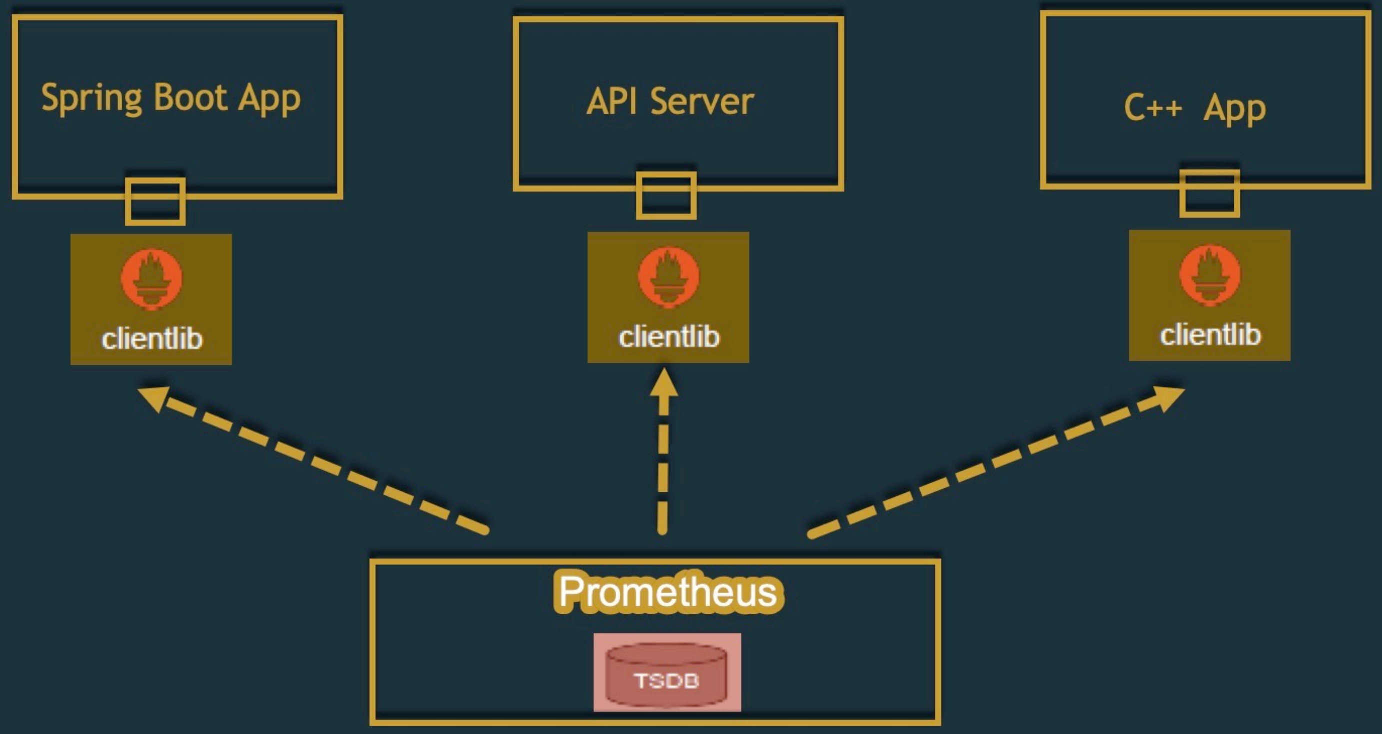
Prometheus

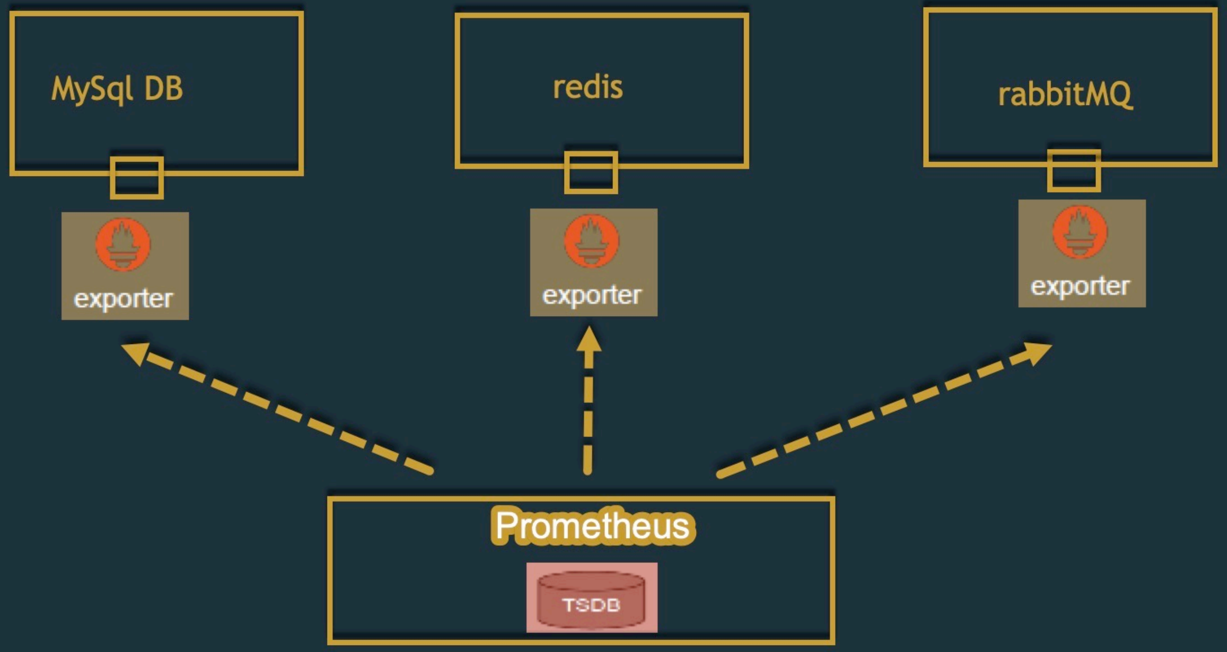
ITS A METRICS BASED MONITORING AND ALERTING STACK

- *Instrumentation*
- *Metrics collection and storage*
- *Querying using PromQL*
- *Alerting*
- *Dashboard*

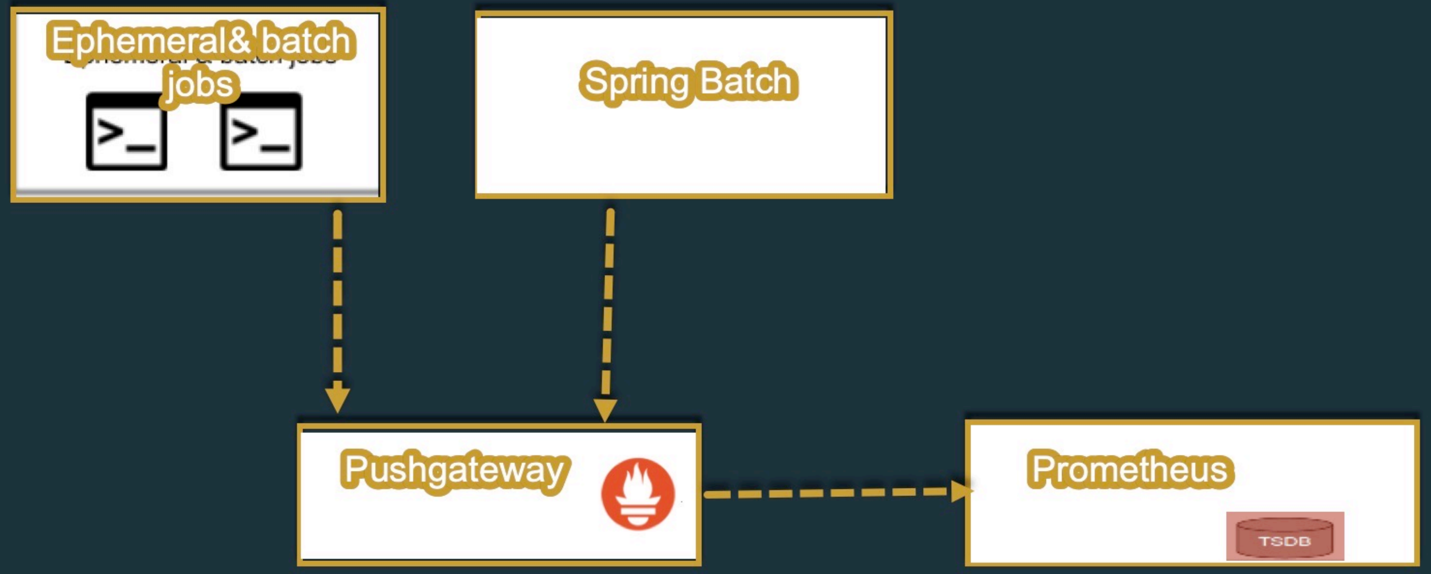
Prometheus


Spring Boot App
API Server
C++ App
clientlib
clientlib
clientlib
Prometheus
TSDB

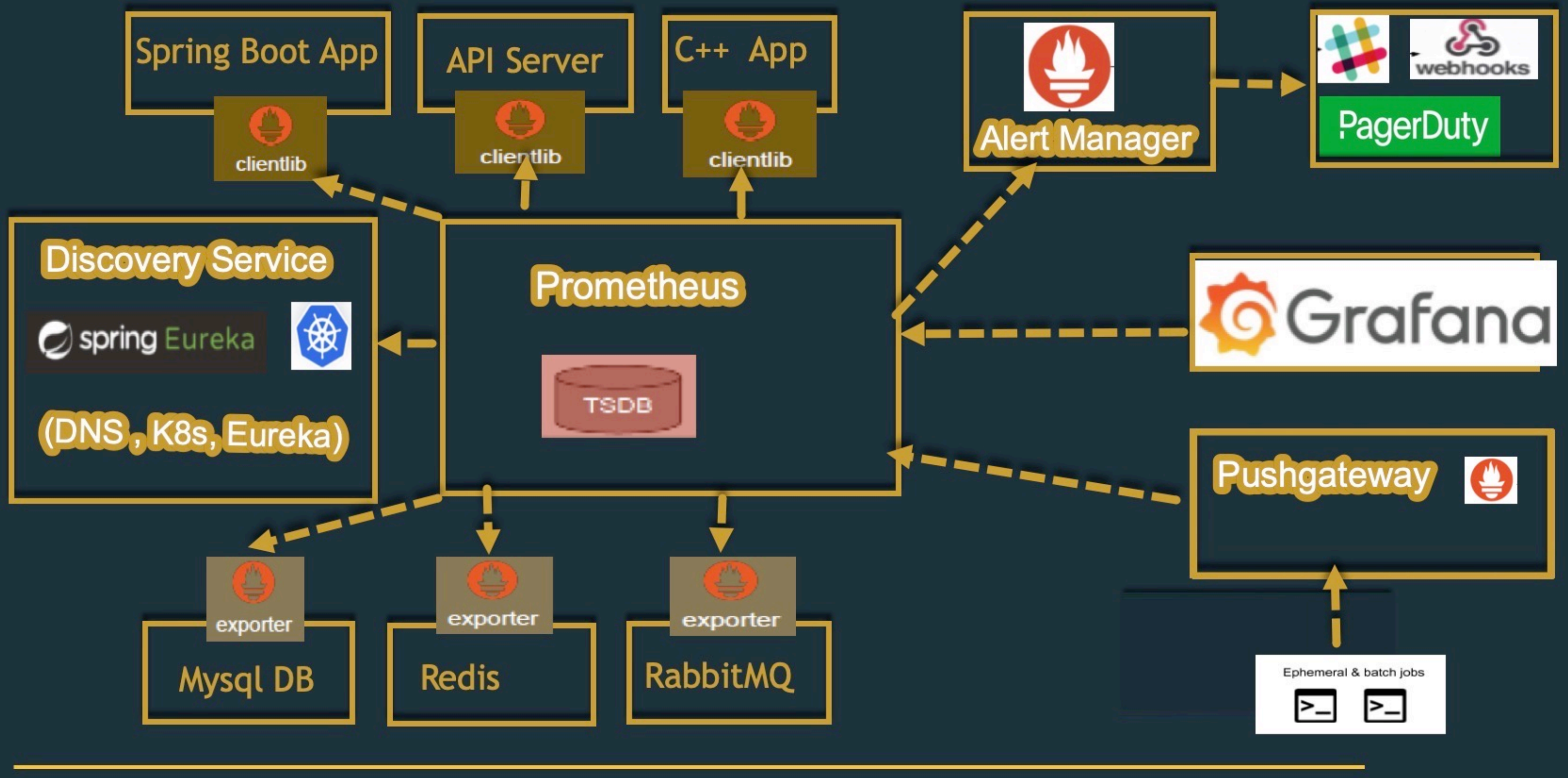
Prometheus


MySql DB
redis
rabbitMQ
exporter
exporter
exporter
Prometheus
TSDB

Prometheus


Ephemeral& batch jobs
Spring Batch
Pushgateway
Prometheus
TSDB

Prometheus


Spring Boot App
clientlib
API Server
clientlib
C++ App
clientlib
Alert Manager
webhooks
PagerDuty
Discovery Service
spring Eureka
(DNS , K8s, Eureka)
Prometheus
TSDB
Grafana
Pushgateway
Ephemeral & batch jobs
exporter
Mysql DB
exporter
Redis
exporter
RabbitMQ

Prometheus

Data Model

```
http_requests_total{job="nginx",instance="1.2.3.4:80",path="/home",status="200"}
```

metric name

labels

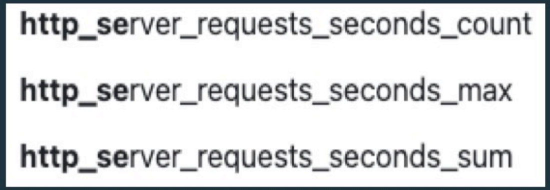
Querying

```
{path="/status", method="GET"}
{path="/", method="GET"}
{path="/api/v1/topics/:topic", method="POST"}
{path="/api/v1/topics, method="GET"}
```

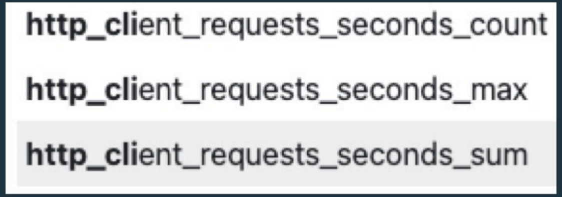
Prometheus

Spring MVC provides excellent monitoring metrics for inbound and outbound HTTP traffic

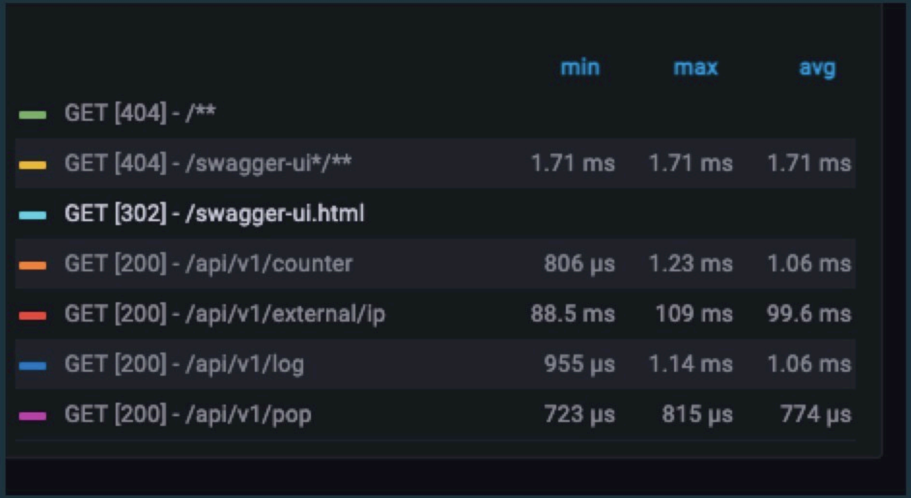
RestTemplate & WebClient to make outbound request are mapped out of the box



```
http_server_requests_seconds_count
http_server_requests_seconds_max
http_server_requests_seconds_sum
```



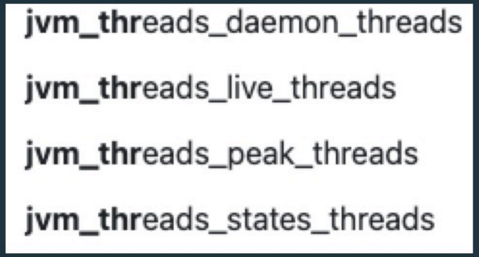
```
http_client_requests_seconds_count
http_client_requests_seconds_max
http_client_requests_seconds_sum
```



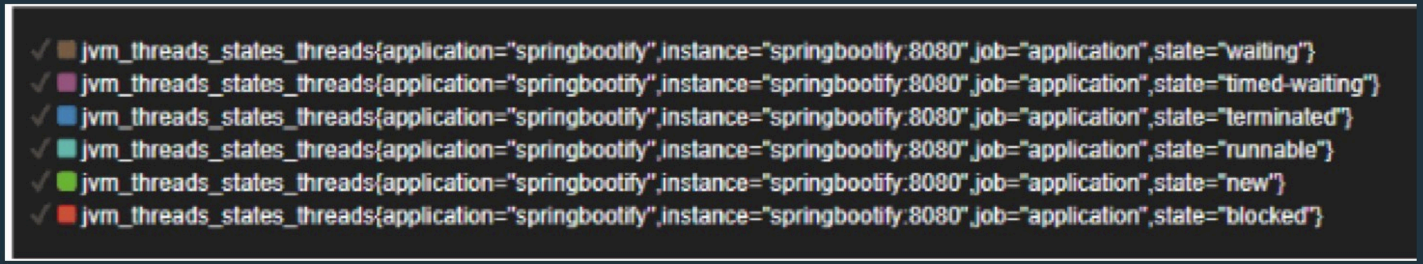
	min	max	avg
🟩 GET [404] - /**			
🟨 GET [404] - /swagger-ui*/**	1.71 ms	1.71 ms	1.71 ms
🟦 GET [302] - /swagger-ui.html			
🟧 GET [200] - /api/v1/counter	806 µs	1.23 ms	1.06 ms
🟥 GET [200] - /api/v1/external/ip	88.5 ms	109 ms	99.6 ms
🟦 GET [200] - /api/v1/log	955 µs	1.14 ms	1.06 ms
🟪 GET [200] - /api/v1/pop	723 µs	815 µs	774 µs

Prometheus

Micrometer provides various types of metrics to monitor JVM



```
jvm_threads_daemon_threads
jvm_threads_live_threads
jvm_threads_peak_threads
jvm_threads_states_threads
```



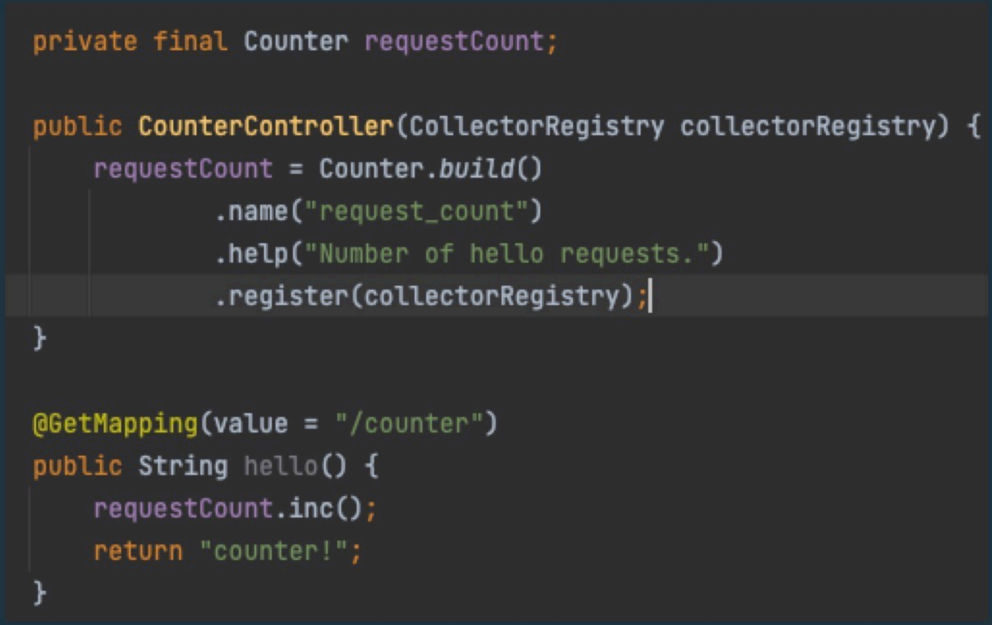
```
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="waiting"}
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="timed-waiting"}
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="terminated"}
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="runnable"}
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="new"}
✓ jvm_threads_states_threads{application="springbootify",instance="springbootify:8080",job="application",state="blocked"}
```

Prometheus

Prometheus provides 4 types of metrics wrapped up in convenient client library

Counters | Gauges | Histograms | Summaries

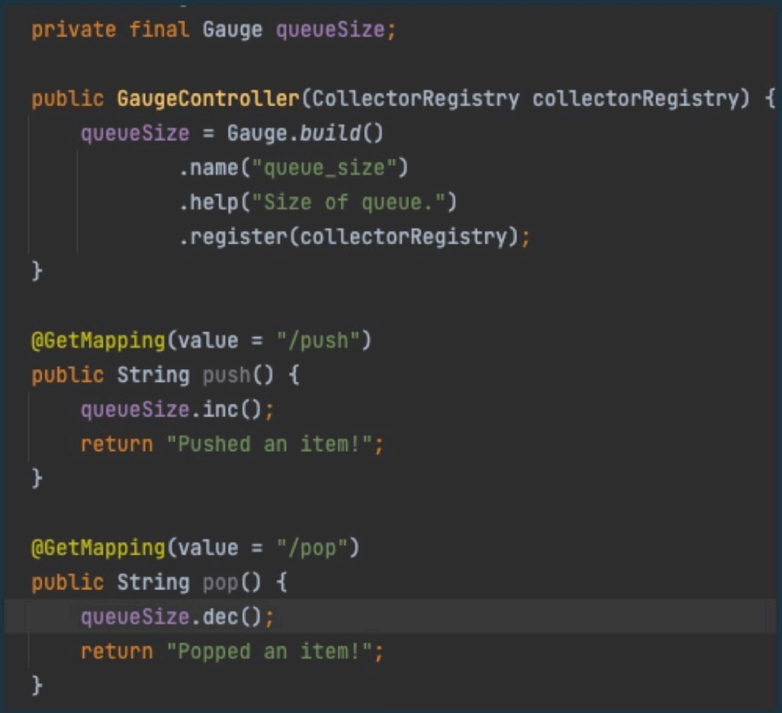
Counters | values that increases such as request count, error count or task completed



```
private final Counter requestCount;

public CounterController(CollectorRegistry collectorRegistry) {
    requestCount = Counter.build()
            .name("request_count")
            .help("Number of hello requests.")
            .register(collectorRegistry);
}

@GetMapping(value = "/counter")
public String hello() {
    requestCount.inc();
    return "counter!";
}
```

Prometheus

Gauges| values that go up as well as down, such as memory usage, item in queues, request in proress



```java
private final Gauge queueSize;

public GaugeController(CollectorRegistry collectorRegistry) {
    queueSize = Gauge.build()
            .name("queue_size")
            .help("Size of queue.")
            .register(collectorRegistry);
}

@GetMapping(value = "/push")
public String push() {
    queueSize.inc();
    return "Pushed an item!";
}

@GetMapping(value = "/pop")
public String pop() {
    queueSize.dec();
    return "Popped an item!";
}
```

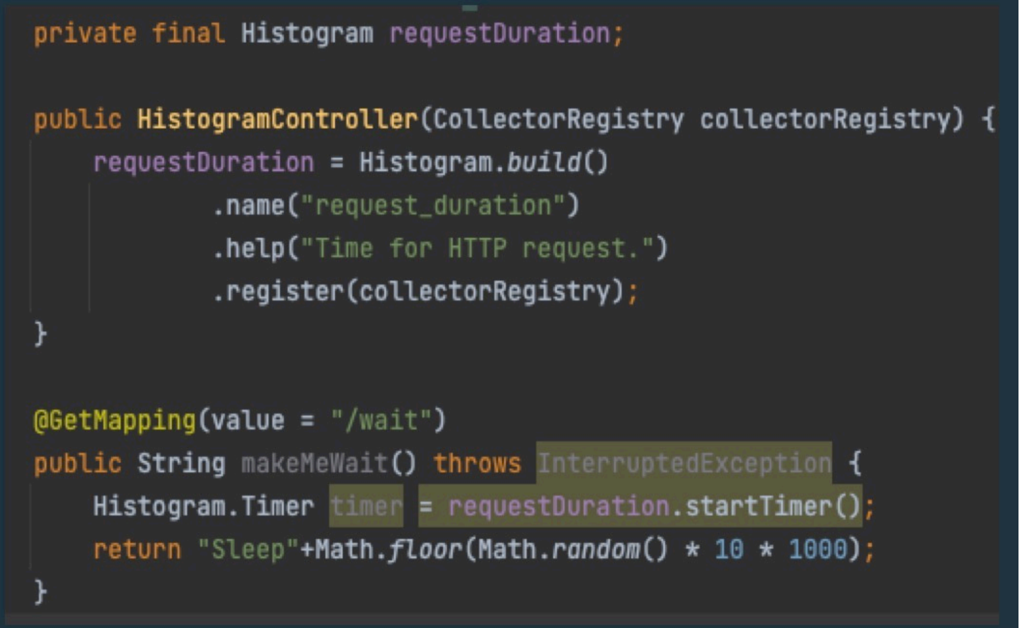
queue_size{**instance**="host.docker.internal:8080", **job**="SpringBoot-application"}

Prometheus

Histogram | measures the frequency of value observations that fall into specific buckets

E.g, measure request duration for a specific HTTP request call using histograms. Rather than storing every duration, prometheus make an approx by storing frequency into specific buckets

Default buckets are .005, .01, .025, .05 ....1, 2.5, 5, 7.5, 10. This is fine tuned for request duration below 10 secs.

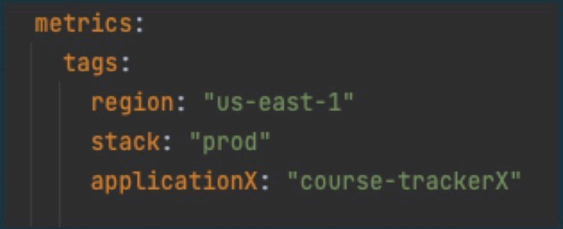


```
private final Histogram requestDuration;

public HistogramController(CollectorRegistry collectorRegistry) {
    requestDuration = Histogram.build()
            .name("request_duration")
            .help("Time for HTTP request.")
            .register(collectorRegistry);
}

@GetMapping(value = "/wait")
public String makeMeWait() throws InterruptedException {
    Histogram.Timer timer = requestDuration.startTimer();
    return "Sleep"+Math.floor(Math.random() * 10 * 1000);
}
```

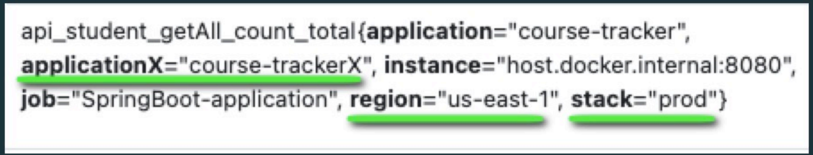
Prometheus

Metric collector Tags

Spring auto configuration enables instrumentation of all Spring Data Repository



```yaml
metrics:
  tags:
    region: "us-east-1"
    stack: "prod"
    applicationX: "course-trackerX"
```

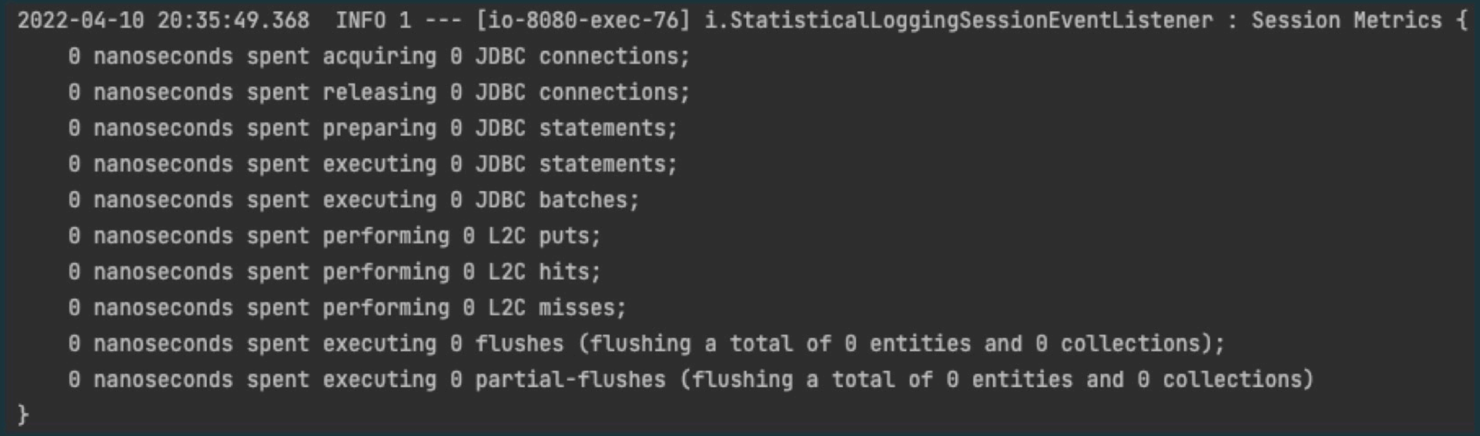


```
api_student_getAll_count_total{application="course-tracker",
applicationX="course-trackerX", instance="host.docker.internal:8080",
job="SpringBoot-application", region="us-east-1", stack="prod"}
```

Prometheus

Metric collector for Hibernate Metrics

Hibernate micrometer enabled statistics can provide valuable information

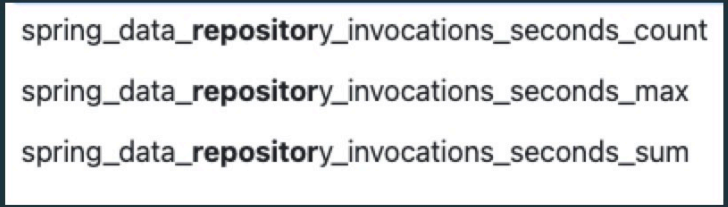


```
2022-04-10 20:35:49.368  INFO 1 --- [io-8080-exec-76] i.StatisticalLoggingSessionEventListener : Session Metrics {
    0 nanoseconds spent acquiring 0 JDBC connections;
    0 nanoseconds spent releasing 0 JDBC connections;
    0 nanoseconds spent preparing 0 JDBC statements;
    0 nanoseconds spent executing 0 JDBC statements;
    0 nanoseconds spent executing 0 JDBC batches;
    0 nanoseconds spent performing 0 L2C puts;
    0 nanoseconds spent performing 0 L2C hits;
    0 nanoseconds spent performing 0 L2C misses;
    0 nanoseconds spent executing 0 flushes (flushing a total of 0 entities and 0 collections);
    0 nanoseconds spent executing 0 partial-flushes (flushing a total of 0 entities and 0 collections)
}
```

Prometheus

Metric collector for Spring Data Repository Metrics

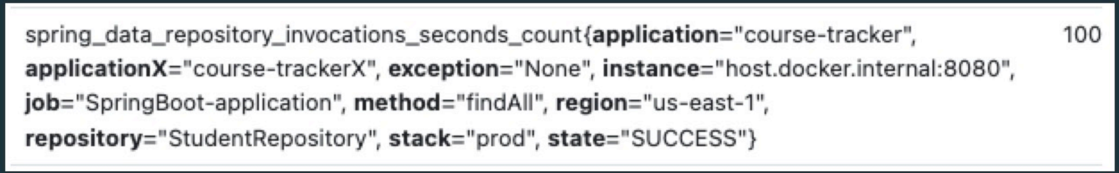
Spring auto configuration enables instrumentation of all Spring Data Repository



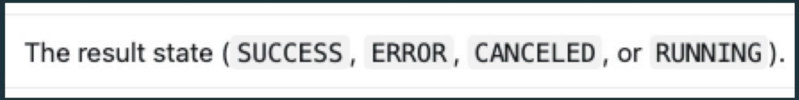
spring_data_**reposito**ry_invocations_seconds_count

spring_data_**reposito**ry_invocations_seconds_max

spring_data_**reposito**ry_invocations_seconds_sum



```
spring_data_repository_invocations_seconds_count{application="course-tracker",        100
applicationX="course-trackerX", exception="None", instance="host.docker.internal:8080",
job="SpringBoot-application", method="findAll", region="us-east-1",
repository="StudentRepository", stack="prod", state="SUCCESS"}
```



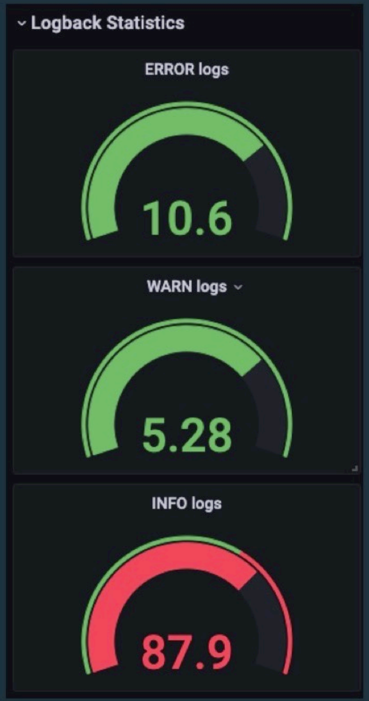
The result state (`SUCCESS`, `ERROR`, `CANCELED`, or `RUNNING`).

Prometheus

Metric collector for log4j appender logging

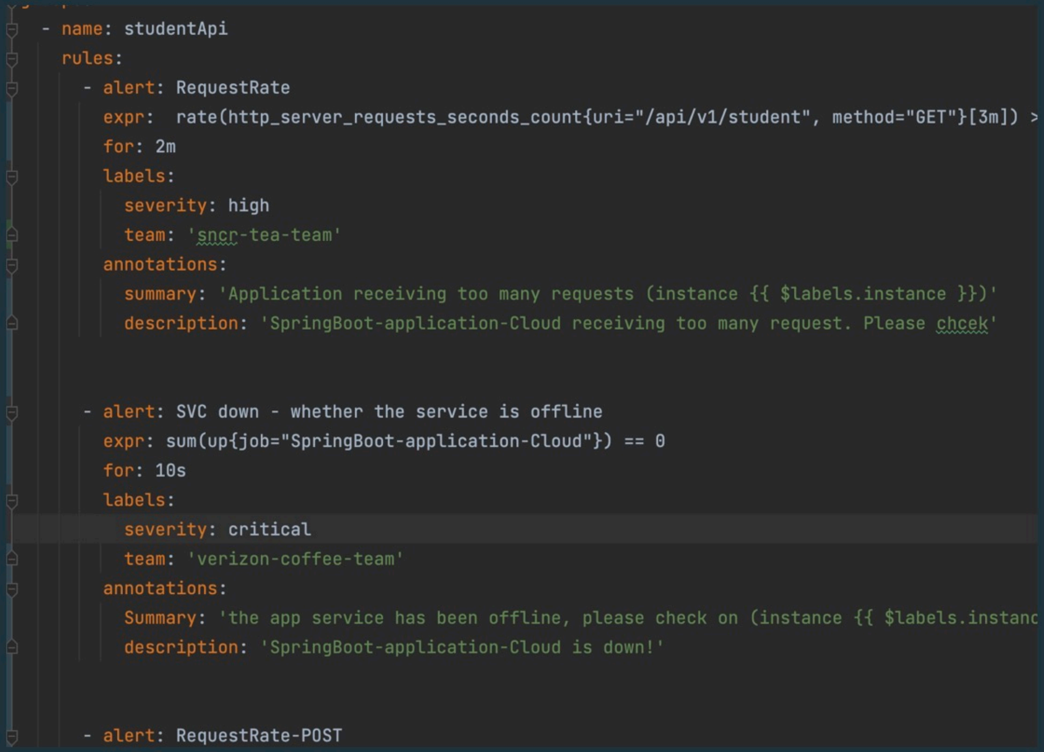
```
implementation 'io.prometheus:simpleclient_log4j2:0.15.0'
```

```
log.info("A INFO Message printed ");
log.trace("A TRACE Message printed ");
log.debug("A DEBUG Message printed ");
log.warn("A WARN Message printed ");
log.error("A ERROR Message printed ");
```



Prometheus

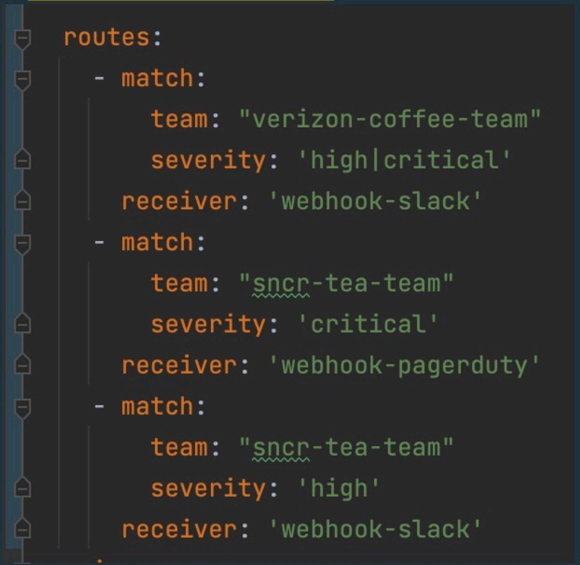
Rules.yml



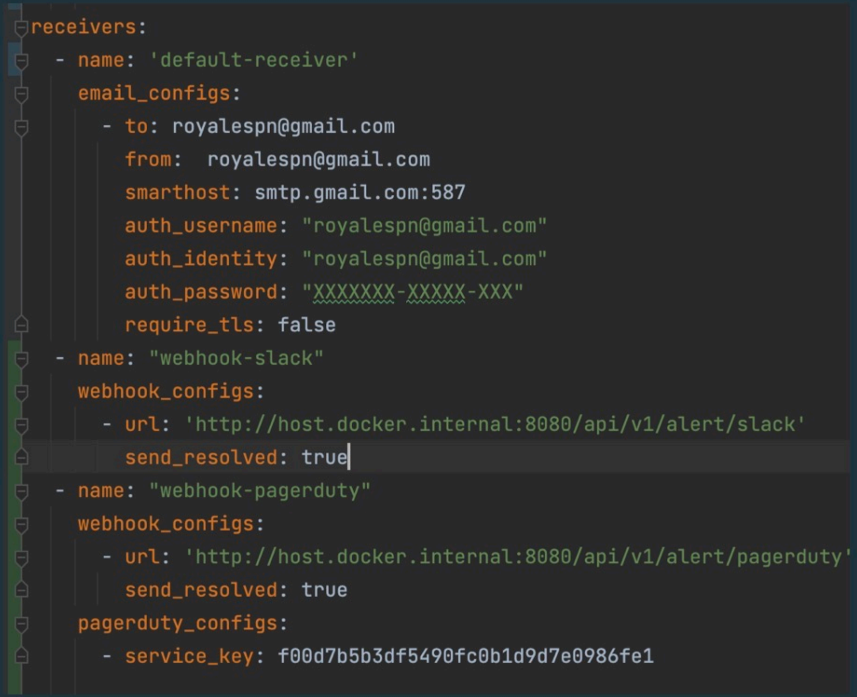
```
- name: studentApi
  rules:
    - alert: RequestRate
      expr:  rate(http_server_requests_seconds_count{uri="/api/v1/student", method="GET"}[3m]) >
      for: 2m
      labels:
        severity: high
        team: 'sncr-tea-team'
      annotations:
        summary: 'Application receiving too many requests (instance {{ $labels.instance }})'
        description: 'SpringBoot-application-Cloud receiving too many request. Please chcek'


    - alert: SVC down - whether the service is offline
      expr: sum(up{job="SpringBoot-application-Cloud"}) == 0
      for: 10s
      labels:
        severity: critical
        team: 'verizon-coffee-team'
      annotations:
        Summary: 'the app service has been offline, please check on (instance {{ $labels.instanc
        description: 'SpringBoot-application-Cloud is down!'


    - alert: RequestRate-POST
```

Prometheus

Rules.yml



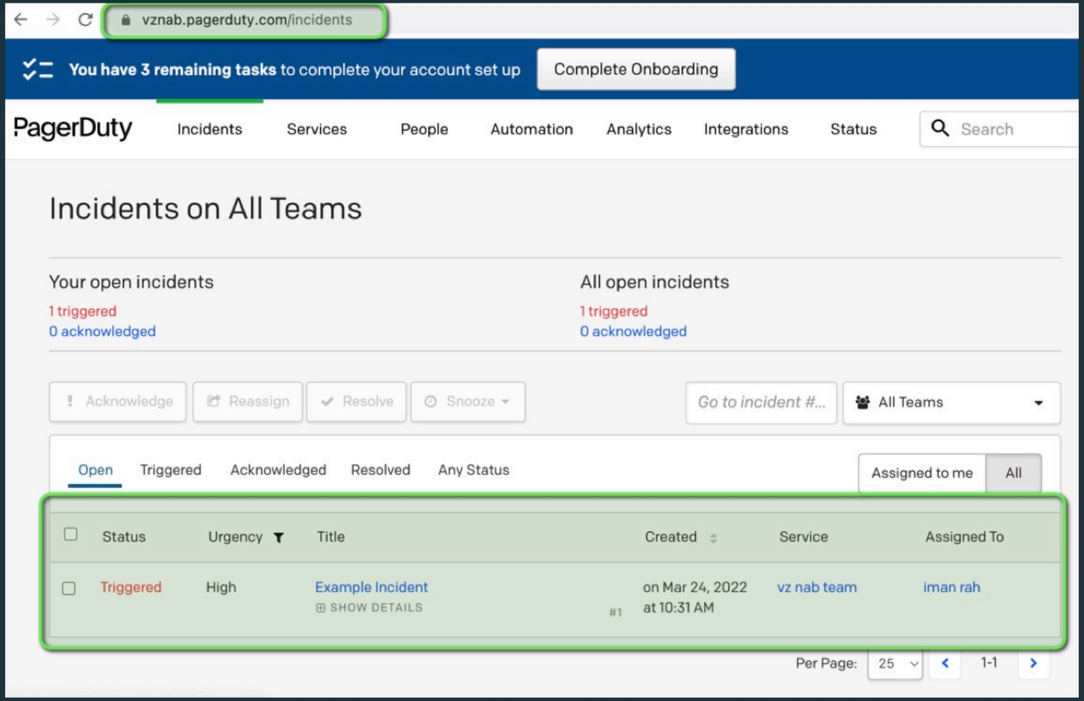
```yaml
routes:
  - match:
      team: "verizon-coffee-team"
      severity: 'high|critical'
    receiver: 'webhook-slack'
  - match:
      team: "sncr-tea-team"
      severity: 'critical'
    receiver: 'webhook-pagerduty'
  - match:
      team: "sncr-tea-team"
      severity: 'high'
    receiver: 'webhook-slack'
```



```yaml
receivers:
  - name: 'default-receiver'
    email_configs:
      - to: royalespn@gmail.com
        from:  royalespn@gmail.com
        smarthost: smtp.gmail.com:587
        auth_username: "royalespn@gmail.com"
        auth_identity: "royalespn@gmail.com"
        auth_password: "XXXXXXX-XXXXX-XXX"
        require_tls: false
  - name: "webhook-slack"
    webhook_configs:
      - url: 'http://host.docker.internal:8080/api/v1/alert/slack'
        send_resolved: true
  - name: "webhook-pagerduty"
    webhook_configs:
      - url: 'http://host.docker.internal:8080/api/v1/alert/pagerduty'
        send_resolved: true
    pagerduty_configs:
      - service_key: f00d7b5b3df5490fc0b1d9d7e0986fe1
```

Prometheus

Pagerduty



vznab.pagerduty.com/incidents

You have 3 remaining tasks to complete your account set up | Complete Onboarding

PagerDuty | Incidents | Services | People | Automation | Analytics | Integrations | Status | Search

Incidents on All Teams

Your open incidents
1 triggered
0 acknowledged

All open incidents
1 triggered
0 acknowledged

Acknowledge | Reassign | Resolve | Snooze | Go to incident #... | All Teams

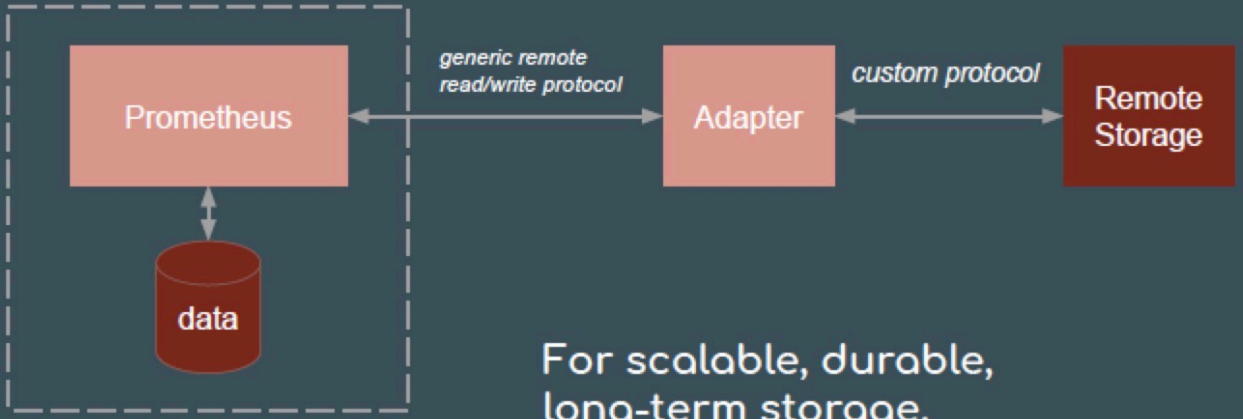
Open | Triggered | Acknowledged | Resolved | Any Status | Assigned to me | All

Status	Urgency	Title	Created	Service	Assigned To
Triggered	High	Example Incident SHOW DETAILS #1	on Mar 24, 2022 at 10:31 AM	vz nab team	iman rah

Per Page: 25 1-1

Prometheus

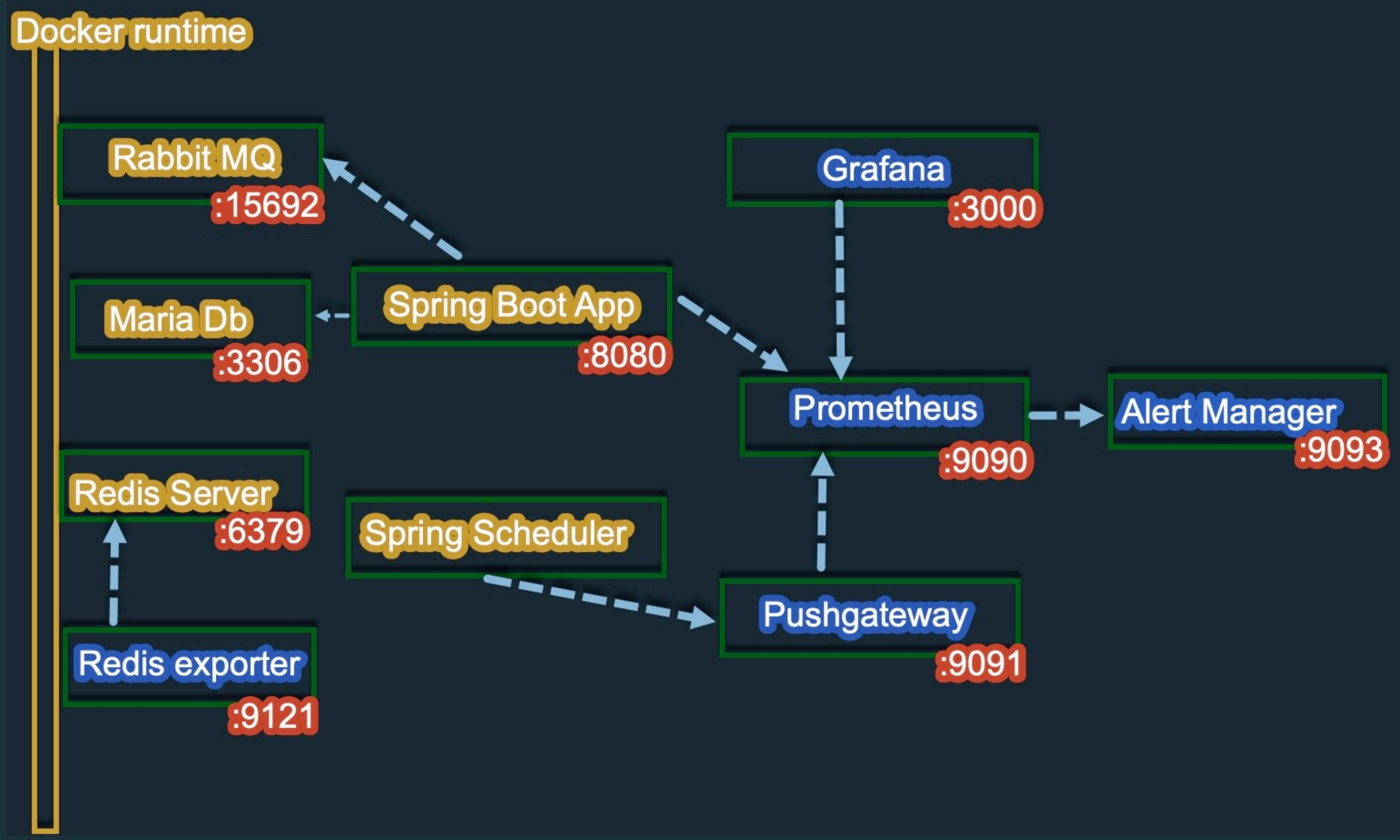
Remote Storage



For scalable, durable, long-term storage.

E.g.: Cortex, InfluxDB

Prometheus


Docker runtime
Rabbit MQ
:15692
Maria Db
:3306
Spring Boot App
:8080
Redis Server
:6379
Redis exporter
:9121
Spring Scheduler
Grafana
:3000
Prometheus
:9090
Alert Manager
:9093
Pushgateway
:9091

Prometheus

swagger API :	http://localhost:8080/swagger-ui/index.html
prometheus:	http://localhost:9090
grafana server:	http://localhost:3000 [admin/admin]
alert manager :	http://localhost:9093
pushgateway :	http://localhost:9091

Metrics endpoints:
-----------------------

redis metrics:	http://localhost:9121/metrics
rabbit-mq:	http://localhost:15692/metrics
springBoot	http://localhost:8080/actuator/prometheus

Grafana Dashboard Import ID:
--------------------------------------

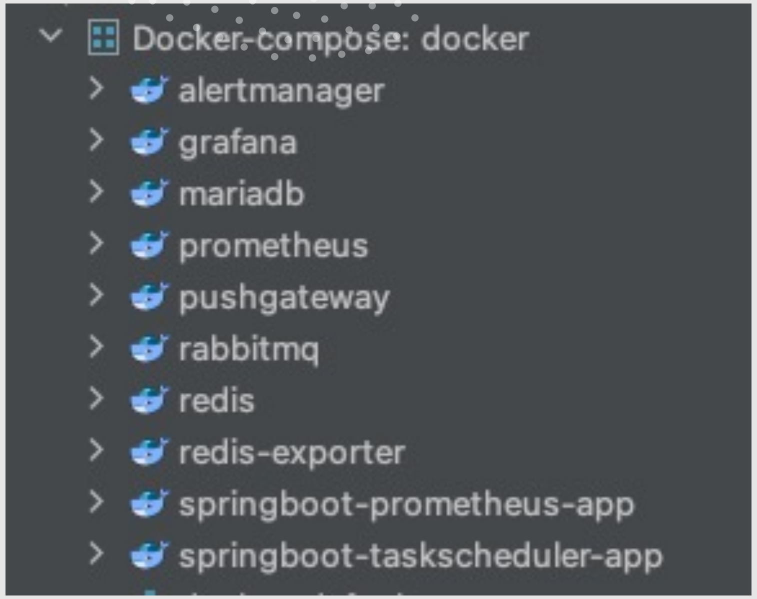
redis: 763
springBoot APM Dashboard: 12900
rabbitMQ: 10991

Run Gatling Script:
----------------------

```
gradlew gatlingRun
```

Docker-compose:
---------------------

```
docker-compose up -d
```



```
Docker-compose: docker
  alertmanager
  grafana
  mariadb
  prometheus
  pushgateway
  rabbitmq
  redis
  redis-exporter
  springboot-prometheus-app
  springboot-taskscheduler-app
```