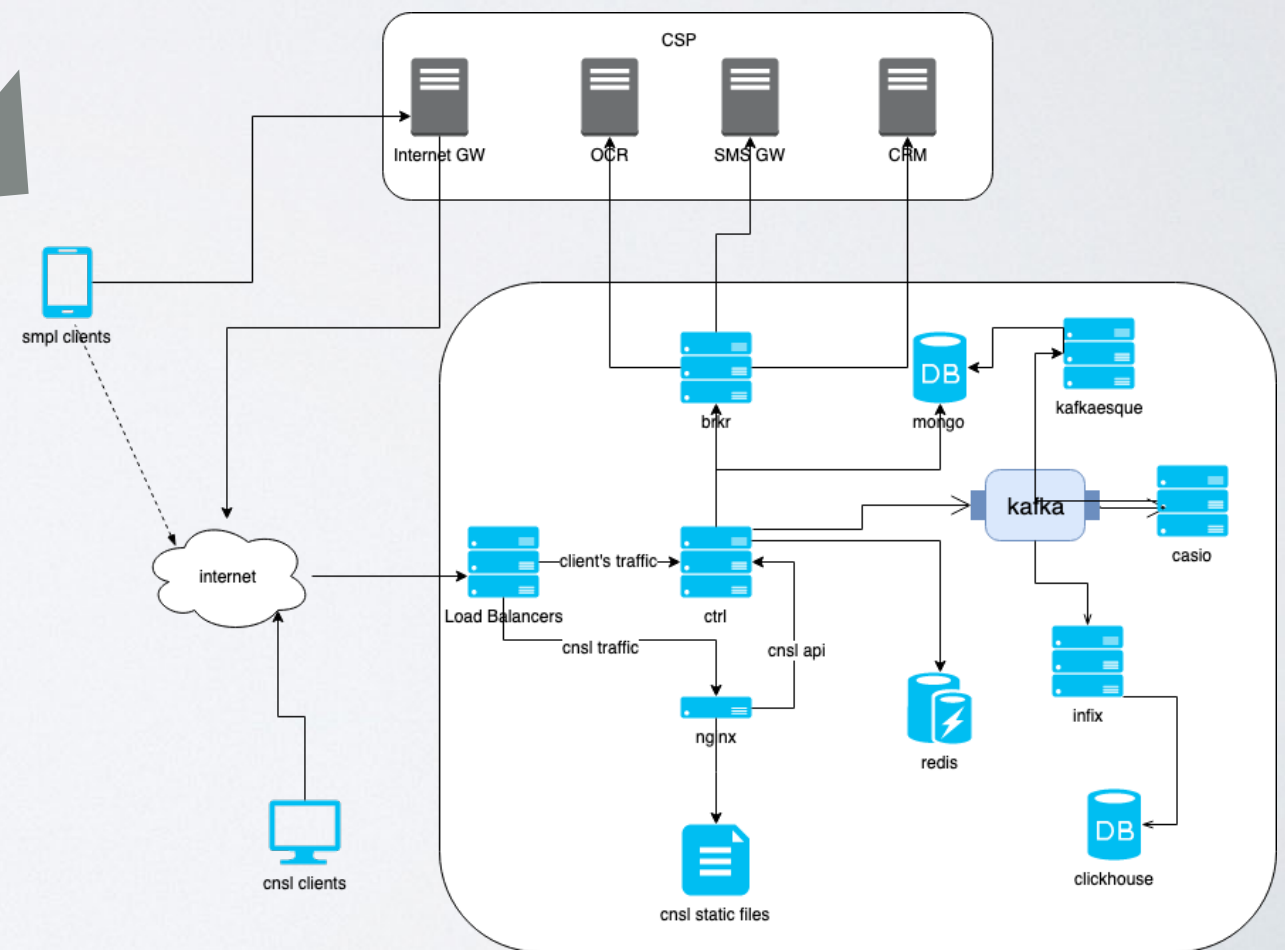
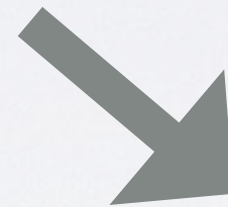
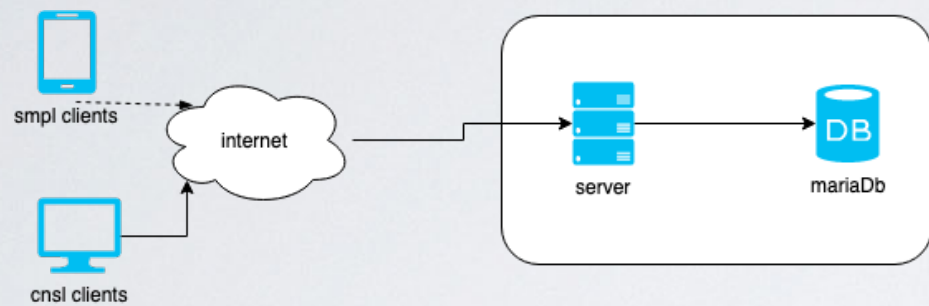


THE PATH TO UNDERSTAND AND DEBUG DISTRIBUTED SYSTEMS

DevOps was partly bringing software development practices to operations

Now we need to bring some **operations practices** to **developers**

COMPLEXITY HAS INCREASED



NEW MINDSET

- Code is not done until running in production
- Devs are responsible to keeps services up
- Test in production

We need new tooling and instrumentation

USER CENTRIC

- Maximize UX
- What are the key features for your users?
- Define SLOs (Service Level Objectives)
- Measure it (error rates, latency...)

KIND OF PROBLEMS

- Known-Knows
- Known-Unknowns
- Unknown-Unknowns

USUAL TOOLSET

- Logging
- Monitoring
- APM

LOGS

STRUCTURED LOGS

```
{  
  "ContainerID": "fe2e868c481b1e2d4d321",  
  "MesosTaskID": "ctrl-blue.cbc2cbd2-dab4",  
  "lvl": "debug",  
  "msg": "brkr-somesystem-http-response",  
  "status": 200,  
  "t": "2019-09-24T08:47:41.960654375Z",  
  "duration": 252.212049  
}
```

- JSON
- Parsable
- Use some Logging library for your language

ADD CONTEXT

Identify

```
{  
  "AccountID": "5c522f9befb45000085a87e0",  
  "RequestID": "17c854",  
  
  "msg": "brkr-somesystem-http-response",  
  "status": 200,  
  "duration": 252.212049,  
  
  "clientHost": "13.73.192.18",  
  "action": "CheckBalance",  
  "callName": "CreditControl",  
  "clientVersion": 100207,  
  "clientOS": "iOS",  
  "clientOSVersion": "ios-12",  
  ...  
}
```

Describe

Context

NAMING CONVENTION

```
{  
  "msg": "brkr-somesystem-http-response",  
  ...  
}
```

- Shows the code hierarchy
- Describes the operation

LOG LEVELS

```
{  
  "level": "[debug|info|warning|error|whatnot]",  
  ...  
}
```

- Two levels:
 - Useful when debugging
 - Something that would require attention

TOOLING

CTRL Daily Summary anna 331

api-purchase-fail-maxactivepackets-error	1
api-purchase-fail-redeem	3
api-sync-fail-customer-info-err_account_not_active	41
api-sync-fail-customer-info-err_incompatible_product	2
api-sync-fail-customer-info-err_temporary_blocked	242
api-sync-fail-customer-info-op_api_failure	5
data-usage-save-event-event-rt-stream-error	1
integration-unreg-crm-error	1
redis-receiver-error	1
smpl-auth-invalid-header-force-unreg	8
smpl-event-bind-json-error	13
smpl-event-streamevent-invalid-poke-event	5
smpl-upload-logs-create-convert-version-atoi	3
sms-poke-worker-error-sending-batch-sms	1
sms-poke-worker-fail-to-send	1
sms-poke-worker-unreg-failed	1
sms-send-smpp-multi-empty-sm	1
sms-send-smpp-send-fail	1

LOGS

- Use a centralised log collector
 - ELK, Loggly, Splunk, ...
- Applications should just write to stdout
 - rsyslog, logspout, ...

LOGS GOTCHAS

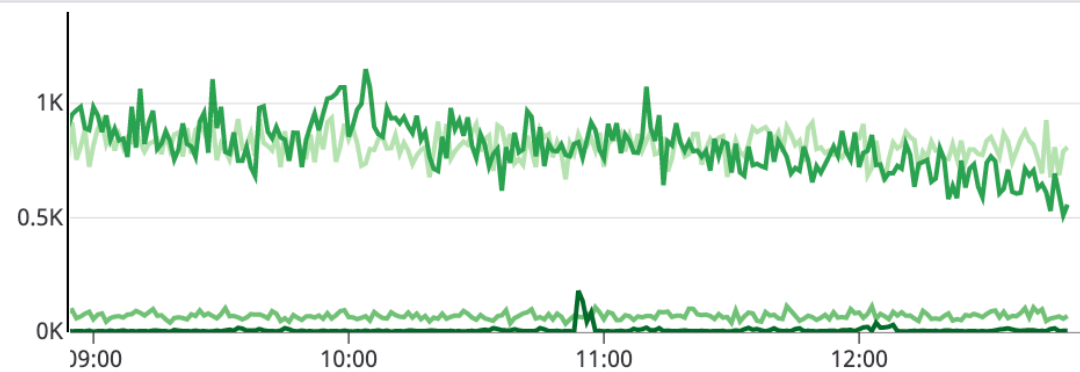
- Can be expensive
- Can kill a server
- Can leak private data or passwords
- Can be illegal (GDPR)

MONITORING

DASHBOARDS

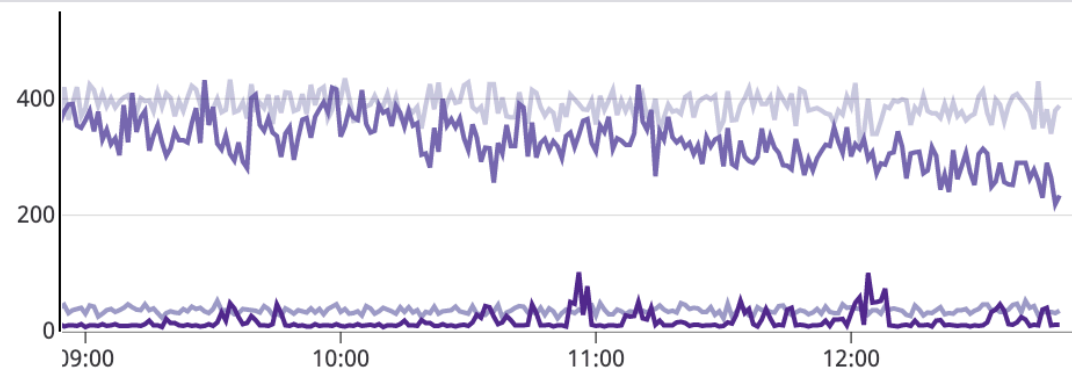
Get commands by name

4h



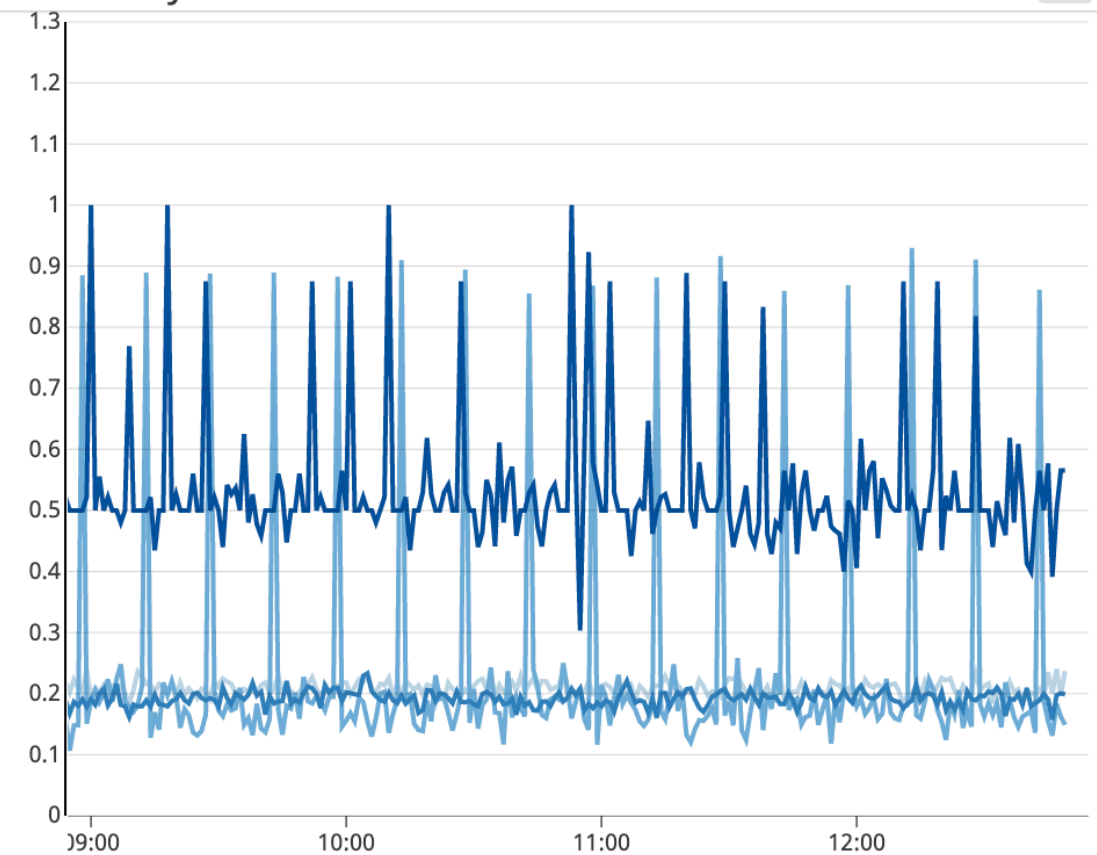
Set commands by name

4h



Hit rate by name

4h





WHAT TO MONITOR?

- System basics
 - Mem, Disk, etc.
- Service specific
 - API endpoints (usage, latency)
 - Error rates
 - Queues
 - ...

ALERTS

- Easy to add many alerts → Noise
- Two priorities
 - Immediate action
 - Can wait for business hours

MONITORING GOTCHAS

- Reactive → New metrics after event (known problems)
- Aggregated → Loose details (evil averages)
- Can be expensive  → Per host charging 
- Monitor everything approach

HEALTH CHECKS

- Each service should be responsible to expose its state
- The best people to create those are the same devs creating the service
- Unified across services

FAST FEEDBACK

GET myservice/health/status

```
[
  {
    "Name": "myDB",
    "Status": "TIMEOUT",
    "StatusTime": "2019-07-19T09:27:56.45Z"
  },
  {
    "Name": "kafka",
    "Status": "OK",
    "StatusTime": "2019-07-19T09:27:56.44Z"
  },
  {
    "Name": "redis",
    "Status": "OK",
    "StatusTime": "2019-07-19T09:27:56.44Z"
  }
]
```



Datadog APP 14:29

Triggered: [oc1-twotowers] CTRL External Health Status Monitor on host:oc1-checks,instance:twotowers_externals_ctrl_status,url:<http://twotowers.ctrl.marathon.l4lb.thisdcos.directory:3000/health/status/external>

CTRL External Health Status Alert. If the alert has been triggered by a 503 code, please expand the content text below the alert to see which of the services CTRL depends on has failed.

@slack-alerts-infra-prod and @opsgenie

Incorrect HTTP return code for url

<http://twotowers.ctrl.marathon.l4lb.thisdcos.directory:3000/health/status/external>.

Expected 1xx or 2xx or 3xx, got 503.

Content: [{"Name":"twodegrees_nz/monetizer","Status":"TIMEOUT","StatusTime":"2019-09-11T12:26:01.545159601Z"},

{"Name":"twodegrees_nz/siebel","Status":"TIMEOUT","StatusTime":"2019-09-11T12:26:01.544916199Z"}],{"

Tags

host:oc1-checks,

instance:twotowers_externals_ctrl_status,

_url:<http://twotowers.ctrl.marathon.l4lb.thisdcos.directory:3000/health/status/external>

dcos.directory:3000/health/status/external

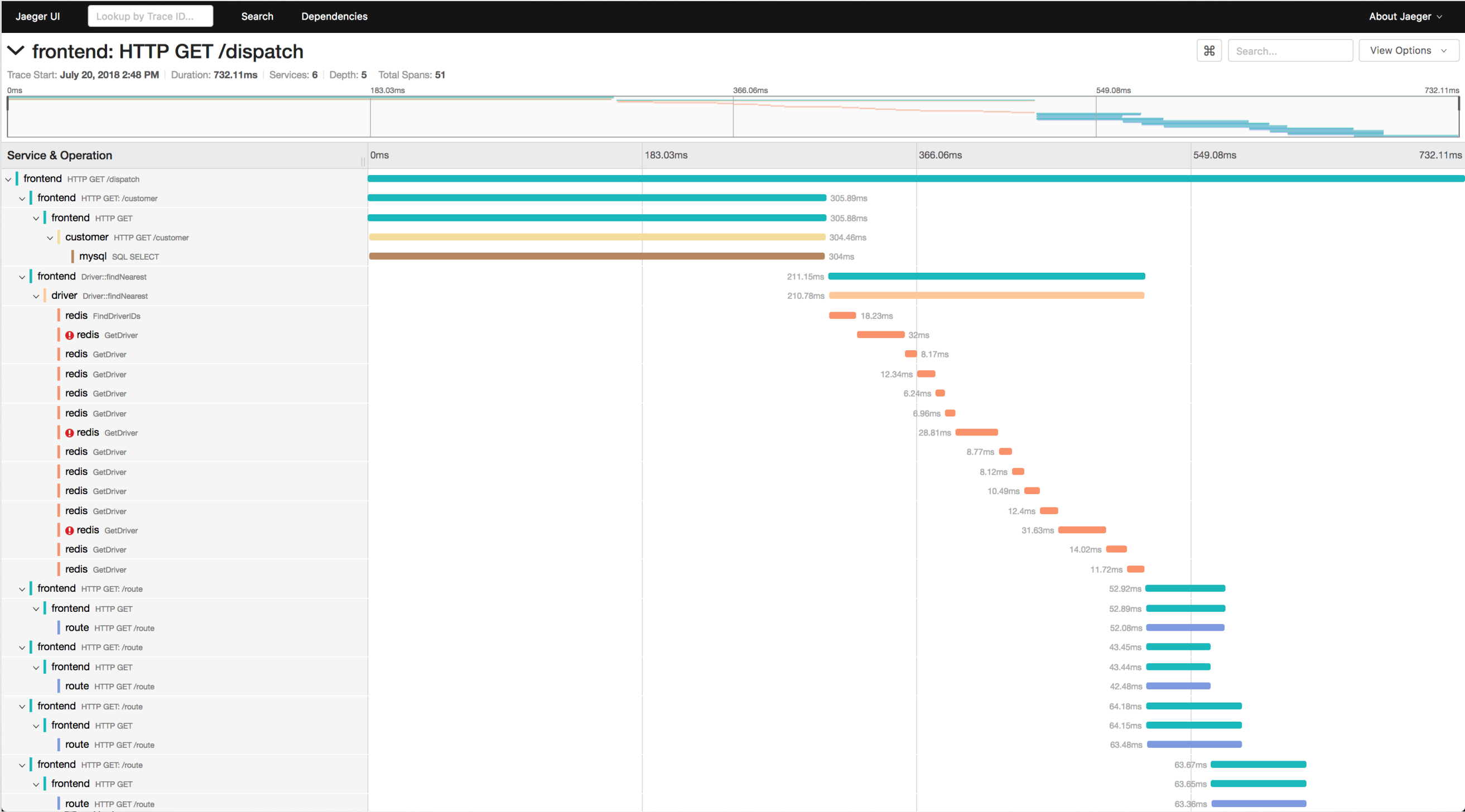
Notified

@slack-alerts-infra-prod, @opsgenie

APM

Application Performance Management

DISTRIBUTED TRACES



GROUPED EVENTS

- Events
 - Duration
 - Context
 - Sub traces
 - Span across services

CODE CHANGES

```
func myFunc(span opentracing.Span, ... ) {  
    ...  
    sp := opentracing.StartSpan(  
        "operation_name",  
        opentracing.ChildOf(span.Context()))  
    defer sp.Finish()  
  
    // read from some mem cache  
    found := false  
    log.Bool("cache-miss", !found)  
  
    ...  
  
    if err != nil {  
        ...  
        sp.SetItem("error", err)  
    }  
    ...  
}
```


APM GOTCHAS

- Painful to add it everywhere (depends on your code)
- Can be expensive (sampling)
- Difficult across boundaries

SO ????

- The needs change as project evolves
- You need to have a mix of solutions
 - Might involve manual correlation
- The trend is a mix of everything
 - New vendors with new solutions

OBSERVABILITY

- It is a path
- You will be testing in production
 - Be prepared!
- Forget the hype
- Serve your users (and your devs)