APPLICATION LOGGING

By Ben Keith

Quoin Inc. May 5, 2016

GOALS OF LOGGING

- Capture info on unexpected errors/exceptions
- Monitor load/traffic
- Debugging
- Security Auditing
- General system status

BASIC METHODS

- File logging
  - Almost universal support/simple
  - Gets the job done
  - Generally persistent
  - Gets unwieldy with multiple application instances
  - Logging multiple services generates a lot of files
  - Essential to rotate files periodically

SYSLOG

- Generic logging "standard"
  - RFC 3164 (from 2001; documents the de facto standard at the time)
  - RFC 5424 (from 2009; attempts to provide a standard for the future)
  - Many older implementations don't conform to either RFC
  - These standards don't specify the transport mechanism
- Fairly simple and easy to use

SYSLOG OLD "STANDARD"

- As described by RFC 3164:

  `<$facility_id*8 + $severity_id>$date $time $hostname $tag[$pid]: $message`

  Ex.

  `<34>Oct 11 22:14:15 mymachine su: 'su root' failed for lonvick on /dev/pts/8`

  - `34 // 8 == 4`: facility id for "security/authorization messages"
  - `34 % 8 == 2`: severity id for CRITICAL level
  - 34 is known as the "priority" value

SYSLOG NEW STANDARD

```
<$facility_id*8 + $severity_id>1 $datetime $hostname $app_name $pid $msg_id
$structured_data $message
```

Example:

```
<34>1 2003-10-11T22:14:15.003Z mymachine.example.com su - ID47 - 'su root'
failed for lonvick on /dev/pts/8
```

SYSLOG

- Multiple implementations
  - `BSD syslog` (multiple implementations; old school)
  - `syslog-ng`
  - `rsyslog` (probably most common now)
- Runs as a server
  - Traditionally receives entries over UDP or a local Unix domain socket
  - But TCP (with TLS/SSL) is now highly recommended, especially for remote logging
  - Many frameworks/languages have built in support for syslog
- Becoming superceded at the system level by the systemd journal
  - But still useful for applications

ADVANTAGES OF SYSLOG

- Simple format
- Fairly easy to understand
- Low resource usage
- Easy to multiplex logs from multiple machines/services

RSYSLOG

- Lightweight but powerful syslog implementation
  - Standard on many Linux distros (before systemd)
- Supports many input and output types
  - Inputs: Unix socket; UDP; TCP; files
  - Outputs: AMQP (RabbitMQ); files; forwarding; most DBs; elasticsearch

RSYSLOG

- Older syslog implementations don't allow multi-line messages
  - "Octet counted" framing allows for newlines embedded in log messages
    - Put the total size in bytes of the entry at the beginning
- Can buffer log entries
  - In memory or on disk for greater durability

LOGSTASH

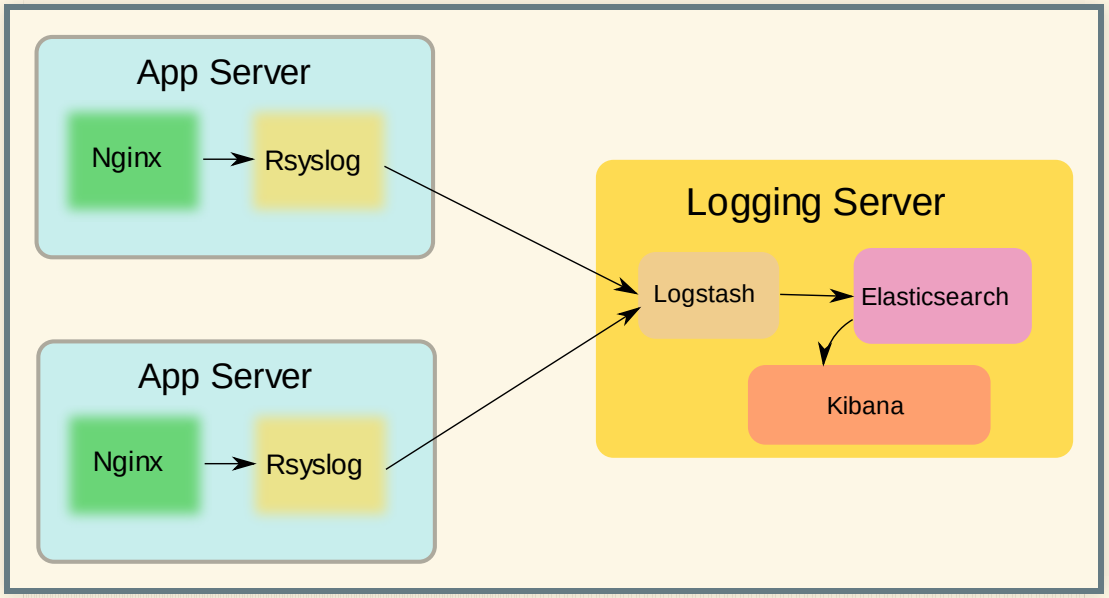
- Accepts logs from many different inputs
- Very powerful built-in filtering/processing abilities
  - Much more powerful here than rsyslog
- Many options for output
- Unfortunately there is no durability built-in yet (supposedly coming later)
  - Also no end-to-end acknowledgment
- Output log data to Elasticsearch for later analysis

```
input {
    syslog {
        port => 1514
        type => "nginx"
    }
}

filter {
    if [type] == "nginx" {
        # Just get rid of syslog's timestamp -- nginx gives its own timestamp
        mutate {
            remove_field => [ "timestamp" ]
        }
        grok {
            match => [ "message" , "%{COMBINEDAPACHELOG}+%{GREEDYDATA:extra_fields}"]
            overwrite => [ "message" ]
        }
```

KIBANA

- Log data viewer/visualizer
- Java app so pretty heavy on memory usage
- Pulls data from elasticsearch


App Server
Nginx
Rsyslog
App Server
Nginx
Rsyslog
Logging Server
Logstash
Elasticsearch
Kibana

AWS CLOUDWATCH LOGS

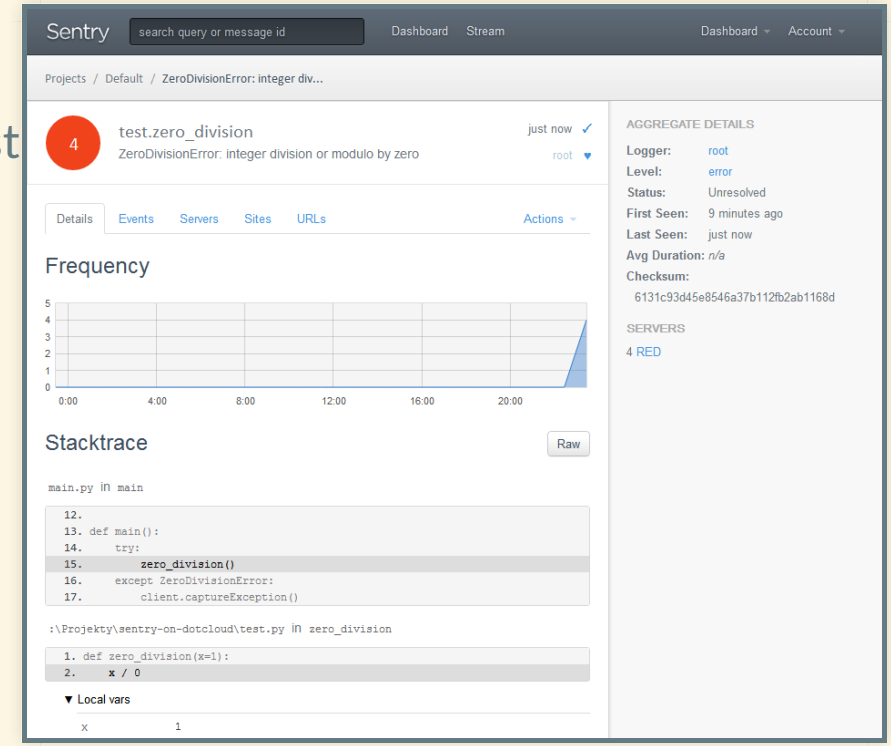
- Very simplistic by itself
- Supports forwarding logs to AWS Lambda routines to filter/process them
  - AWS Lambda routines can be written in Node.js, Python, or Java
  - No built-in filtering/processing config like Logstash

ALTERNATIVES

- Fluentd
- Graylog2

SENTRY

- Started as a Django exception reporting tool
- Now supports multiple frameworks
- Automatically get the full context of the request



GENERAL TIPS

- Log all errors
  - Including unhandled exceptions!
- Log function params especially in error logs
- Have an alert mechanism for error messages
  - Hipchat
  - Email
- Don't forget log rotation if you use files
- Periodically check to make sure logging is still working right
- Filter out extraneous logs unless you really need it
- Use an active health check service since logs aren't always helpful with that

THE END