Open Source Application Performance Monitoring (APM) Tools

Dr. Andreas Brunnert
RETIT GmbH

Motivation

The amount of open source** APM tools* for Java has grown dramatically in the last years:

PinPoint

Jaeger

Zipkin

inspectIT

Glowroot

elasticAPM

Apache Skywalking

Stage-monitor

Haystack

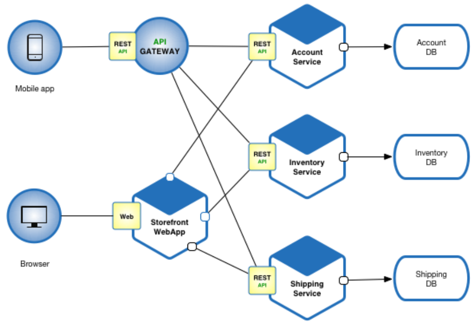
JavaMelody

* Tools that are available as open source on Github and support the storage, processing and visualization of application traces and (optionally) metrics
** All these tools are available under the Apache License 2.0 (Elastic APM also applies the Elastic License)

Motivation

Motivation

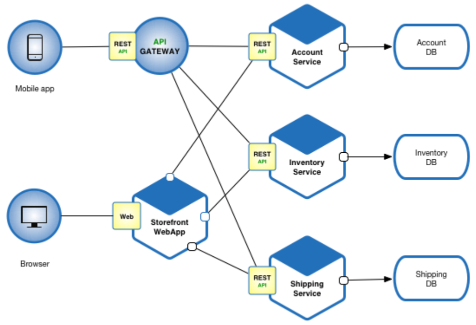
Complexity increase in modern software systems



Services might need to interact with each other in ways that might not be obvious at the time of development or deployment.

Motivation

Complexity increase in modern software systems



Services might need to interact with each other in ways that might not be obvious at the time of development or deployment.

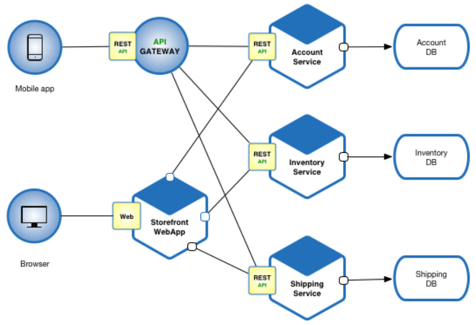
Growing importance of IT for more business models

100 MILLISECONDS
amazon.com
Increased revenue by 1% for every 100 milliseconds of improvement.
SOURCE: Amazon
$

Downtimes or bad software performance have a direct impact on revenue.

Motivation

Complexity increase in modern software systems



Services might need to interact with each other in ways that might not be obvious at the time of development or deployment.

Growing importance of IT for more business models

100
MILLISECONDS
amazon.com
Increased revenue by 1% for every 100 milliseconds of improvement.
SOURCE: Amazon
$

Downtimes or bad software performance have a direct impact on revenue.

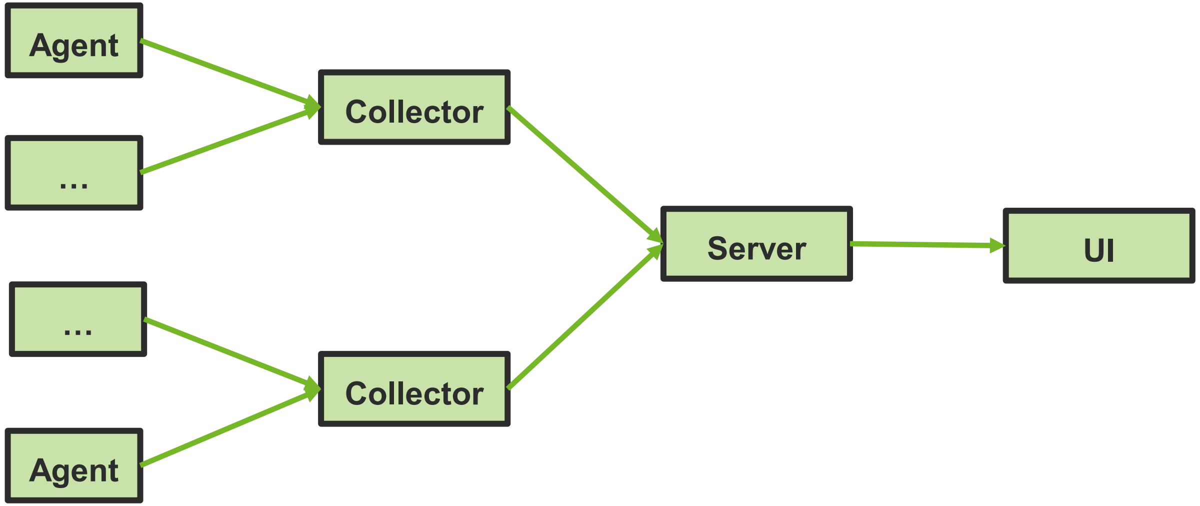
Development of tracing standards

OPENTRACING OpenCensus

OpenTelemetry

Which allow to easily exchange the tracing tool in use. Furthermore, they reduce the effort for each vendor.

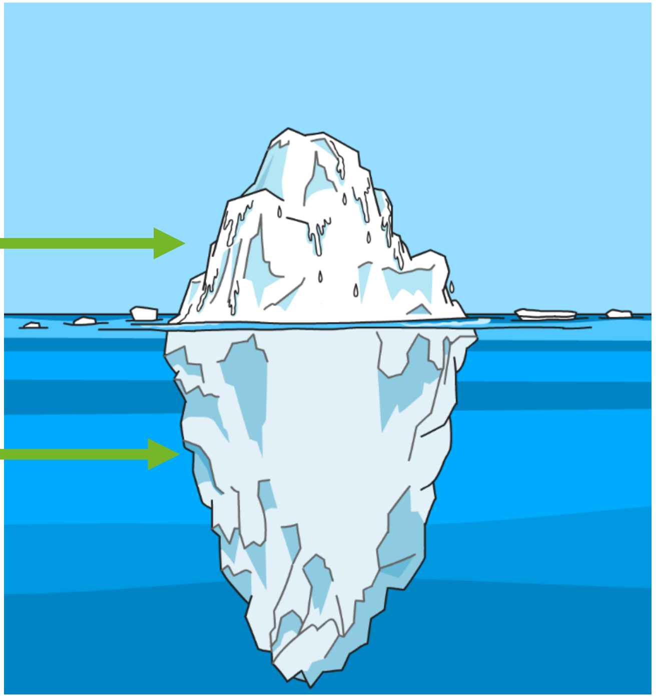
Context - Anatomy of an APM Solution



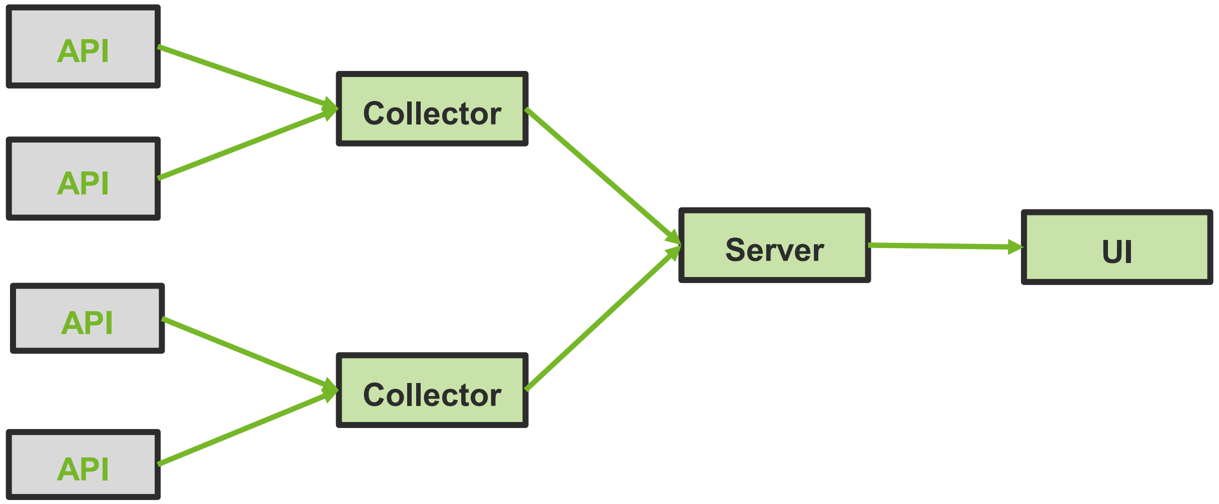
Context - Code and Effort Distribution

UI + Server + Collectors

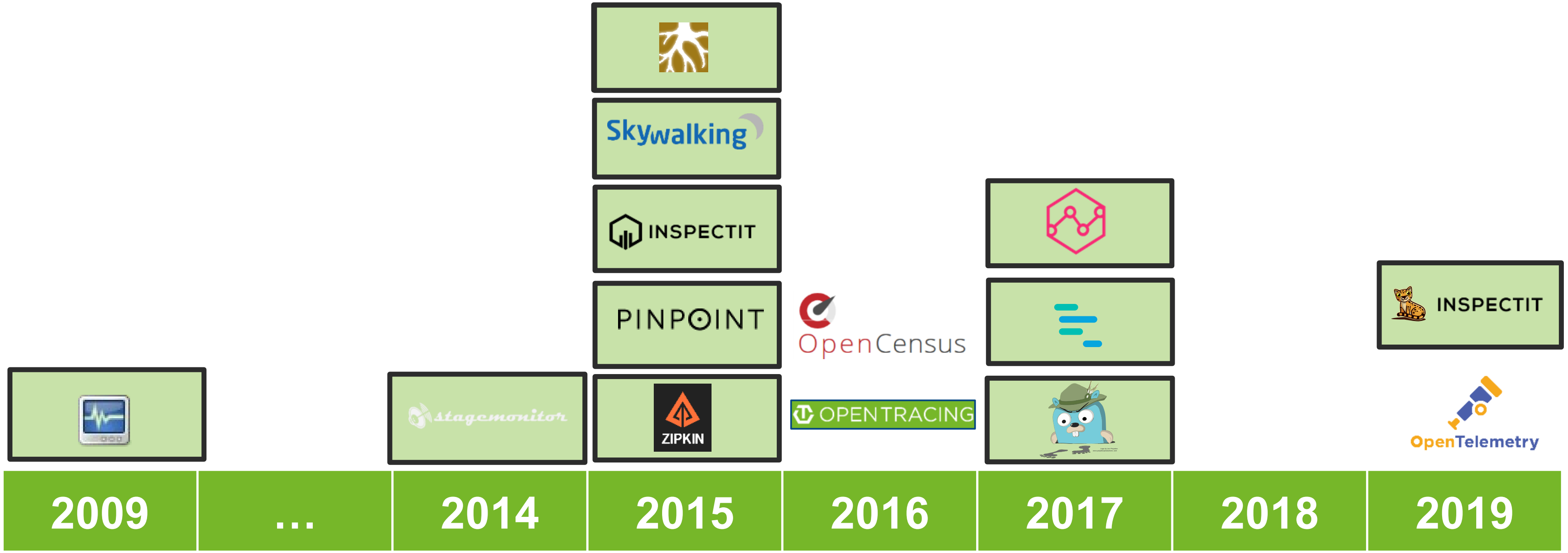
Agents



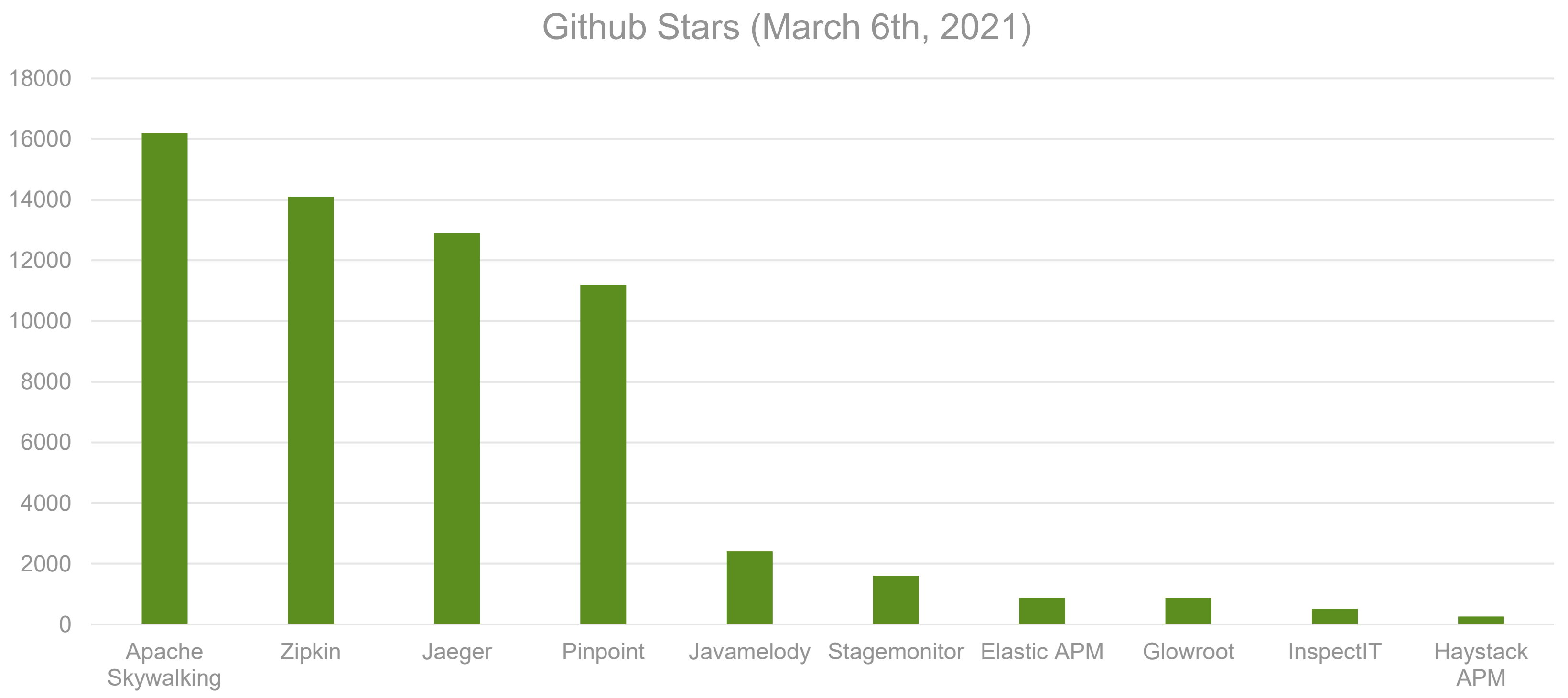
Context - Scope of Open Source APM Solutions



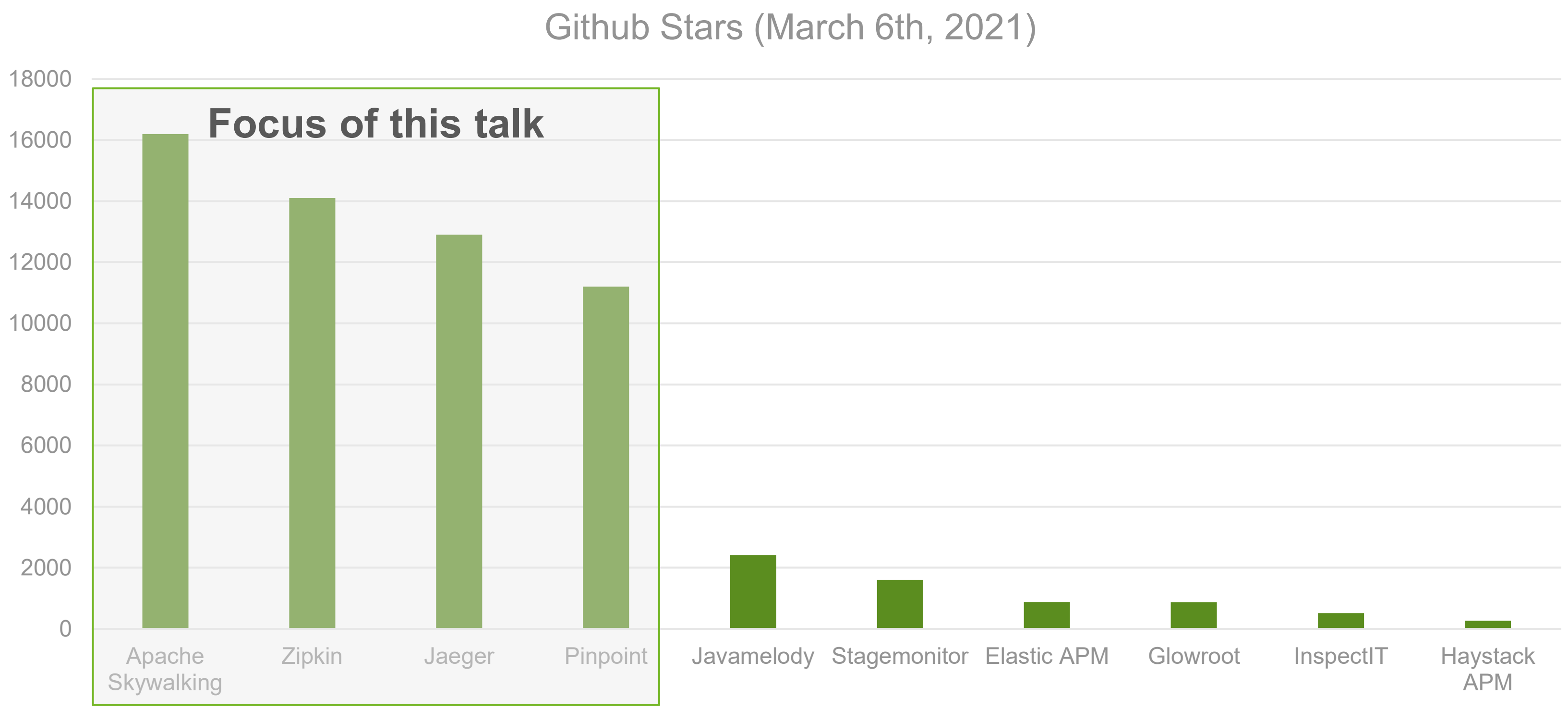
A brief timeline of tool availability



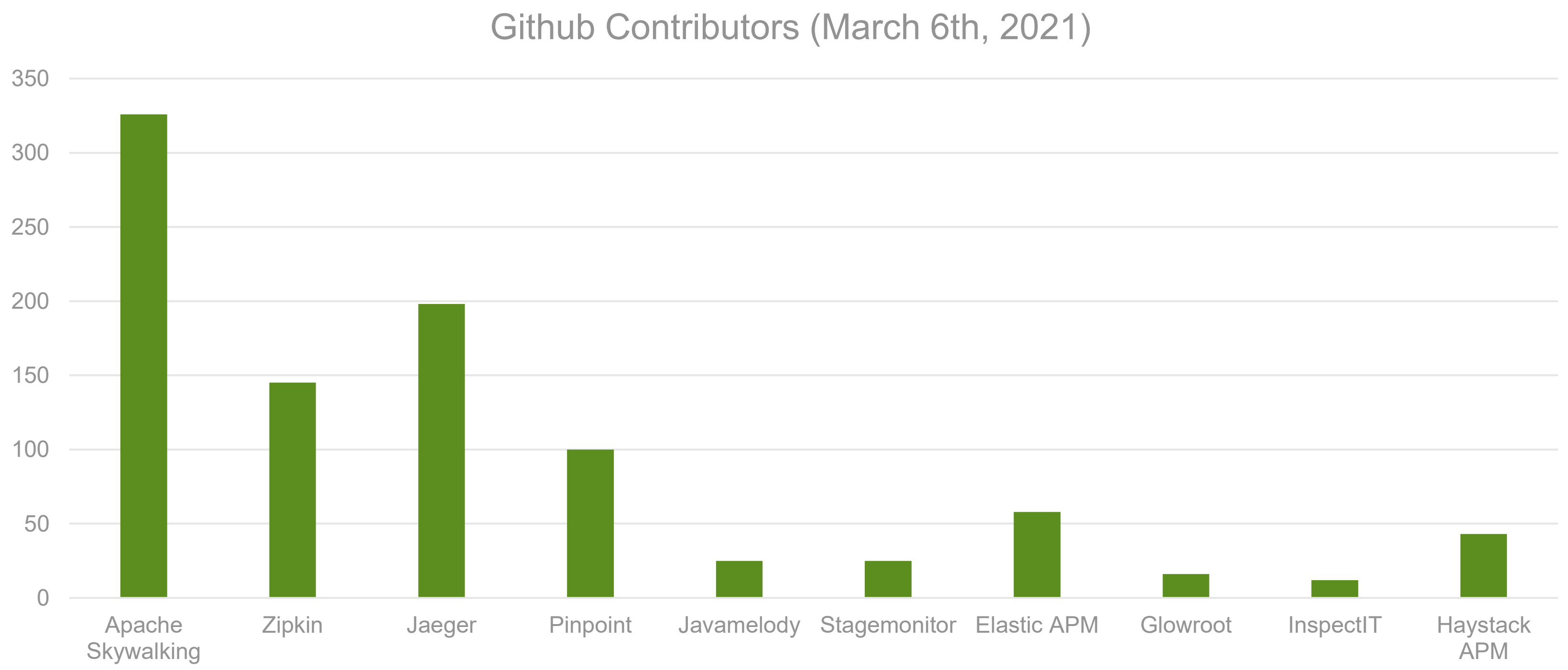
A ranking of GitHub stars



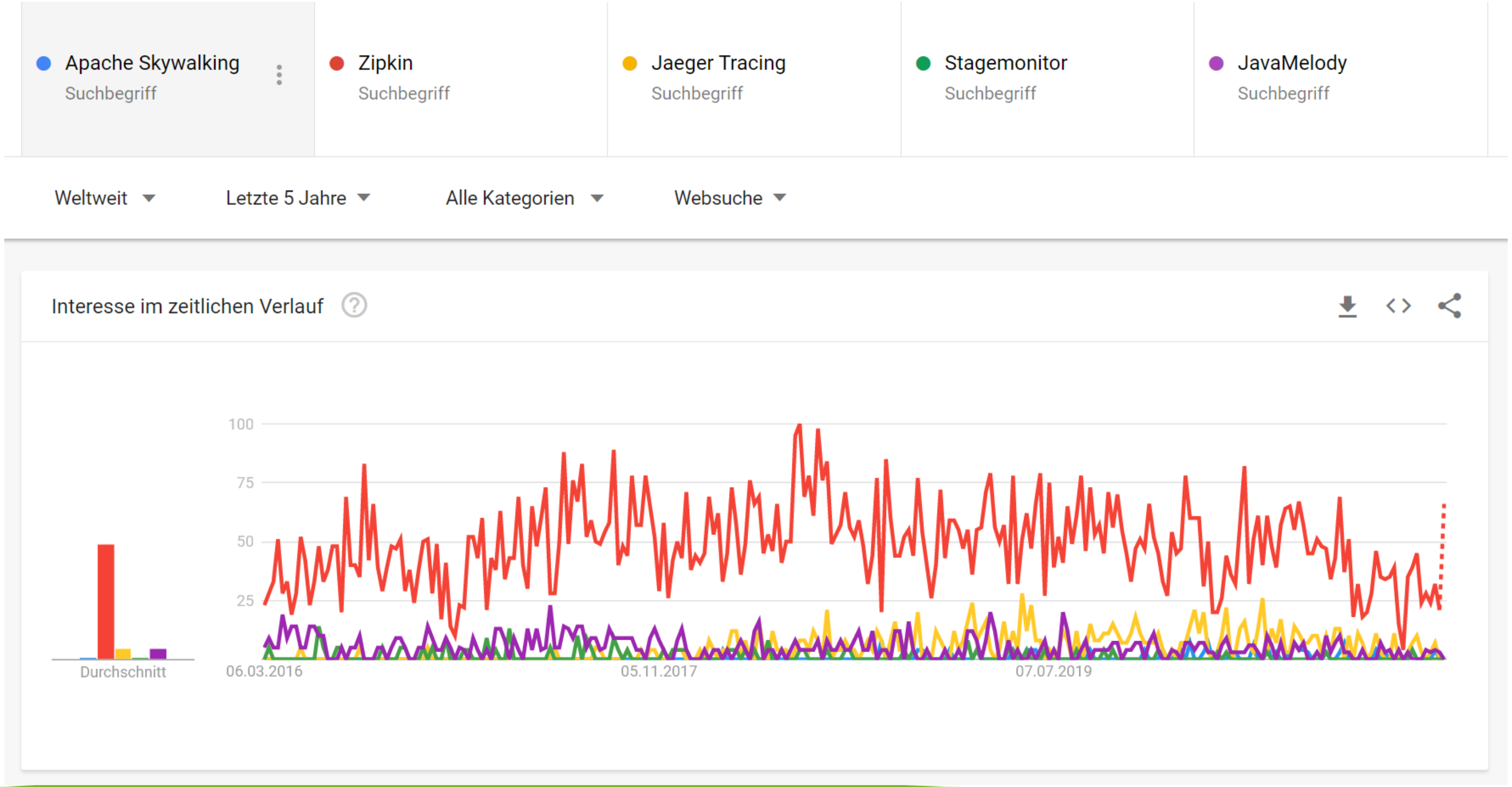
A ranking of GitHub stars



A ranking of GitHub contributors



Google Trends Analysis



● Apache Skywalking
Suchbegriff

● Zipkin
Suchbegriff

● Jaeger Tracing
Suchbegriff

● Stagemonitor
Suchbegriff

● JavaMelody
Suchbegriff

Weltweit ▾ Letzte 5 Jahre ▾ Alle Kategorien ▾ Websuche ▾

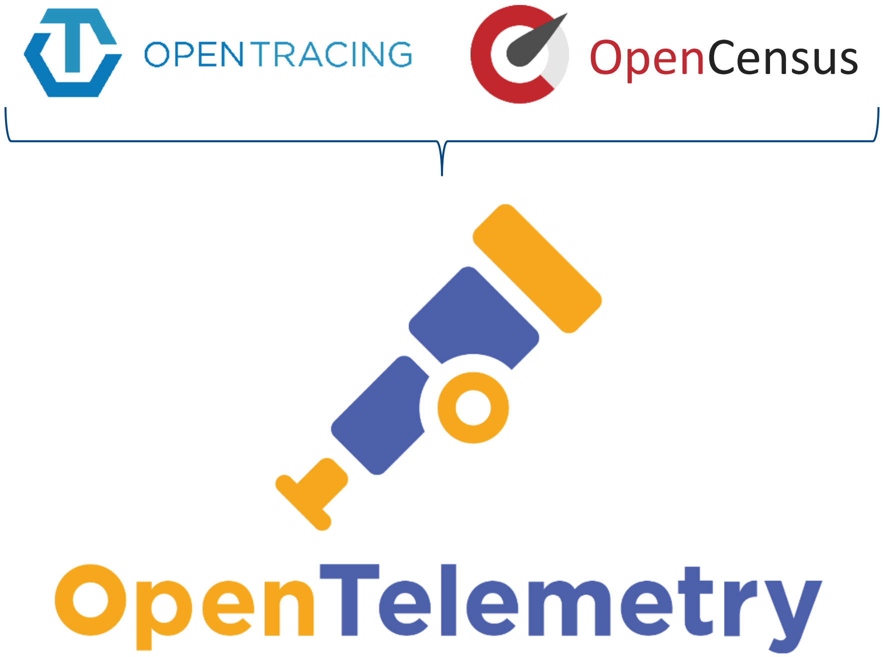
Interesse im zeitlichen Verlauf

Open Source “Standards”

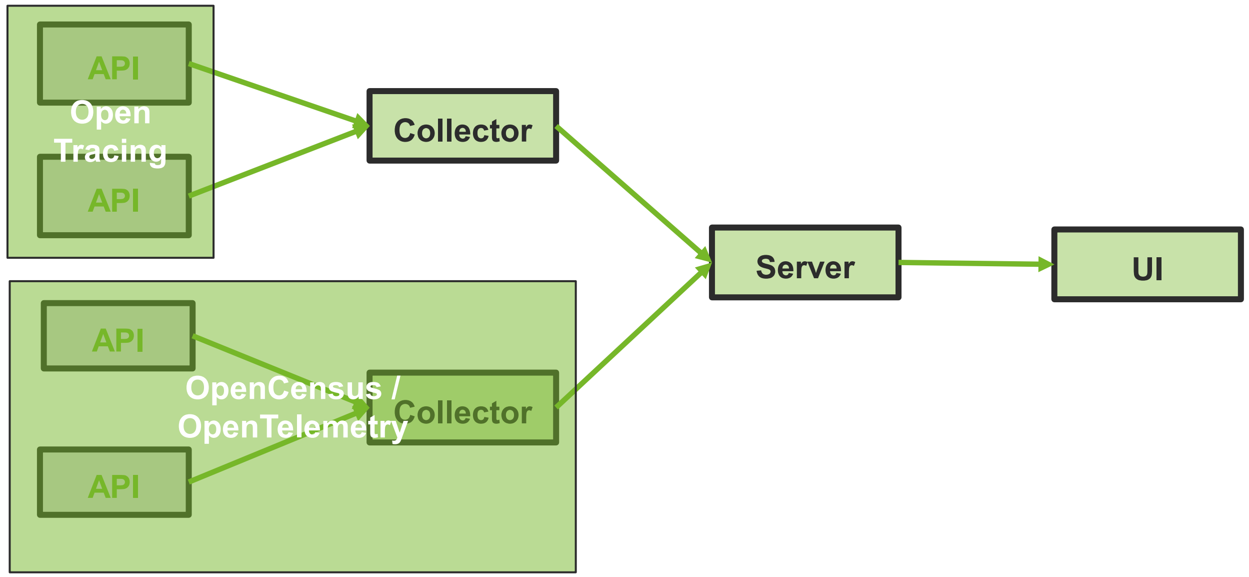
https://opentelemetry.io

- Observability Framework
  - Traces
  - Metrics
  - (Logs – still in incubation)
- Collect telemetry data, forward to analysis tool
- One API and SDK per language
- Licensing: Apache 2.0
- RETIT blog post on OpenCensus & OpenTracing:

  https://www.retit.de/open-application-performance-monitoring-apm-standards-opentracing-and-opencensus-2



Open Source “Standards” - Scope



OpenTelemetry Java Auto-Instrumentation

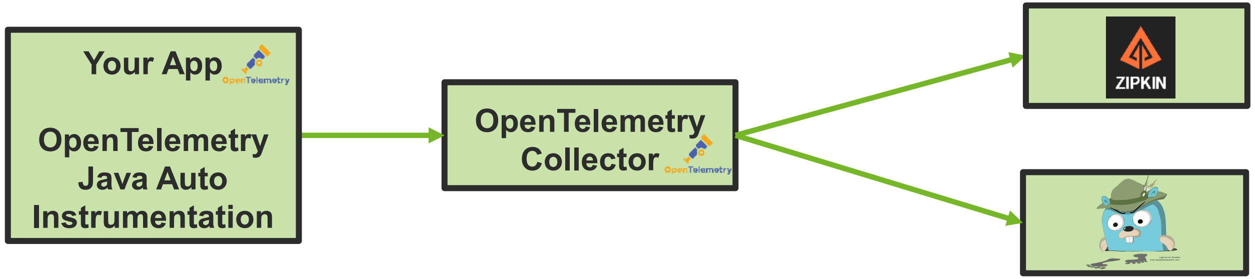
For Java-Enterprise Applications (e.g., based on Java EE,Spring, Quarkus) it is no longer required to manually instrument your application ([1], 1.0.0 release on March 6th, 2021):

- Supports all major frameworks in the Java Enterprise Space [2]

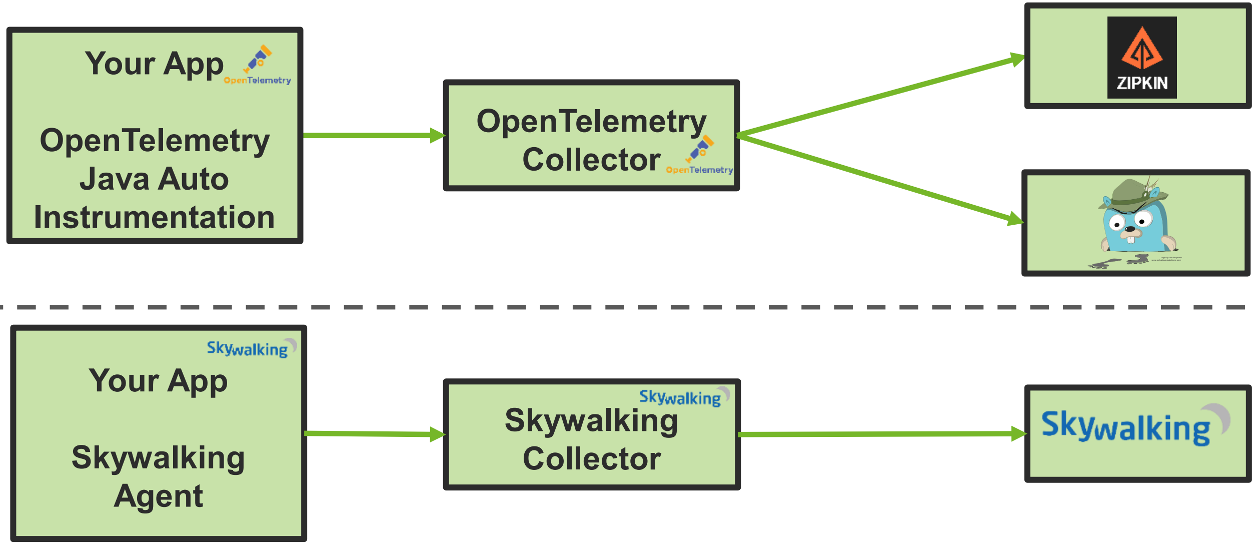
```
java -javaagent:path/to/opentelemetry-javaagent-all.jar -jar myapp.jar
```

[1] https://github.com/open-telemetry/opentelemetry-java-instrumentation
[2] https://github.com/open-telemetry/opentelemetry-java-instrumentation/blob/main/docs/supported-libraries.md

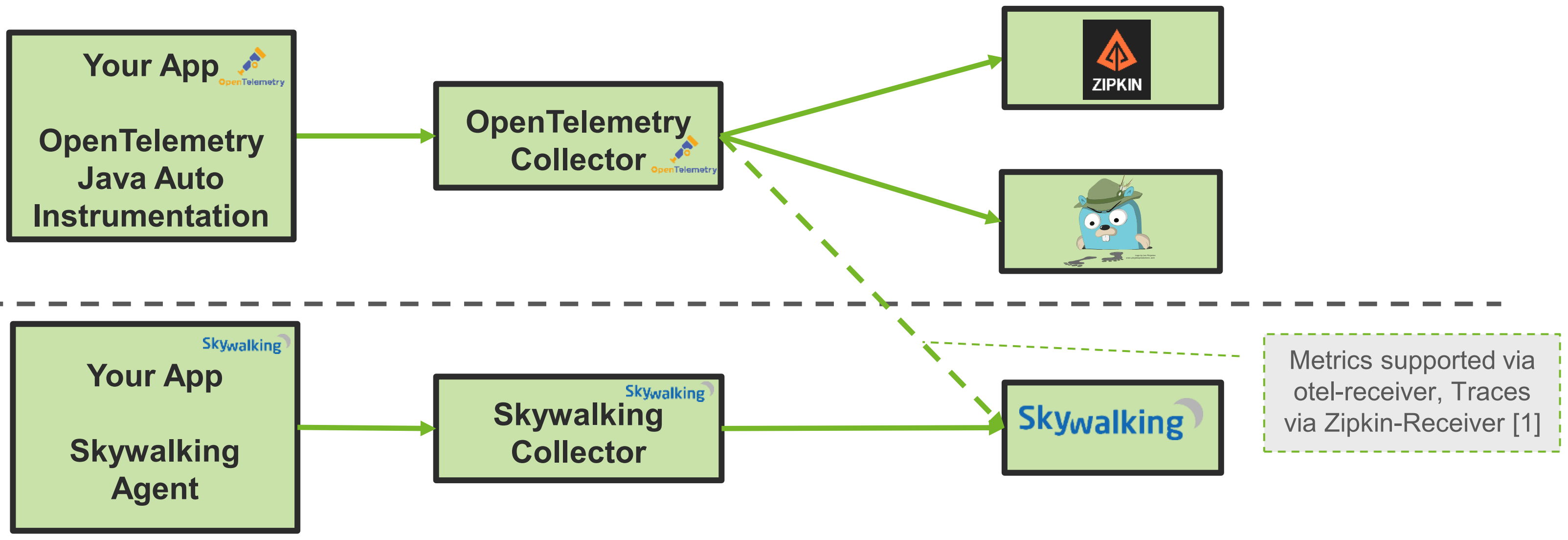
Possible Setups to get Started



Possible Setups to get Started

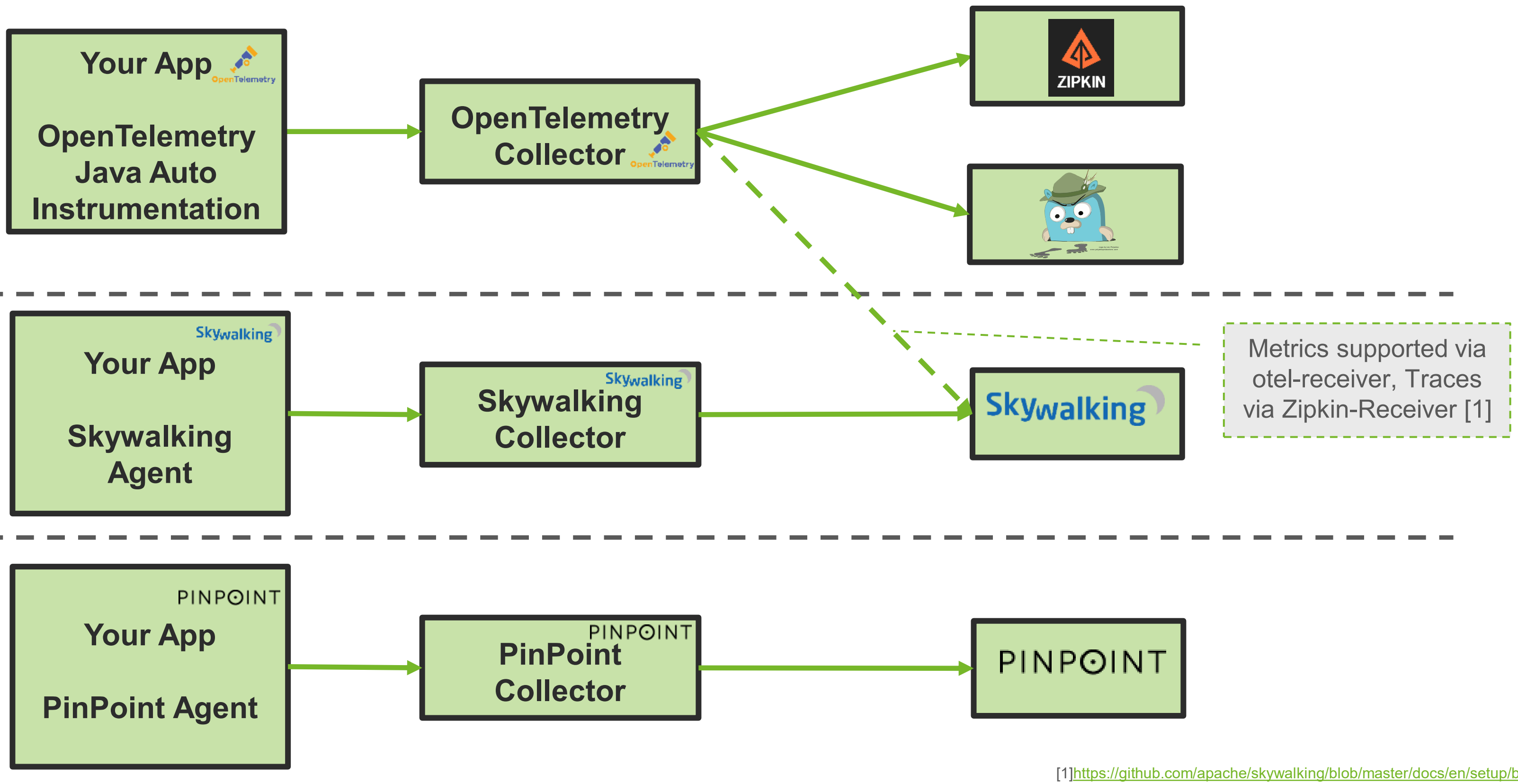


Possible Setups to get Started



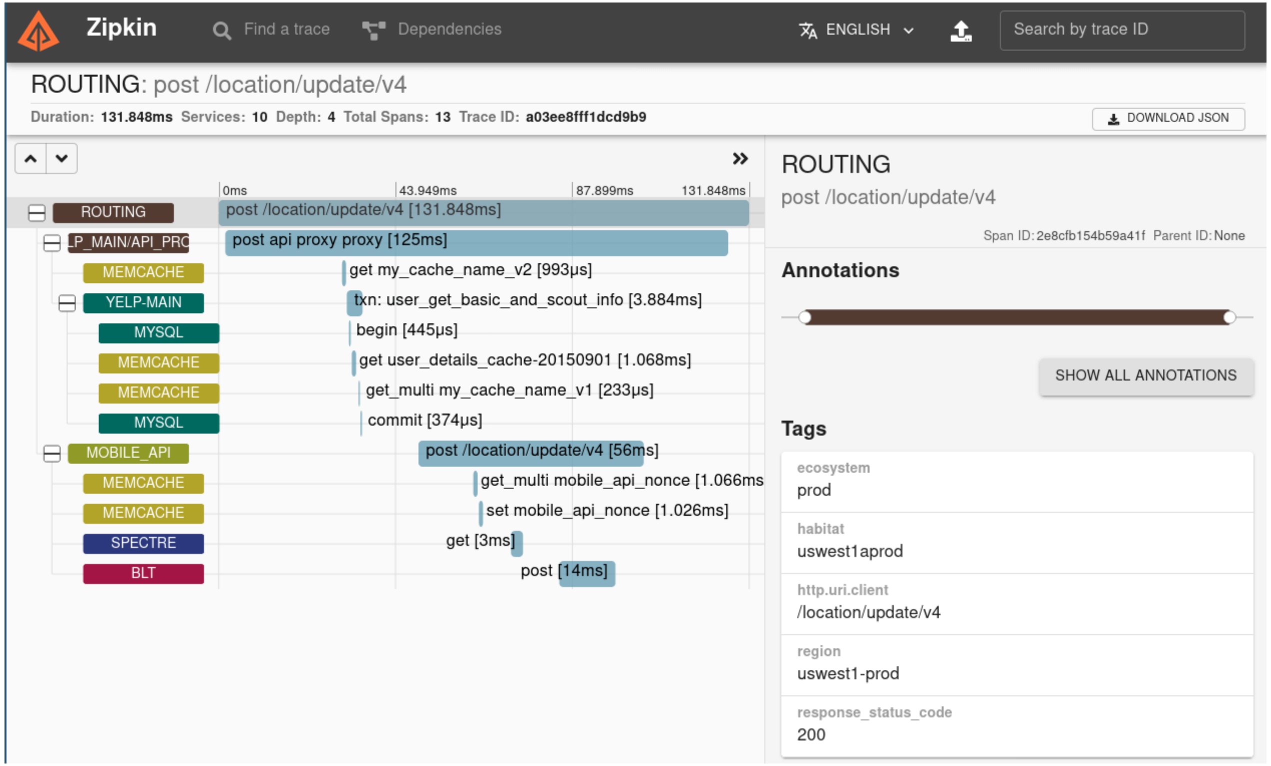
[1]https://github.com/apache/skywalking/blob/master/docs/en/setup/backend/backend-receivers.md#opentelemetry-receiver

Possible Setups to get Started



[1]https://github.com/apache/skywalking/blob/master/docs/en/setup/backend/backend-receivers.md#opentelemetry-receiver

ZIPKIN (https://zipkin.io/)



ROUTING: post /location/update/v4

Duration: 131.848ms Services: 10 Depth: 4 Total Spans: 13 Trace ID: a03ee8fff1dcd9b9

0ms | 43.949ms | 87.899ms | 131.848ms

ROUTING	post /location/update/v4 [131.848ms]
LP_MAIN/API_PRC	post api proxy proxy [125ms]
MEMCACHE	get my_cache_name_v2 [993μs]
YELP-MAIN	txn: user_get_basic_and_scout_info [3.884ms]
MYSQL	begin [445μs]
MEMCACHE	get user_details_cache-20150901 [1.068ms]
MEMCACHE	get_multi my_cache_name_v1 [233μs]
MYSQL	commit [374μs]
MOBILE_API	post /location/update/v4 [56ms]
MEMCACHE	get_multi mobile_api_nonce [1.066ms
MEMCACHE	set mobile_api_nonce [1.026ms]
SPECTRE	get [3ms]
BLT	post [14ms]

ROUTING

post /location/update/v4

Span ID: 2e8cfb154b59a41f Parent ID: None

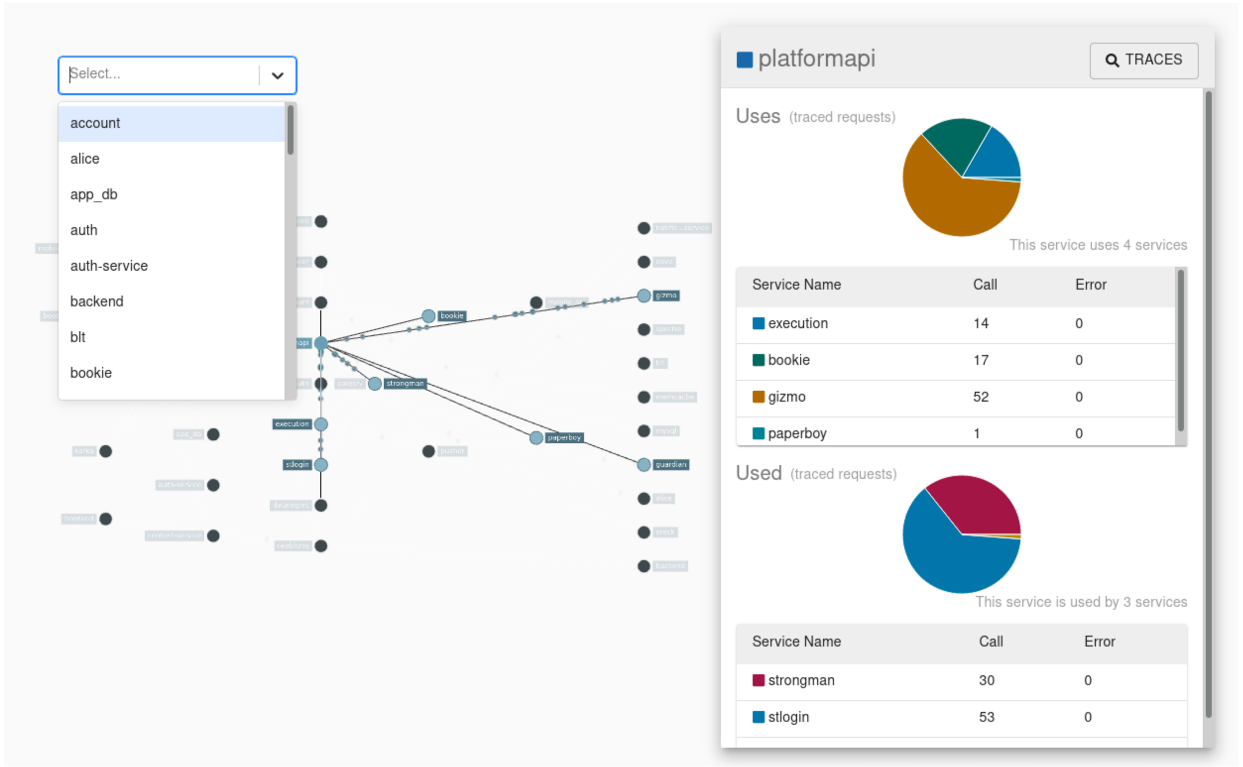
Annotations

Tags

ecosystem	prod
habitat	uswest1aprod
http.uri.client	/location/update/v4
region	uswest1-prod
response_status_code	200

Trace Details - Source: https://zipkin.io/public/img/web-screenshot.png

ZIPKIN (https://zipkin.io/)



Uses (traced requests)

This service uses 4 services

Service Name	Call	Error
🟦 execution	14	0
🟩 bookie	17	0
🟧 gizmo	52	0
🟦 paperboy	1	0

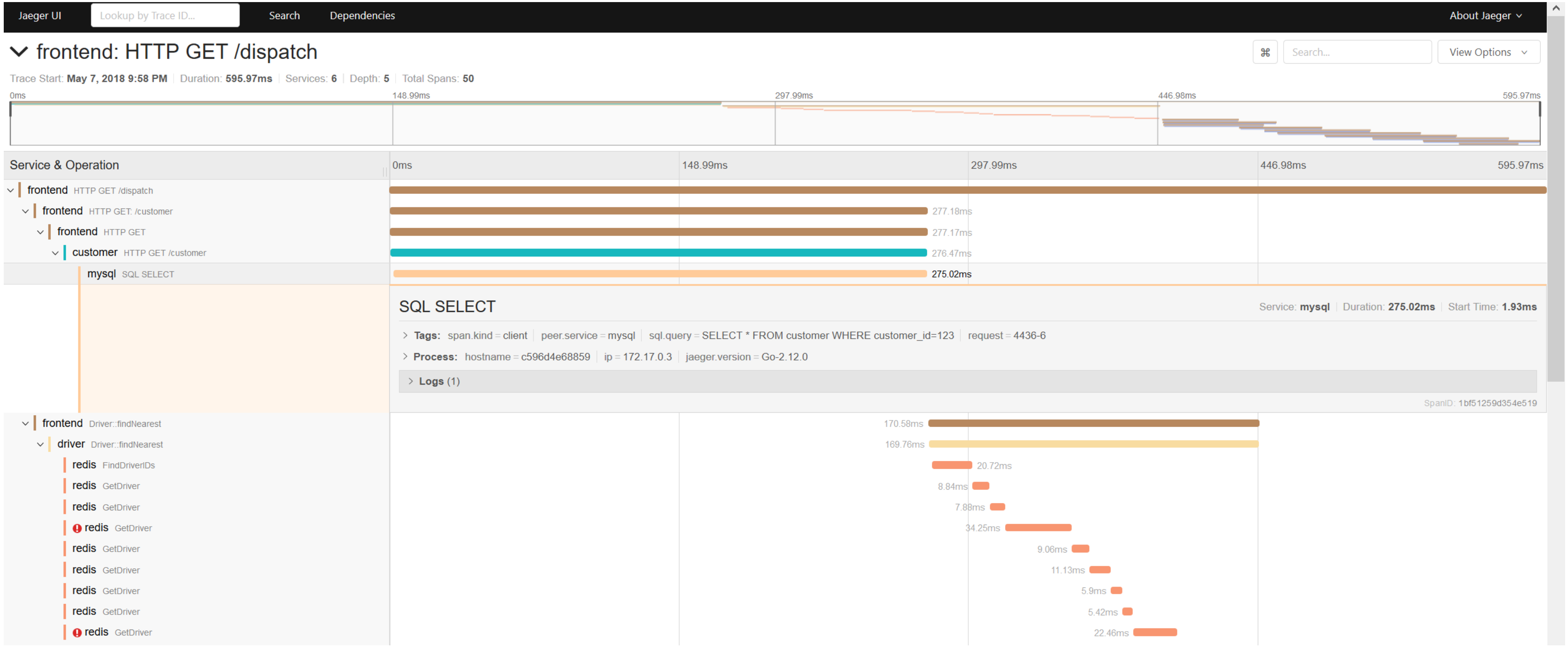
Used (traced requests)

This service is used by 3 services

Service Name	Call	Error
🟥 strongman	30	0
🟦 stlogin	53	0

Dependency Graph Source: https://zipkin.io/public/img/dependency-graph.png

Jaeger (https://www.jaegertracing.io/)



Jaeger UI | Lookup by Trace ID... | Search | Dependencies | About Jaeger

frontend: HTTP GET /dispatch

Trace Start: **May 7, 2018 9:58 PM** | Duration: **595.97ms** | Services: **6** | Depth: **5** | Total Spans: **50**

Service & Operation	Duration
frontend HTTP GET /dispatch	
frontend HTTP GET: /customer	277.18ms
frontend HTTP GET	277.17ms
customer HTTP GET /customer	276.47ms
mysql SQL SELECT	275.02ms
frontend Driver::findNearest	170.58ms
driver Driver::findNearest	169.76ms
redis FindDriverIDs	20.72ms
redis GetDriver	8.84ms
redis GetDriver	7.88ms
redis GetDriver	34.25ms
redis GetDriver	9.06ms
redis GetDriver	11.13ms
redis GetDriver	5.9ms
redis GetDriver	5.42ms
redis GetDriver	22.46ms

SQL SELECT — Service: **mysql** | Duration: **275.02ms** | Start Time: **1.93ms**

Tags: span.kind = client | peer.service = mysql | sql.query = SELECT * FROM customer WHERE customer_id=123 | request = 4436-6

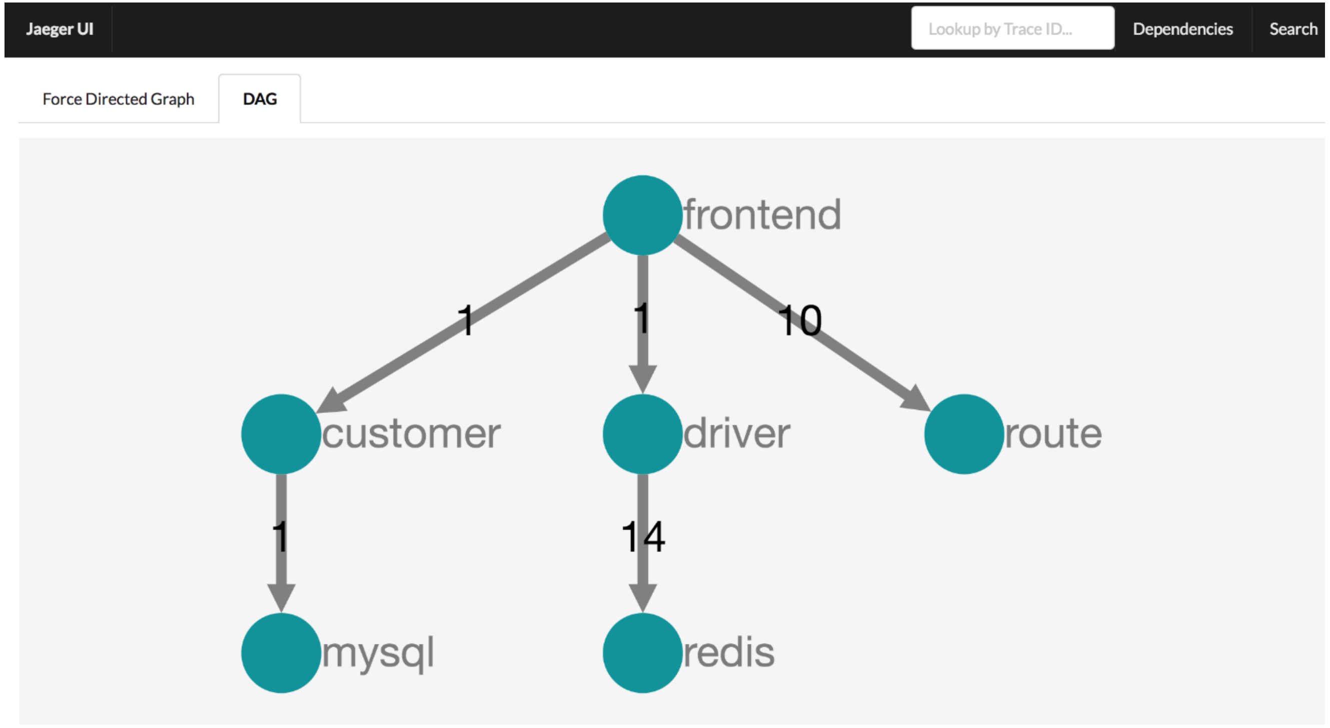
Process: hostname = c596d4e68859 | ip = 172.17.0.3 | jaeger.version = Go-2.12.0

Logs (1)

SpanID: 1bf51259d354e519

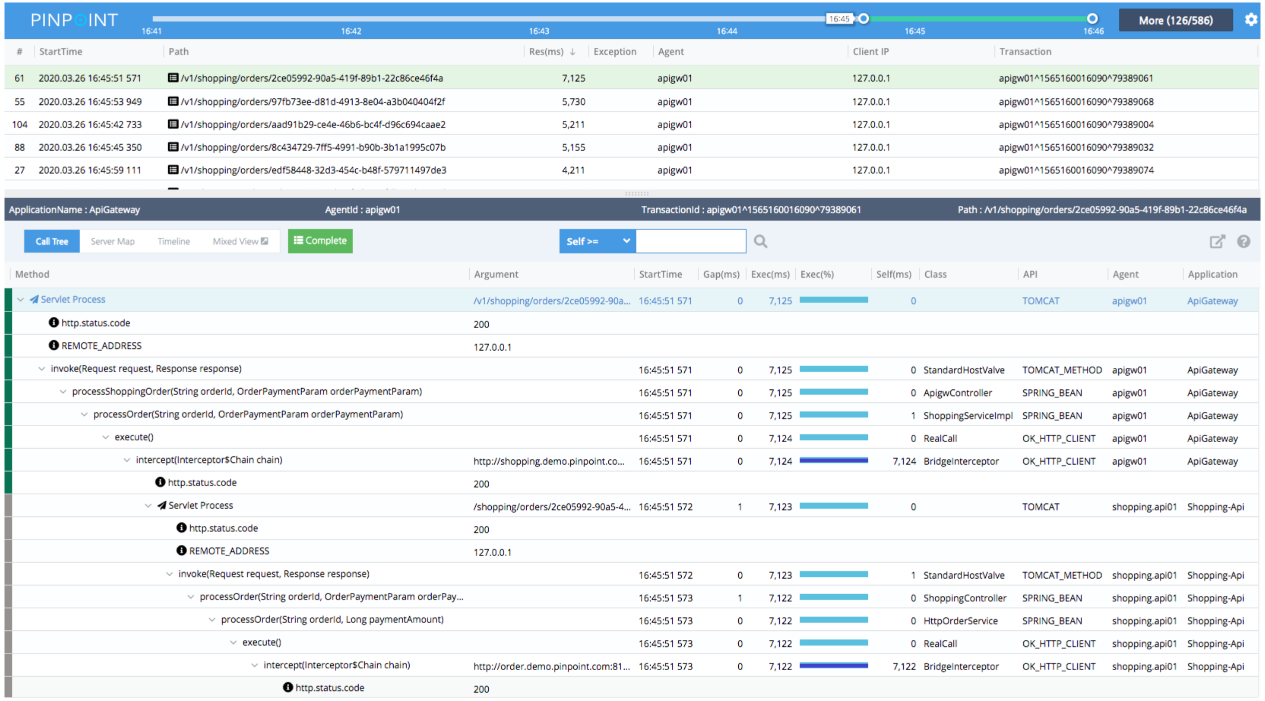
Trace Details Source: https://medium.com/opentracing/take-opentracing-for-a-hotrod-ride-f6e3141f7941

Jaeger (https://www.jaegertracing.io/)



Dependency Graph Source: https://medium.com/opentracing/take-opentracing-for-a-hotrod-ride-f6e3141f7941

PINPOINT (https://pinpoint-apm.github.io/pinpoint/)



PINPOINT — More (126/586)

#	StartTime	Path	Res(ms) ↓	Exception	Agent	Client IP	Transaction
61	2020.03.26 16:45:51 571	/v1/shopping/orders/2ce05992-90a5-419f-89b1-22c86ce46f4a	7,125		apigw01	127.0.0.1	apigw01^1565160016090^79389061
55	2020.03.26 16:45:53 949	/v1/shopping/orders/97fb73ee-d81d-4913-8e04-a3b040404f2f	5,730		apigw01	127.0.0.1	apigw01^1565160016090^79389068
104	2020.03.26 16:45:42 733	/v1/shopping/orders/aad91b29-ce4e-46b6-bc4f-d96c694caae2	5,211		apigw01	127.0.0.1	apigw01^1565160016090^79389004
88	2020.03.26 16:45:45 350	/v1/shopping/orders/8c434729-7ff5-4991-b90b-3b1a1995c07b	5,155		apigw01	127.0.0.1	apigw01^1565160016090^79389032
27	2020.03.26 16:45:59 111	/v1/shopping/orders/edf58448-32d3-454c-b48f-579711497de3	4,211		apigw01	127.0.0.1	apigw01^1565160016090^79389074

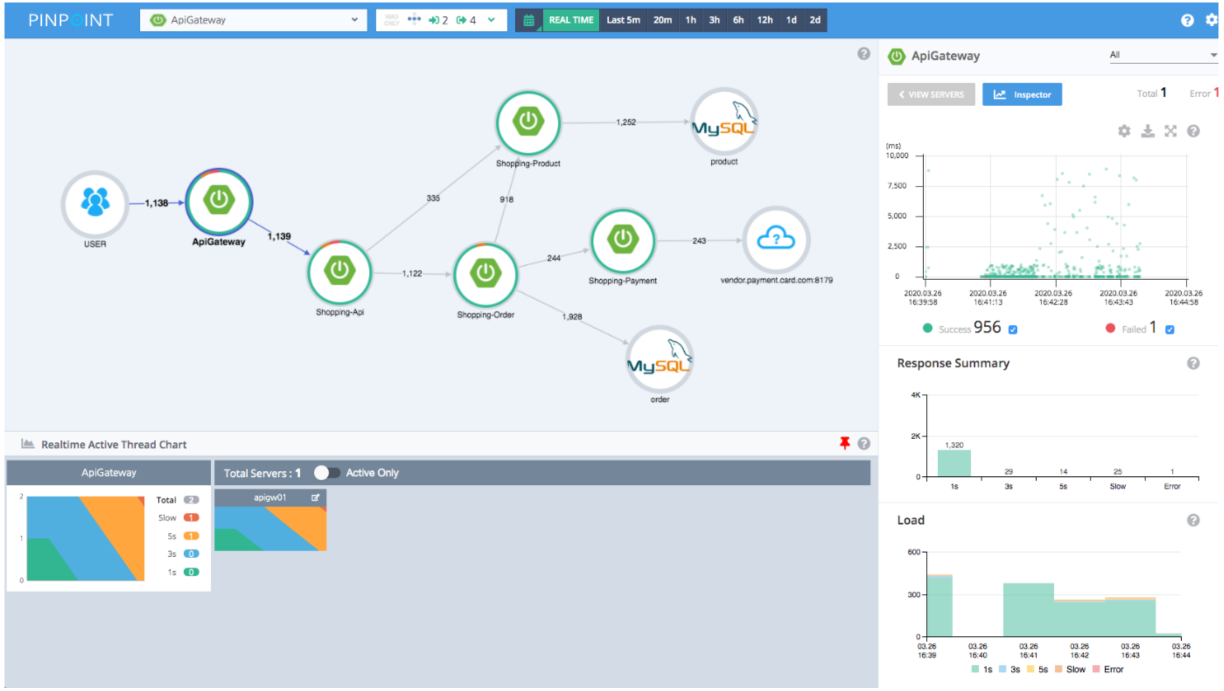
ApplicationName : ApiGateway AgentId : apigw01 TransactionId : apigw01^1565160016090^79389061 Path : /v1/shopping/orders/2ce05992-90a5-419f-89b1-22c86ce46f4a

Call Tree | Server Map | Timeline | Mixed View | Complete | Self >=

Method	Argument	StartTime	Gap(ms)	Exec(ms)	Exec(%)	Self(ms)	Class	API	Agent	Application
Servlet Process	/v1/shopping/orders/2ce05992-90a...	16:45:51 571	0	7,125		0		TOMCAT	apigw01	ApiGateway
http.status.code	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		16:45:51 571	0	7,125		0	StandardHostValve	TOMCAT_METHOD	apigw01	ApiGateway
processShoppingOrder(String orderId, OrderPaymentParam orderPaymentParam)		16:45:51 571	0	7,125		0	ApigwController	SPRING_BEAN	apigw01	ApiGateway
processOrder(String orderId, OrderPaymentParam orderPaymentParam)		16:45:51 571	0	7,125		1	ShoppingServiceImpl	SPRING_BEAN	apigw01	ApiGateway
execute()		16:45:51 571	0	7,124		0	RealCall	OK_HTTP_CLIENT	apigw01	ApiGateway
intercept(Interceptor$Chain chain)	http://shopping.demo.pinpoint.co...	16:45:51 571	0	7,124		7,124	BridgeInterceptor	OK_HTTP_CLIENT	apigw01	ApiGateway
http.status.code	200									
Servlet Process	/shopping/orders/2ce05992-90a5-4...	16:45:51 572	1	7,123		0		TOMCAT	shopping.api01	Shopping-Api
http.status.code	200									
REMOTE_ADDRESS	127.0.0.1									
invoke(Request request, Response response)		16:45:51 572	0	7,123		1	StandardHostValve	TOMCAT_METHOD	shopping.api01	Shopping-Api
processOrder(String orderId, OrderPaymentParam orderPay...		16:45:51 573	1	7,122		0	ShoppingController	SPRING_BEAN	shopping.api01	Shopping-Api
processOrder(String orderId, Long paymentAmount)		16:45:51 573	0	7,122		0	HttpOrderService	SPRING_BEAN	shopping.api01	Shopping-Api
execute()		16:45:51 573	0	7,122		0	RealCall	OK_HTTP_CLIENT	shopping.api01	Shopping-Api
intercept(Interceptor$Chain chain)	http://order.demo.pinpoint.com:81...	16:45:51 573	0	7,122		7,122	BridgeInterceptor	OK_HTTP_CLIENT	shopping.api01	Shopping-Api
http.status.code	200									

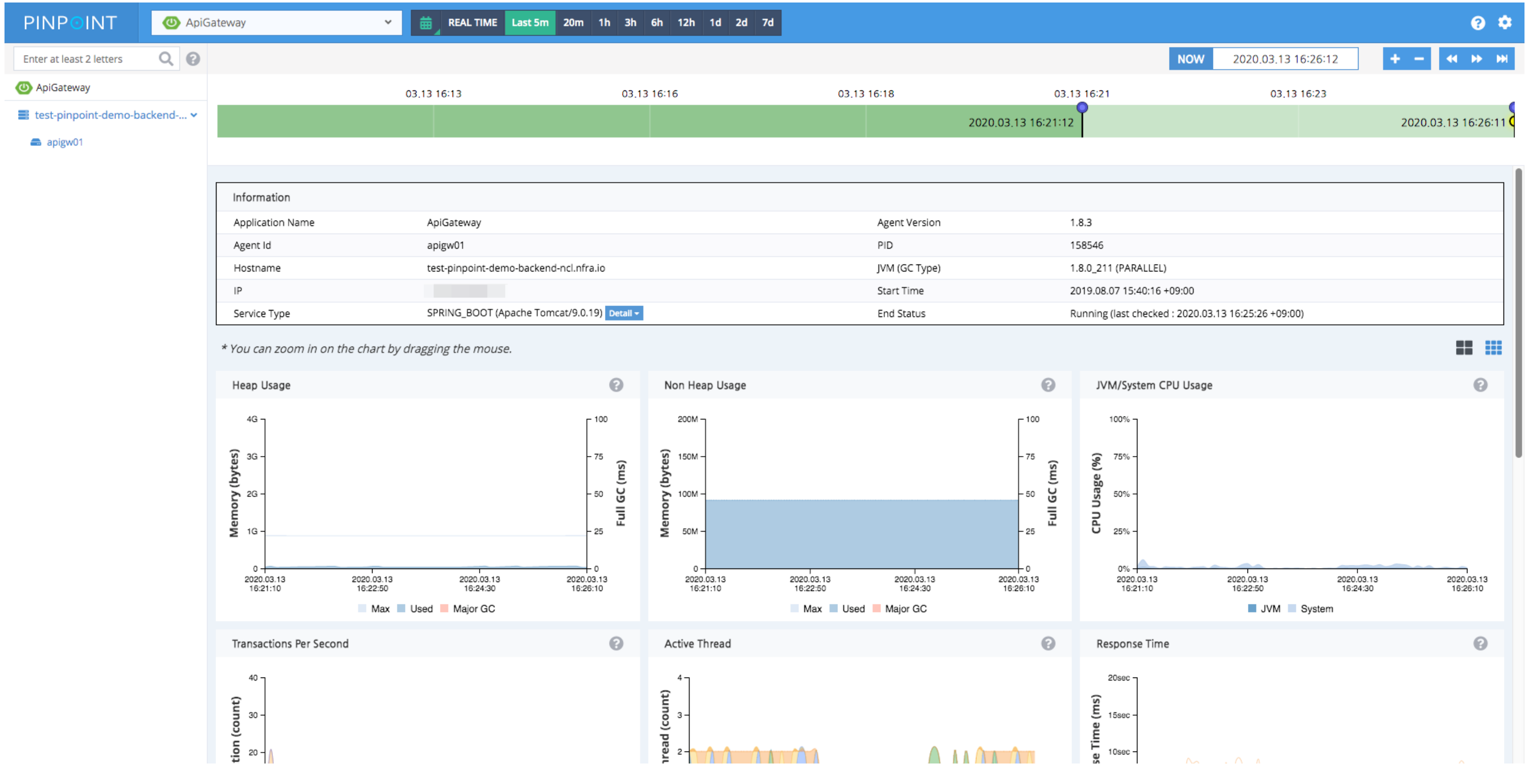
Trace Details Source: https://github.com/pinpoint-apm/pinpoint/blob/master/doc/images/ss_call-stack.png

PINPOINT (https://pinpoint-apm.github.io/pinpoint/)



Dependency Graph Source: https://github.com/pinpoint-apm/pinpoint/blob/master/doc/images/ss_server-map.png

PINPOINT (https://pinpoint-apm.github.io/pinpoint/)



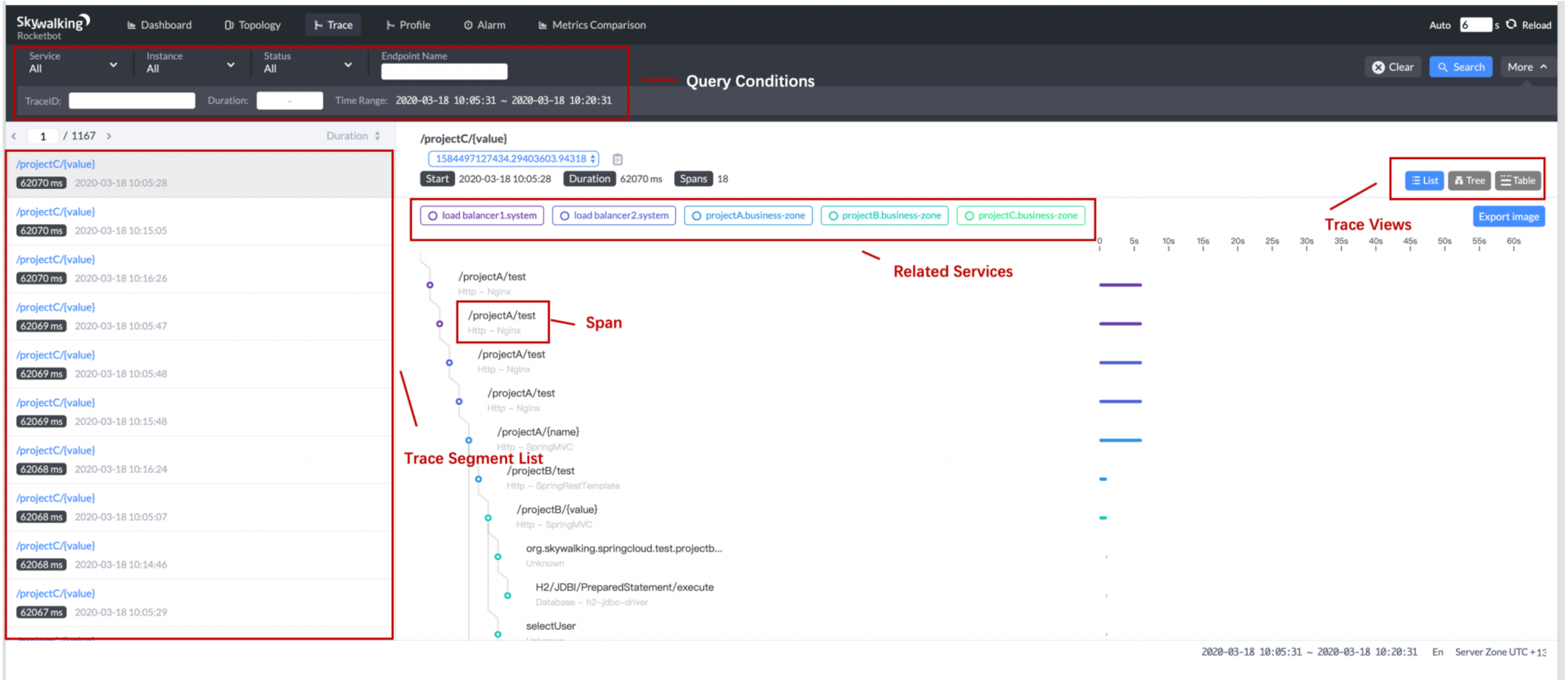
Information

Application Name	ApiGateway	Agent Version	1.8.3
Agent Id	apigw01	PID	158546
Hostname	test-pinpoint-demo-backend-ncl.nfra.io	JVM (GC Type)	1.8.0_211 (PARALLEL)
IP	[redacted]	Start Time	2019.08.07 15:40:16 +09:00
Service Type	SPRING_BOOT (Apache Tomcat/9.0.19) Detail	End Status	Running (last checked : 2020.03.13 16:25:26 +09:00)

** You can zoom in on the chart by dragging the mouse.*

Metric Details Source: https://github.com/pinpoint-apm/pinpoint/blob/master/doc/images/ss_inspector.png

Apache Skywalking (https://skywalking.apache.org/)



Service: All
Instance: All
Status: All
Endpoint Name:
TraceID:
Duration:
Time Range: 2020-03-18 10:05:31 ~ 2020-03-18 10:20:31

1 / 1167 Duration

Endpoint	Duration	Start
/projectC/{value}	62070 ms	2020-03-18 10:05:28
/projectC/{value}	62070 ms	2020-03-18 10:15:05
/projectC/{value}	62070 ms	2020-03-18 10:16:26
/projectC/{value}	62069 ms	2020-03-18 10:05:47
/projectC/{value}	62069 ms	2020-03-18 10:05:48
/projectC/{value}	62069 ms	2020-03-18 10:15:48
/projectC/{value}	62068 ms	2020-03-18 10:16:24
/projectC/{value}	62068 ms	2020-03-18 10:05:07
/projectC/{value}	62068 ms	2020-03-18 10:14:46
/projectC/{value}	62067 ms	2020-03-18 10:05:29

/projectC/{value}
1584497127434.29403603.94318
Start 2020-03-18 10:05:28 Duration 62070 ms Spans 18

load balancer1.system | load balancer2.system | projectA.business-zone | projectB.business-zone | projectC.business-zone

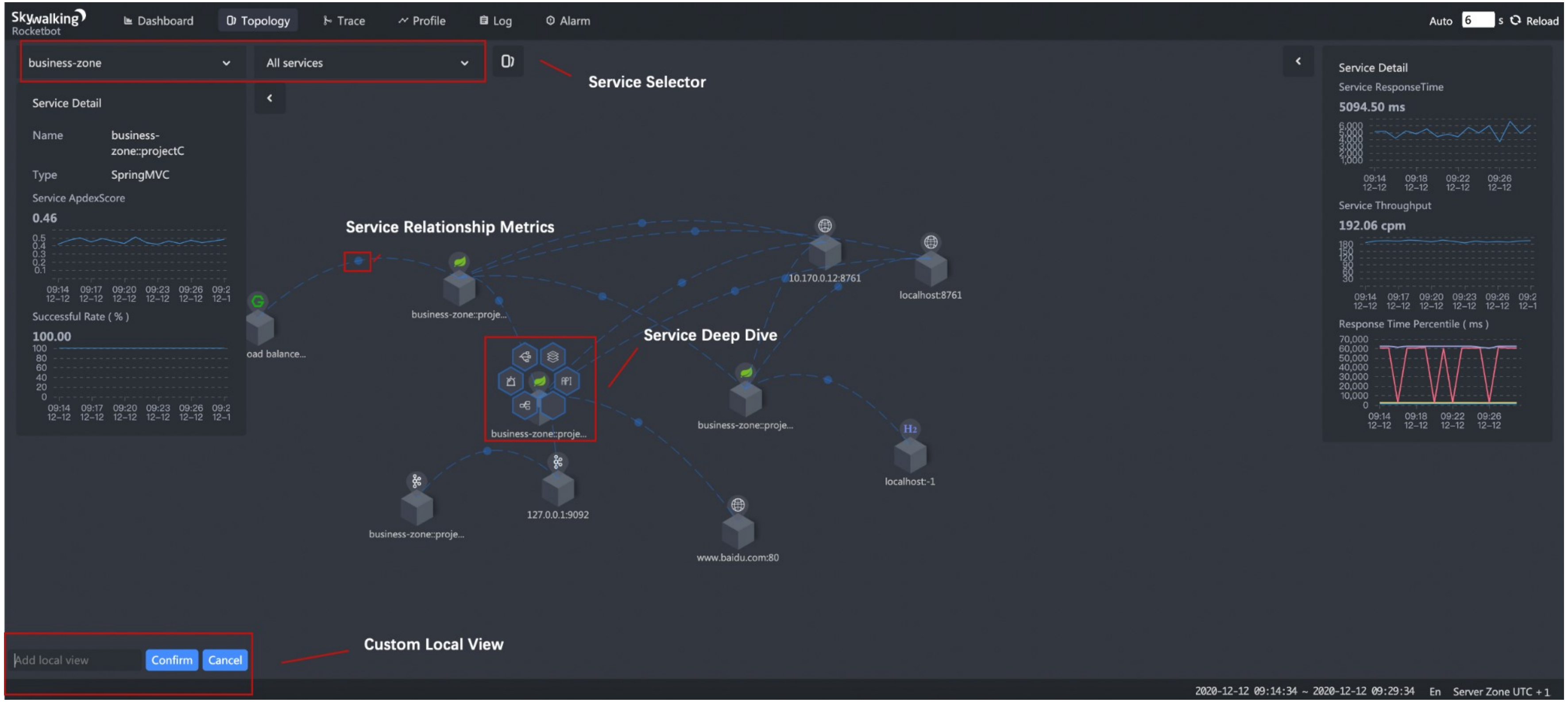
0 5s 10s 15s 20s 25s 30s 35s 40s 45s 50s 55s 60s

- /projectA/test — Http – Nginx
- /projectA/test — Http – Nginx
- /projectA/test — Http – Nginx
- /projectA/test — Http – Nginx
- /projectA/{name} — Http – SpringMVC
- /projectB/test — Http – SpringRestTemplate
- /projectB/{value} — Http – SpringMVC
- org.skywalking.springcloud.test.projectb... — Unknown
- H2/JDBI/PreparedStatement/execute — Database – h2-jdbc-driver
- selectUser

2020-03-18 10:05:31 ~ 2020-03-18 10:20:31 En Server Zone UTC +13

Trace Details - Source: http://skywalking.apache.org/docs/main/latest/en/ui/readme/

Apache Skywalking (https://skywalking.apache.org/)



Service Detail

Name	business-zone::projectC
Type	SpringMVC

Service ApdexScore

0.46

Successful Rate (%)

100.00

Service Detail

Service ResponseTime

5094.50 ms

Service Throughput

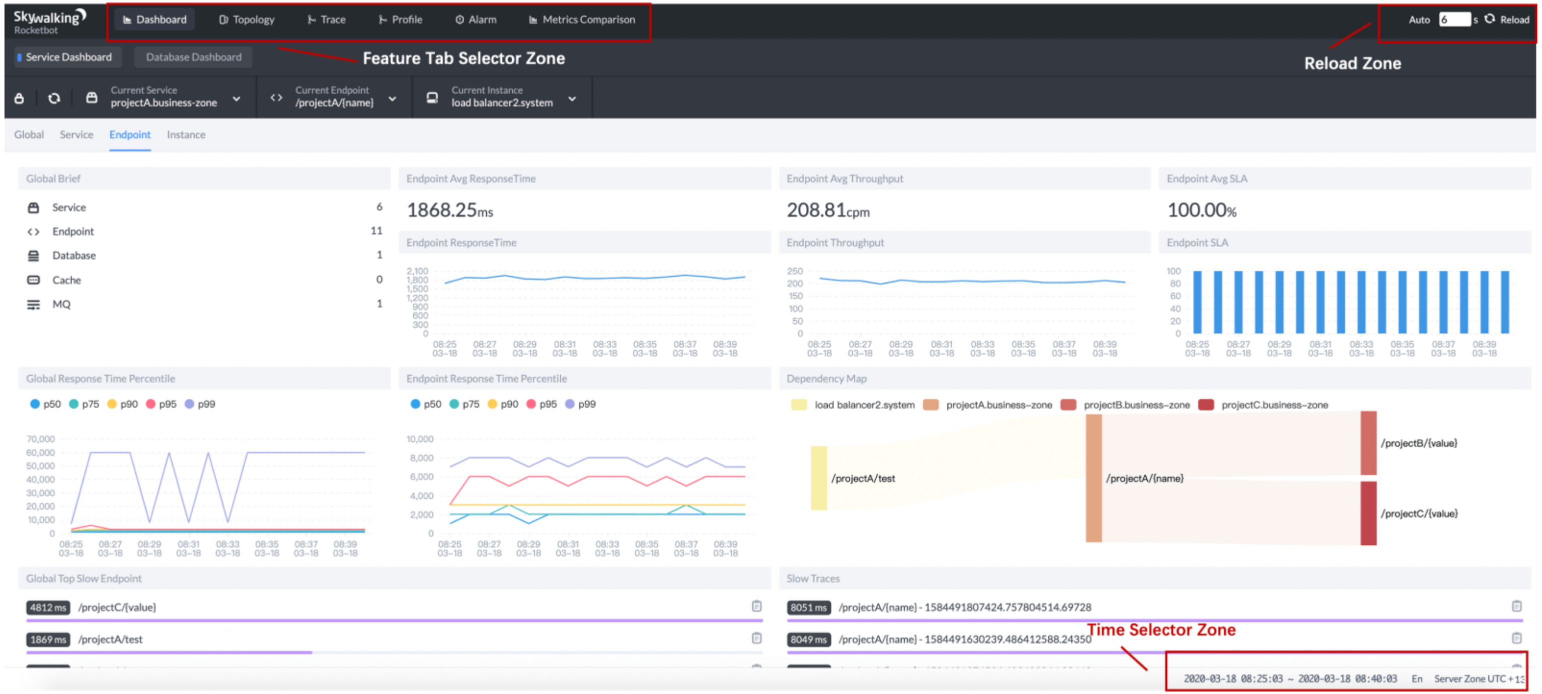
192.06 cpm

Response Time Percentile (ms)

2020-12-12 09:14:34 ~ 2020-12-12 09:29:34 En Server Zone UTC + 1

Dependency Graph Source: http://skywalking.apache.org/docs/main/latest/en/ui/readme/

Apache Skywalking (https://skywalking.apache.org/)



Service 6
Endpoint 11
Database 1
Cache 0
MQ 1

1868.25ms

208.81cpm

100.00%

4812 ms /projectC/{value}

1869 ms /projectA/test

8051 ms /projectA/{name} - 1584491807424.757804514.69728

8049 ms /projectA/{name} - 1584491630239.486412588.24350

2020-03-18 08:25:03 ~ 2020-03-18 08:40:03 En Server Zone UTC +13

Dashboard - Source: http://skywalking.apache.org/docs/main/latest/en/ui/readme/

What are reasons for a proprietary alternative?

- There is also cost associated with setting up and maintaining an open source APM solution (taken from https://sematext.com/blog/performance-monitoring-comparison-build-vs-buy/) :

- **Build Your Own Monitoring System — Cost Scenario**

  - Hourly rate: 100 € (ballpark figure; could be much higher)
  - Installation: 2 hours (very optimistic)
  - Configuration: 8 hours (very optimistic)
  - Maintenance: 2 hours/month (optimistic)
  - Upgrading: 2 days (i.e., ~20 hours)/year (IF all goes well!)
  - # of servers to run this configuration: 3 (monitoring 10 total servers)
  - Cost per server (hardware): 1,000 € each (i.e., 3,000 € total)

  - Total Cost in Year 1: 6,200 €
  - Total Cost in Year 2: 3,200 € (not including any additional server purchases)
  - Total Cost in Year 3: 3,200 € (at least, though most likely higher)

What are reasons for a proprietary alternative?

- **Easier problem resolution**:
  - You do have someone to investigate and fix issues
  - Less risk in production as tools are (mostly) more thoroughly tested

- **Broader technology support**:
  - Developing agents is very time consuming and, thus, costly – the open-source community cannot spend the same amount of manpower into this effort for each and every version of a technology (e.g., supporting Tomcat, 5,6,7,8, …)

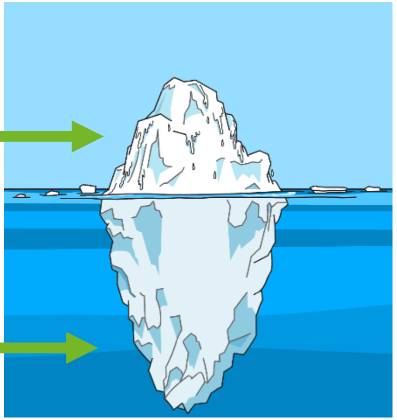
- **You can plan ahead**:
  - Vendors typically communicate the time until which a software version is supported and support the transition phase as well, this is not always the case for open source software

What are reasons for a proprietary alternative?

Remember: Code and Effort distribution of an APM Solution

UI + Server + Collectors

Agents



- Some things might change, as some open-source projects (e.g., istio/Ingress/ WildFly) are already supporting OpenTracing, OpenCensus or OpenTelemetry natively
- Furthermore, there are default implementations for Spring Boot or Quarkus to automatically capture traces that can be packaged in your application
- Additionally, the OpenTelemetry Java Auto Instrumentation is also simplifying the adoption

Thank you!

Dr. Andreas Brunnert
brunnert@retit.de

RETIT
Resource Efficient Technologies & IT Systems