Log Servers

Defensive Technologies

Pierre-Philipp Braun <p.braun@innopolis.ru>

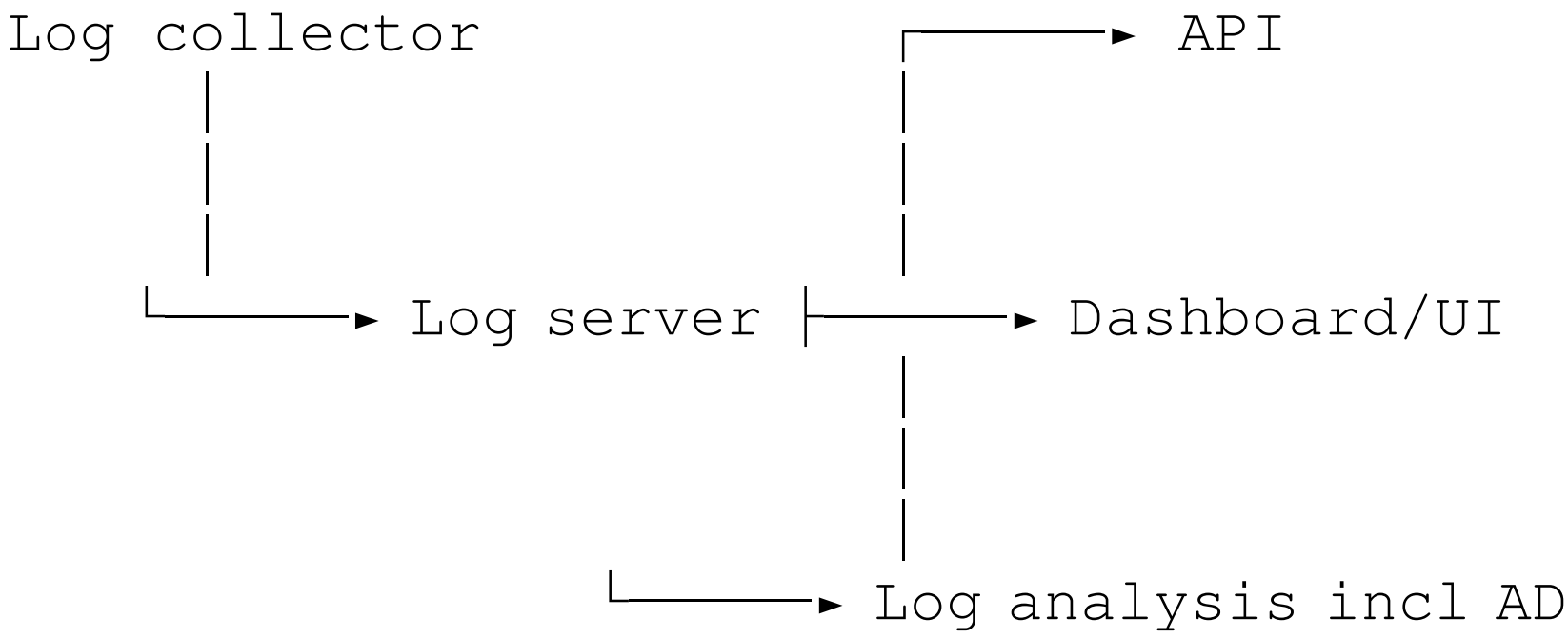
Table of contents

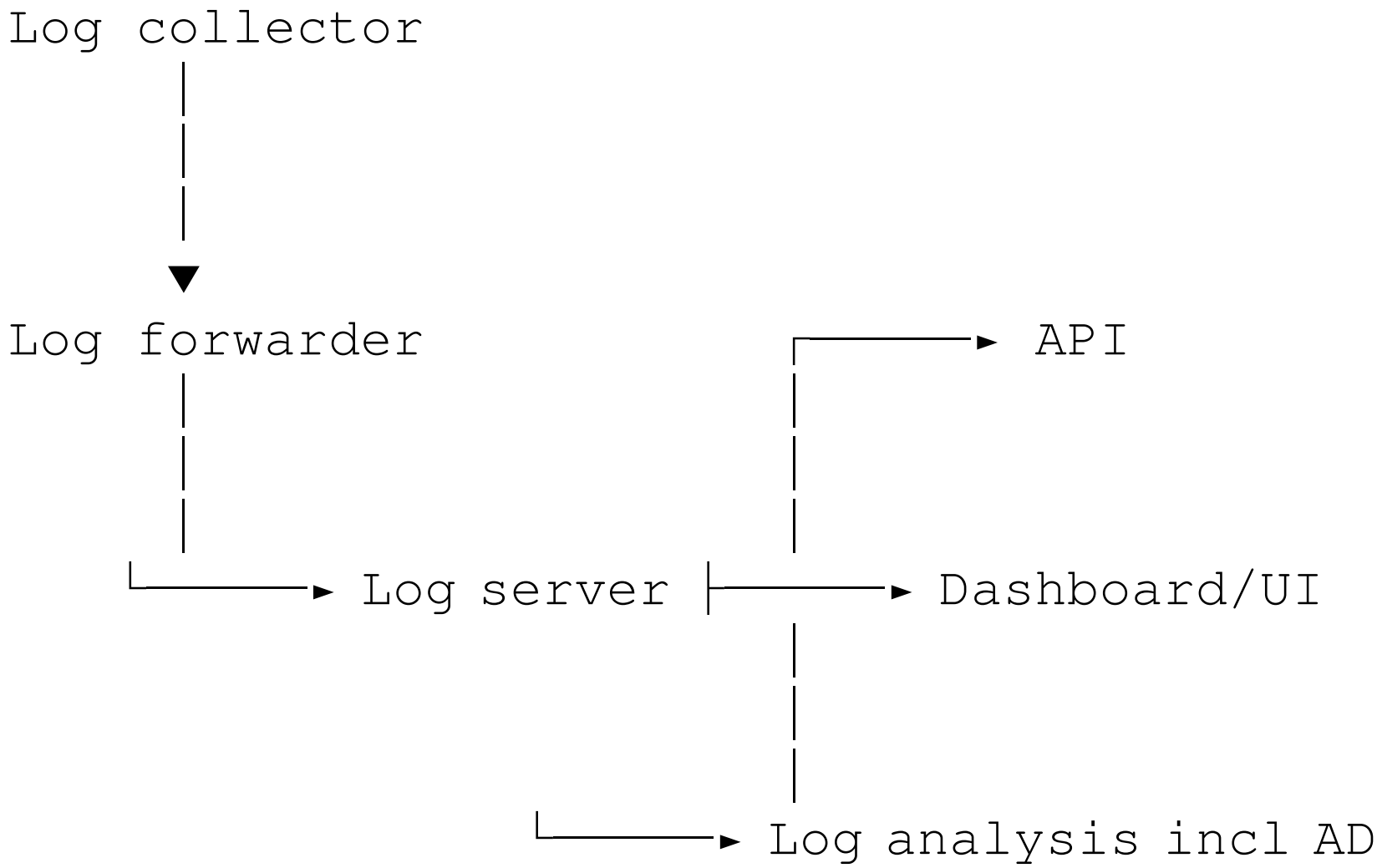
Overview & Use-cases

what are logs good for if nobody looks at them?…

what do with those lonely log files?…

overall architecture




Log collector
Log forwarder
API
Log server
Dashboard/UI
Log analysis incl AD

use-cases

- CDN logs (streaming or live-archives?)
- reverse-proxy logs
- application logs
- SIEM logs
  - suricata alert logs
  - suricata events logs (misc)
  - suricata flow logs
  - auth logs (incl. ssh)
  - sshguard & embbedded defense
  - cloud audit-trails
  - falco, tetragon, …
- *anything else?*

what kind of log sources are we dealing with exactly?…

and what kind of log formats are we dealing with within?…

==> sources:

```
misc one-liner log files
json log files
systemd
container logs
```

==> formats:

```
(various formats - typically has a level field)
json
```

// Questions on overall architecture?

Products & Architectures

what indexing engines are there?…

==> log servers / indexing engines

- Elasticsearch v7 (a nice legacy)
- Elasticsearch v9
- Opensearch
- Grafana Loki
- Victoria Logs

Maps

Maps rocks more on OS than on ELK7…

Licensing

ELK9 has three license modes

mode 1 (combo)

- GNU Affero General Public License v3.0 only
- Server Side Public License, v 1
- Elastic License 2.0

mode 2 - an “Apache License 2.0” compatible license

mode 3 - Elastic License 2.0

Anomaly Detection

- Random Cut Forest by AWS

Totally different log server architectures

- OS datastream shards & replicas
- –vs– VLogs split daemons (no indices!)

what log collectors are there?…

==>

log collectors

```
file-d (Ozontech)
filebeat
fluent-bit
promtail
vector
...?
```

tpsvp

but you also have log forwarders

```
fluentd
logstash
...?
```

// Questions on what to deploy?

Templates Settings & Mappings

that goes for ELK/OS only

settings

```
"index": {
   "number_of_shards": "${index_shards}",
   "number_of_replicas": "${index_replicas}"
}
```

mappings

a good start

```
{
  "numeric_detection": true,
  "dynamic": "true",
  "date_detection": false,
  "properties": {
    <MAPPINGS HERE>
```

mapping GeoIP coordinates

```
"source_location": {
  "ignore_malformed": true,
  "ignore_z_value": true,
  "type": "geo_point"
},
"destination_ip": {
  "ignore_malformed": true,
  "type": "ip"
},
```

mapping IP & IP ranges

```
"source_ip": {
  "ignore_malformed": true,
  "type": "ip"
},
"source_ip_range": {
  "ignore_malformed": true,
  "type": "ip_range"
},
```

improving (auto-)numeric fields

```
"EdgeResponseStatus": {
  "ignore_malformed": true,
  "fields": {
    "keyword": {
      "type": "keyword"
    }
  },
  "type": "short"
},
```

tpsvp

`long` by default?

```
"ClientASN": {
  "ignore_malformed": true,
  "fields": {
    "keyword": {
      "type": "keyword"
    }
  },
  "type": "integer"
},
```

`long` by default?

```
"gzip_ratio": {
  "ignore_malformed": true,
  "type": "float"
},
```

difficult corner-case (undo auto-numeric when dealing with hex)

```
"tcp.tcp_flags_ts": {
  "fields": {
    "keyword": {
      "type": "keyword"
    }
  },
  "type": "text"
}
```

yet another corner-case (non-compliant numeric fields) – needs to match data type in the app!

```
"principal": {
    "fields": {
        "keyword": {
            "type": "keyword"
        }
    },
    "type": "text"
},
```

// Questions on templates?

Index Lifecycle

rollovers & deletion

first things first

what's a datastream?…

==> eased indices' rolling instead of an index alias

```
datastream-name
.ds-stream-index001
.ds-stream-index002
...
```

ELK ISL & OS ISM

heavy-duty datastreams' rules of thumb:

- 30 to 50 GiB per primary shard
- less than 200 millions docs per primary shard

e.g. for CDN stream or reverse-proxy logs

```
# heavy-duty data-stream maximum power:
#
#  6 primaries                  multiple of the amount
#  50gb shards                  should trigger first
#  6 x 199 Mil cap (1 194 Mil)  just in case there'
#  1 week limit (just in case)  just in case it's
#
# retention for ... is 3 monthes
```

(terraform/opensearch custom module goes)

```
# policy
policy_name    = "cdn-name"
rollover_size  = "50gb"
rollover_docs  = 1194000000
rollover_after = "7d"
delete_after   = "90d"
```

cont.

```
# template
template_name  = "cloudflare"
template_file  = "json/template.json"
index_shards   = 3
index_replicas = 1
```

cont.

```
# shared
index_patterns     = ["cdn-name-*"]
template_priority = 0
```

condition coordination is –OR–!

```
"name": "hot",
"actions": [
  {
    "rollover": {
   "min_primary_shard_size": "${rollover_siz
    "min_doc_count": ${rollover_docs},
   "min_index_age": "${rollover_after}",
    }
  }
]
```

clean-up after a while – warm idles till deletion

```
"name": "warm",
"transitions": [
  {
    "state_name": "delete",
    "conditions": {
   "min_rollover_age": "${delete_after}"
    }
  }
]
```

tpsvp

cont.

```
"state_name": "delete",
"actions": [
  {
    "delete": {}
  }
]
```

// Questions on policy mgmt?

The Easy-peasy Alternative (VLogs)

easy and lighter compared to ELK/OS

what goes there as…

- *indices/datastreams?*
- *template settings?*
- *template mappings?*
- *policy mgmt?*
- *client and endpoint authentication?*

- *indices/datastreams?*

==> no indices, just some manual slices and indexing hint

```
victoria-logs-prod -storageDataPath /data/vlogs
suricata
victoria-logs-prod -storageDataPath /data/vlogs
...

/insert/jsonline?_stream_fields=...
```

the `_msg` field situation

- VLogs assumes a verbose log
- need to define the field for that log message
- *but* we don't do that, we do json properly during log collection!

- the `_msg` field offers easy search
- so let's do that for selecting *kind-of-indices*!

the trick I use for defining the default log field – e.g. with fluentbit

```
[FILTER]
    name modify
    match eve
   add sensor suricata@{{inventory_hostname_sho
```

cont.

```
[OUTPUT]
    name http
    match eve
   uri /insert/jsonline?_stream_fields=sensor&_
```

LogsQL – therefore it's easy to show a specific "index"…

```
"suricata@that-node"
```

as `_msg` is default field

- *template settings?*

==> no replicas, eventually setup cluster mode for HA with

```
writes - vlstorage + vlinsert
reads  - vlstorage + vlselect

victoria-logs-prod -storageNode=node1:9491,node
```

==> or eventually setup DR with

```
writes - log collector dual destination
reads  - dns switch over
```

tpsvp

==> no shards

(not sure cluster mode helps here)

- *template mappings?*

==> nothing

`automatic field type detection (no IP, IP range nor GeoIP)`

- *policy mgmt?*

==> nothing, just run-time parms

```
-retentionPeriod=90d
-retention.maxDiskSpaceUsageBytes=100GiB
-retention.maxDiskUsagePercent=80
```

…per storage path

- *client and endpoint authentication?*

==> skipping ssl and l7 auth, using ACLs instead

```
(using vmauth + nftables instead of VLogs Enterp
```

```
# vmauth-select
iif $nic ip saddr @query_subnets tcp dport 8427 a
```

```
# vmauth-insert
iif $nic ip saddr @ingest_subnets tcp dport 8429
```

…yes there's a risk for DoS here, but no data leakage

vmauth multitenancy setup – ingestion

```
# suricata has dedicated vlogs instance to ease
 # /data/vlogs-suricata/
 - url_prefix: "http://127.0.0.1:9401"
   src_paths:
     - "/insert/[^/]+"
   src_headers:
     - "ProjectID: 1"
   headers:
     - "ProjectID: 1"
```

– query

```
  - url_prefix: "http://127.0.0.1:9401"
    src_paths:
      #- "/select/[^/]+"
      - ".*"
    src_headers:
      - "ProjectID: 1"
    headers:
      - "ProjectID: 1"
```

and setup grafana datasource accordingly

// Questions on Vlogs?

This is the end