zipnish Documentation

Release latest

August 02, 2016

Contents

Microservice monitoring tool based on Varnish-Cache. Currently it only supports Varnish 4. Zipnish piggybacks the VSL and stores a bunch of data in a similar way as Zipkin does.

There is a log-reader component responsible for fetching data from VSL and having it stored into a MySql database, furthermore there is an available UI that will display in a hierachical manner all requests going through varnish to your services. Both of these components share the same MySql instance.

CHAPTER 1

Prerequisites

Following packages are required for running Zipnish:

- simplemysql
- flask
- sqlalchemy
- flask_sqlalchemy
- mysql-python

CHAPTER 2

Installation

Install with pip:

```
$ sudo pip install zipnish
```

CHAPTER 3

Configuration

A configuration file found at `/etc/zipnish/zipnish.cfg` is required with a structure as described below:

```
[Database]
# Db settings for the MySql connection.

# MySql host
host = 192.168.59.103

# Database name
db_name = microservice

# User name
user = zipnish

# Password
pass = secret

# Connection keep-alive
keep_alive = true

[Cache]
# Defines which cache to fetch logs from.
# Name of the cache (same value sent via the -n argument)
# name = demo

[Log]
# Path to the daemon logfile.
log_file = /var/log/zipnish/zipnish.log

# Valid log_levels are: DEBUG, INFO, WARNING, ERROR
log_level = DEBUG
```

For convenience purposes, there is a docker image available which handles setting up Mariadb database along with a test user.

CHAPTER 4

Run Zipnish

Considering that your Varnish instance is properly configured in relation to your services, after installing Zipnish there are two commands available:

Run the logreader:

```
$ zipnish-logreader
```

Run the ui (by default port 5000):

```
$ zipnish-ui
```

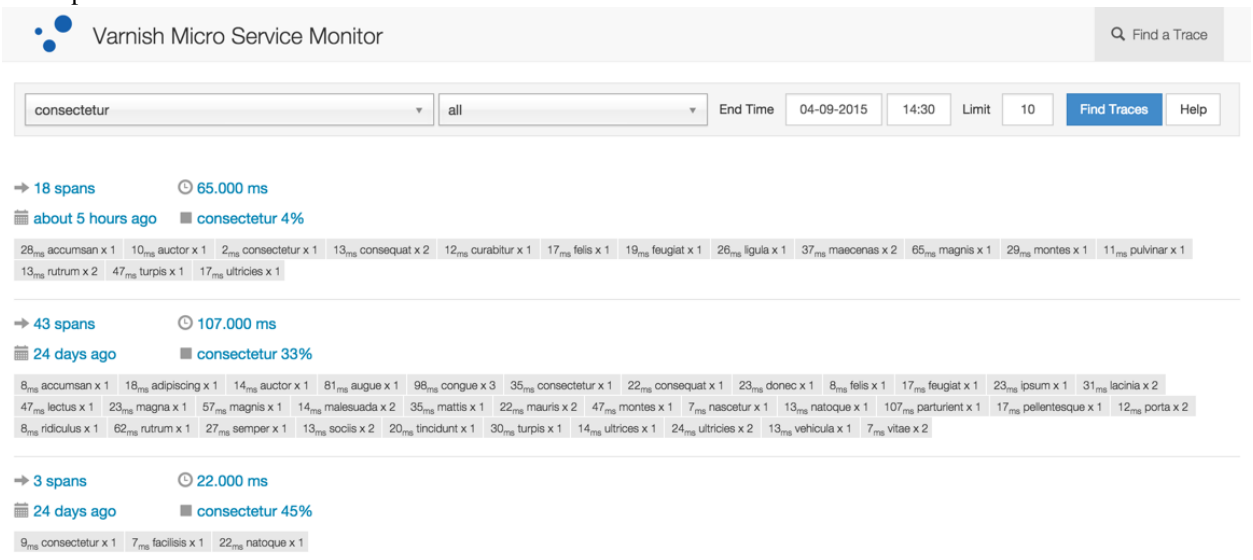
Contents:

4.1 Zipnish UI

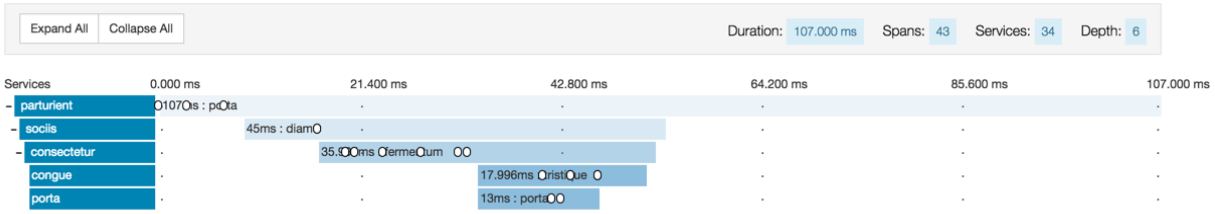
Zipnish UI is a Flask based app meant to give an overview of the timestamps handled by the logreader.

Screenshots:

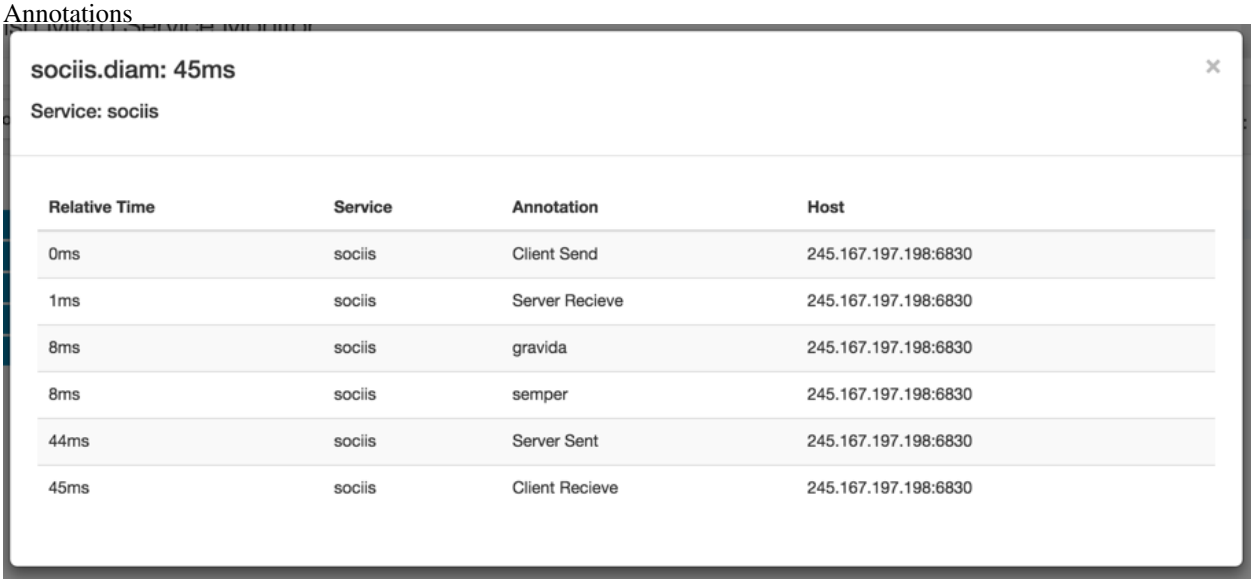
Lookup



Search Results



Annotations



4.2 VCL how to's

In order for Zipnish to grab its required headers there are a few changes that are required in your VCL. There are two main scenarios to be handled here:

1. Caching disabled:

```
vcl 4.0;

backend DemoMicroservice {
    .host = "127.0.0.1";
    .port = "9999";
}

# disable caching - see further down for another example with caching enabled
sub vcl_recv {
    if (req.url ~ "^/DemoService") {
       set req.backend_hint = DemoMicroservice;
       return (pass);
    }
}
```

2. Caching enabled:

```
vcl 4.0;

backend DemoMicroservice {
```

```
    .host = "127.0.0.1";
    .port = "9999";
}

# disable caching - see further down for another example with caching enabled
sub vcl_recv {
    if (req.url ~ "^/DemoService") {
        set req.backend_hint = DemoMicroservice;
    }
}

sub vcl_deliver {
    # add the response headers if this is a cache hit
    if (obj.hits > 0) {
        if (req.http.x-varnish-trace) {
            set resp.http.x-varnish-trace = req.http.x-varnish-trace;
        } else {
            set resp.http.x-varnish-trace = req.http.x-varnish;
        }
        set resp.http.x-varnish-parent = req.http.x-varnish;
    }
}
```

4.3 Docker

Unless you have a MySql instance at hand, this Ubuntu based image will spawn a Mariadb instance. This instance will be the one used by both the logreader and the ui. Check the IP of your docker setup and update the zipnish.cfg accordingly.

The image relies on two extra files: `database.sql` and `init-db.sh`. Both these files are available in the `/docker` folder within the zipnish source code. The two extra files are responsible for the creation and initialisation of an user and db tables that Zipnish will use further down the line. Browse to this folder and run the following commands:

```
$ docker pull mariusm/ubuntu-mariadb
$ docker run -d -p 3306:3306 mariusm/ubuntu-mariadb
```

A database and db user with the following credentials will be available, if you're going to use this db instance, make sure that zipnish.cfg reflects these settings:

user = zipnish

pass = secret

db_name = microservice

To quickly check that the container is up and running, you can connect to it directly with a Mysql client. Retrieve the IP of your docker setup and connect to the mariadb instance as follows:

```
$ mysql -u zipnish -h "your docker ip" --paswword=secret
```

On a MacOs machine you can simply run the following command:

```
$ mysql -u zipnish -h $(boot2docker ip) --password=secret
```

4.4 Examples

This section will provide a short example on how to extend an endpoint in order for Zipnish to be aware of it.

Full example code found here: https://github.com/varnish/zipnish/blob/master/logreader/test/server.py

Zipnish requires three headers to be available per request basis:

- X-Varnish - Request id assigned by Varnish.
- X-Varnish-Trace - The id that has been assigned by Varnish to the first incoming request.
- X-Varnish-Parent - The id of the parent request which has triggered the current request.

```
def do_GET(self):
    x_varnish_header = self.headers['x-varnish']
    trace_header = x_varnish_header

    if 'x-varnish-trace' in self.headers:
        trace_header = self.headers['x-varnish-trace']

    headers = {'x-varnish-trace': trace_header,
               'x-varnish-parent': x_varnish_header}
...
```

In this specific example there is a very simple web server that handles basic GET requests based on a configuration found in server.yaml:

```
---
traces:
    - trace:
        - url: /api/articles
        - span: /api/auth
        - span: /api/titles
        - span: /api/images
        - span: /api/correct
        - span: /api/compose
```

The test server is exposed through port 9999, our vcl configuration has a backend the points to this server:

```
vcl 4.0;

backend default {
    .host = "127.0.0.1";
    .port = "9999";
}

# For simplicity reason disable caching, see vcl how to's for enabled caching.
sub vcl_recv {
       return (pass);
}
```

Given that Varnish has its default settings, the request below:

```
$ curl -is http://localhost:6081/api/articles
```

wil have the following output:

```
HTTP/1.1 200 OK
Server: BaseHTTP/0.3 Python/2.7.9
Date: Tue, 10 May 2016 08:43:44 GMT
```

```
Content-type: text/html
X-Varnish: 11
Age: 0
Via: 1.1 varnish-v4
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
```

and server output:

```
127.0.0.1 - - [10/May/2016 08:43:43] "GET /api/auth HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 08:43:43] "GET /api/titles HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 08:43:44] "GET /api/images HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 08:43:44] "GET /api/correct HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 08:43:44] "GET /api/compose HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 08:43:44] "GET /api/articles HTTP/1.1" 200 -
```

The scenario is as follows:

1. A client does a request to the test server asking for **/articles**
2. In order to serve **/articles**, subsequent calls are required to other endpoints such as **/auth**, **/titles** ...etc. For demo purposes these subsequent calls are handled by the same server, what is important to notice is that all sub-requests go through Varnish as well. A random sleep time has been added for each request in order to simulate some "hard work".
3. The application server decorates the subsequent requests with the required headers, as shown in the code above.
4. Zipnish-logreader picks up its required data from VSL as these requests go through.
5. While data gets written in the MySql database, Zipnish-UI will be able to represent how requests have been issued and how much time each of them has taken.

4.5 Give Zipnish a try

This section aggregates all other chapters in the documentation and will provide a guide for setting up a working environment with Zipnish. As a side-note, all steps below have been run on a Centos7 machine.

A fresh Centos7 VM requires the following packages:

```
$ sudo yum -y install mariadb-devel
$ sudo yum -y install python-devel
$ sudo yum -y install python-pip
```

1. **Start an application server**

   Clone the git repo:

   ```
   $ git clone https://github.com/varnish/zipnish.git
   ```

   Start the application server:

   ```
   $ cd zipnish/logreader/test
   $ python server.py &
   ```

   This will spawn a lite web server listening on port `9999`, the endpoints available in this server are as defined in the server.yaml file located in the same folder.

2. **Install, configure and start Varnish**

   Zipnish requires Varnish 4, earlier versions are not supported.

```
$ sudo yum install -y varnish
```

   Update `/etc/varnish/default.vcl` file with the following content:

```
vcl 4.0;

# Default backend definition. Set this to point to your content server.
backend default {
    .host = "127.0.0.1";
    .port = "9999";
}

sub vcl_recv {
    return(pass);
}
```

   Start Varnish:

```
$ sudo service varnish start
```

   Or reload, if Varnish has already been installed:

```
$ sudo service varnish reload
```

   Unless otherwise specified, Varnish will listen on port `6081`. For simplicity reasons `vcl_recv()` will pass all requests, refer to the VCL section in order to have caching enabled. Notice that the default backend points to the server that has just been spawned previously.

3. **Check that Varnish and the backend are set correctly**

   Issue the following request against Varnish:

```
$ curl -is http://localhost:6081/api/articles
```

   Expected output:

```
127.0.0.1 - - [10/May/2016 11:26:54] "GET /api/auth HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 11:26:54] "GET /api/titles HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 11:26:54] "GET /api/images HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 11:26:55] "GET /api/correct HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 11:26:55] "GET /api/compose HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2016 11:26:55] "GET /api/articles HTTP/1.1" 200 -

HTTP/1.1 200 OK
Server: BaseHTTP/0.3 Python/2.7.9
Date: Tue, 10 May 2016 11:26:55 GMT
Content-type: text/html
X-Varnish: 32803
Age: 0
Via: 1.1 varnish-v4
Transfer-Encoding: chunked
Connection: keep-alive
Accept-Ranges: bytes
```

4. **Configure a MariaDb instance**

   Install docker:

```
$ sudo yum -y install docker
```

Pull and run the following container for setting up a MariaDb instance:

```
$ docker pull mariusm/ubuntu-mariadb
$ docker run -d -p 3306:3306 mariusm/ubuntu-mariadb
```

Once created, the container will host a mariadb instance with a `microservice` database and a user with the following credentials:

user = zipnish

pass = secret

5. **Install and configure Zipnish**

   Zipnish is available in Pypi, thus run the following command to install it:

```
$ sudo pip install zipnish
```

   Create a `/etc/zipnish/zipnish.cfg` with a content similar as described in configuration. Retrieve the docker container IP and update the mysql host accordingly in the .cfg file.

   Create the log folder:

```
$ sudo mkdir -p /var/log/zipnish
$ sudo chown -R $(whoami): /var/log/zipnish
```

6. **Run**

   Start the log-reader:

```
$ zipnish-logreader &
```

   Start the zipnish UI:

```
$ zipnish-ui &
```

   Issue a test request to generate tracking data:

```
$ curl -is http://localhost:6081/api/articles
```

7. **Browse the UI**

   Open a browser and navigate to http://127.0.0.1:5000

4.6 Changes