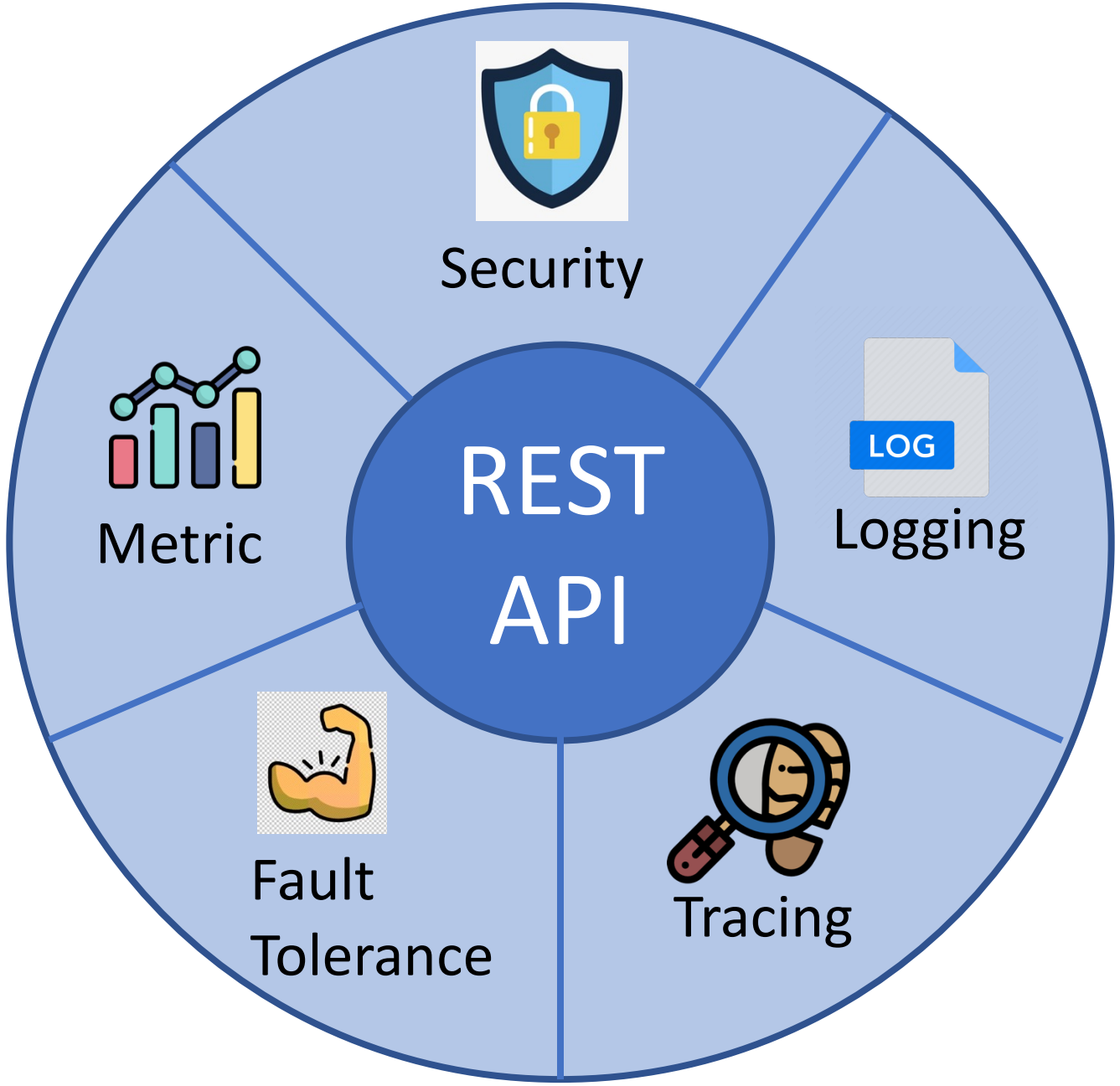
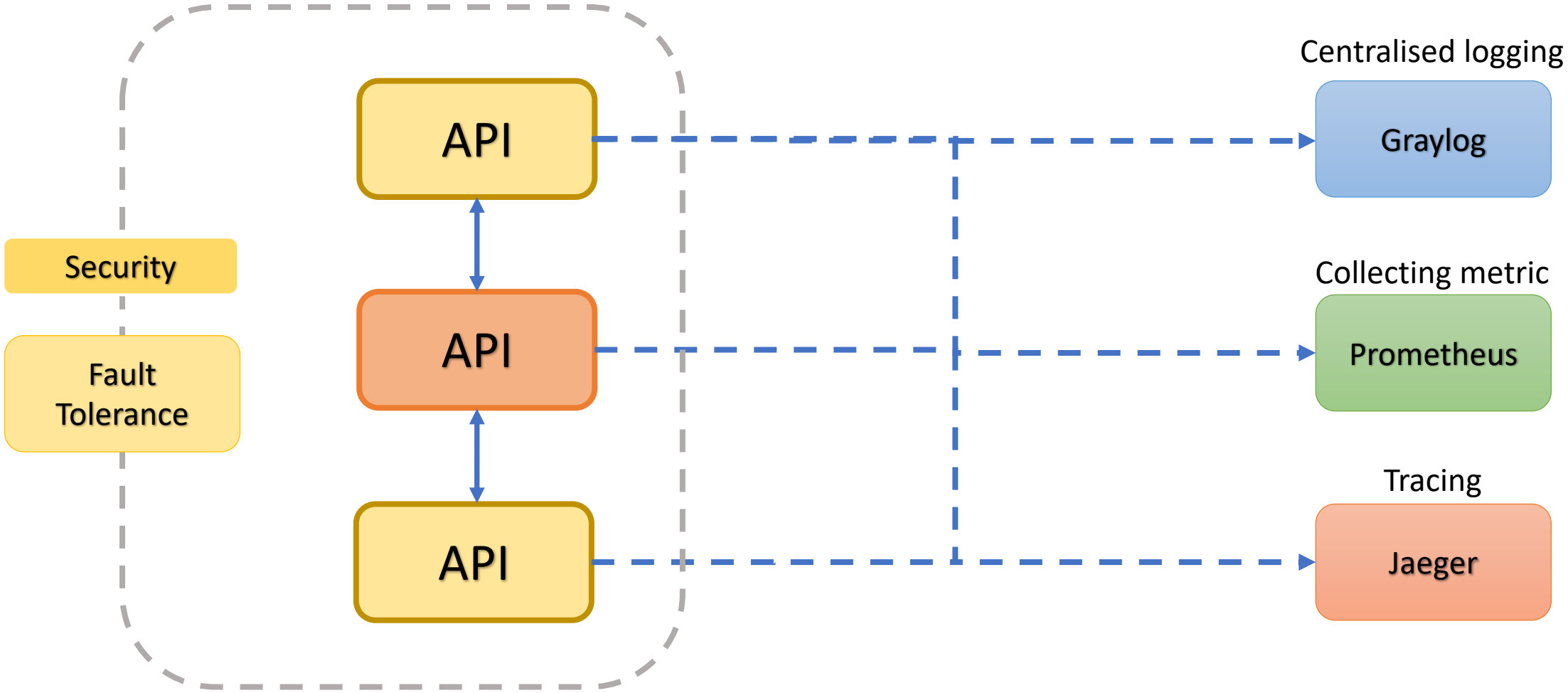
My Standard
API Application Example

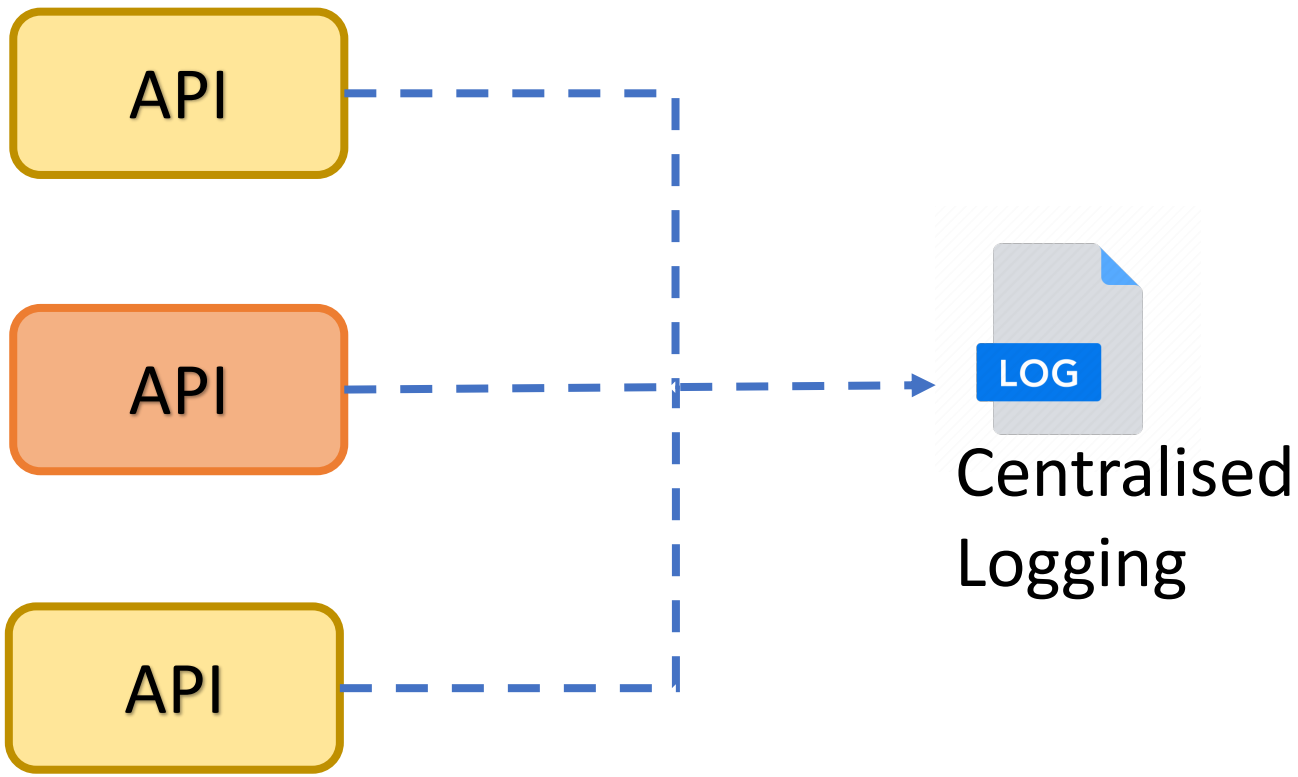
By TC 2022

Standard API Application

- When creating a standard API application in today’s Cloud environment, what are the considerations?


Security
Logging
LOG
Tracing
Fault
Tolerance
Metric
REST
API


Security
Fault Tolerance
API
API
API
Centralised logging
Graylog
Collecting metric
Prometheus
Tracing
Jaeger


API
API
API
LOG
Centralised
Logging

graylog Search Streams Alerts Dashboards Enterprise System 1

0 in 0 out

From: 2022-04-10 05:53:53.675 Until: 2022-04-10 07:49:39.307

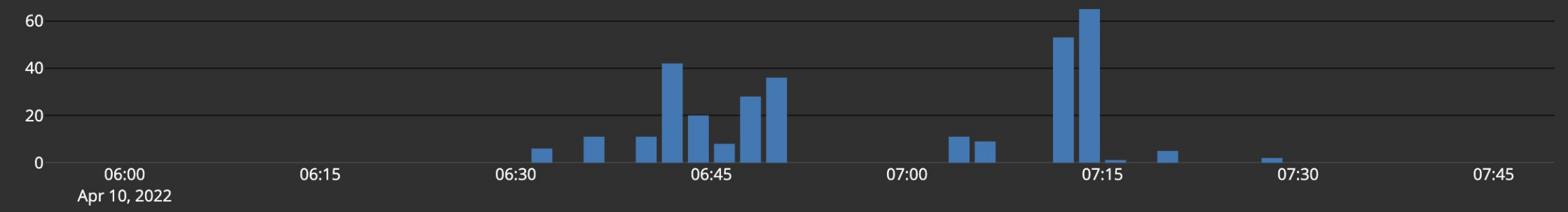
Select streams the search should include. Searches in all streams if...

Not updating

Type your search query here and press enter. E.g.: ("not found" AND http) OR http_response_c

Save Load Share

Message Count



All Messages

2022-04-10 07:14:06.364 +00:00 universent
Returning all data.

2022-04-10 07:14:06.320 +00:00 universent
Returning request from Service 2.

2022-04-10 07:14:06.299 +00:00 universent
Fetching data from external service 2

2022-04-10 07:14:06.292 +00:00 universent
Returning request from Service 1.

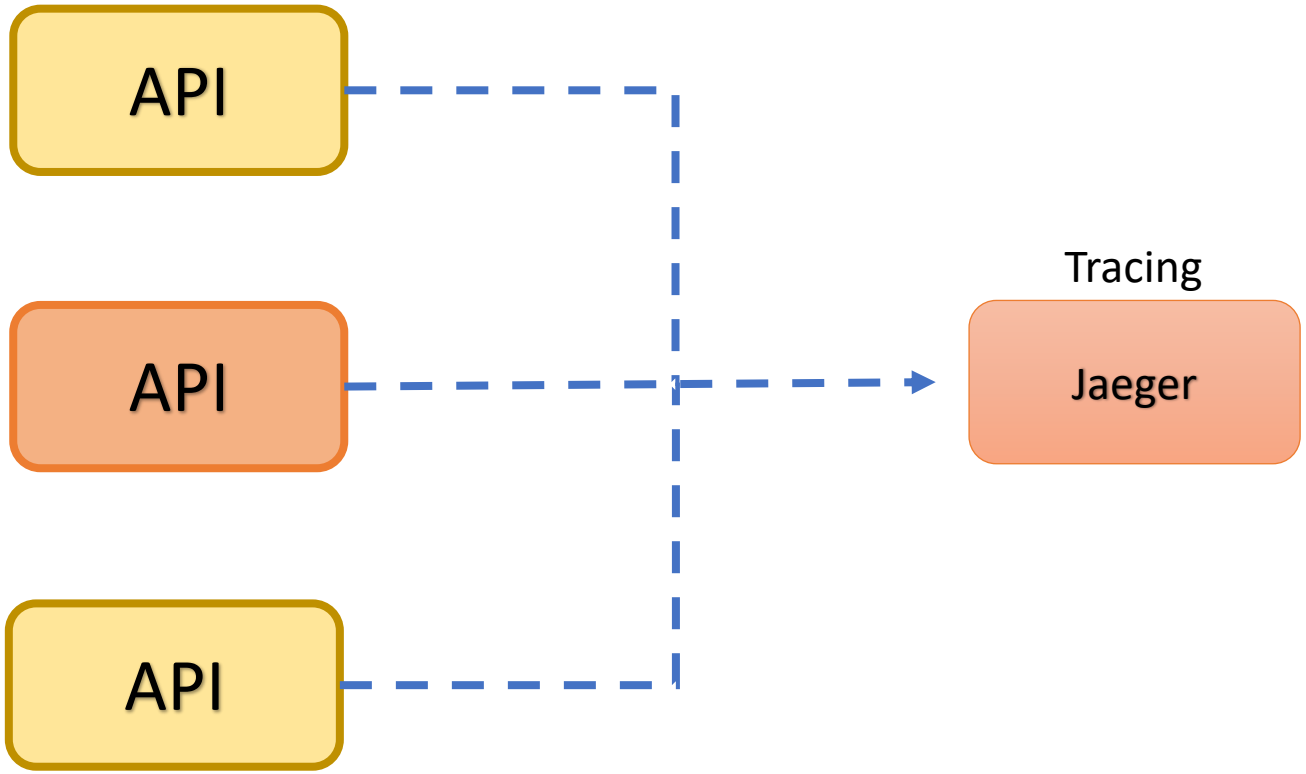
2022-04-10 07:14:06.284 +00:00 universent
Fetching data from external service 1

2022-04-10 07:14:05.577 +00:00 universent
Returning all data.

Log from API-2

Log from API-1

Log from API-0

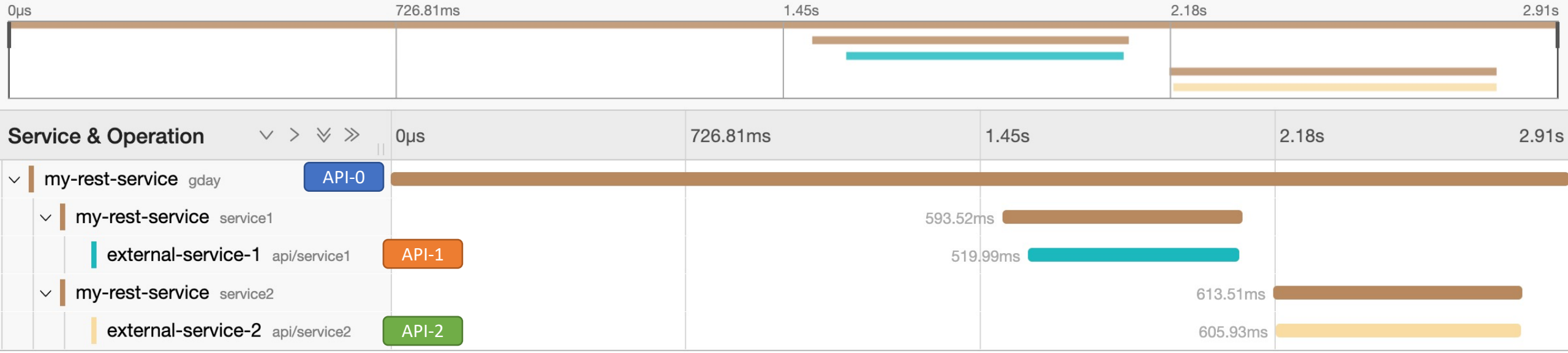

API
API
API
Tracing
Jaeger

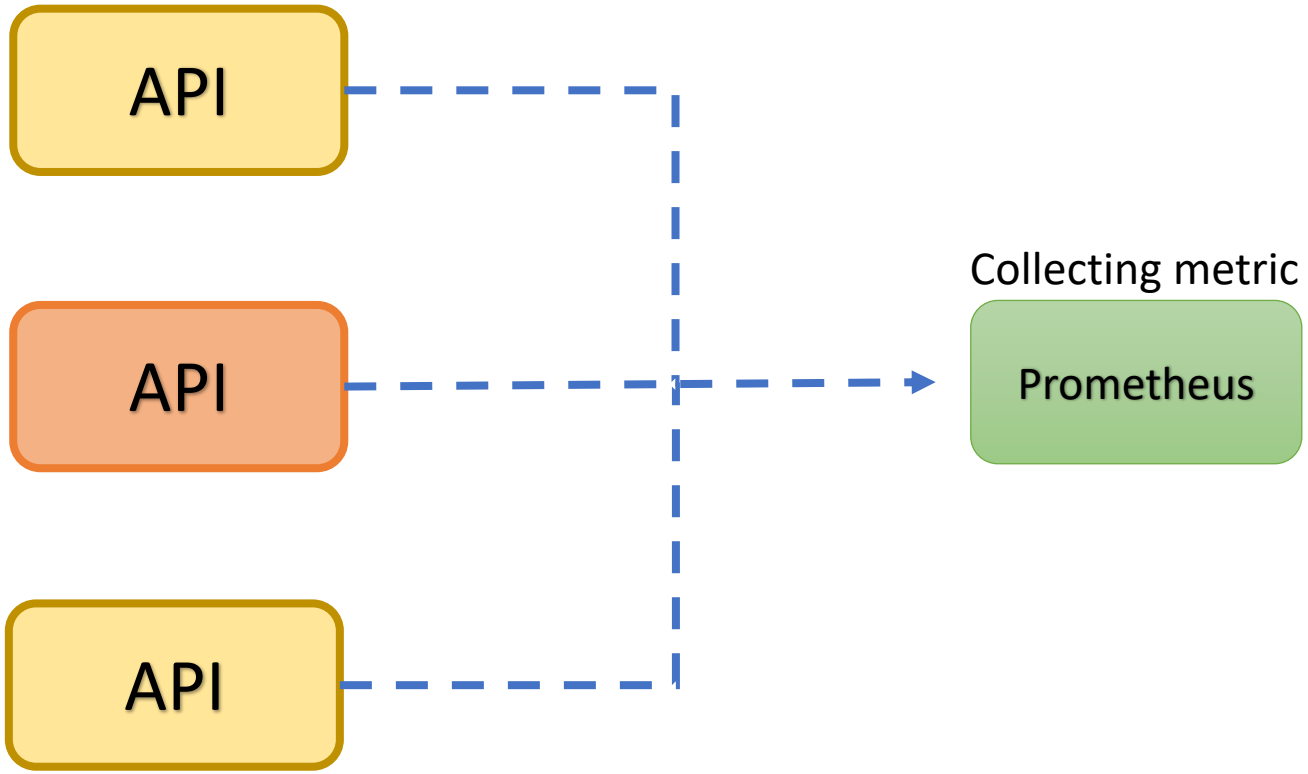
GET http://localhost:8080/gday Send

JAEGER UI Search Compare System Architecture Lookup by Trace ID... About Jaeger

my-rest-service: gday 510c696 Find... Trace Timeline

Trace Start **April 12 2022, 17:05:23.741** Duration **2.91s** Services **3** Depth **3** Total Spans **5**




API
API
API
Collecting metric
Prometheus

Prometheus Alerts Graph Status Help

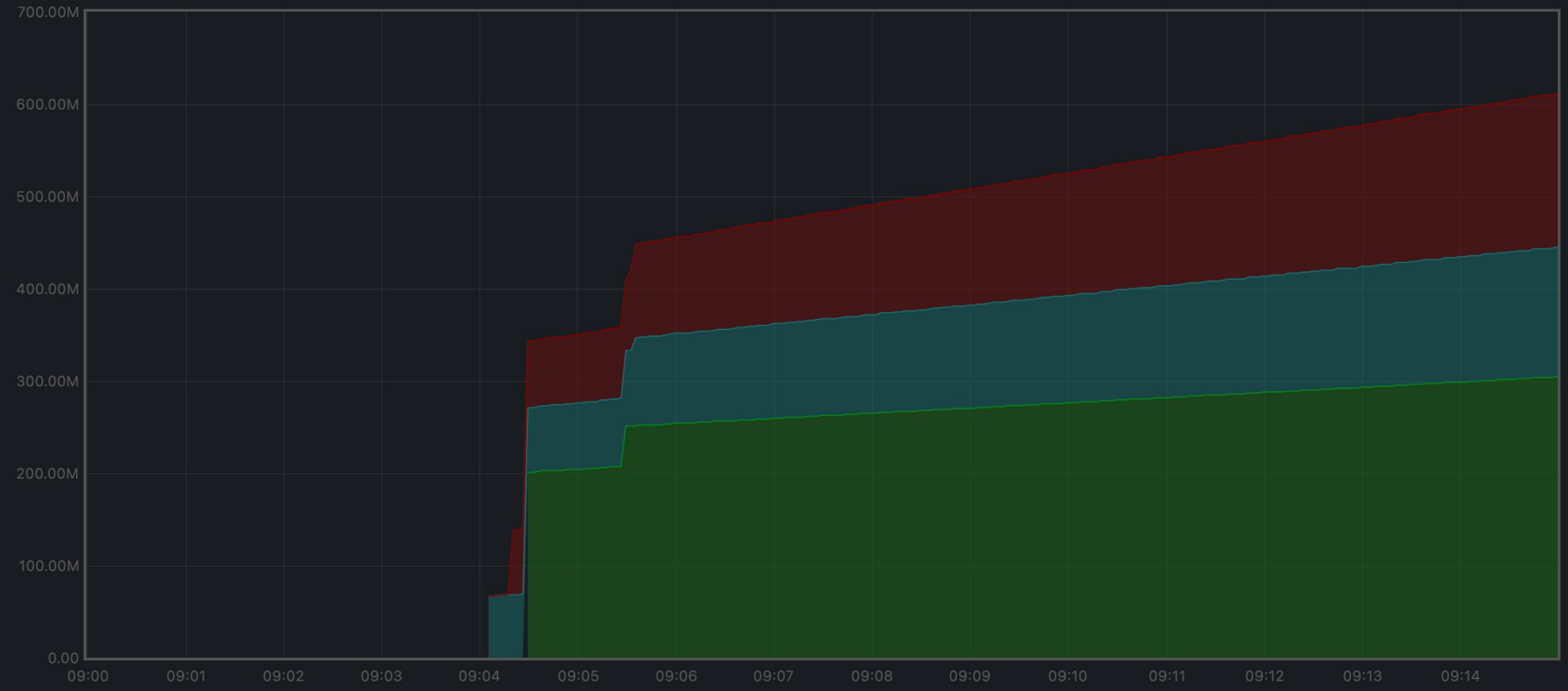
base_memory_usedHeap_bytes

Execute

Table Graph

Load time: 10ms Resolution: 3s Result series: 3

15m End time Res. (s) Show Exemplars



base_memory_usedHeap_bytes{**instance**="localhost:8080", **job**="my_sample_api"}

base_memory_usedHeap_bytes{**instance**="localhost:8081", **job**="service_api_1"}

base_memory_usedHeap_bytes{**instance**="localhost:8082", **job**="service_api_2"}

Click: select series, CMD + click: toggle multiple series

Prometheus Alerts Graph Status Help

All Unhealthy Collapse All

Filter by endpoint or labels

API-0

my_sample_api (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8080/q/metrics	UP	instance="localhost:8080" job="my_sample_api"	1.453s ago	6.684ms	

prometheus (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:9090/metrics	UP	instance="localhost:9090" job="prometheus"	14.724s ago	3.626ms	

API-1

service_api_1 (1/1 up) show less

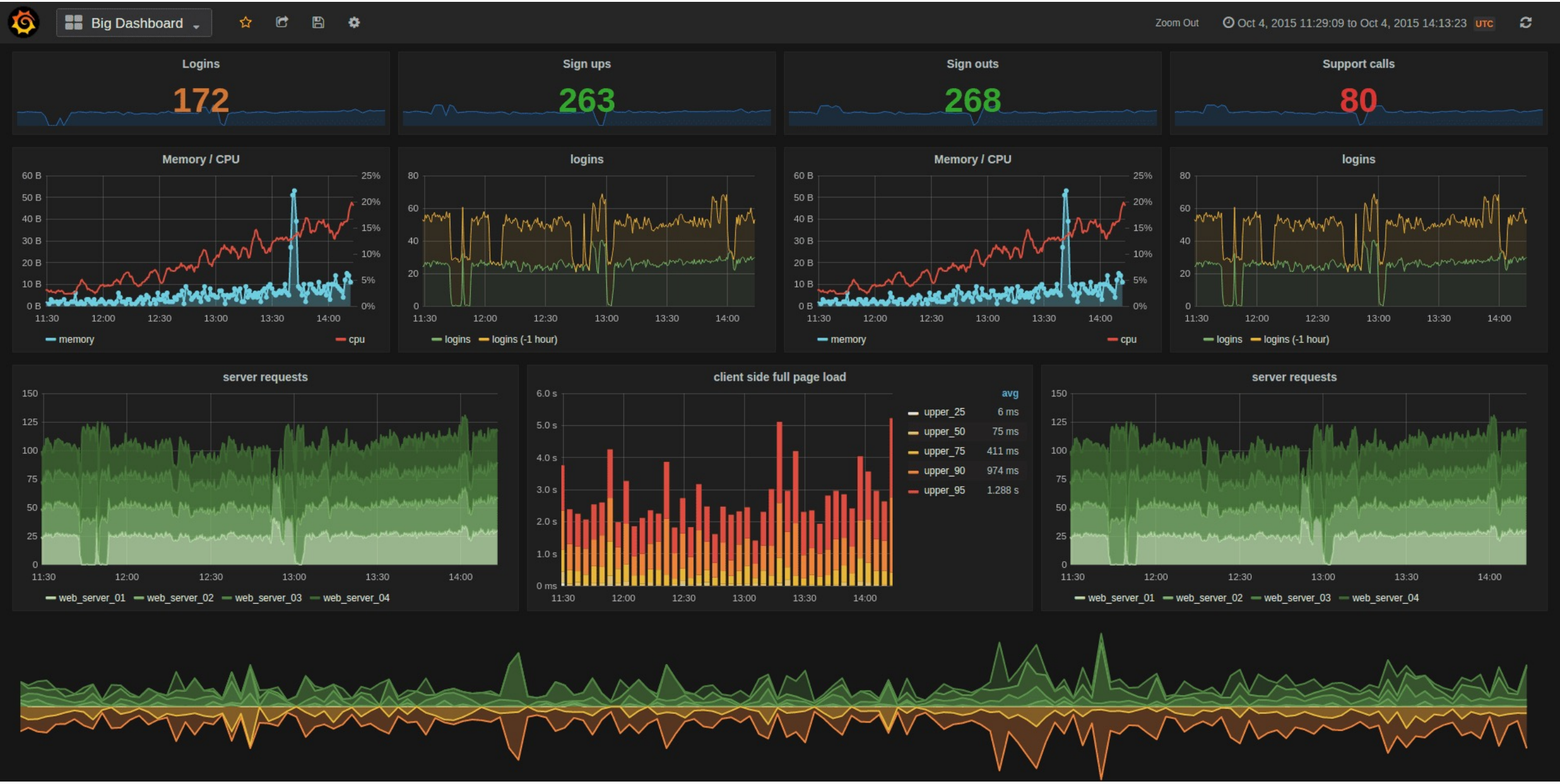
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8081/q/metrics	UP	instance="localhost:8081" job="service_api_1"	5.526s ago	3.195ms	

API-2

service_api_2 (1/1 up) show less

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://localhost:8082/q/metrics	UP	instance="localhost:8082" job="service_api_2"	5.103s ago	3.066ms	

Grafana example



General / Service overview
Last 3 hours
logins
Last 20 minutes
logins
logins (-1 hour)
Memory / CPU
cpu
memory
Free disk space
sda1
sda2
sda3
sda5
sda6
sda7
69.9 GB
24.2 GB
20.3 GB
18.1 GB
89.8 GB
73.0 GB
Server Request / s
web_server_01
web_server_02
web_server_03
web_server_04
Free disk space
80.6 GB
sda1
21.9 GB
sda2
54.5 GB
sda3
68.0 GB
sda5
29.7 GB
sda6
4.21 GB
sda7
Server CPU usage
Backend-ops-01
Backend-ops-02
Backend-ops-03
Backend-ops-04
Frontend-web-01
Frontend-web-02
< 10%
10%+
60%+
70%+
client side full page load
avg
upper_25 2.03 ms
upper_50 138 ms
upper_75 612 ms
upper_90 1.20 s
upper_95 1.70 s
Flow readings
time flow1 flow2 flow3
Jun 8th, 2021 11:23:43 50% 50% 50%
Jun 8th, 2021 11:25:43 54.0% 49.7% 44.5%
Jun 8th, 2021 11:27:43 53.0% 49.7% 50.2%
Jun 8th, 2021 11:29:43 49.4% 51.8% 54.6%
Jun 8th, 2021 11:31:43 51.1% 52.7% 57.5%
Jun 8th, 2021 11:33:43 50.2% 56.8% 62.9%
Jun 8th, 2021 11:35:43 53.1% 61.3% 65.0%
Jun 8th, 2021 11:37:43 50.6% 59.7% 68.9%
Jun 8th, 2021 11:39:43 52.4% 66.0% 65.8%
Jun 8th, 2021 11:41:43 51.5% 64.9% 72.5%
Jun 8th, 2021 11:43:43 54.0% 69.0% 79.5%
Jun 8th, 2021 11:45:43 52.8% 75.9% 81.9%
Jun 8th, 2021 11:47:43 57.1% 78.2% 86.8%

Security Example

GET Health | GET Health | GET Hello | GET Bye | No Environment

New Collection / Bye

Save

GET http://localhost:8080/bye

Send

Params | Auth | Headers (8) | Body | Pre-req. | Tests | Settings | Cookies

Type

Basic Auth

The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables ↗

Username: user

Password: ••••••••

Show Password

GET http://localhost:8080/bye Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Type

No Auth

This request does not use any authorization. Learn more about authorization ↗

Body 401 Unauthorized 23 ms 89 B Save Response

Pretty Raw Preview Visualize Text

1

Fault Tolerance Example

```
@Retry(maxRetries = 1)
@Fallback(fallbackMethod = "fallbackMethod")
WorldClock getNow(){}

public WorldClock fallbackMethod() {
    return new WorldClock();
}
```

```
@CircuitBreaker(requestVolumeThreshold = 4,
                failureRatio=0.75,
                delay = 1000)
WorldClock getNow(){}
```

```
@Bulkhead(5)
@Retry(maxRetries = 4,
       delay = 1000,
       retryOn = BulkheadException.class)
WorldClock getNow(){}
```

Prerequisite:

JAEGER

Prometheus

graylog

graylog_stack > docker-compose.yml

```yaml
version: "3.2"

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.8.2
    ports:
      - "9200:9200"
    environment:
      ES_JAVA_OPTS: "-Xms512m -Xmx512m"
    networks:
      - graylog

  mongo:
    image: mongo:4.0
    networks:
      - graylog

  graylog:
    image: graylog/graylog:4.2
    ports:
      - "9000:9000"
      - "12201:12201"
      - "12201:12201/udp"
      - "1514:1514"
      - "1514:1514/udp"
    environment:
      GRAYLOG_PASSWORD_SECRET: "somepasswordpepper"
      GRAYLOG_HTTP_EXTERNAL_URI: "http://127.0.0.1:9000/"
      GRAYLOG_ROOT_PASSWORD_SHA2: "8c6976e5b5410415bde908bd4dee15dfb167a9c
    entrypoint: /usr/bin/tini -- wait-for-it elasticsearch:9200 -- /docker
    networks:
      - graylog
    depends_on:
      - elasticsearch
      - mongo

networks:
  graylog:
    driver: bridge
```

opentelemetry_stack > docker-compose.yml

```yaml
version: "2"
services:

  # Jaeger
  jaeger-all-in-one:
    image: jaegertracing/all-in-one:latest
    ports:
      - "16686:16686"
      - "14268"
      - "14250"
  # Collector
  otel-collector:
    image: otel/opentelemetry-collector:latest
    command: ["--config=/etc/otel-collector-config.yaml"]
    volumes:
      - ./otel-collector-config.yaml:/etc/otel-collector-config.yaml
    ports:
      - "13133:13133" # Health_check extension
      - "4317:4317"   # OTLP gRPC receiver
      - "55680:55680" # OTLP gRPC receiver alternative port
    depends_on:
      - jaeger-all-in-one
```

..t-application (-zsh)

```
→ quarkus-rest-application brew install prometheus
```

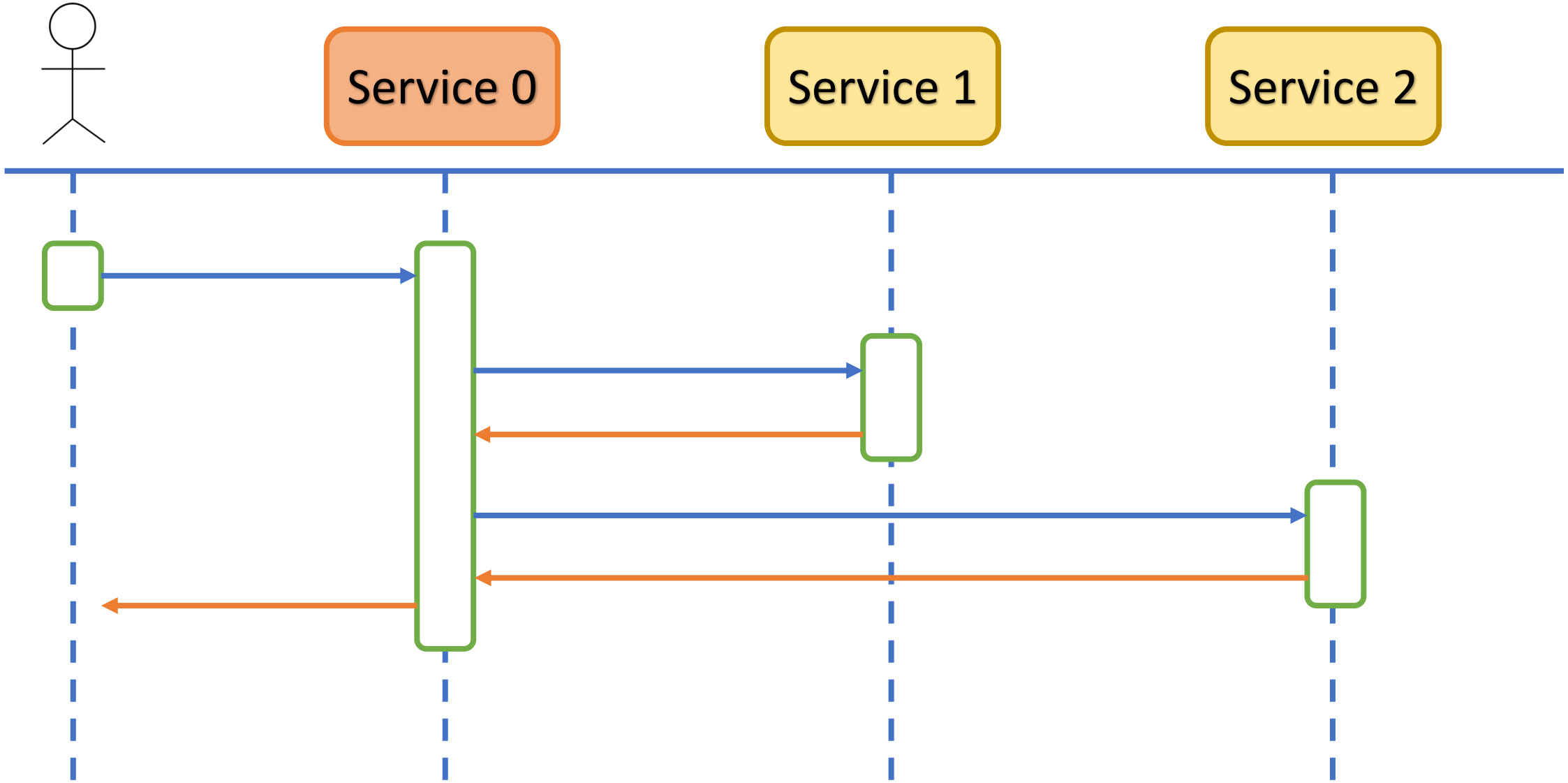
opentelemetry_stack > ! otel-collector-config.yaml

```yaml
receivers:
  otlp:
    protocols:
      grpc:
        endpoint: otel-collector:4317
  otlp/2:
    protocols:
      grpc:
        endpoint: otel-collector:55680

exporters:
  jaeger:
    endpoint: jaeger-all-in-one:14250
    insecure: true

processors:
  batch:

extensions:
  health_check:

service:
  extensions: [health_check]
  pipelines:
    traces:
      receivers: [otlp, otlp/2]
      processors: [batch]
      exporters: [jaeger]
```

quarkus-rest-service-2 > prometheus_stack > ! prometheus.yml

```yaml

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries sc
  - job_name: "prometheus"

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ["localhost:9090"]

  - job_name: "my_sample_api"
    scrape_interval: 5s
    metrics_path: "/q/metrics"
    static_configs:
      - targets: ["localhost:8080"]

  - job_name: "service_api_1"
    scrape_interval: 5s
    metrics_path: "/q/metrics"
    static_configs:
      - targets: ["localhost:8081"]

  - job_name: "service_api_2"
    scrape_interval: 5s
    metrics_path: "/q/metrics"
    static_configs:
      - targets: ["localhost:8082"]
```

Design:



Requirement: Quarkus CLI

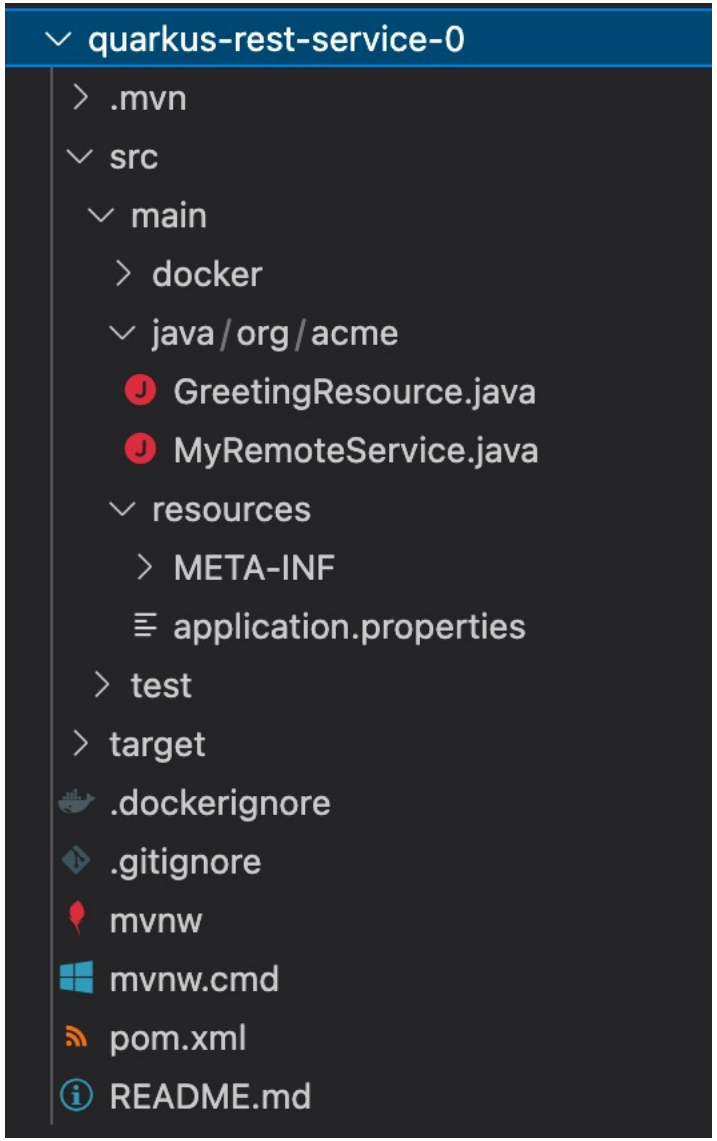
```
brew install quarkusio/tap/quarkus
```

Bootstrap the application

```
quarkus create app org.acme:quarkus-rest-service-0 \
--extension=resteasy-jackson,\
smallrye-metrics,\
logging-gelf,\
opentelemetry-exporter-otlp,\
elytron-security-properties-file,\
rest-client,\
rest-client-jackson
```

..est-service-0 (-zsh)

```
➜  my-standard-applications quarkus create app org.acme:quarkus-rest-service-0 \
> --extension=resteasy-jackson,smallrye-metrics,logging-gelf,opentelemetry-exporter-otlp,elytron-securi
ty-properties-file,rest-client,rest-client-jackson
Looking for the newly published extensions in registry.quarkus.io
-----------
selected extensions:
- io.quarkus:quarkus-logging-gelf
- io.quarkus:quarkus-opentelemetry-exporter-otlp
- io.quarkus:quarkus-resteasy-jackson
- io.quarkus:quarkus-rest-client-jackson
- io.quarkus:quarkus-rest-client
- io.quarkus:quarkus-elytron-security-properties-file
- io.quarkus:quarkus-smallrye-metrics


applying codestarts...
📚  java
🔨  maven
📦  quarkus
📝  config-properties
🔧  dockerfiles
🔧  maven-wrapper
🚀  rest-client-codestart
🚀  resteasy-codestart

-----------
[SUCCESS] ✅  quarkus project has been successfully generated in:
--> /Users/tcong/git/testlab/my-standard-applications/quarkus-rest-service-0
-----------
Navigate into this directory and get started: quarkus dev
➜  my-standard-applications cd quarkus-rest-service-0
```



quarkus-rest-service-0 > src > main > java > org > acme > GreetingResource.java > GreetingResource

```java
package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;

@Path("/hello")
public class GreetingResource {

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String hello() {
        return "Hello RESTEasy";
    }
}
```

ServiceData.java 1 × REAMDE.md prometheus.yml

quarkus-rest-service-0 > src > main > java > org > acme > ServiceDa

```java
package org.acme;

public class ServiceData {
  private String name;
  private String description;

  public String getName() {
    return name;
  }

  public String getDescription() {
    return description;
  }

  public void setDescription(String description) {
    this.description = description;
  }

  public void setName(String name) {
    this.name = name;
  }

}
```

GreetingResource.java

src > main > java > org > acme > GreetingResource.java > GreetingResource

```java

@Path("/hello")
public class GreetingResource {

    @Inject
    Logger logger;

    @Inject
    @RestClient
    ExternalService1 service1;

    @Inject
    @RestClient
    ExternalService2 service2;

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public List<ServiceData> hello() {
        logger.info(message: "Received request.");

        logger.info(message: "Getting service data from service1.");
        ServiceData serviceData1 = service1.getData();

        logger.info(message: "Getting service data from service2.");
        ServiceData serviceData2 = service2.getData();

        logger.info(message: "Returning service data.");
        return List.of(serviceData1, serviceData2);
    }
}
```

ExternalService1.java 1 | REAMDE.md | prometheus.yml

quarkus-rest-service-0 > src > main > java > org > acme > ExternalService1.java > ...

```java
package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;

@RegisterRestClient(configKey = "external-api-1")
public interface ExternalService1 {

  @GET
  @Path("/service1")
  ServiceData getData();

}
```

ExternalService2.java 1 | REAMDE.md | prometheus.yml

quarkus-rest-service-0 > src > main > java > org > acme > ExternalService2.java > ...

```java
package org.acme;

import javax.ws.rs.GET;
import javax.ws.rs.Path;

import org.eclipse.microprofile.rest.client.inject.RegisterRestClient;


@RegisterRestClient(configKey = "external-api-2")
public interface ExternalService2 {

  @GET
  @Path("/service2")
  ServiceData getData();

}

```

application.properties

src > main > resources > application.properties > ...

```
# OpenTelemetry properties
quarkus.application.name=quarkus-rest-service-0
quarkus.opentelemetry.enabled=true
quarkus.opentelemetry.tracer.exporter.otlp.endpoint=http://localhost:4317
#
# Centralised Logging with GELF
quarkus.log.handler.gelf.enabled=true
quarkus.log.handler.gelf.host=localhost
quarkus.log.handler.gelf.port=12201
#
# Security Setting
# Enable BASIC Auth
quarkus.http.auth.basic=true
# Users in properties filess
quarkus.security.users.file.enabled=true
quarkus.security.users.file.users=test-users.properties
quarkus.security.users.file.roles=test-roles.properties
quarkus.security.users.file.realm-name=MyRealm
quarkus.security.users.file.plain-text=true
#
# Your configuration properties
quarkus.rest-client.external-api-1.url=http://localhost:8081/api
quarkus.rest-client.external-api-1.scope=javax.inject.Singleton
# Your configuration properties
quarkus.rest-client.external-api-2.url=http://localhost:8082/api
quarkus.rest-client.external-api-2.scope=javax.inject.Singleton
```

Securing the endpoint

```java
    @RolesAllowed("User")
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public List<ServiceData> hello() {
        logger.info(message: "Received request.");

        logger.info(message: "Getting service data from service1.");
        ServiceData serviceData1 = service1.getData();

        logger.info(message: "Getting service data from service2.");
        ServiceData serviceData2 = service2.getData();

        logger.info(message: "Returning service data.");
        return List.of(serviceData1, serviceData2);
    }
```

test-roles.properties × application.properti

src > main > resources > test-roles.properties

```properties
admin=Admin
user=User
```

Bootstrap Service 1

```
quarkus create app org.acme:quarkus-rest-service-1 \
--extension=resteasy-jackson,\
smallrye-metrics,\
logging-gelf,\
opentelemetry-exporter-otlp
```

Bootstrap Service 2

```
quarkus create app org.acme:quarkus-rest-service-2 \
--extension=resteasy-jackson,\
smallrye-metrics,\
logging-gelf,\
opentelemetry-exporter-otlp,\
smallrye-fault-tolerance
```

REAMDE.md | DemoResource.java | prometheus.yml

quarkus-rest-service-1 > src > main > java > au > com > versent > DemoResource.j

```java
@Path("/api")
public class DemoResource {

  @Inject
  Logger logger;

  @GET
  @Path("/service1")
  public ServiceData handleRequest() {
    ServiceData data = new ServiceData();
    data.setName(name: "REST Service 1");
    data.setDescription(description: "Description of service 1");
    logger.info(message: "Returning request from Service 1.");
    return data;
  }
}
```

REAMDE.md | DemoResource.java | prometheus.yml

quarkus-rest-service-2 > src > main > java > au > com > versent > DemoResource.java

```java
  @GET
  @Path("/service2")
  @Produces(MediaType.APPLICATION_JSON)
  @Retry(maxRetries = 8)
  public Response handleRequest() {

    ServiceData data = new ServiceData();
    data.setName(name: "REST Service 2");
    data.setDescription(description: "Description of service 2");

    maybeFail(failureLogMessage: "Failed to generate Service Data.");
    logger.info(message: "Returning request from Service 2.");

    return Response.ok(data).build();
  }

  private void maybeFail(String failureLogMessage) {
    if (new SecureRandom().nextBoolean()) {
      logger.error(failureLogMessage);
      throw new RuntimeException(message: "Resource failure.");
    }
  }
}
```

application.properties

quarkus-rest-service-1 > src > main > resources > application.properties > ...

```
quarkus.http.port=8081
quarkus.application.name=external-service-1
quarkus.opentelemetry.enabled=true
quarkus.opentelemetry.tracer.exporter.otlp.endpoint=http://localhost:4317
# Centralised Logging with GELF
quarkus.log.handler.gelf.enabled=true
quarkus.log.handler.gelf.host=localhost
quarkus.log.handler.gelf.port=12201
```

application.properties

quarkus-rest-service-2 > src > main > resources > application.properties > ...

```
quarkus.http.port=8082
quarkus.application.name=external-service-2
quarkus.opentelemetry.enabled=true
quarkus.opentelemetry.tracer.exporter.otlp.endpoint=http://localhost:4317
# Centralised Logging with GELF
quarkus.log.handler.gelf.enabled=true
quarkus.log.handler.gelf.host=localhost
quarkus.log.handler.gelf.port=12201
```

To live demo, or not to live demo?

Question?