

How we built a versionator in 30m using datadog + puppet

Yotam Gafni, Data pipeline in Similarweb
yotam.gafni@gmail.com

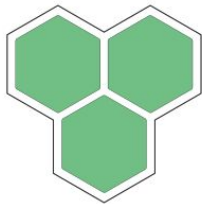


The versionator

TAPI PROXY versions

TAPI versions

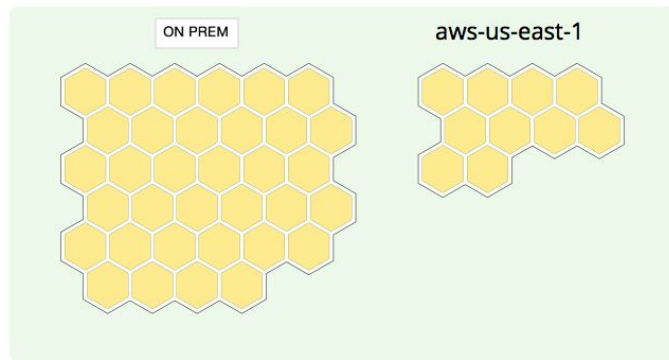
production



staging



production



staging



Why do we need a versionator ?

- Two significant production issues due to deployment not reaching all machines
 - A tale of snappy compression
 - The story of a client bug
- Teamcity and generally deployment has issues
 - Left out test versions in production
 - On-prem vs Amazon

Datadog in a nutshell

- The agent
- Tags
- Building a dashboard

Let's take that to puppet

hiera/role/tracker-ng.yaml

```
+sg_base::packages::install:  
+ datadog:  
+   ensure: 0.13.0  
+   provider: pip
```

modules/roles/manifests/tracker-ng.pp

```
+   cron { 'tapi_versionator':  
+       command => "python  
/opt/tapi/versionator/tapi_versionator.py",  
+       user    => root,  
+       minute  => '*/1'  
+   }
```

modules/sg_tracker_ng/manifests/init.pp

```
+   "/opt/tapi/versionator":  
+   source =>  
'puppet:///modules/sg_tracker_ng/versionator',  
+   ensure => directory,  
+   mode    => 0755,  
+   require => File['/opt/tapi'];
```

The code

```
modules/sg_tracker_ng/files/versionator
```

```
+import os
```

```
+import datetime
```

```
+
```

```
+from datadog.dogstatsd.base import DogStatsd
```

```
+
```

```
+
```

```
+dogs = DogStatsd(port=9125)
```

```
+last_version_raw = os.path.realpath("/opt/tapi/deployment/latest").split("/")[-1]
```

```
+version_timestamp = last_version_raw.split(".")[1]
```

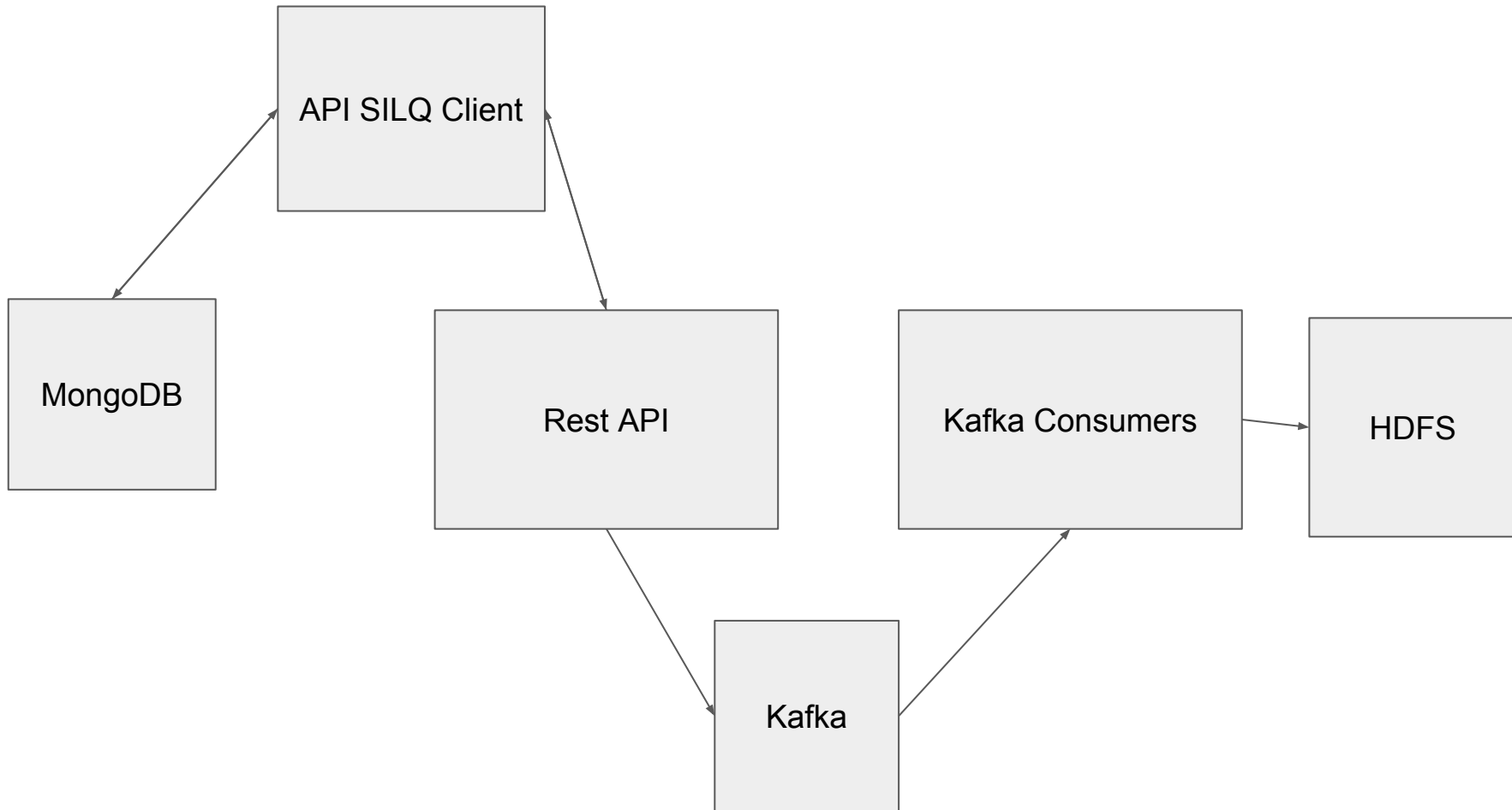
```
+last_version =
```

```
datetime.datetime.fromtimestamp(int(version_timestamp)).strftime('%y%m%d%H%M%S')
```

```
+dogs.gauge("versionator.tapiversion", last_version)
```

Lesson learnt using API SILQ

- Over 100 Integration tests for our ingest pipeline



Important features

- Requests coloring & filtering
- Ignore changing fields in end-data
- Custom GUI
- Running test by test or whole suite
- Tests are implemented via configuration Json + Textual post data that can later be obfuscated / compressed
- Expected results

Technical Highlights

- HDFS `ls` is heavier than read
- Pre-HDFS Pyro module
- Latency Benchmark

Issues

- Built-In Inconsistencies
 - First request of the day logic
 - Cache behavior
 - Random sampling
- Inconsistencies over time
 - Changing configs
 - Added/Removed consumers

Future plans

- Teamcity integration, giving up on Expected results