Application Monitoring & Measurement

Story time

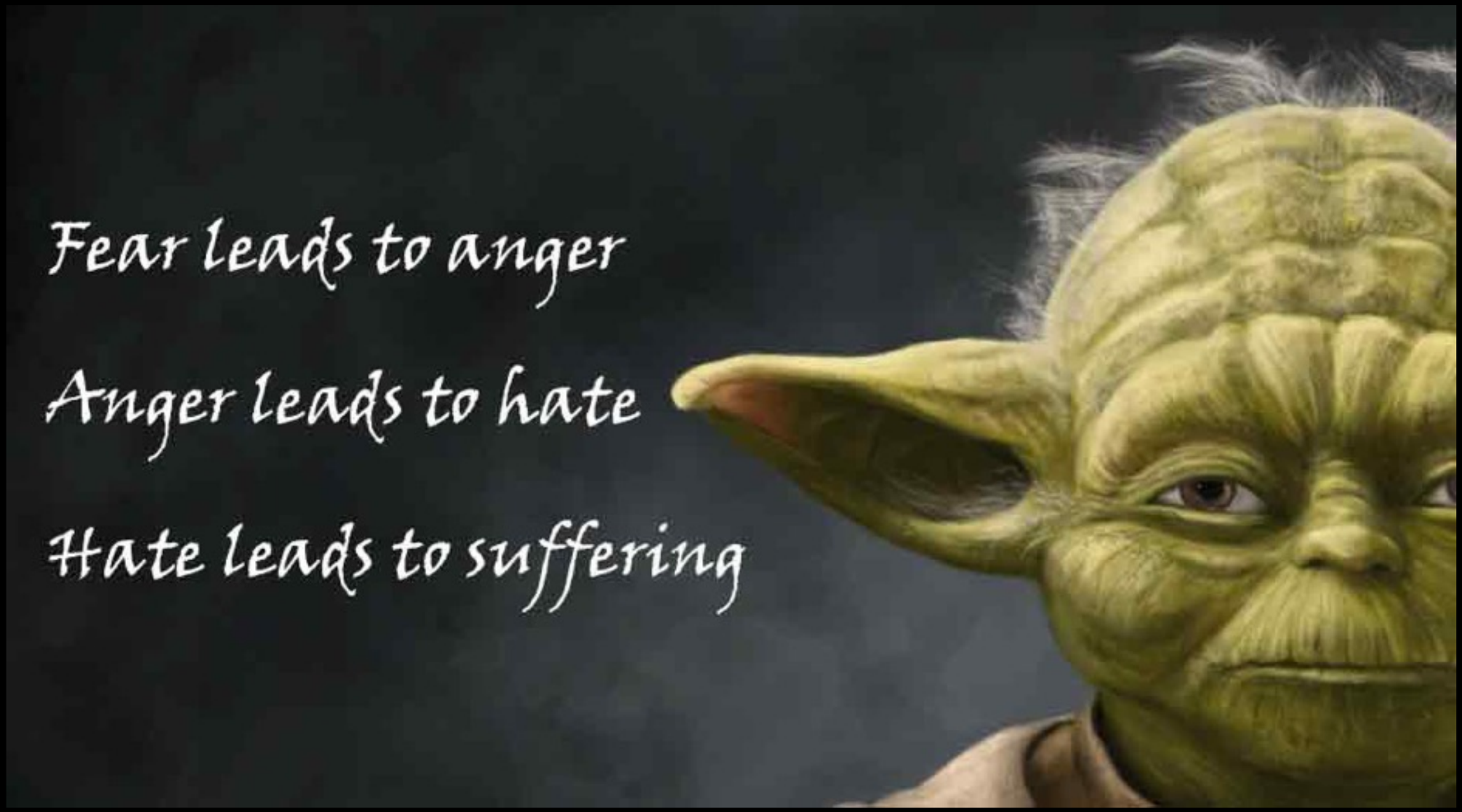
In Your Future…

Like a Boss

Life is beautiful…

…then you get Users

And you don’t know what’s wrong



Knowledge is power

- Why does Customer X have 3% more timeouts at 4:00am?
- How many customers have >10% error response from Web Service Y (in the last 5 min, no the last 30 days!)?
    - Can I get notified when this happens?
- Why was Customer Z's API POST slow?
- We used tools you built to quickly debug an issue, and verify the fix without asking you any questions

Essential Data

App Events

What's happening in your application?

Classify messages with Log Levels.

`FATAL`: Severe error event, most likely a crash. Alert!

`ERROR`: Error / Exception. Alert.

`WARN`: Recoverable issue, less severe. Alert?

`INFO`: Informative, e.g. Server coming back online.

`DEBUG`: Should be everything you need to debug an issue.

App Events

```
$log = new Logger(__CLASS__);
try {…}
catch (Exception $e) {
  $log->error(“Hodor! {$e}”);
} finally {
  # Kidding, don’t do this…
  $log->fatal(“Oh hai, on-call Ops friend! Conrad for President!!!”);
}
####
timestamp loglevel=ERROR Some_Class
“Hodor! …useful stack trace telling you what line of code blew up…”
```

Web Server

Web servers provide useful access / error data

```
127.0.0.1 - frank [10/Oct/2000:13:55:36
-0700] "GET /apache_pb.gif HTTP/1.0" 200
2326

[Wed Oct 11 14:32:52 2000] [error]
[client 127.0.0.1] client denied by
server configuration: /export/home/live/
ap/htdocs/test
```

App Performance

How’s your application performing?

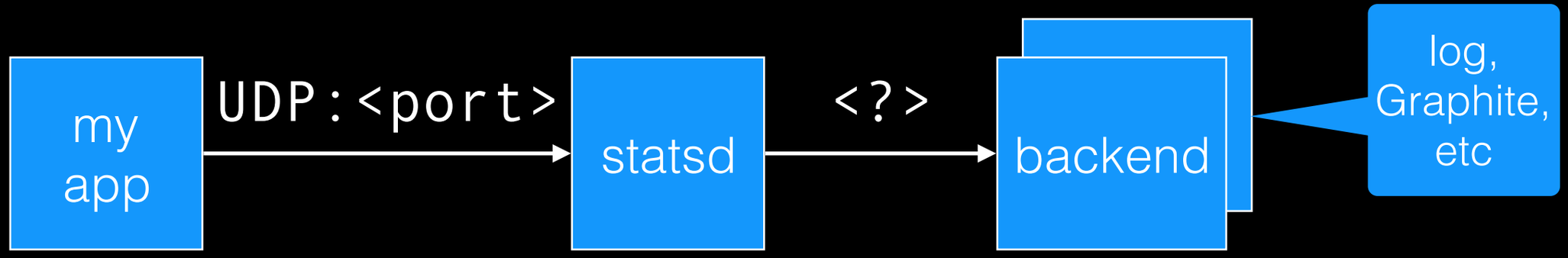
Timers - Run time of key (all?) functions in call stack.

Counters - Cache hits / misses, Batch sizes, whatever.

```
{"total":"500|ms", "some_db_func":"150|ms",
"some_util_func":"75;100,85|ms", “a_counter":"42|
c", "cache_hits":"2|c", "batch_size":"900|c"}
```

App Performance

`Statsd` example



Moving on…

Largely a solved problem

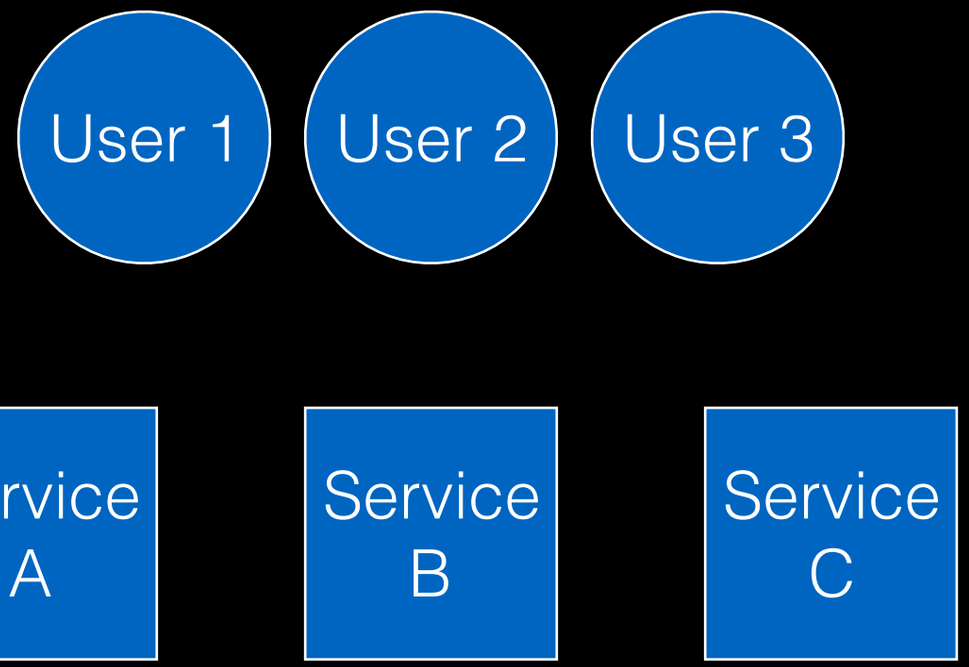
Some effort to instrument performance

Stats aggregation daemons are shiny

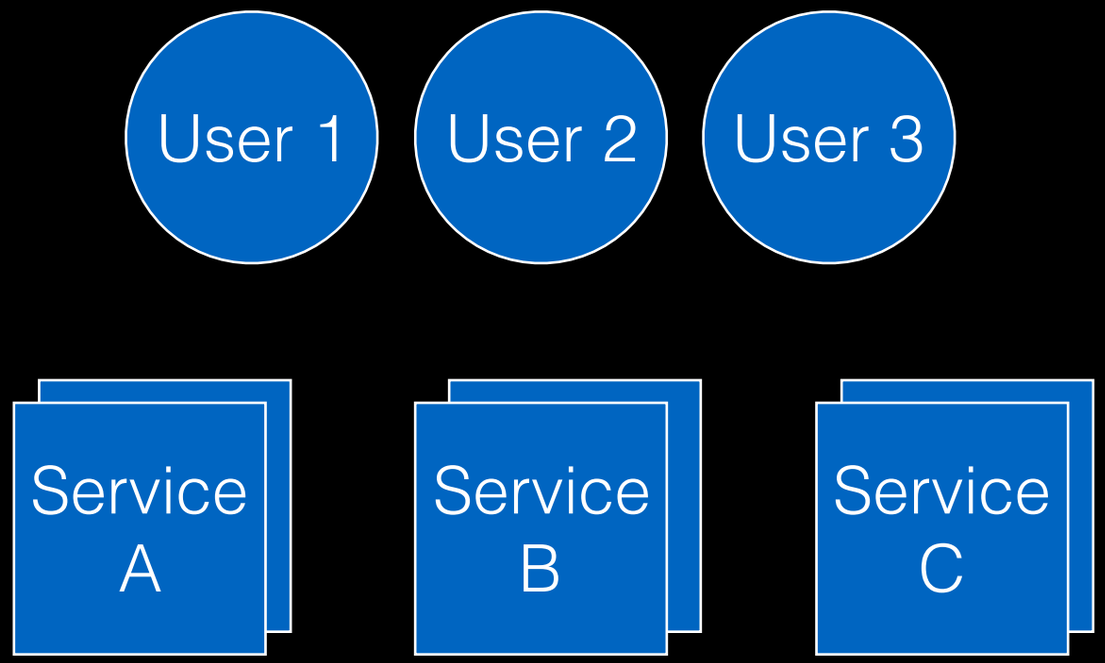
Valuable information is waiting to be discovered

Build it to be measured from the start

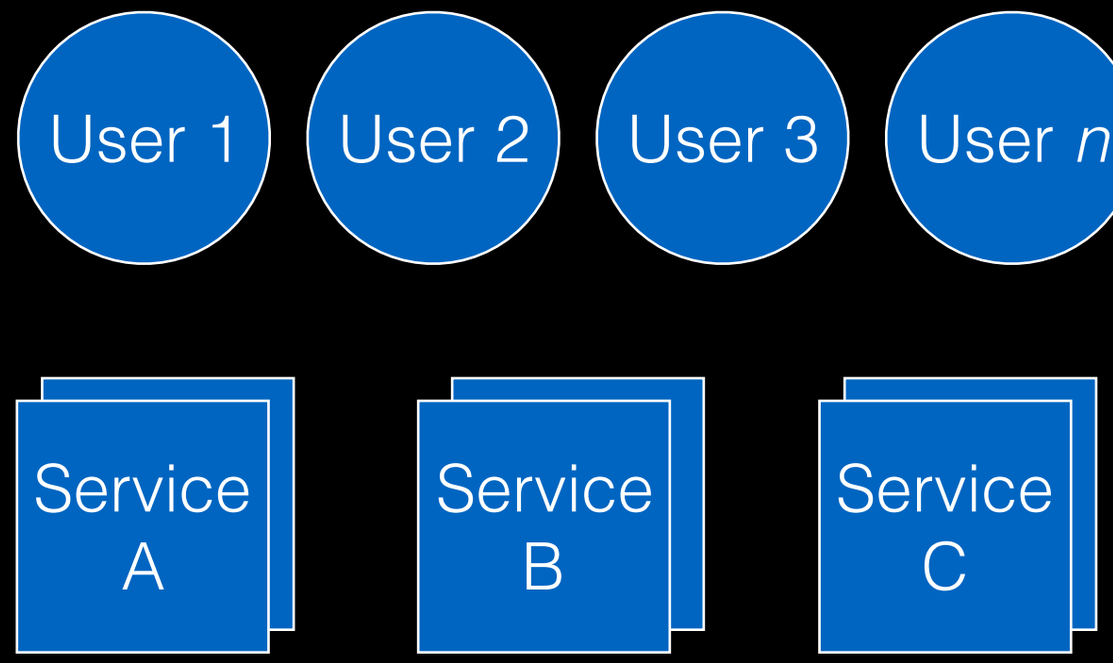
Mapped Diagnostic Context




```
"User did something."
"User did something else."
"User did something."
"User did something."
"User is happy."
"User did something else."
"Software went boom. User is unhappy."
"User is happy."
```




```
"User 1 did something."
"User, 1, did something else."
"User 3 did something."
"User 2 did something."
"User [user-2] is happy."
"User, 3, did something else."
"Software went boom. User is unhappy."
"User [user-3] is happy."
```




```
user=1 ip=1.0.0.0 Service_A "User did something."
user=1 ip=1.0.0.0 Service_B "User did something else."
user=3 ip=3.0.0.0 Service_A "User did something."
user=2 ip=2.0.0.0 Service_A "User did something."
user=2 ip=2.0.0.0 Service_C "User is happy."
user=3 ip=3.0.0.0 Service_B "User did something else."
user=1 ip=1.0.0.0 Service_C "Software went boom…”
user=3 ip=3.0.0.0 Service_C "User is happy."
```

Moving on…

DRY & Consistent

What could be in MDC?

timestamp (may get this other ways)
thread / process id
customer / user identifiers like ID, IP address
host, event author (may get this other ways)

OK, now what?

Time to `curl`, `grep`, `awk` and mangle those log files!

```
 $2 <= 0.1 {na=na+1} ($2 > 0.1) && ($2 <= 0.2) {nb = nb+1}
($2 > 0.2) && ($2 <= 0.3) {nc = nc+1} ($2 > 0.3) && ($2 <=
0.4) {nd = nd+1} ($2 > 0.4) && ($2 <= 0.5) {ne = ne+1} ($2
 > 0.5) && ($2 <= 0.6) {nf = nf+1} ($2 > 0.6) && ($2 <= 0.7)
   {ng = ng+1} ($2 > 0.7) && ($2 <= 0.8) {nh = nh+1} ($2 >
    0.8) && ($2 <= 0.9) {ni = ni+1} ($2 > 0.9) {nj = nj+1}
    END {print na, nb, nc, nd, ne, nf, ng, nh, ni, nj, NR}
```

Kidding

CEO – Application Management

localhost:8000/en-US/app/appmgmt/ceo

splunk> Application Management v2.0

Administrator | App | Manager | Alerts | Jobs | Logout

About | IT Operations | Executive Dashboards | Reports | Help | About

CEO | Actions

RT Transaction Volume real-time

Real-time transaction volume across all tiers.

18

RT Shopping Cart Value real-time

Real-time average shopping cart values.

$118.00

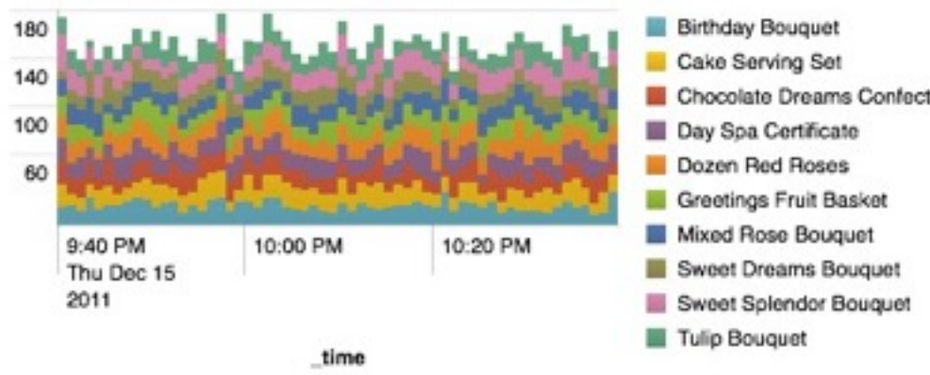
RT Visitor Count real-time

Real-time concurrent users browsing website.

60

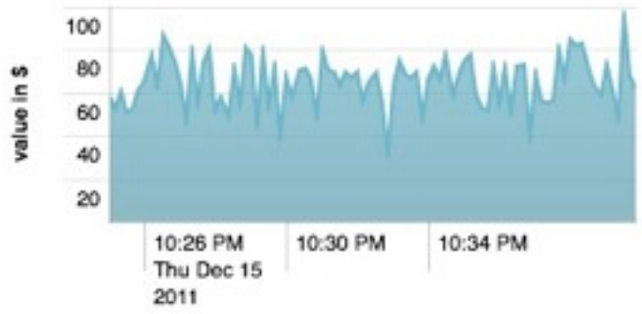
Visitor Location 3m ago

Top Items Sold 5m ago



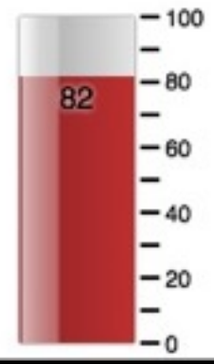
Abandoned Baskets 5m ago

Average value of abandoned shopping carts on website via logout or expired sessions over the last 60 minutes.



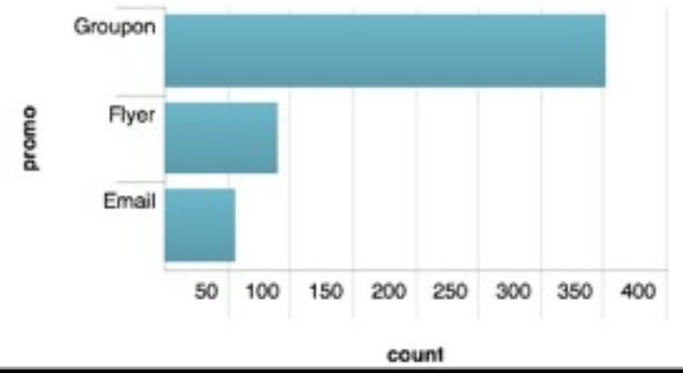
% Coupon Usage real-time

Real-time percentage of shoppers using coupons vs paying full price.

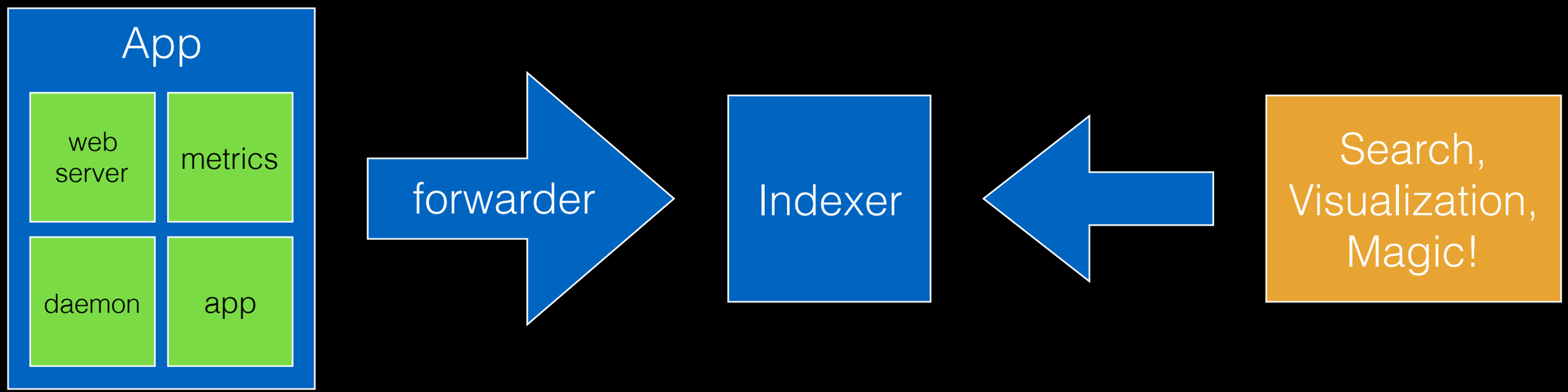


Top Promotions 3m ago

Promotional program popularity over the past 24 hours.



Splunk Overview



Resources

Apache Logging (Log4x)

Measure Everything (Etsy post & Statsd)

ELK (Elasticsearch, Logstash, Kibana)

New Relic

Splunk