A New Hope

Monitoring data.fao.org

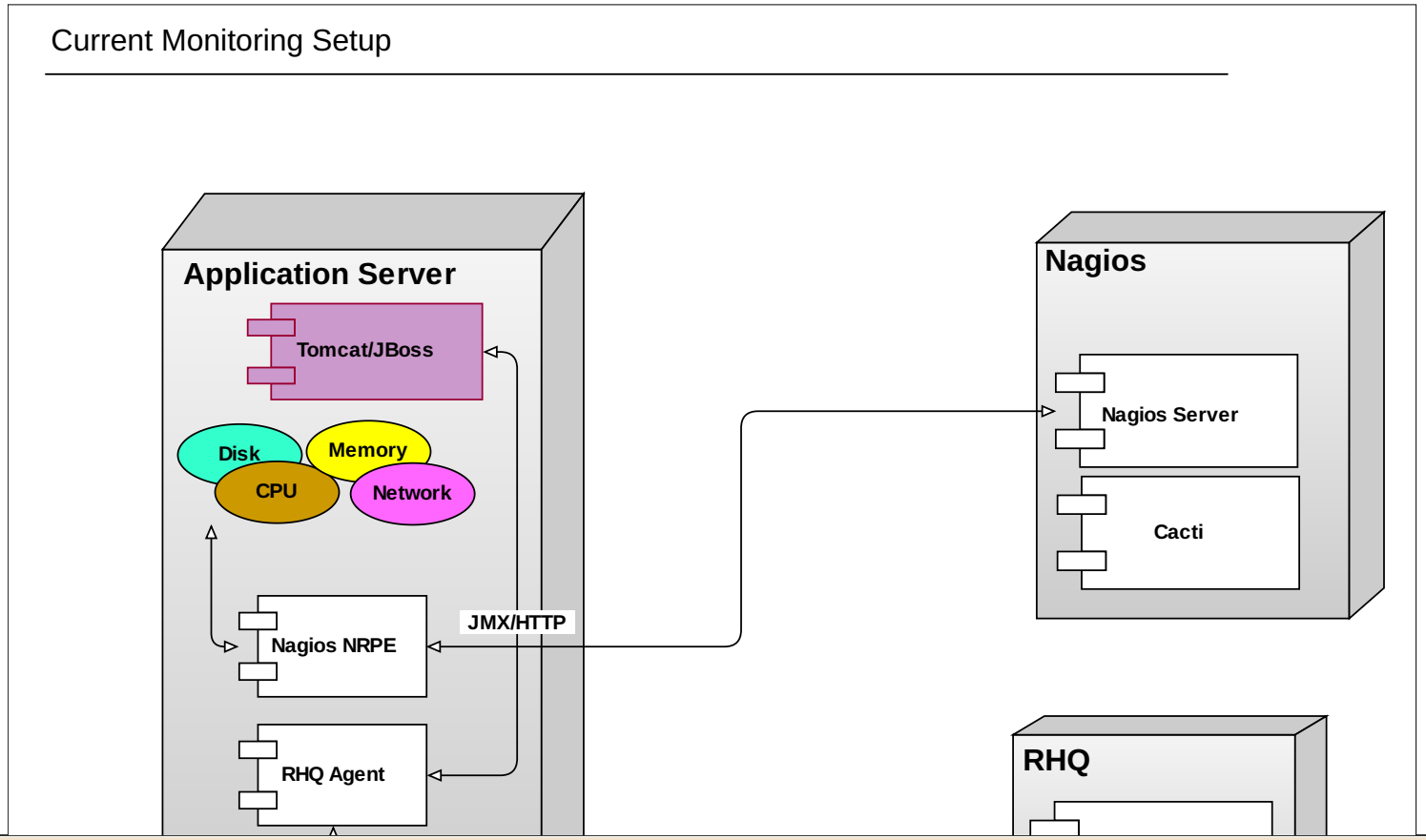
Overview

- What we have now
- Why it sucks
- What might be better
- What you can do

We have a complicated application stack

systematic monitoring is critical to untangling it

We currently use Nagios and RHQ



Downsides

- Graphing capabilities in Nagios+Cacti very limited
- Nagios is a pain to configure, even with Chef
- RHQ is powerful but very inflexible

More RHQ Issues

- The RHQ agent consumes non-trivial amount of RAM and CPU
- Extending it requires you to write Java class and a Maven pom.xml
- Small user community despite being a relatively mature project
- No easy way to access data in RHQ to create dashboards

That Makes me Want to...

A New Hope

- A more flexible monitoring system
- We will continue to use Nagios for alerting
- The future use of RHQ is uncertain

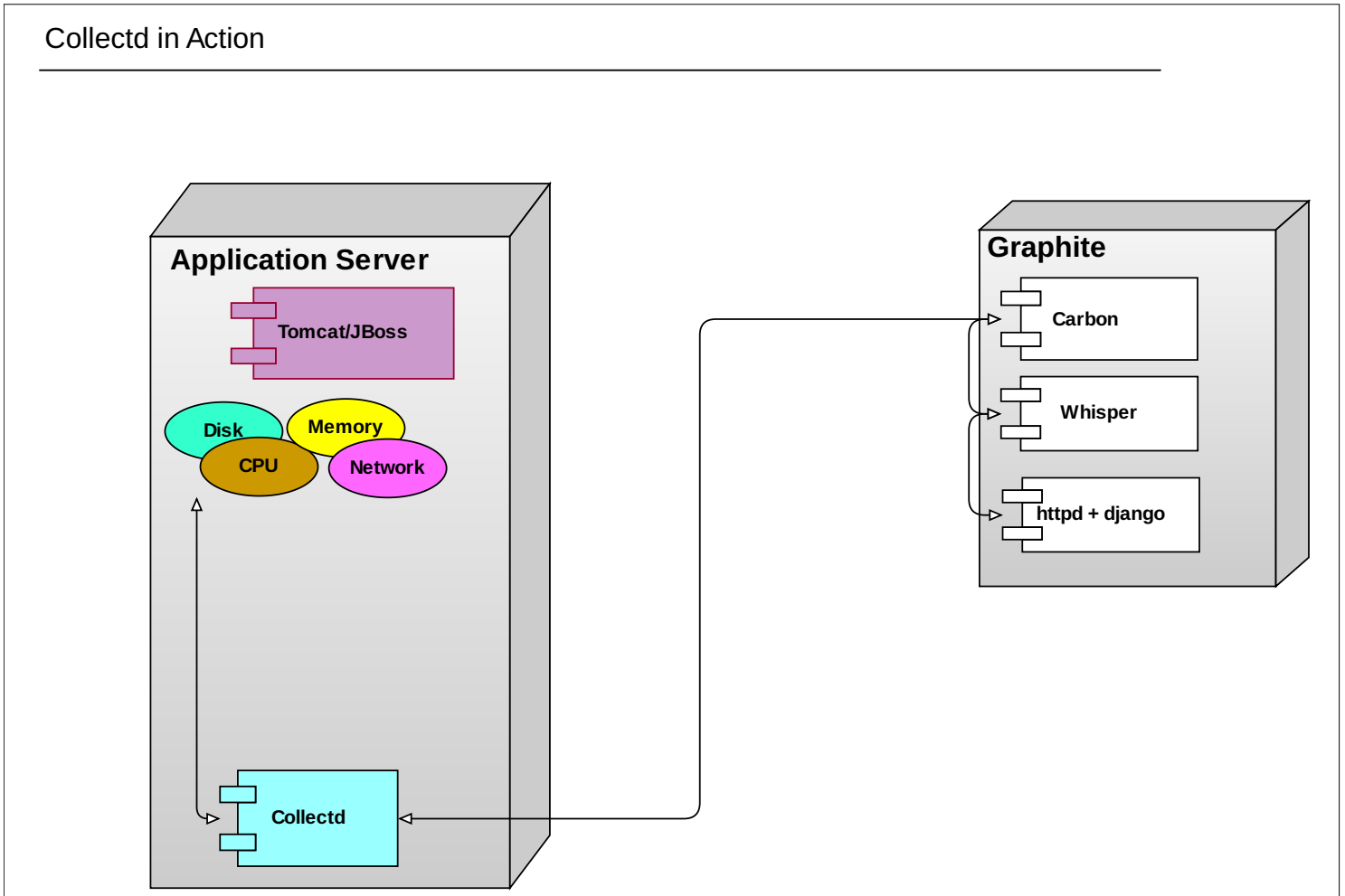
The DevOps Community has (mostly) converged on a toolchain for monitoring

What is a metric?

- a name
- a value
- a timestamp, typically the UNIX epoch time

stats.haproxy.data_fao_org.request_duration 330 74857843

Collectd for system metrics



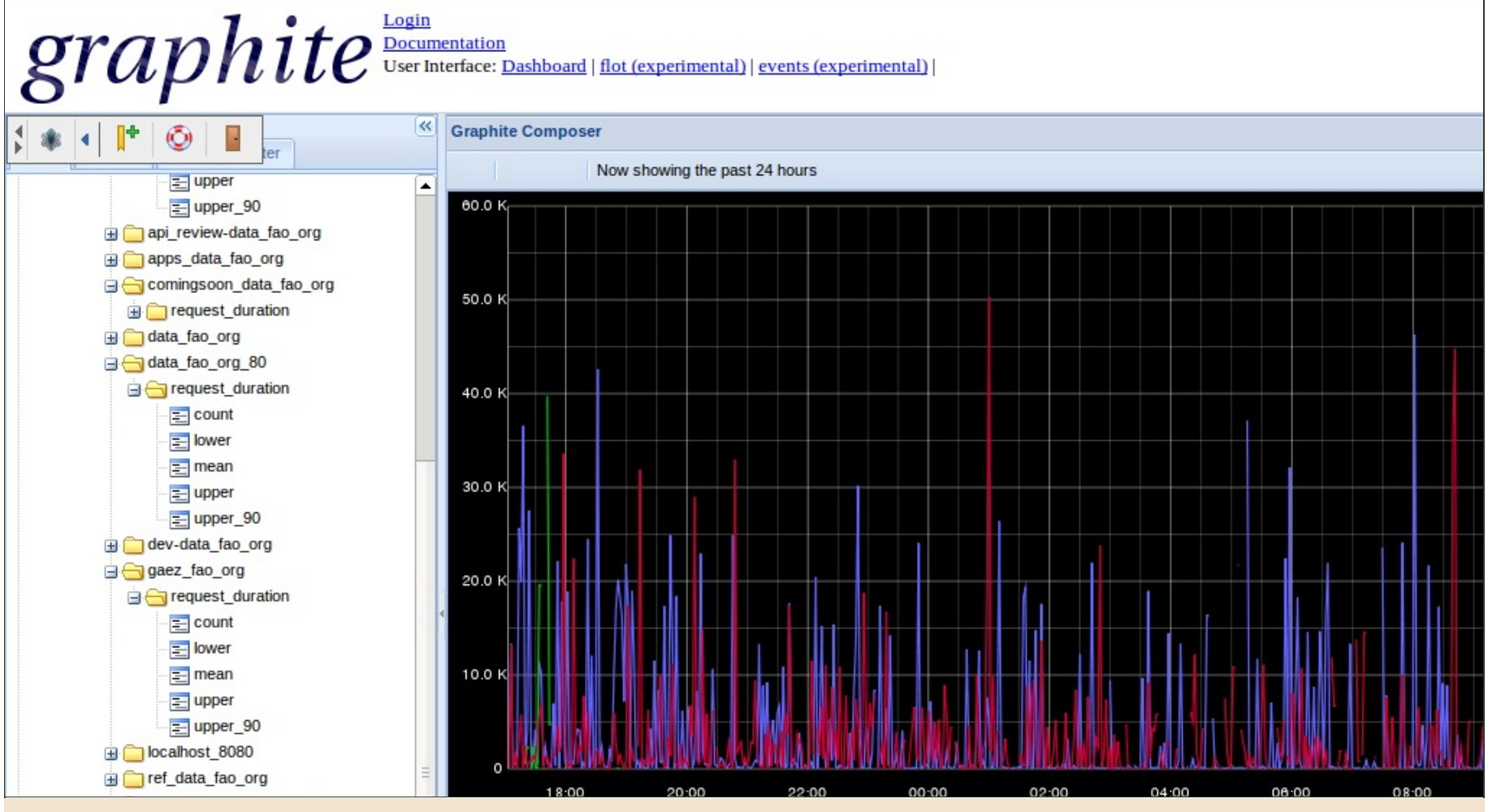
Collectd is good

- lightweight C daemon
- monitors continuously, not once every 5 minutes
- Best for OS-level metrics such as CPU, Disk, Memory, etc.

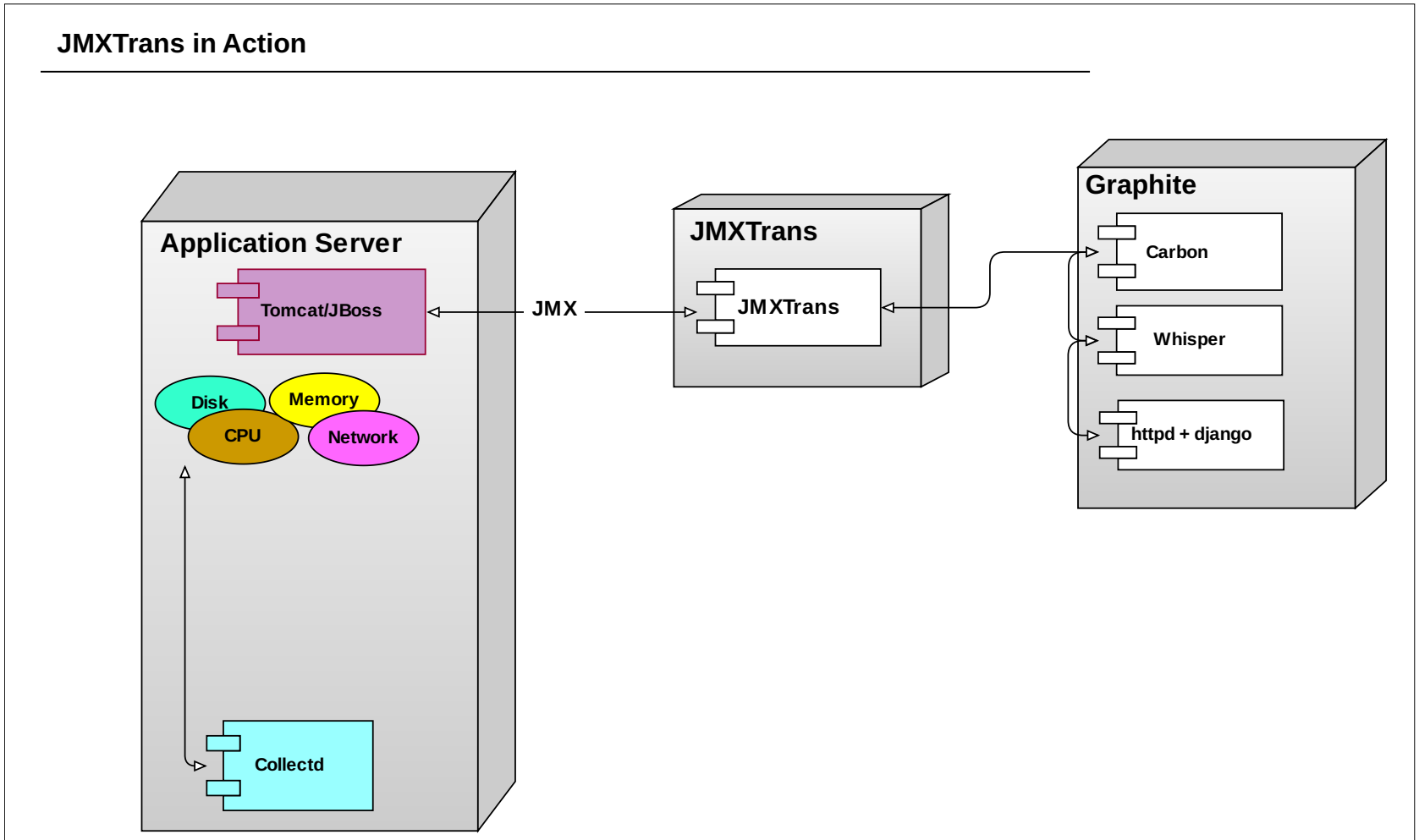
What's Graphite?

- Time Series Database (Whisper)
- Rendering Engine
- Dashboard (Graphite-Web)
- data relay and aggregation (Carbon)

Here is a taste



Let's take a look at that JVM



JMXTrans

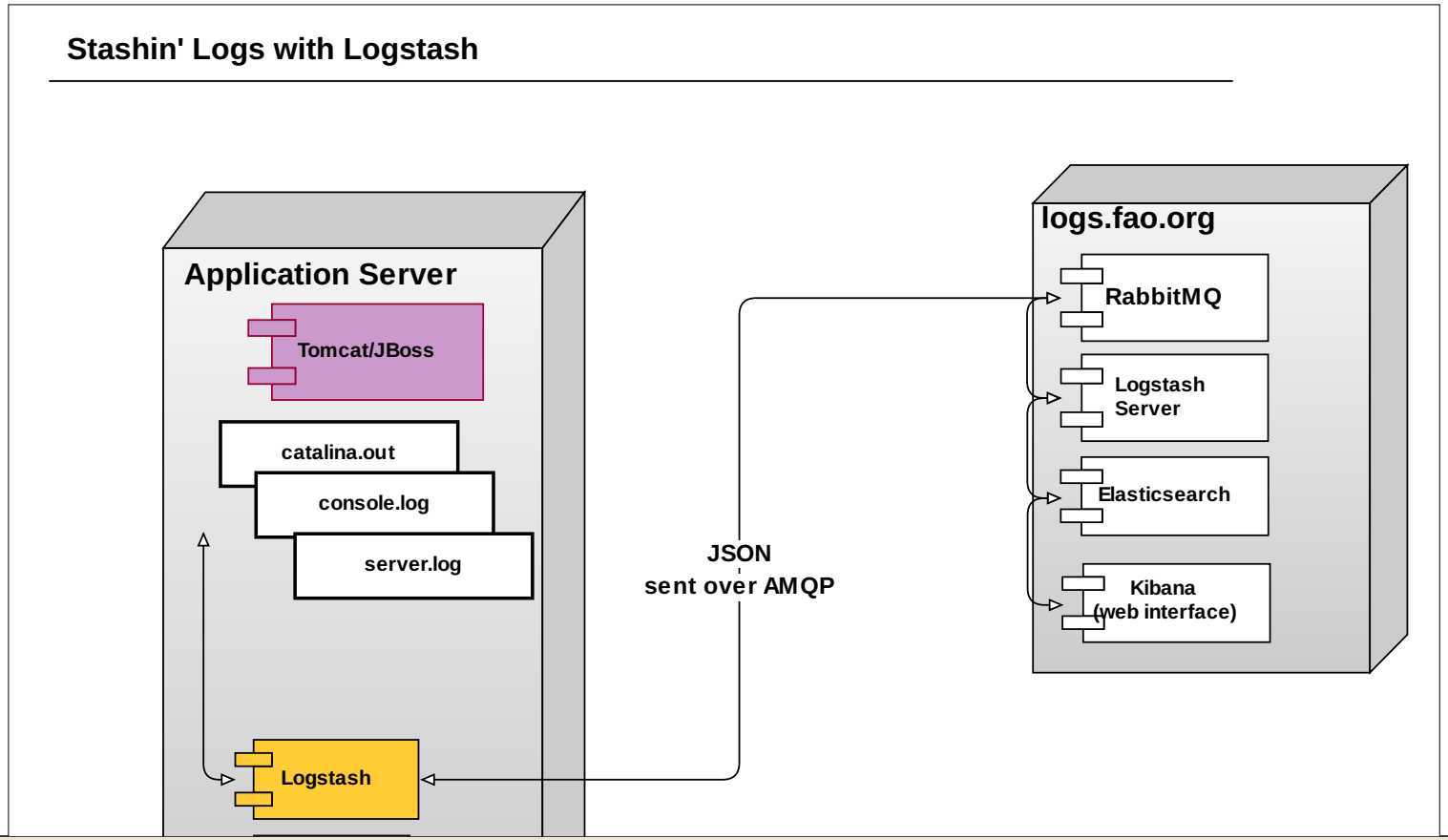
- Is just a connector for transporting JMX data
- No agent involved

What about those logs?

We don't only care about metrics, we also care about important events.

Would be nice to scrape metrics from logs though.

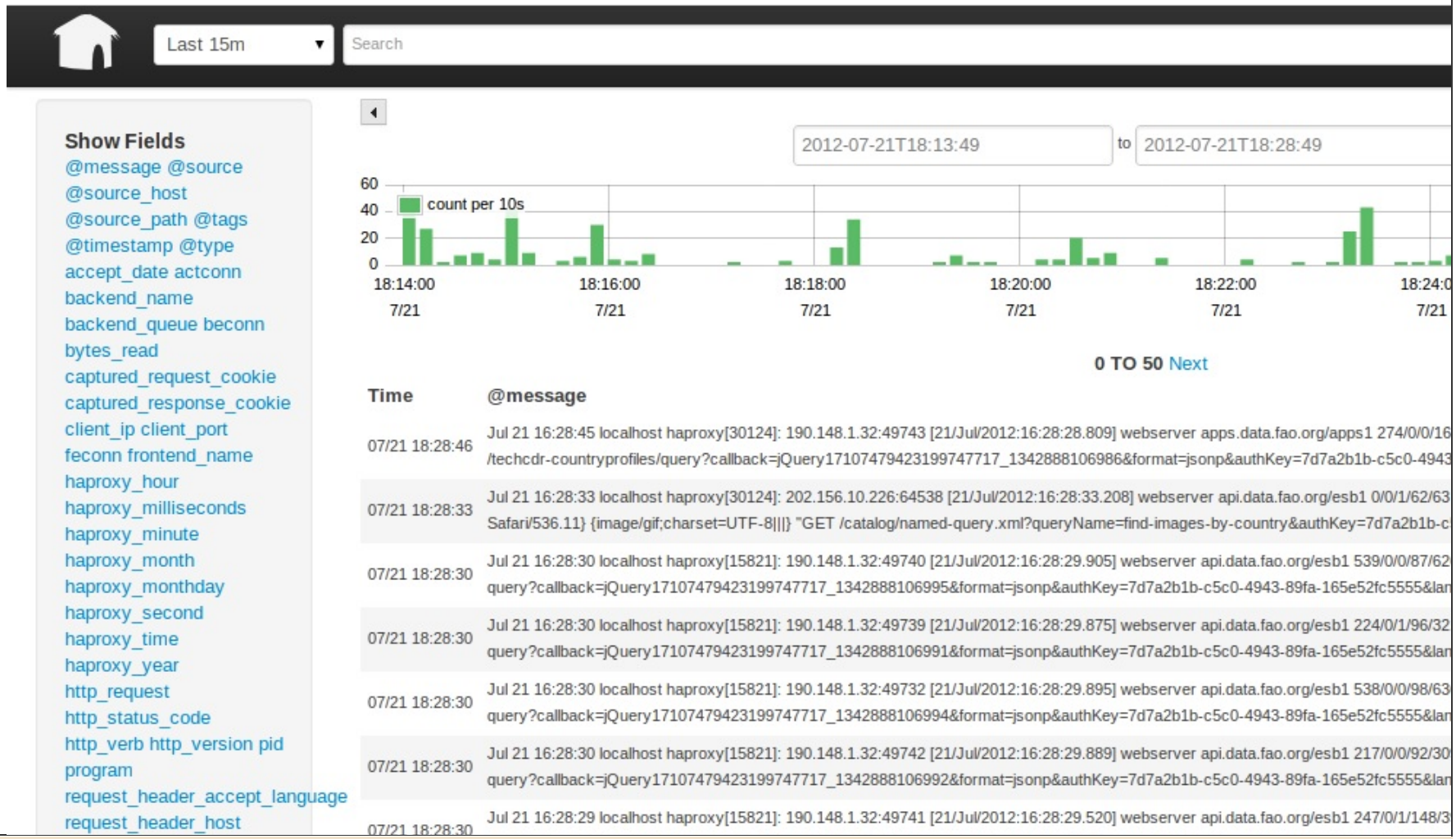
Show me the graphic already!



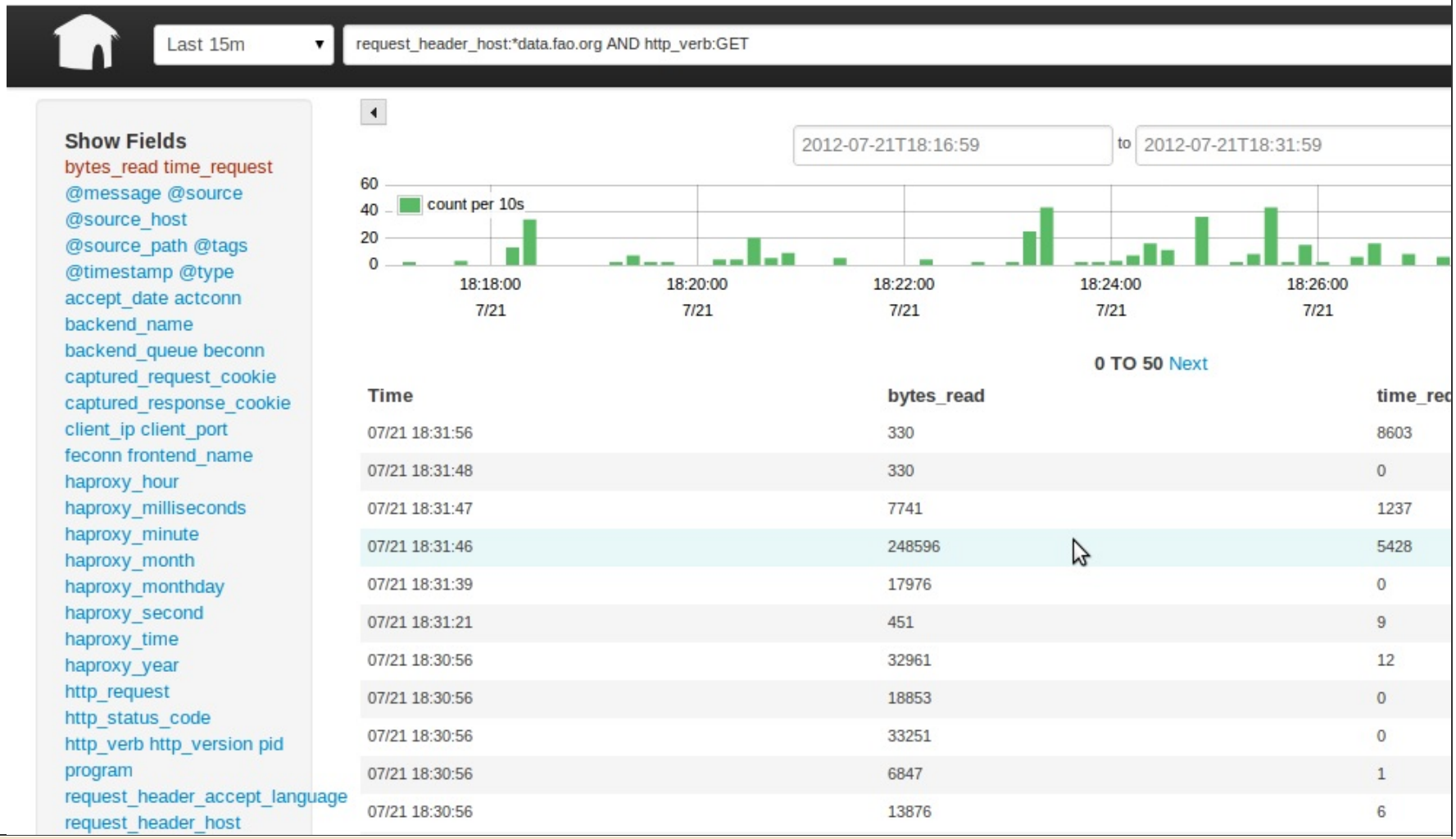
Logstash can more than just ship logs

- Win
  - index by field
  - shape data
  - add new fields and tags to entries
  - Elasticsearch backend is awwwes0me
- Con - The agent isn't so light on resource usage

Logs don't have to be Ugly



We can filter the data



UNIX Tail in your browser

request_header_host:*data.fao.org AND http_verb:GET

Time	bytes_read	time_request
07/21 18:40:56	32961	7
07/21 18:40:56	33251	0
07/21 18:40:56	6847	0
07/21 18:40:56	18853	0
07/21 18:40:56	30254	186
07/21 18:40:56	1291	32
07/21 18:40:56	3029	19
07/21 18:40:56	13876	25
07/21 18:40:56	2876	16
07/21 18:40:56	7256	0
07/21 18:40:56	628	25
07/21 18:40:55	2610	987
07/21 18:40:55	3598	950

Elasticsearch Rocks

- We can use Lucene Parser Syntax to construct queries
- Watch out though, don't use quotes, the 1st example here works, the second doesn't

> *status_code:40* AND request_header_host:*fao_org*

> *status_code:"40*" AND request_header_host:"*fao_org"*

But this is not really enough

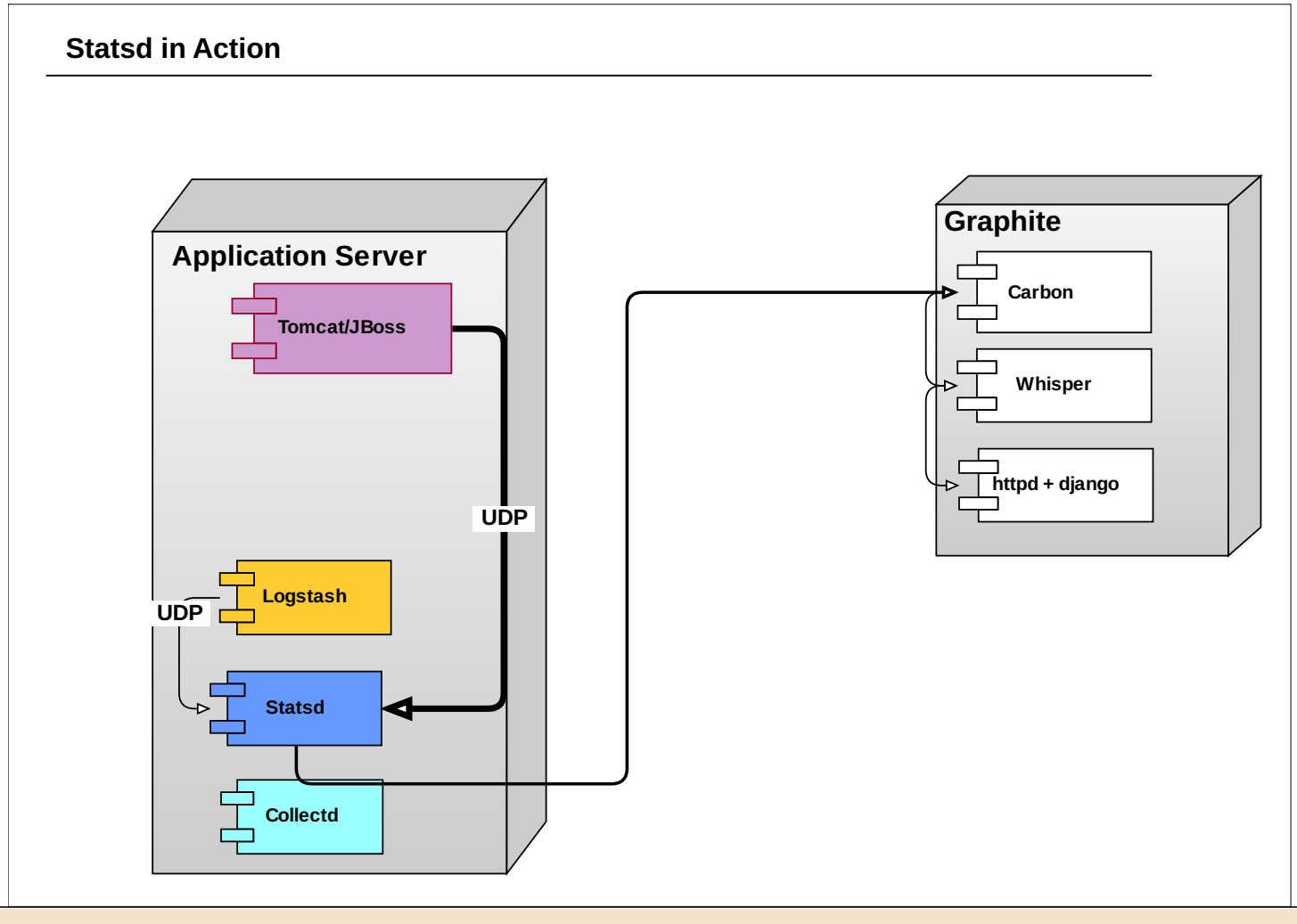
Ideally, your application should be instrumented from the inside

No one knows your code better than you

Use the Force, Luke

Use Statsd

Show me a graphic



Enter Statsd

- a small local daemon that your application send metrics to over UDP
- I use Pete Fritchman's Ruby implementation
- Statsd ships the metrics it receives to graphite
- Types of metrics
  - gauges

So about the code

Here is some sample java code

Java # embed raw

```java
* Example usage:

StatsdClient client = new StatsdClient("statsd.example.com", 8125);
// increment by 1
client.increment("foo.bar.baz");
// increment by 10
client.increment("foo.bar.baz", 10);
// sample rate
client.increment("foo.bar.baz", 10, .1);
// increment multiple keys by 1
client.increment("foo.bar.baz", "foo.bar.boo", "foo.baz.bar");
// increment multiple keys by 10 -- yeah, it's "backwards"
client.increment(10, "foo.bar.baz", "foo.bar.boo", "foo.baz.bar");
// multiple keys with a sample rate
client.increment(10, .1, "foo.bar.baz", "foo.bar.boo", "foo.baz.bar");
```

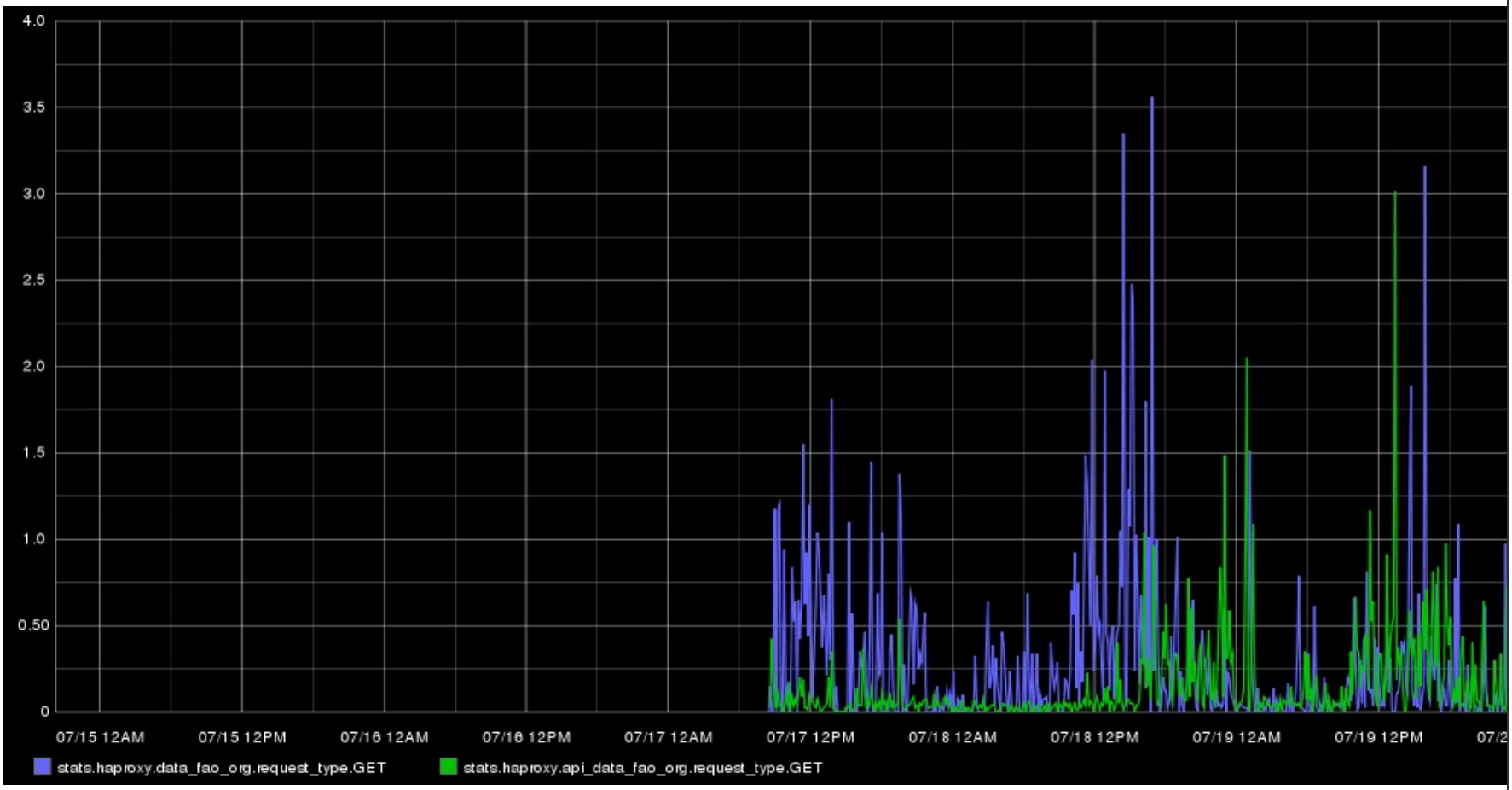
delete this gist

Metrics, Metrics Everywhere

Coda Hale gave an excellent talk about how his team at yammer uses metrics

He also created an excellent java library that you can use together with statsd

Now, Graphite Demo



You will get for free

- System-level monitoring with collectd
- JMX monitoring w/ JMXtrans
- log aggregation on request via logstash+elasticsearch+kibana
- All those data points in graphite

But if you are serious about performance . . .

. . . You will

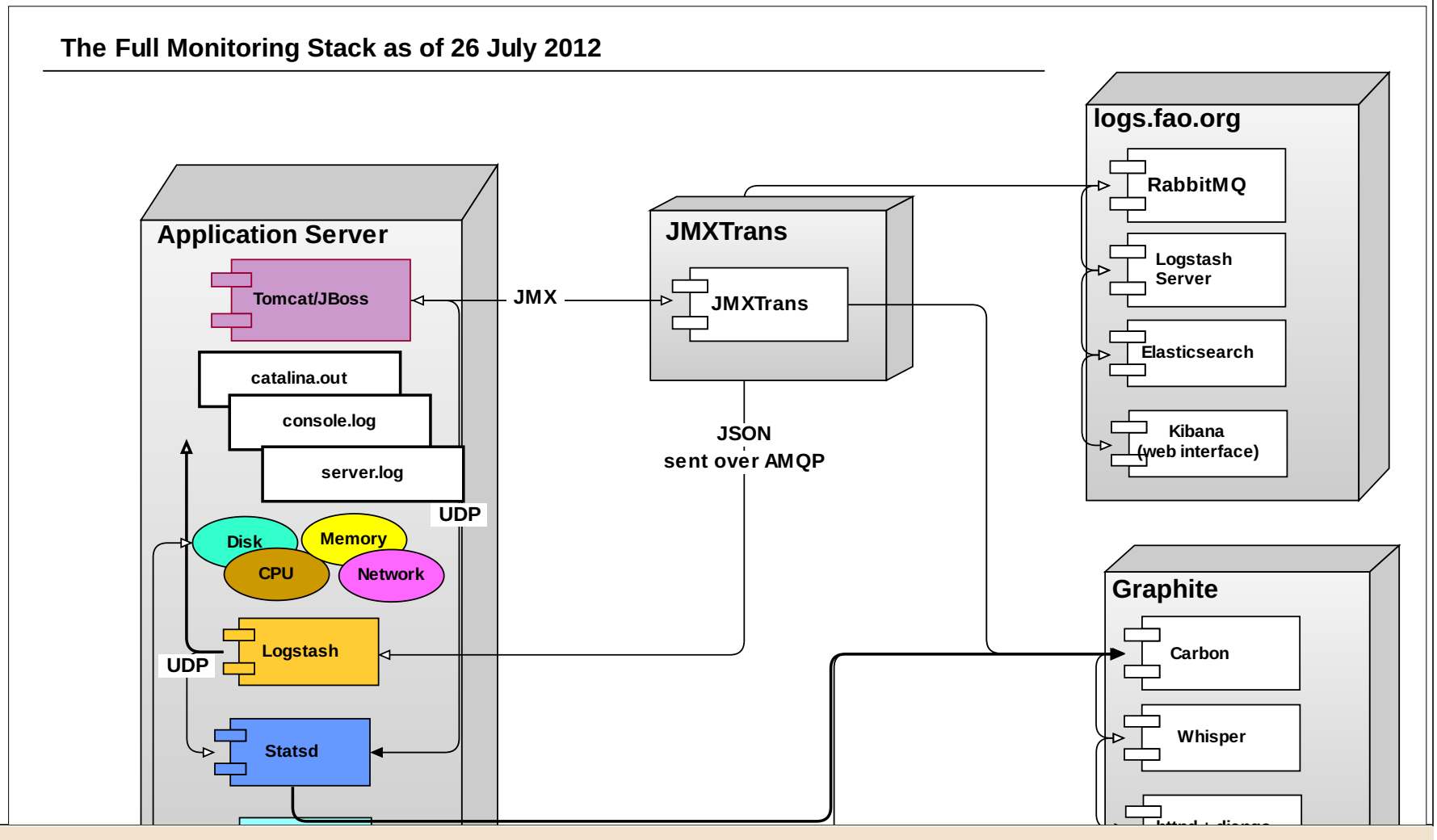
- instrument your code with the statsd java client or Coda Hale's metrics library
- Create custom graphs in graphite

- Solve performance problems the way real engineers do, with *data*

Questions?

Ask away

The Full Stack



Further Resources

- Logstash
- Collectd
- statsd and Pete Fritchman's ruby-statsd
- graphite
- Coda Hale's excellent talk and library java library

Be sure to listen to the FeedFightShow!