log everything with logstash and elasticsearch

Peter Hoffmann

About Me

- Follow me on Twitter @peterhoffmann
- Get the Slides github.com/hoffmann/
- Visit my Blog peter-hoffmann.com
- Software Developer at blueyonder

log everything

When your application grows beyond one machine you need a central space to log, monitor and analyze what is going on. Logstash and elasticsearch store your logs in a structured way. Kibana is a web fronted to search and aggregate your logs.

Logging Overview

Producer	Transport	Route/Filter	Storage	Analysis
Frontend	syslog	**logstash**	elasticsearch	**kibana**
API/Backend	**gelf**	graylog2-server		graylog2-web-interface
Auth Server	redis			elasticsearch head
Database	rabbitmq			python/pyes
Operation System	logfile/grok			

Logging Chain

Transport | Route/Filter | Store | Analyze

Gelf | **Logstash** | **Elasticsearch** | **Kibana**

Transport: GELF

- Graylog Extended Logging Format
- JSON over UDP (nonblocking)
- Compression and Chunking
- Structured Extensible Key/Value Message Format
- Graypy Python Handler (clients for all other major programming languages available too)

Required Fields

```
{
  "version": "1.1",
  "host": "example.org",
  "short_message": „Hello Logstash",
  "full_message": "Backtrace here\n\nmore stuff",
  "timestamp": 1385053862.3072,
  "level": 7,
  "_facility": "root.logger",
  „_request_id": "XX-",
  "_env": "bar"
}
```

Graypy

- Gelf Handler for Standardlib Logging
- `pip install graypy`

```
import logging
import graypy

log = logging.getLogger(__name__)
log.setLevel(logging.DEBUG)

handler = graypy.GELFHandler('127.0.0.1', 12201)
log.addHandler(handler)

log.debug('Hello Logstash')
```

Route/Filter: Logstash

- Logstash is a tool for receiving, processing and outputting logs.

- written in JRuby runs in the JVM
- based on Pipes/Filter Pattern
- Jordan Sissel creator of logstash is now employed by the elastic search company

Running Logstash

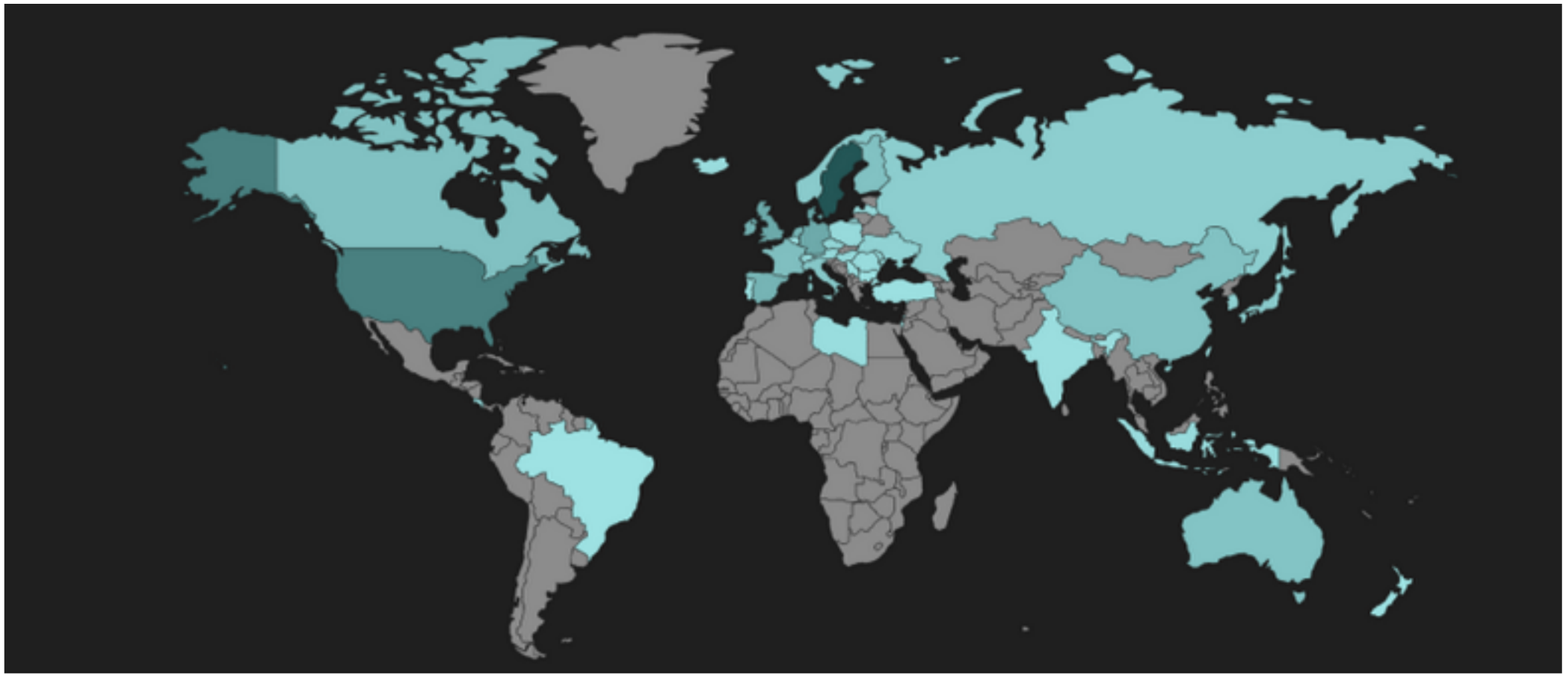
```
input {
  gelf {
      port => 12201
  }
}
filter {
   if [loglevel] == "debug" {
    drop { }
  }
}
output {
  elasticsearch {
    host => localhost
  }
}
```

bin/logstash -f mylogstash.conf

Analysis: Kibana

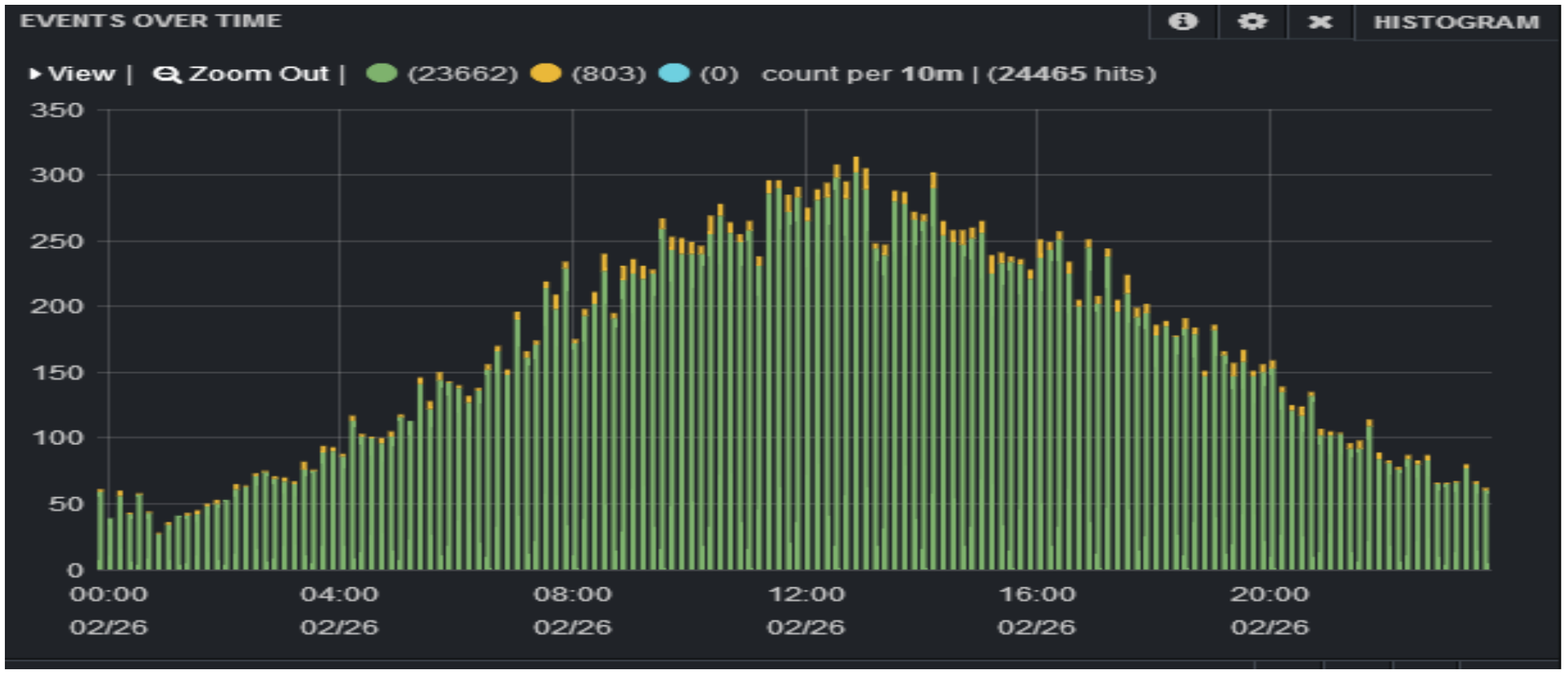
- Single page Javascript application to search and analyze time based data in elasticsearch
- rich set of visualizations
- powerful search syntax
- create and share dashboards

Bettermap



- **Bettermap** uses geographic coordinates to create clusters of markers on map

Histogram

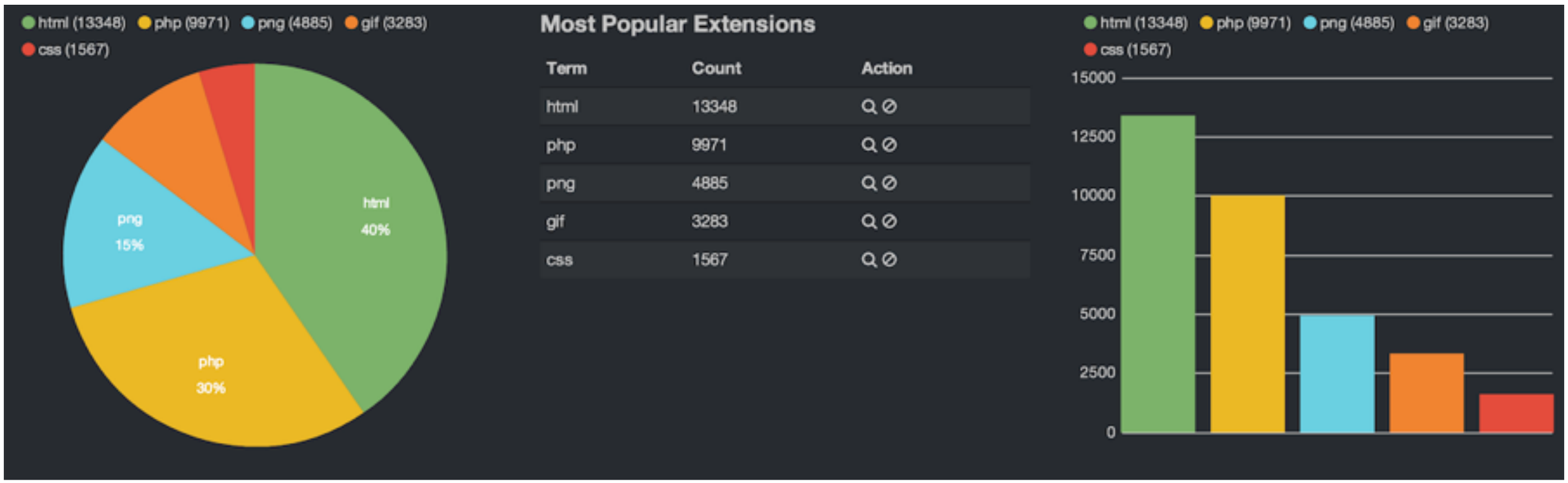


- **histogram** is for display of time charts. It displays event counts, mean, min, max and total of numeric fields.

Sparklines

- **sparklines** panel shows tiny time charts. The purpose of these is to show the shape of the time series in a compact manner.

Terms



- **terms** panel shows table, bar chart or pie chart based on the results of an Elasticsearch terms facet.

Logging Patterns

- Adding Context
- RequestID
- CorrelationID
- Finger Crossed Handler

Adding Context I

```
import logging
import graypy

log = logging.getLogger(__name__)
log.setLevel(logging.DEBUG)

handler = graypy.GELFHandler('127.0.0.1', 12201)
log.addHandler(handler)

log.debug('Hello Logstash', extra={'env': 'bar'})
```

Adding Context II

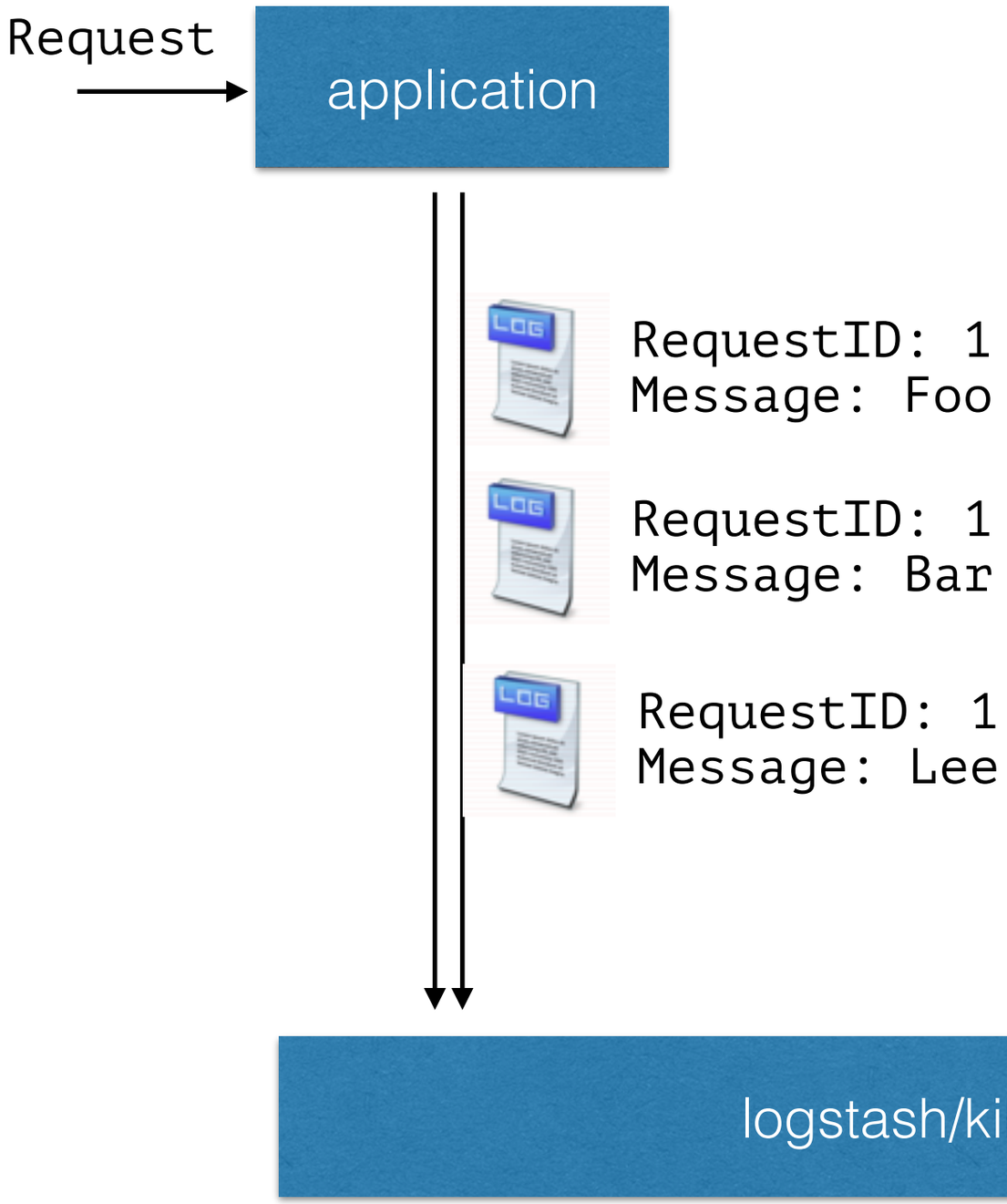
```
class UserFilter(logging.Filter):
    def __init__(self):
        self.username = "John"

    def filter(self, record):
        record.username = self.username
        return True

log.addFilter(UsernameFilter())

log.debug('Hello logstash’)
log.info('all good’)
```

Request ID

- a `Request ID` lets you correlate all log messages for a request

- the identifier is generated at the beginning of a request

- all logging messages within the request have this field


Request
application
RequestID: 1
Message: Foo
RequestID: 1
Message: Bar
RequestID: 1
Message: Lee
logstash/kibana

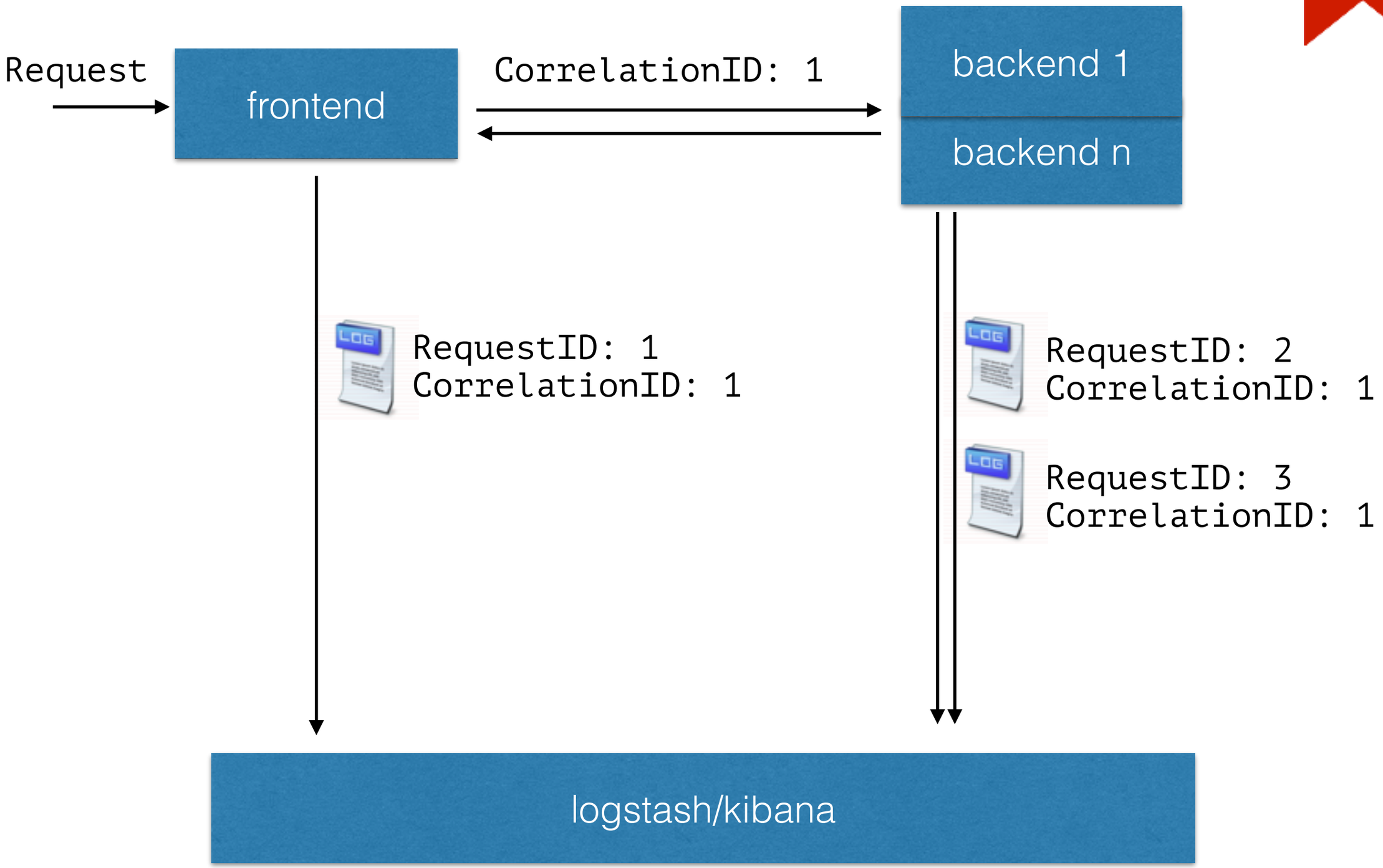
```
from flask import g
import uuid

@app.before_request
def before_request():
    g.request_id = uuid.uuid4().get_hex()

class RequestIDFilter(logging.Filter):
    def filter(self, record):
        record.RequestID = g.request_id
        return True

log.addFilter(RequestIDFilter())
```

Correlataion ID

- a `Correlation ID` lets you correlate log messages from different applications and systems

- generated at the entry application

- passed to backend applications via the HTTP Header Field `X-Correlation-ID`


Request
frontend
CorrelationID: 1
backend 1
backend n
RequestID: 1
CorrelationID: 1
RequestID: 2
CorrelationID: 1
RequestID: 3
CorrelationID: 1
logstash/kibana

```
from flask import g, request
import uuid

@app.before_request
def before_request():
    g.correlation_id = request.headers.get('X-
Correlation-ID','')

class CorrelationIDFilter(logging.Filter):
    def filter(self, record):
        record.CorrelationID = g.correlation_id
        return True

log.addFilter(CorrelationIDFilter())
```

Finger Crossed Handler

This handler wraps another handler and will log everything in memory until a certain level (action_level, defaults to ERROR) is exceeded. When that happens the fingers crossed handler will activate forever and log all buffered records as well as records yet to come into another handled which was passed to the constructor.

pocoo.org/projects/logbook/

```
import logbook
import sys

def calc(a,b):
    logbook.info("a = %s" %a)
    logbook.info("b = %s" %b)
    return a/b

err = logbook.StderrHandler()
handler = logbook.FingersCrossedHandler(err)
handler.push_application()

try:
    a, b = sys.argv[1:]
    print calc(int(a),int(b))
except:
    logbook.exception()
```

Blue Yonder is hiring!









Please visit our booth at EuroPython 2014