Let's Visualize Your Spring Boot Applications

Acroquest Technology Co., LTD.
Shin Tanimoto (@cero_t)

Acroquest Technology

Who am I?

- 谷本 心 (Shin Tanimoto)
  - Acroquest Technology (in Japan)
  - Board member of Japan Java User Group (JJUG)
    - System Architect / Troubleshooter
    - Microservices, Elasticsearch
    - Pokemon: Team Mystic
    - twitter: @cero_t
    - facebook: shin.tanimoto

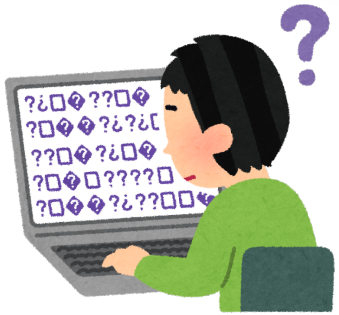
I ask you ...

If you have any question, ask me by

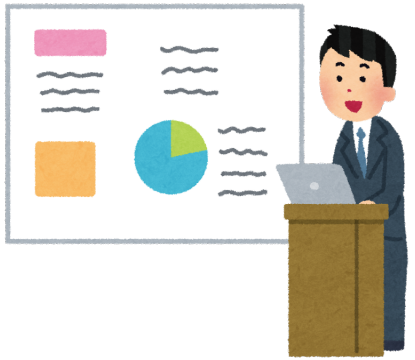
- twitter with mention to **@cero_t** or hashtag **#s1p_vis**
- **http://sli.do/** with event code **#5463**

Seeing is believing

Seeing is deceiving?



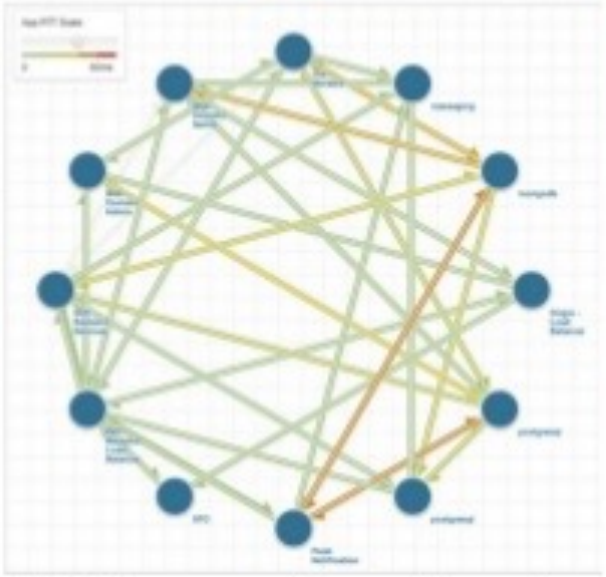
You can’t even believe without seeing



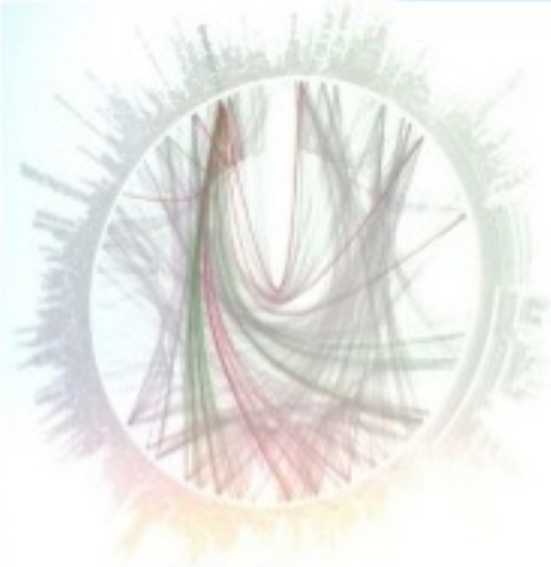
Seeing is getting insights

“Death Star” Architecture Diagrams

Netflix



Gilt Groupe (12 of 450)



Twitter

As visualized by Appdynamics, Boundary.com and Twitter internal tools

Let's Visualize Your Spring Boot Applications

Acroquest Technology Co., LTD.
Shin Tanimoto (@cero_t)

Acroquest Technology

Table of contents

1. Tools
2. Are your services working well?
3. Are your services enough comfortable?
4. How do your microservices look like?
5. Do your services contribute to your business?

#1

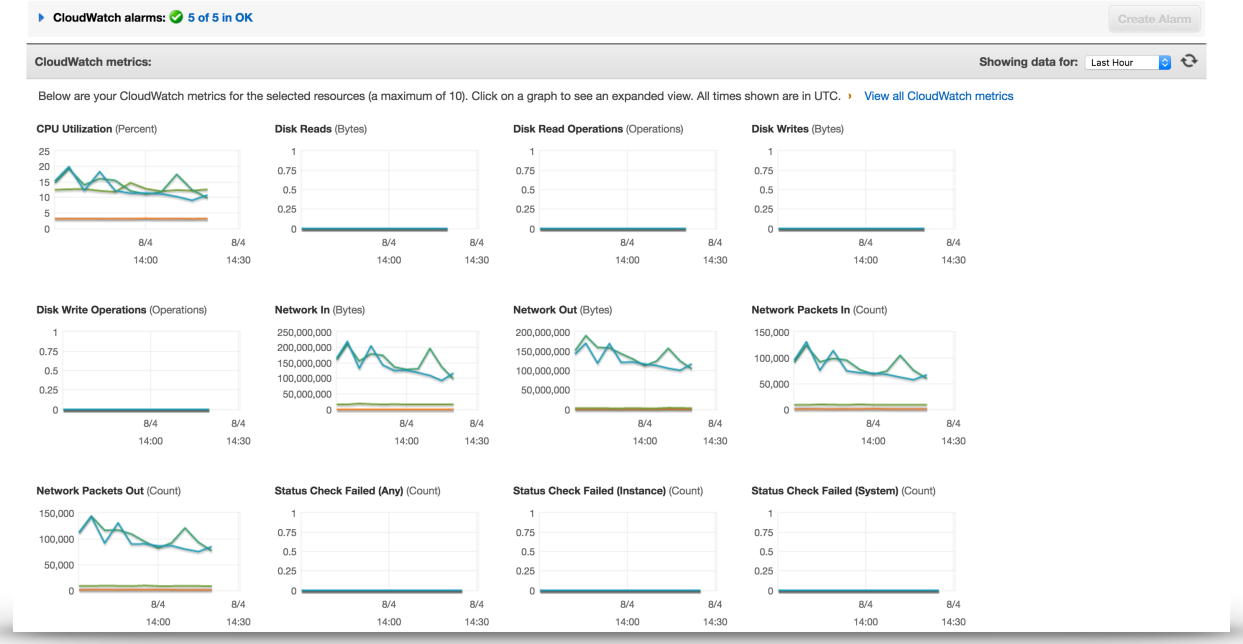
Tools

1. Tools

1. AWS CloudWatch
   - for monitoring and alerting
2. Elasticsearch + Logstash + Kibana + Beats
   - for monitoring and visualizing logs
3. Spring Cloud Sleuth + Zipkin
   - for visualizing microservices

1. Tools

1. AWS CloudWatch



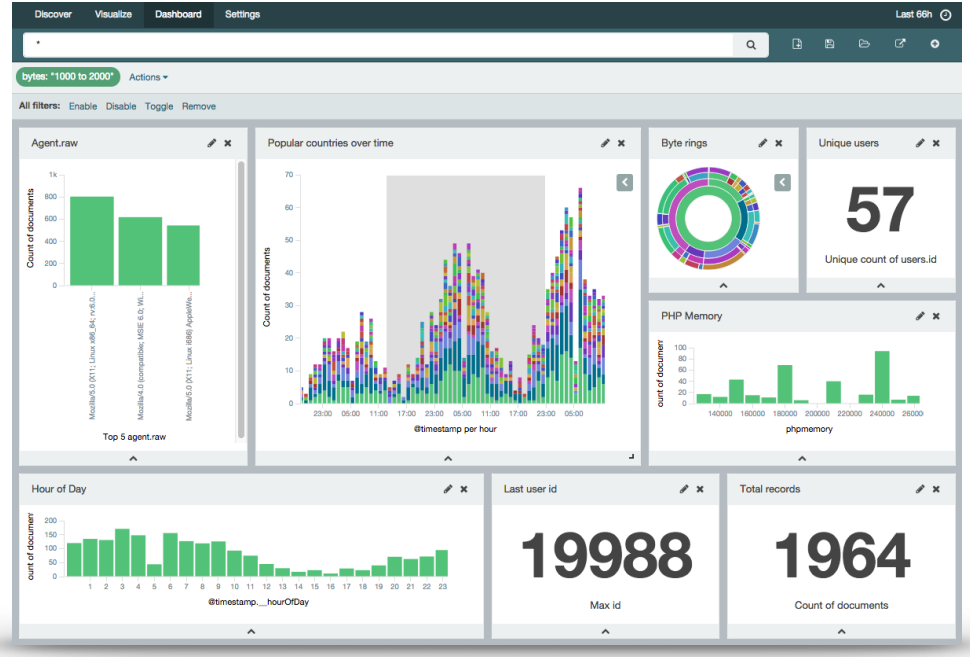
1. Tools

1. AWS CloudWatch
   - Built-in monitoring service for AWS.
   - Custom metrics can be collected by aws-cli or SDK on each server.
   - Logs can be collected by “awslogs" agent on each server.

   - Datadog can be alternative.

1. Tools

2. Elasticsearch + Logstash + Kibana + Beats

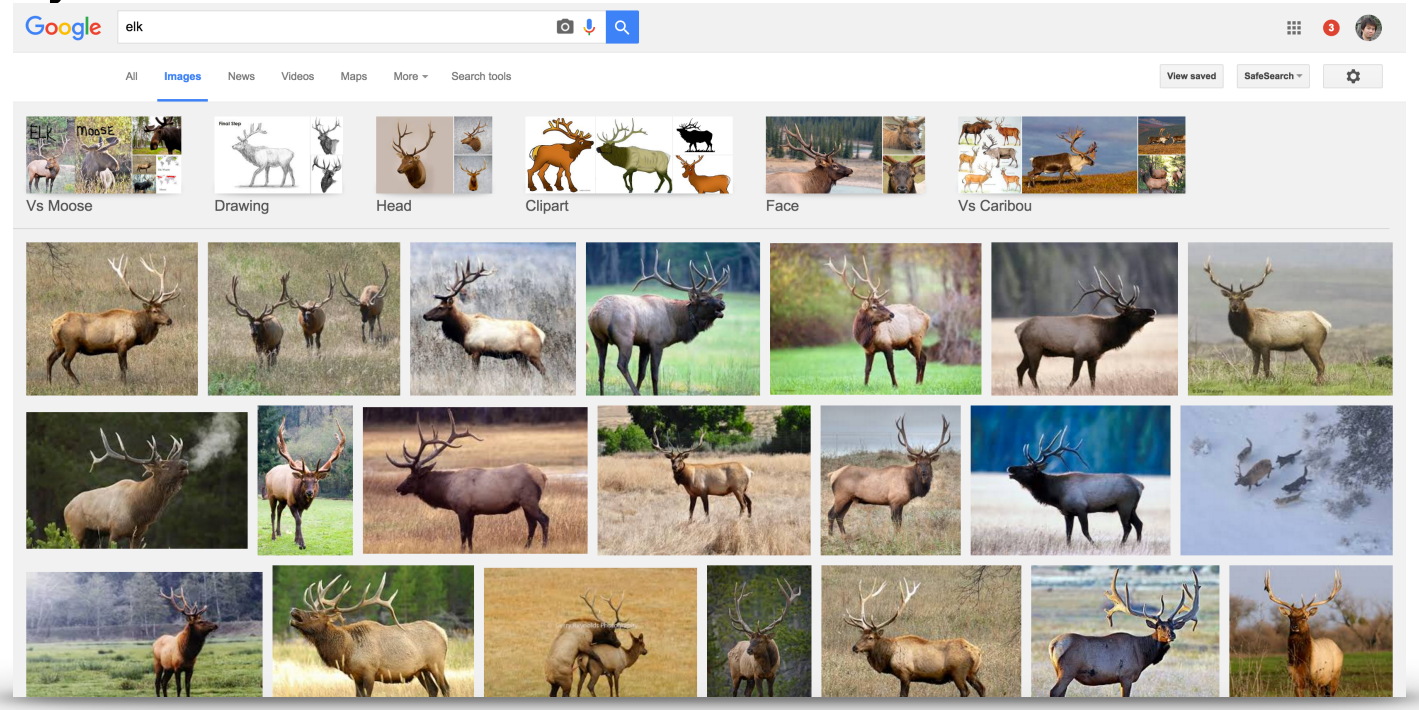


1. Tools

2. Elasticsearch + Logstash + Kibana + Beats
   - Elasticsearch : Full-text search engine
   - Logstash : Data collector, processor, sender
   - Kibana : Visualization platform
   - Beats : Data shipper

1. Tools

They are so called “ELK”.



1. Tools

ELK-bee?

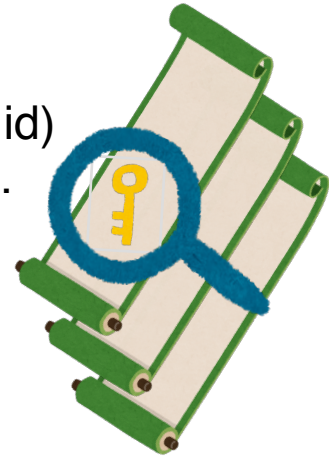
1. Tools

Elastic stack

1. Tools

3. Spring Cloud Sleuth + Zipkin

- Spring Cloud Sleuth is a distributed tracing solution.
  - Embeds the trace id (correlation id) and the span id (message id) into the HTTP request headers, RabbitMQ headers, and so on.
  - Embeds those ids into the logs through SLF4J MDC.
  - Send trace data to Zipkin or Spring Cloud Stream.



- Zipkin is a distributed tracing system.
  - Collects and lookup trace data.
  - Zipkin UI shows the data.

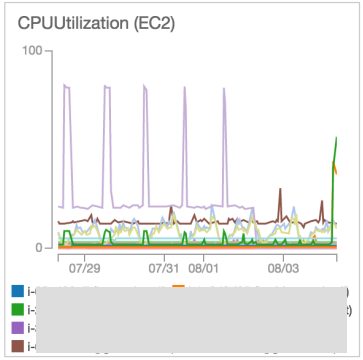
#2

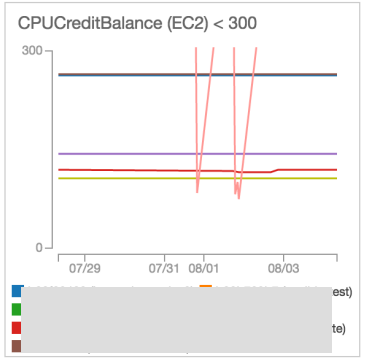
Are your services working well?

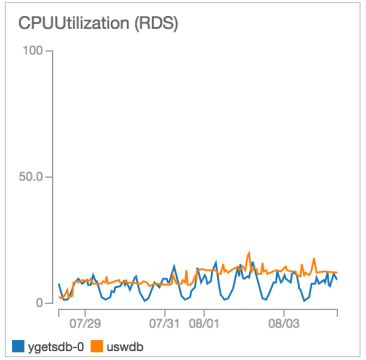
2. Are your services working well?

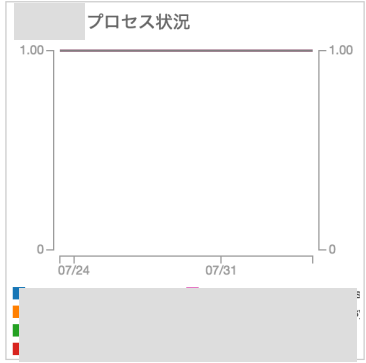
1. What to do?
   - Per server monitoring
     (1) CPU usage
     (2) Physical memory usage / available
     (3) Disk space usage / available
   - Per service / middleware monitoring
     (1) Process status
     (2) Health check
   - Put them on the same dashboard.
   - Create alert notifications of them.

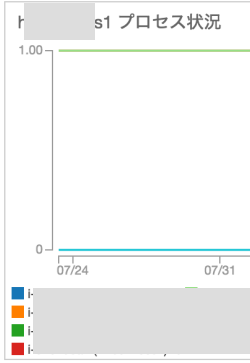
2. Are your services working well?

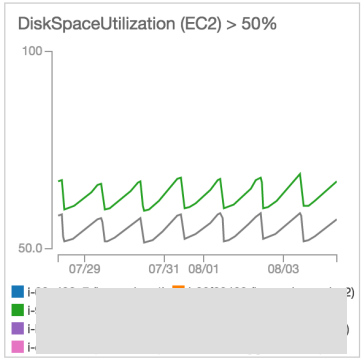


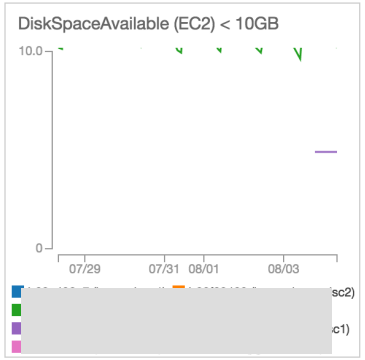


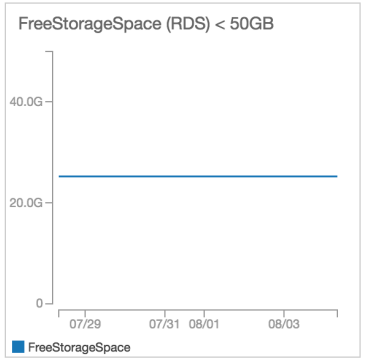


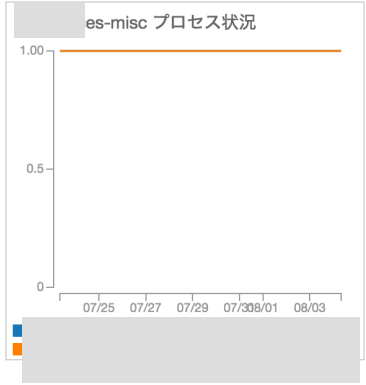


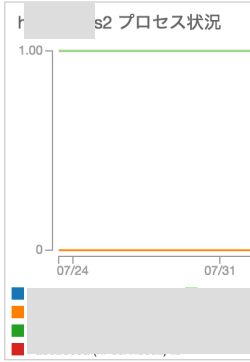












2. Are your services working well?

2. Why do this?
   (1) To see if servers are woking well now.
   (2) To see if servers were woking well in the past.
   (3) To see if servers will be working well in the near future.

2. Are your services working well?

3. How to do?
   - I believe you already know to setup monitoring.
   - Do you know topboat of elastic stack?

2. Are your services working well?

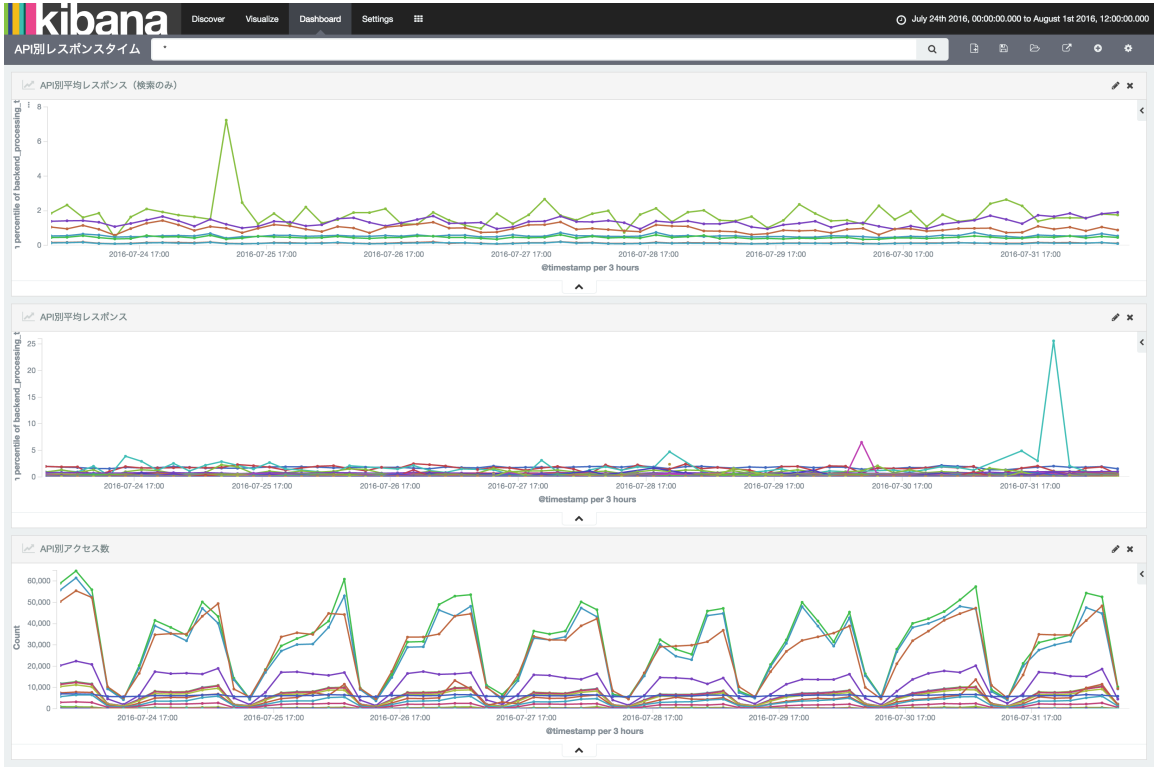
#3

Are your services enough comfortable?

3. Are your services enough comfortable?

1. What to do?
   - Monitor the web access
     (1) Access count
     (2) Response time (Per public API)
     (3) Response code
   - Monitor the datasource
     (1) Query count
     (2) Query response time

3. Are your services enough comfortable?



Avg. of response time per API (Search only)

Avg. of response time per API (All)

Access count per API

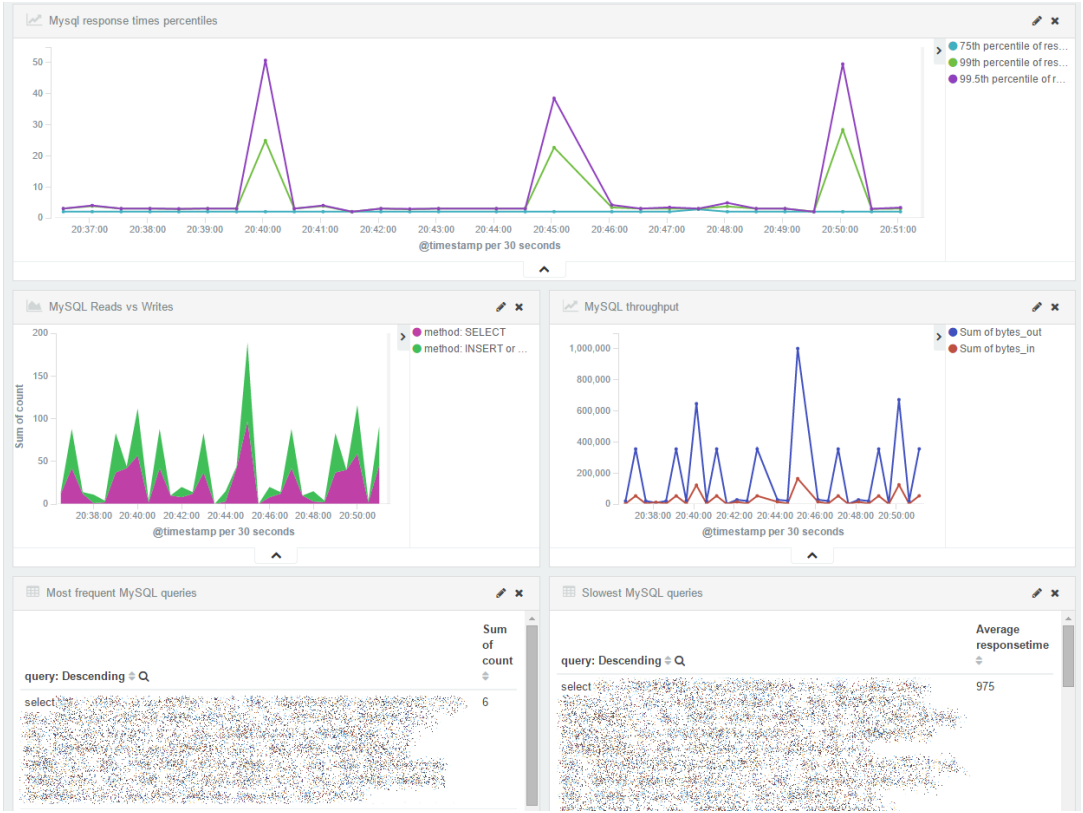
3. Are your services enough comfortable?

2. Why do this?
   - To see if the visitors feel comfortable or annoyed.

3. Are your services enough comfortable?

3. How to do?
   - If you use ELB, just use ELB monitoring on AWS CloudWatch.
   - Collect access logs using Logstash or Filebeat and forward them to Elasticsearch.
     And visualize them on Kibana.
   - Do you know packetbeat?

3. Are your services enough comfortable?



#4

How do your microservices look like?

4. How do your microservices look like?

1. What to do?
   - Trace microservices request by Spring Cloud Sleuth
   - Send trace data to Zipkin and visualize them

4. How do your microservices look like?

2. Why do this?
   (1) To get a grasp of your system looks like.
   (2) When modifying some microservices, to confirm the influence range.
   (3) To find unused services.

4. How do your microservices look like?

3. How to do?
   (1) Go to Spring Initializr web site and find “sleuth” and “zipkin”.
   (2) Add sleuth dependencies to your microservices.
   (3) Create zipkin server using zipkin dependencies.

4. How do your microservices look like?

Project Metadata

Artifact coordinates

Group

com.example

Artifact

demo

Generate Proje

Don't know what to look for? Want more options? Switch to the full version.

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

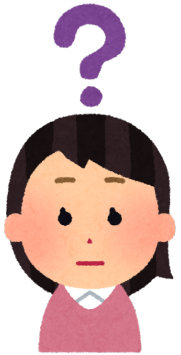
sleuth

Sleuth
Distributed tracing via logs with spring-cloud-sleuth

Zipkin Client
Distributed tracing with an existing Zipkin installation and spring-cloud-sleuth-zipkin. Alternatively, consider Sleuth Stream.

Sleuth Stream
Marshals Spring Cloud Sleuth Spans over a Spring Cloud Stream binder

Zipkin Stream
Consumes span data in messages from Spring Cloud Sleuth Stream and writes them to a Zipkin store



4. How do your microservices look like?

Project Metadata

Artifact coordinates

Group

com.example

Artifact

demo

Generate Proje

Don't know what to look for? Want more options? Switch to the full version.

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

zipkin Client

Zipkin Client
Distributed tracing with an existing Zipkin installation and spring-cloud-sleuth-zipkin. Alternatively, consider Sleuth Stream.

Zipkin Stream
Consumes span data in messages from Spring Cloud Sleuth Stream and writes them to a Zipkin store

Zipkin UI
add the Zipkin UI module to the Zipkin server to get a Zipkin service that accepts Spans and provides visualization

Zipkin Server
Consumes span data over HTTP and writes them to a span store

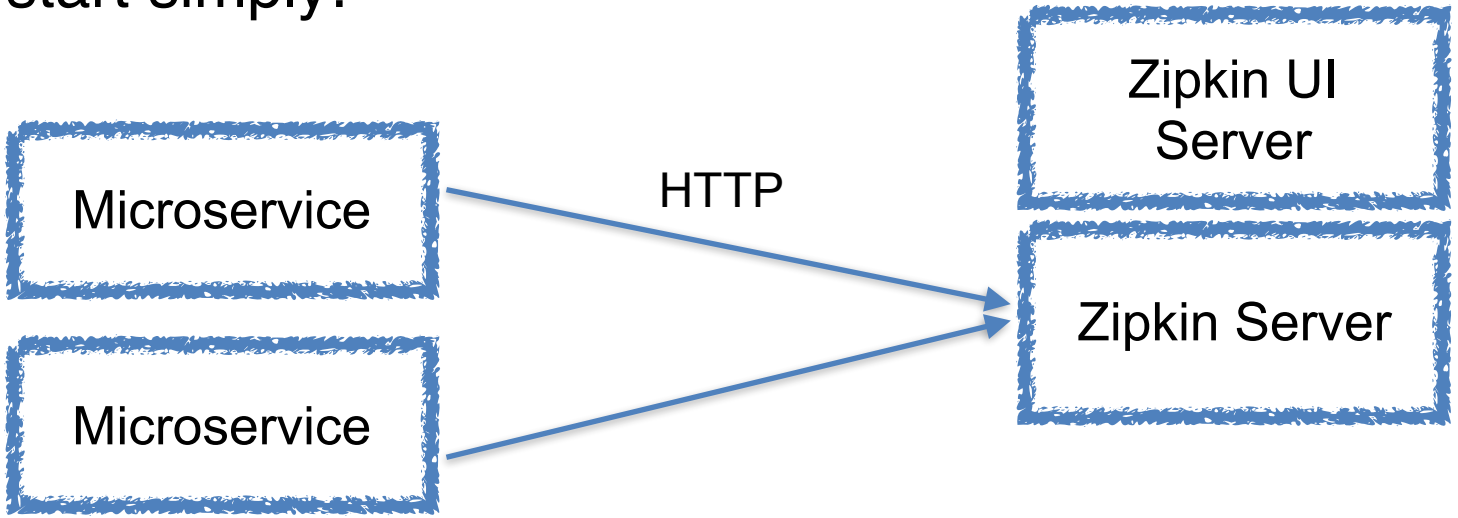
4. How do your microservices look like?

Sleuth and Zipkin dependencies

Name on Initializr	artifactId
Sleuth	spring-cloud-starter-sleuth
Sleuth Stream	spring-cloud-sleuth-stream
Zipkin Client	spring-cloud-starter-zipkin
Zipkin Stream	spring-cloud-sleuth-zipkin-stream
Zipkin UI	spring-boot-starter zipkin-autoconfigure-ui
Zipkin Server	spring-boot-starter zipkin-server

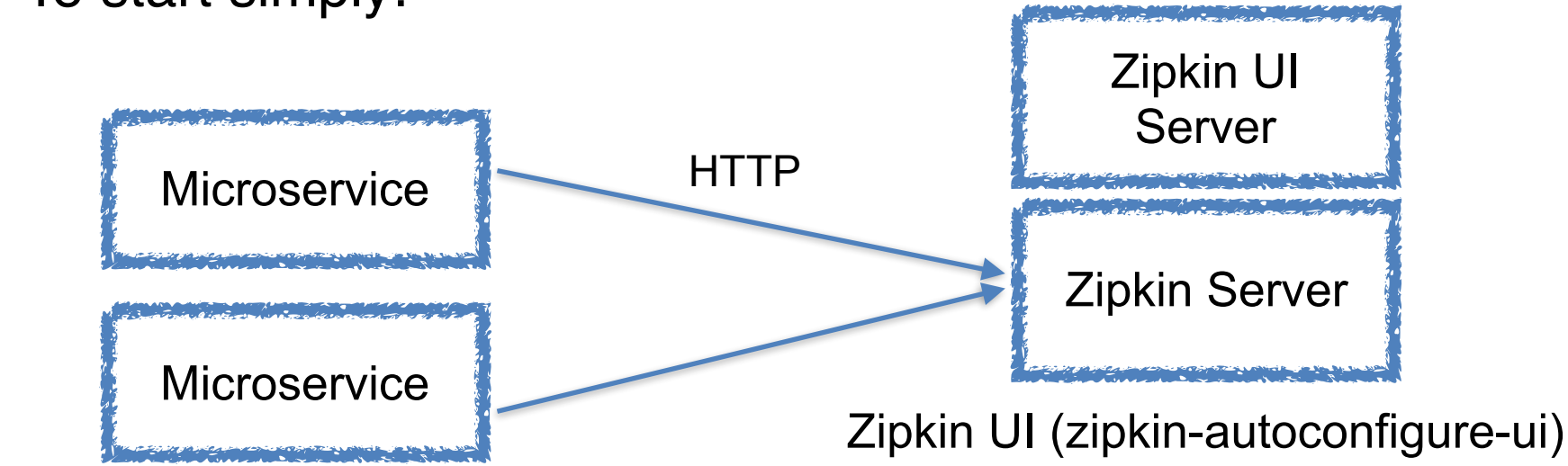
4. How do your microservices look like?

To start simply:



4. How do your microservices look like?

To start simply:



4. How do your microservices look like?

To use RabbitMQ as a middle buffer:

4. How do your microservices look like?

To use RabbitMQ as a middle buffer:

Sleuth Stream
(spring-cloud-sleuth-stream)
Stream Rabbit
(spring-cloud-starter-stream-rabbit)

Zipkin UI
(zipkin-autoconfigure-ui)
Zipkin Stream
(spring-cloud-sleuth-zipkin-stream)
Stream Rabbit
(spring-cloud-starter-stream-rabbit)

4. How do your microservices look like?

To use RabbitMQ as a middle buffer:

Sleuth Stream
(spring-cloud-sleuth-stream)

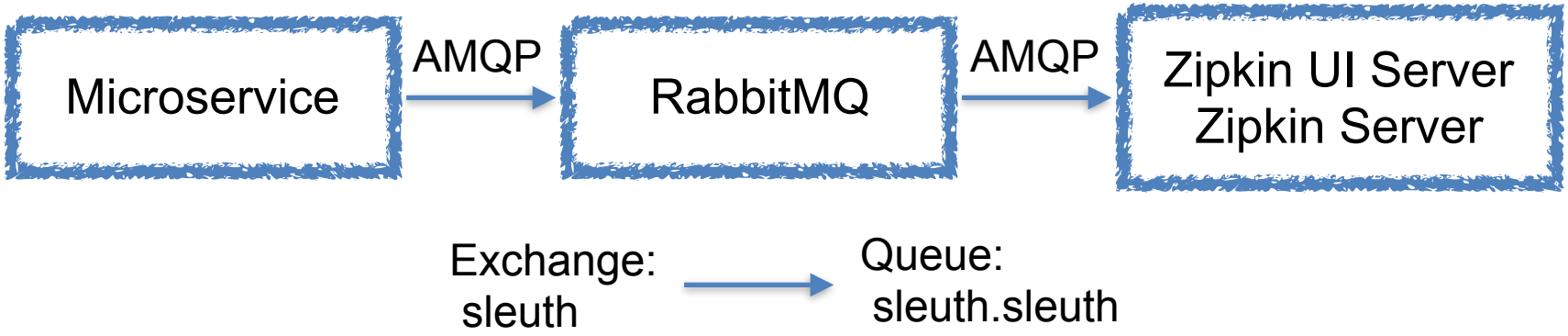
spring-cloud-stream-**binder**-rabbit

Zipkin UI
(zipkin-autoconfigure-ui)
Zipkin Stream
(spring-cloud-sleuth-zipkin-stream)

spring-cloud-stream-**binder**-rabbit

4. How do your microservices look like?

The exchange and the queue are created and bound automatically.



4. How do your microservices look like?

Sleuth and Zipkin dependencies

Name on Initializr	artifactId	HTTP	AMQP
Sleuth	spring-cloud-starter-sleuth		
Sleuth Stream	spring-cloud-sleuth-stream		Producer
Zipkin Client	spring-cloud-starter-zipkin	Client	
Zipkin Stream	spring-cloud-sleuth-zipkin-stream		Consumer
Zipkin UI	spring-boot-starter zipkin-autoconfigure-ui	Server	Consumer
Zipkin Server	spring-boot-starter zipkin-server	Server	
Stream Rabbit	spring-cloud-starter-stream-rabbit		Both

4. How do your microservices look like?

Sleuth and Zipkin dependencies

Name on Initializr	artifactId	HTTP	AMQP
Sleuth	spring-cloud-starter-sleuth		

Adds trace and span ids to the logs

4. How do your microservices look like?

Zipkin server application (HTTP):

```
@SpringBootApplication
@EnableZipkinServer
public class ZipkinApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZipkinApplication.class, args);
    }
}
```

4. How do your microservices look like?

Zipkin server application (Stream):

```
@SpringBootApplication
@EnableZipkinStreamServer
public class ZipkinApplication {
    public static void main(String[] args) {
        SpringApplication.run(ZipkinApplication.class, args);
    }
}
```

4. How do your microservices look like?

No source code modification in microservices!

4. How do your microservices look like?

4. How do your microservices look like?

- Note
  (1) Spring Cloud Sleuth does NOT support RabbitTemplate (spring-boot-starter-amqp).
  (2) It supports Stream Rabbit (spring-cloud-starter-stream-rabbit).

#5

Do your services contribute to your business?

5. Do your services contribute to your business?

1. What to do?
   - Collect access logs in Elasticsearch.
   - Session ID or user id should be contained in the access log.
   - Visualize logs using Kibana.

5. Do your services contribute to your business?

5. Do your services contribute to your business?

2. Why do this?
   (1) To see how your system contribute to.
   (2) To make your boss or manager
   to invest more $$$ to the developers!

At last

Summary

Summary

- Use server monitoring and process monitoring.
- Gathering the system performance and business performance into one dashboard shows the relationship of them!
- Zipkin is cool, enough cool.

By the way,

Yesterday I found a big news.

By the way,

The Netflix Tech Blog: Vizceral Open Source

http://techblog.netflix.com/2016/08/vizceral-open-source.html

Let's have fun visualization your microservices!

Acroquest Technology

Infrastructures Evolution

Summary

- Sample code: https://github.com/cero-t/e-commerce-example
- Clip arts: http://www.irasutoya.com/