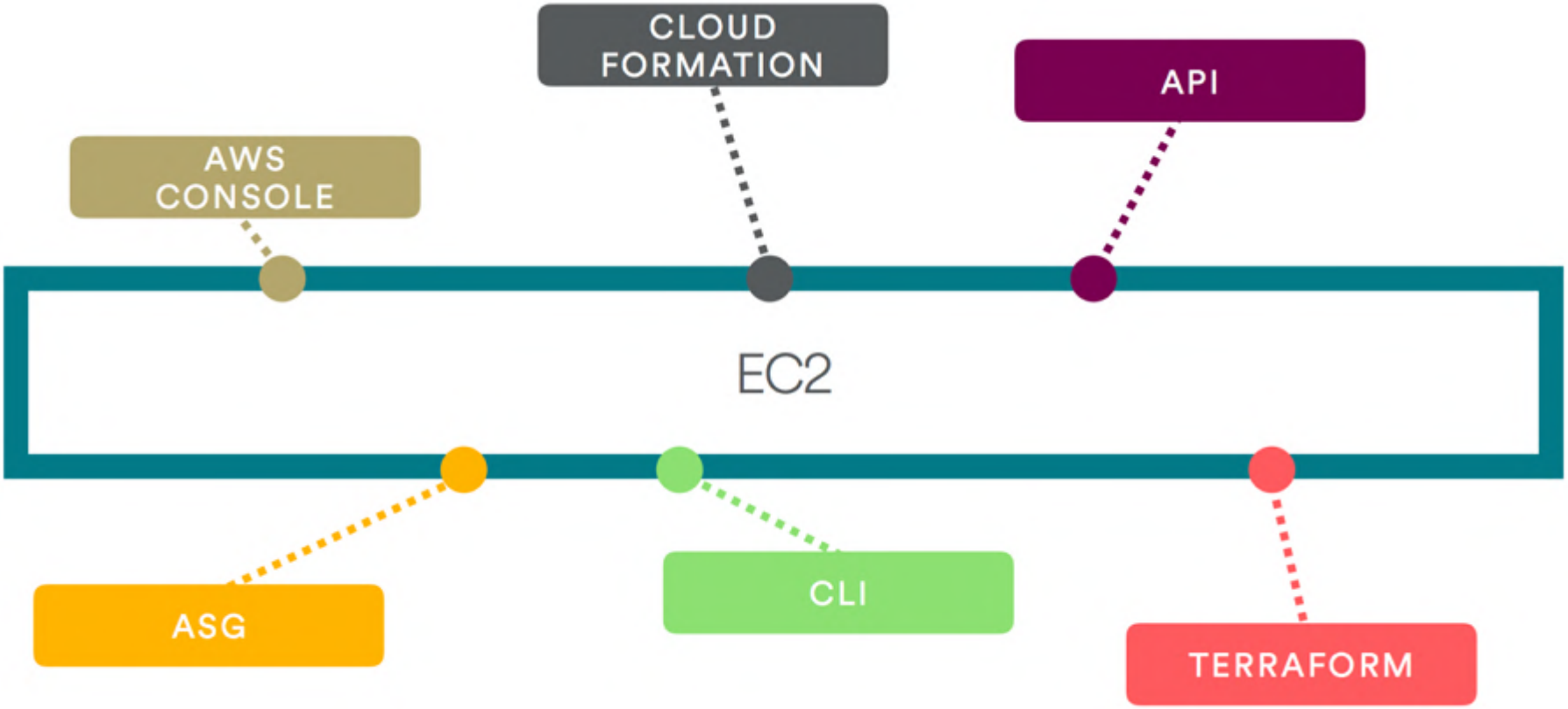

EC2
CLI


API
EC2
CLI


API
AWS
CONSOLE
EC2
CLI


API
AWS
CONSOLE
EC2
ASG
CLI


CLOUD FORMATION
API
AWS CONSOLE
EC2
ASG
CLI


CLOUD FORMATION
API
AWS CONSOLE
EC2
ASG
CLI
TERRAFORM

Requirements for Each Integration

Monitoring

Alerting

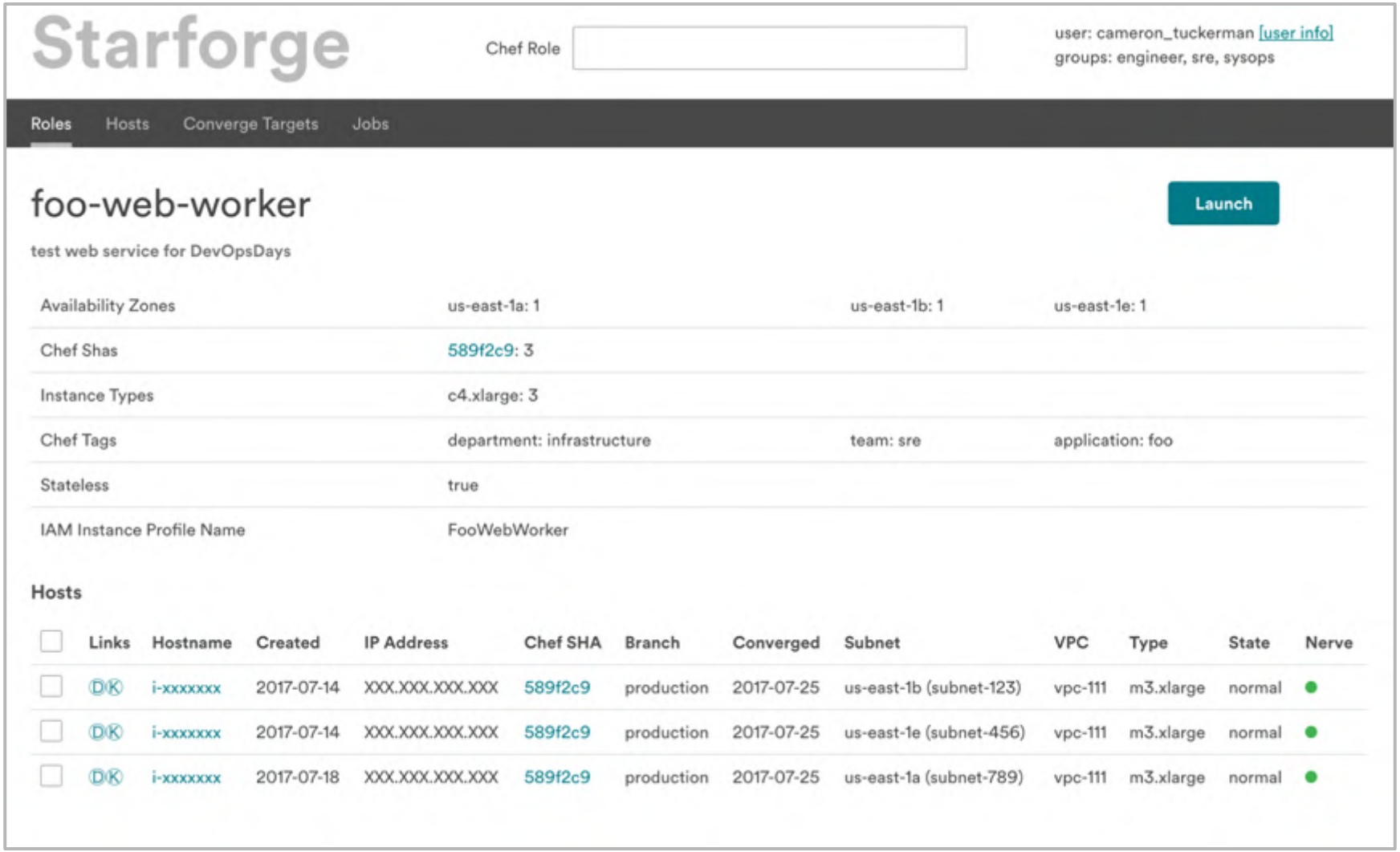
Security Approval

Auditing

Version Upgrades

Access Control

...

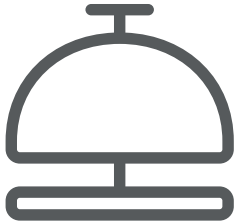

Starforge
Chef Role
user: cameron_tuckerman [user info]
groups: engineer, sre, sysops
Roles
Hosts
Converge Targets
Jobs
foo-web-worker
Launch
test web service for DevOpsDays
Availability Zones
us-east-1a: 1
us-east-1b: 1
us-east-1e: 1
Chef Shas
589f2c9: 3
Instance Types
c4.xlarge: 3
Chef Tags
department: infrastructure
team: sre
application: foo
Stateless
true
IAM Instance Profile Name
FooWebWorker
Hosts
Links
Hostname
Created
IP Address
Chef SHA
Branch
Converged
Subnet
VPC
Type
State
Nerve
D K
i-xxxxxxx
2017-07-14
XXX.XXX.XXX.XXX
589f2c9
production
2017-07-25
us-east-1b (subnet-123)
vpc-111
m3.xlarge
normal
D K
i-xxxxxxx
2017-07-14
XXX.XXX.XXX.XXX
589f2c9
production
2017-07-25
us-east-1e (subnet-456)
vpc-111
m3.xlarge
normal
D K
i-xxxxxxx
2017-07-18
XXX.XXX.XXX.XXX
589f2c9
production
2017-07-25
us-east-1a (subnet-789)
vpc-111
m3.xlarge
normal

Reliability

Three Pillars of Reliability

Uptime Measurement

Every team at any time should be able to *confidently* say whether their service is working properly or not.



Alerting & Detection

Defense-in-depth: our users are protected from bugs and regressions by multiple layers of opinionated alerts.

Incident Response

Engineers can coordinate across teams, investigate problems in systems they don't fully understand, and keep stakeholders up-to-date.

1. Uptime

Identify quantifiable metrics which are related to the health of their services, called *(Service Level Indicators or SLI)*

Make public and easily discoverable promises about the behavior of your service using your SLIs *(Service Level Objectives or SLO)*

Teams review their services current SLIs and compare them to their published SLOs to make tradeoffs between reliability improvements and new features --- SLOs encode the tradeoff between moving fast and breaking things *(Error budgets)*

1. Uptime

2. Alerting

Alerting philosophy should be opinionated --- engineers know what kind of alerts to write and when to write them

Alerts (like configuration) should be code

Practice defense in depth --- protect your users from bugs and regressions with layers of alerts like a security team protects employees from being compromised with layers of defenses

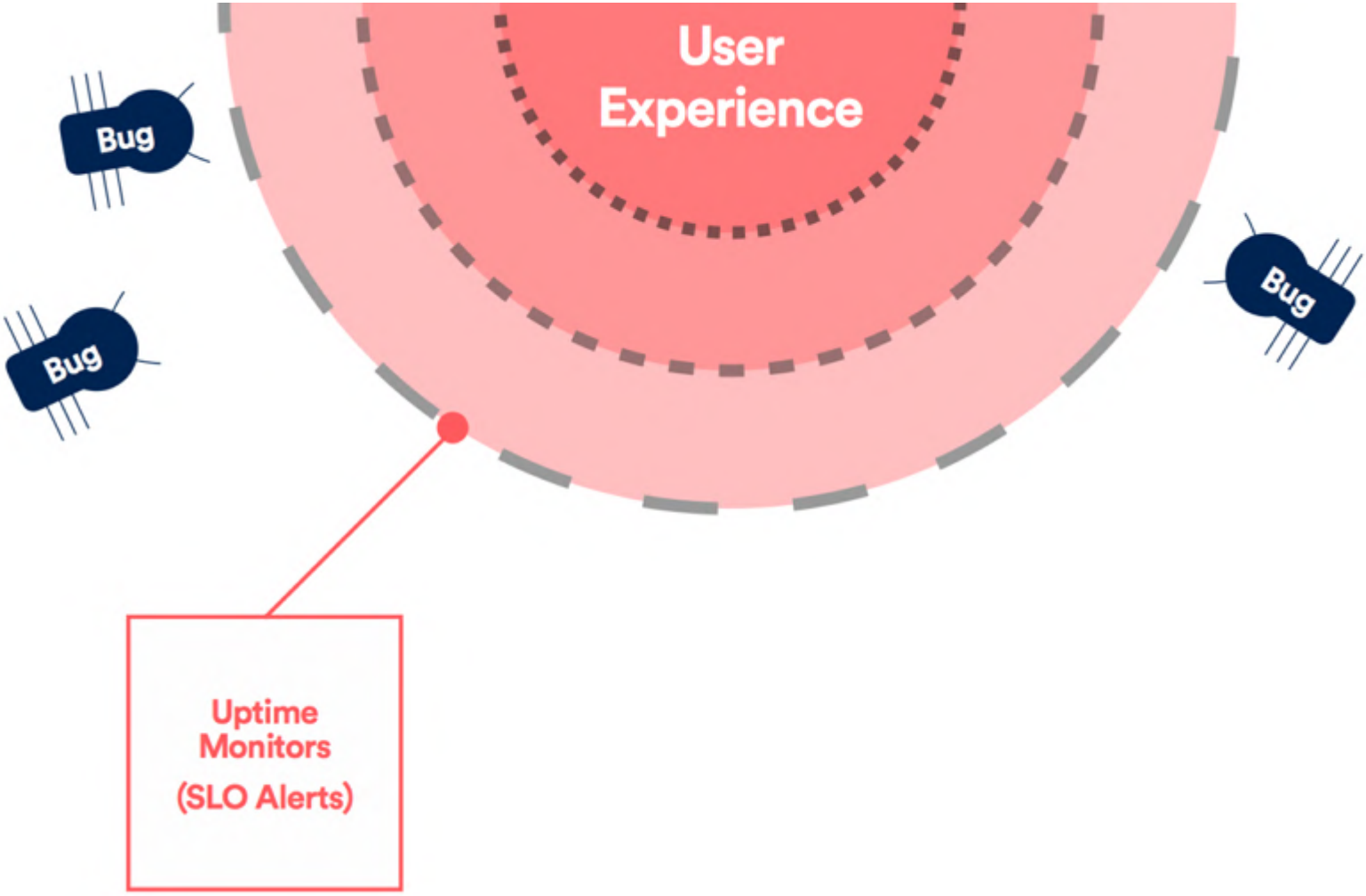
Example Alert

HIGH CPU USAGE

Able to query internal data stores to create alerts for different applications automatically (inserts data into the @hostinfo variable).

Write alerts once and have them be automatically enabled for every new service your team builds.

```
name '#{@hostinfo[:role]} high average load'

message <<EOF
#{@hostinfo[:role]} has high average load!

https://dashboards/#{@hostinfo[:role]}
https://deploys/#{@hostinfo[:role]}
EOF

notify.groups [@hostinfo[:team]]

applies { !@hostinfo[:role].include? 'test' }

metric.query <<EOF
  avg(last_5m):
    avg:system.load{role:#{@hostinfo[:role]}}
  > 0.9
EOF
```


User
Experience
Bug
Bug
Bug
Uptime
Monitors
(SLO Alerts)

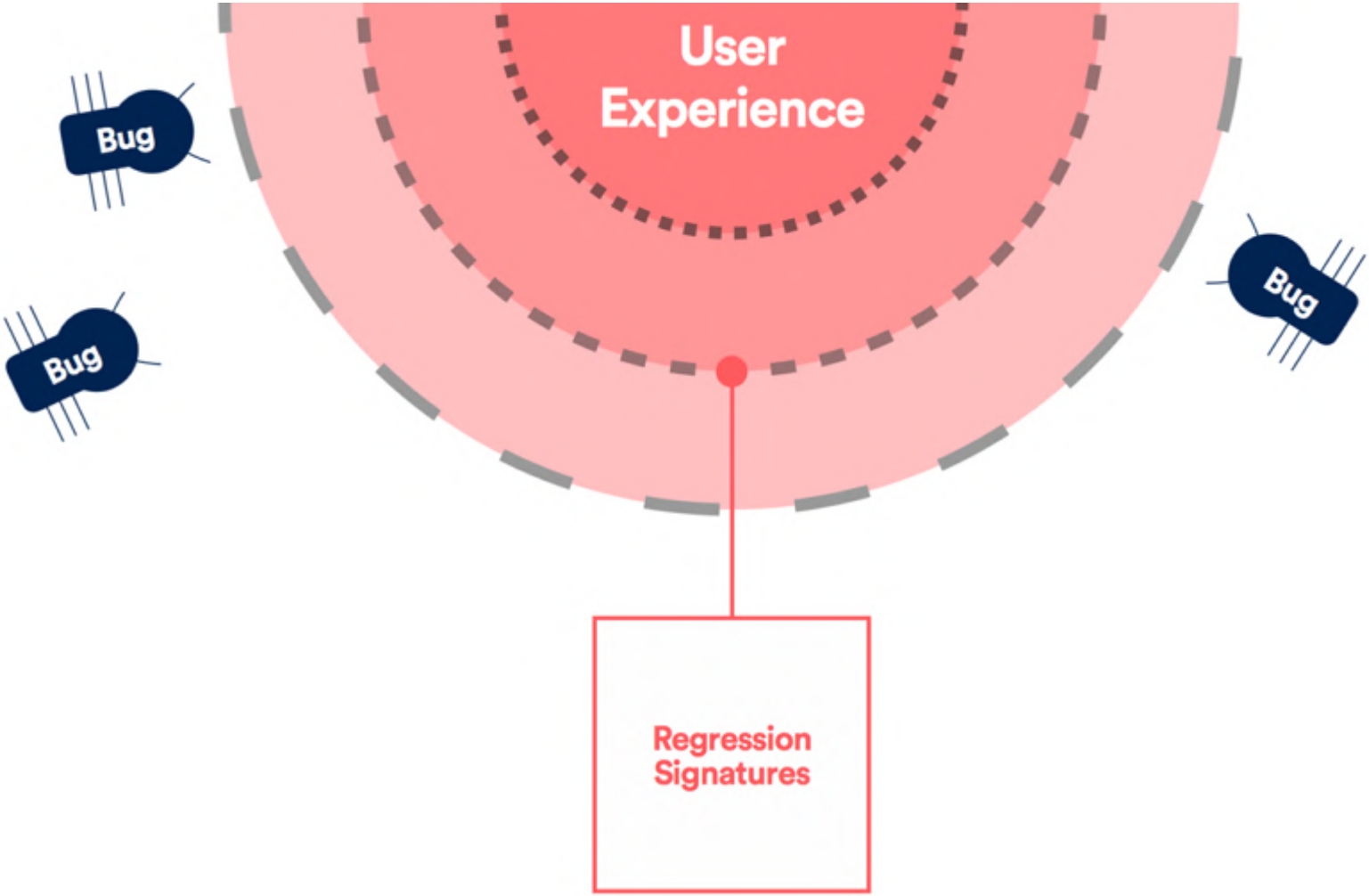
Uptime Monitors

SERVICE LEVEL OBJECTIVE VIOLATIONS

If a service has a contract to behave a certain way (e.g. p99 latency less than some threshold) alert the team if that contract is violated.

Can also automatically notify the users of that service so they can take mitigating steps as well.

```
name 'Foo Service - Latency SLO Violation'

message <<EOF
Foo Service Latency SLO Violation Notification

Service Level Objective:
5 min avg p99 service-side latency < 750 ms
EOF

notify.groups ['foo-team', 'bar-team']

metric.query <<EOF
  avg(last_5m):
    avg:foo_service.request.p99
  > 750
EOF
```


User Experience
Bug
Bug
Bug
Regression Signatures

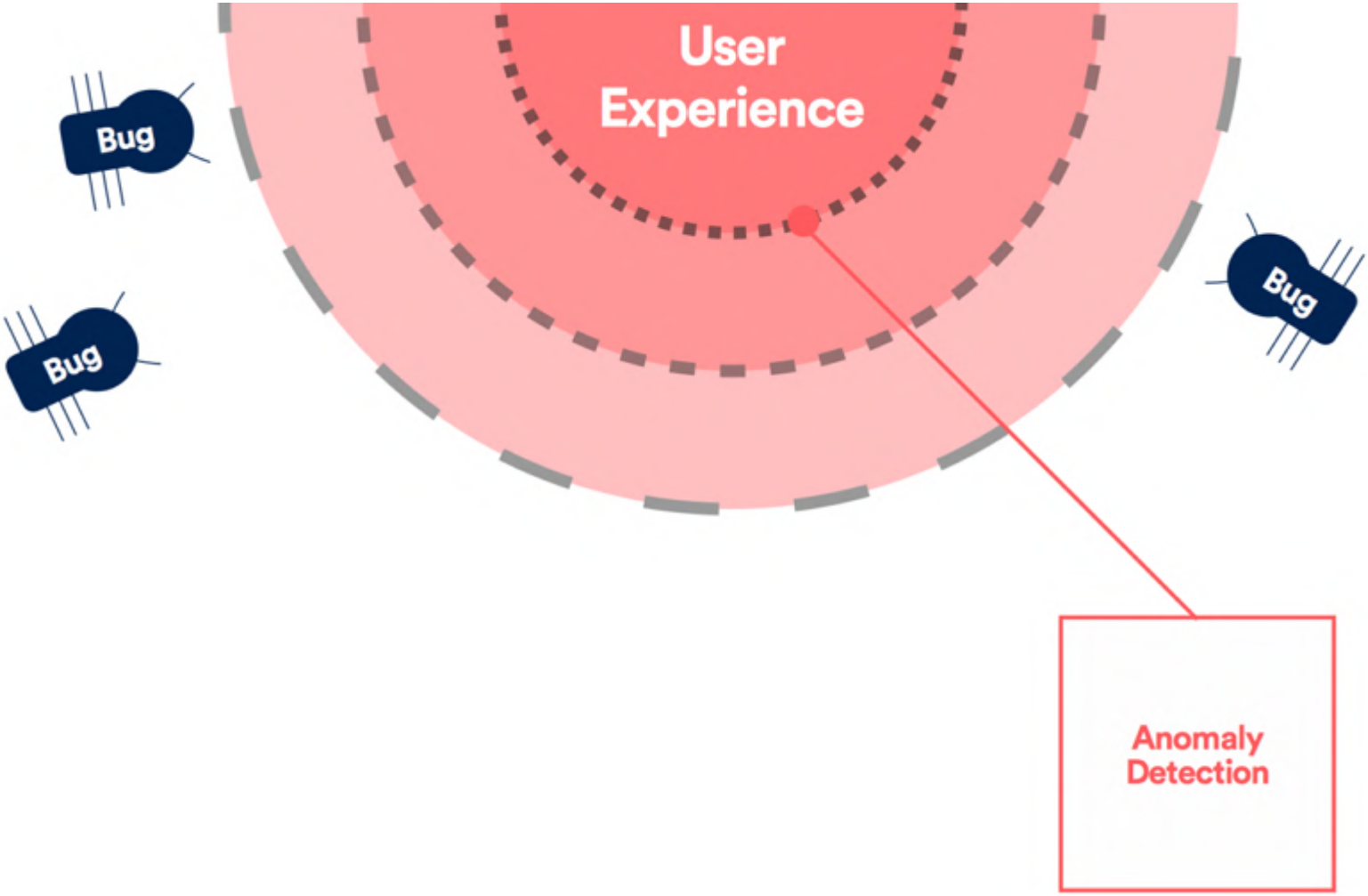
Regression Signatures

PREVIOUS INCIDENT IMMUNIZATION

If an incident occurred and there wasn't an alert that caught it, you can add a metric and alert to cover the exact case of the incident.

If the incident ever reoccurs, the team will be notified immediately!

```
name '[regression] response missing required field'

message <<EOF
Regression Signature Alert, incident #1754

A required field in the /api/v2/baz endpoint is
missing, which can cause crashes in the mobile
applications.

https://postmortems/1754
EOF

notify.groups ['baz-team']

metric.query <<EOF
  sum(last_1m):
    sum:baz_api_response_missing_field
  > 1
EOF

notify_no_data false
```


User
Experience
Bug
Bug
Bug
Anomaly
Detection

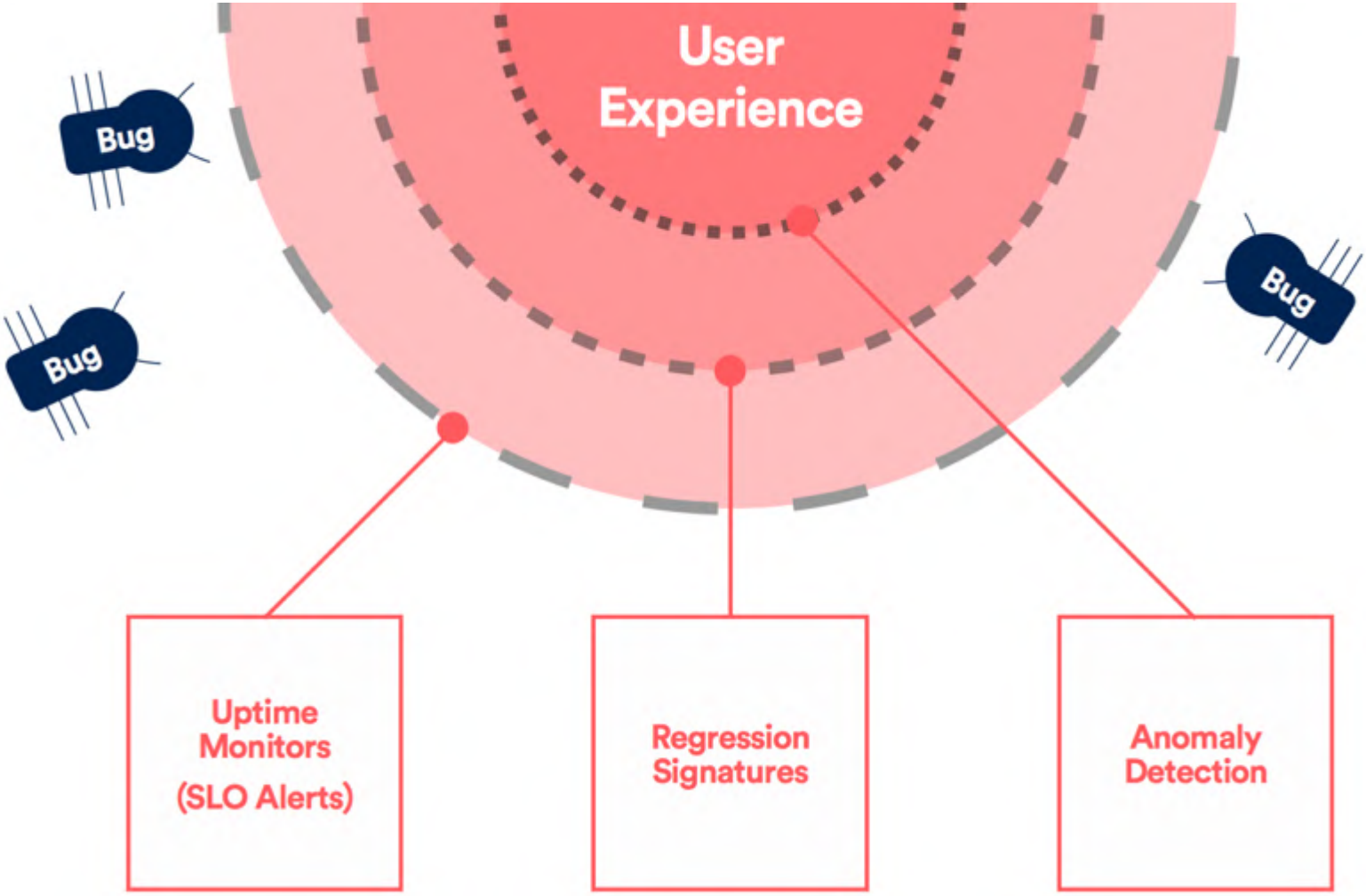
Anomaly Detection

BUSINESS METRIC AND SERVICE LEVEL METRIC ANOMALY DETECTION

As the last level of defense, you can look for anomalies in your business and system metrics, e.g. number of logins.

The anomaly detection can be advanced (e.g. machine-learning based) or simple (e.g. percentage changes week-over-week).

```
name '[anomaly] Logins dropped versus 7 days ago'

message <<EOF
For a specific platform (for example web, android, or
iOS), logins have dropped significantly this hour
compared to the previous week. This alert should be
investigated with high priority!

https://dashboards/business-metrics
https://deploys/web-service
EOF

notify.groups ['signup-login-team']

metric.query <<EOF
  pct_change(sum(last_1h), 7d_ago):
    sum:logins by {platform}
  < -25
EOF
```


User Experience
Bug
Bug
Bug
Uptime Monitors (SLO Alerts)
Regression Signatures
Anomaly Detection

1. Uptime

2. Alerting

3. Response

Incident Reporter Tool

- Mid-Incident
  - Engineers can effectively coordinate, even across teams
  - Stakeholders (upstream clients, management, employees) are kept aware of updates
  - Working on a Slack integration so responders can stay in chat but keep the company up-to-date
- Post-Incident
  - Blameless postmortem process
  - Consistent impact measurement (management sees that better incident response + corrective actions matters to the bottom line)
  - Easily search past incidents/postmortems

Incident Reporter

Open Incident

Incidents

Search

Ongoing Incidents (1)

Start Time	Title
2017-07-25 11:16	Bad thing that is currently happening

Concluded Incidents (3)

Start Time	Title	Duration
2017-08-01 00:10 UTC	incident 1	2 minutes
2017-08-02 05:55 UTC	incident 2	11 minutes
2017-08-03 14:05 UTC	incident 3	4 minutes

Future of Ops

Future of Ops

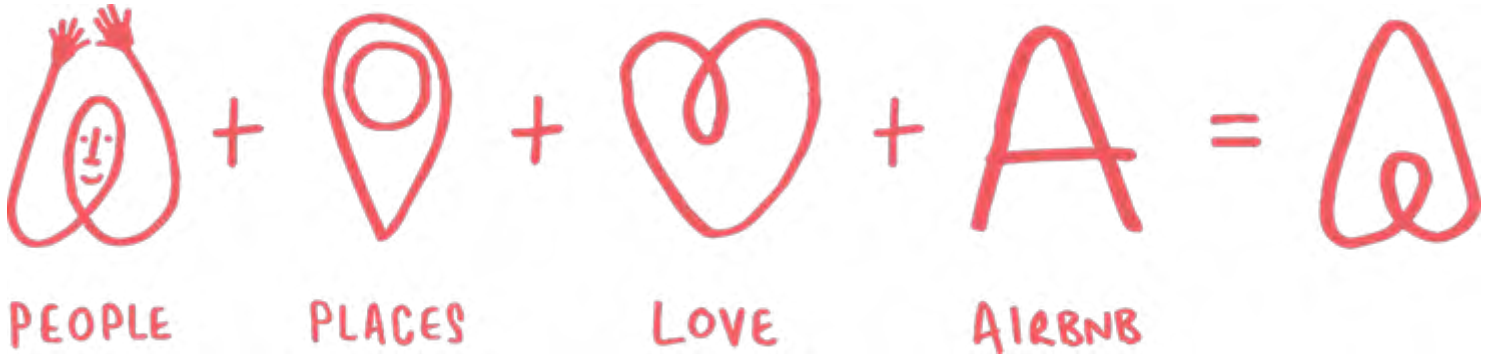
People-First On-call

Pager-Life Balance: Ensure that more involved, tenured engineers aren't always the ones waking up at 3 AM to put out fires

Learning/Growth Focused: Continuing education and learning opportunities for on-call engineers

Evaluation Metrics: Engineers should know where they can improve and should be recognized for excellent work

Intelligent Scheduling: In DevOps when every team has at least two on-call rotations, how can we schedule around lives outside of work (and responsibilities inside of work)?


PEOPLE
PLACES
LOVE
AIRBNB

高效运维社区
GreatOPS Community

DevOpsDays Shanghai

会议

- 8月18日 DevOpsDays 上海
- 全年 DevOps China 巡回沙龙
- 11月17日 DevOps金融上海

培训

- EXIN DevOps Master 认证培训
- DevOps 企业内训
- DevOps 公开课
- 互联网运维培训

咨询

- 企业DevOps 实践咨询
- 企业运维咨询

商务经理：刘静女士
电话 / 微信：13021082989
邮箱：liujing@greatops.com