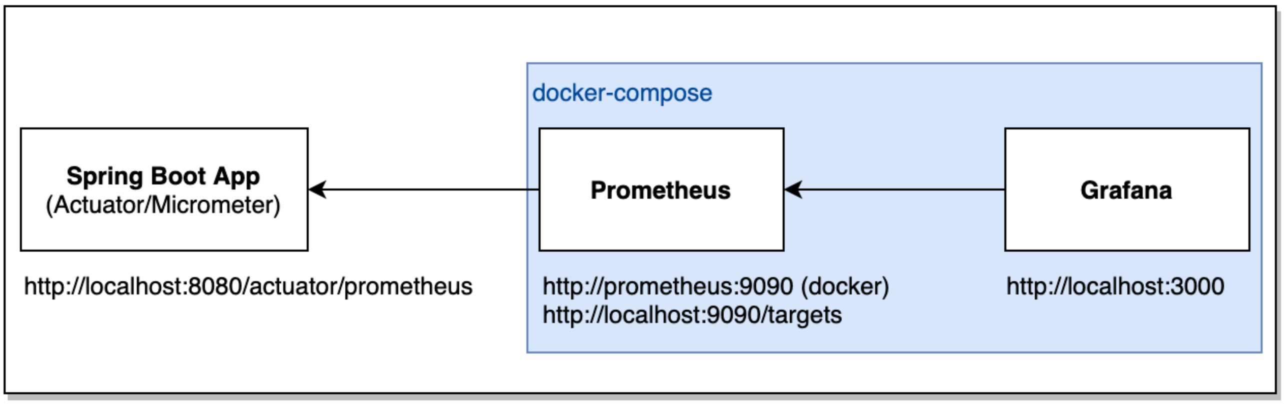
Setup Overview

Architecture


docker-compose
Spring Boot App
(Actuator/Micrometer)
Prometheus
Grafana
http://localhost:8080/actuator/prometheus
http://prometheus:9090 (docker)
http://localhost:9090/targets
http://localhost:3000

Prometheus

- is scraping the http://localhost:8080/actuator/prometheus

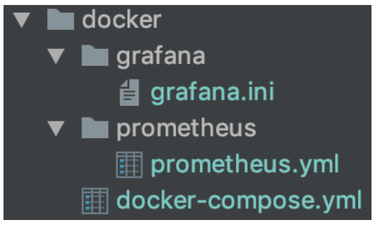
- Browser -> http://localhost:9090/targets

Grafana

- refers to the prometheus docker-compose service http://prometheus:9090
- which is the internal DNS that docker-compose offers, so no need to do local host

- Browser -> http://localhost:3000
- login: admin/admin

Docker setup



```
docker
  grafana
    grafana.ini
  prometheus
    prometheus.yml
  docker-compose.yml
```

docker-compose.yml

- 2 Services: Prometheus & Grafana
- Network
- Volumes

prometheus.yml

- URI to scrape (**host.docker.internal:8080***)
- Actuator path (/actuator/prometheus)
- Scrape interval seconds

* Spring Boot App URI. localhost won't work here because we'll be connecting to the HOST machine from the docker container. You must specify the network IP address.

grafana.ini

empty

Micrometer - create Metrics

Publish Metrics in Spring Boot

use **MeterRegistry** to add metrics to micrometer

```
public class TodoMetricHandlerInterceptor implements HandlerInterceptor {
    private final MeterRegistry meterRegistry;

    public TodoMetricHandlerInterceptor(final MeterRegistry meterRegistry) {
        this.meterRegistry = meterRegistry;
    }

    @Override
    public void afterCompletion(final HttpServletRequest request, final
HttpServletResponse response, final Object handler, final Exception ex) {
        final String pathKey = "api_".concat(request.getMethod())
                .concat(request.getRequestURI());
        meterRegistry.counter(pathKey).increment();
    }
}
```

Creates a counter for each Endpoint and increments it at each call
NOTE: HandlerInterceptor has nothing to do with Metrics, its just a way to intercept REST calls

Metric types

Counter, Timer, Gauge, DistributionSummary
https://www.baeldung.com/micrometer

Micrometer data export

Spring Boot Actuator **auto-configures** and registers every Micrometer registry based on dependencies

```
<dependency>
    <groupId>io.micrometer</groupId>
    <artifactId>micrometer-registry-prometheus</artifactId>
</dependency>
```

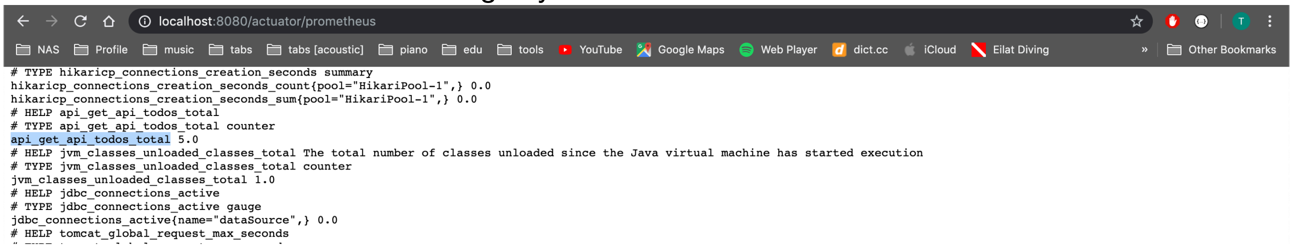
for Prometheus, the actuator configures the **/actuator/prometheus** endpoint

other Registries for data export

- micrometer-registry-jmx

Export for Prometheus Server

- Metrics are exposed to /actuator/prometheus
- Also metrics created with MeterRegistry are included



```
# TYPE hikaricp_connections_creation_seconds summary
hikaricp_connections_creation_seconds_count{pool="HikariPool-1",} 0.0
hikaricp_connections_creation_seconds_sum{pool="HikariPool-1",} 0.0
# HELP api_get_api_todos_total
# TYPE api_get_api_todos_total counter
api_get_api_todos_total 5.0
# HELP jvm_classes_unloaded_classes_total The total number of classes unloaded since the Java virtual machine has started execution
# TYPE jvm_classes_unloaded_classes_total counter
jvm_classes_unloaded_classes_total 1.0
# HELP jdbc_connections_active
# TYPE jdbc_connections_active gauge
jdbc_connections_active{name="dataSource",} 0.0
# HELP tomcat_global_request_max_seconds
```

NOTE: metrics for endpoints are listed after the endpoint was called for the first time

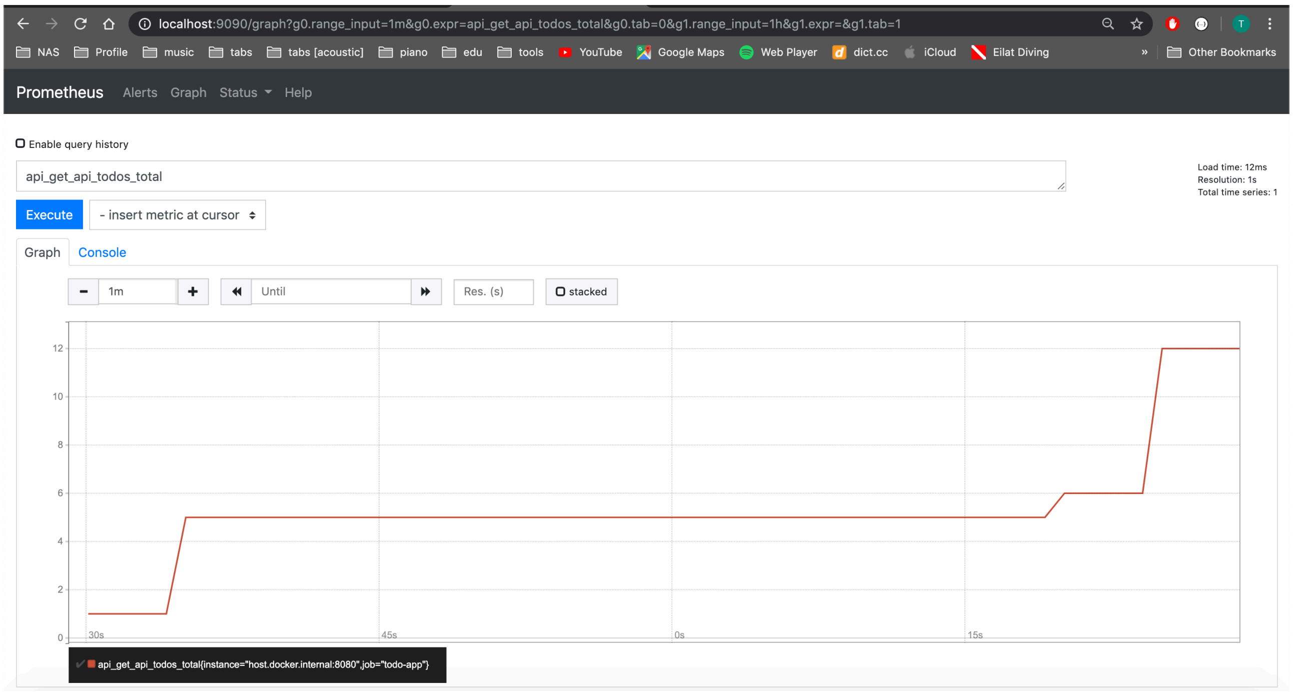
Prometheus - scrape metrics

Prometheus

- is scraping the http://localhost:8080/actuator/prometheus
- Browser: http://localhost:9090/targets

Graphs

Enter the name of a metric



localhost:9090/graph?g0.range_input=1m&g0.expr=api_get_api_todos_total&g0.tab=0&g1.range_input=1h&g1.expr=&g1.tab=1

☐ Enable query history

api_get_api_todos_total

Load time: 12ms
Resolution: 1s
Total time series: 1

1m

☐ stacked

api_get_api_todos_total{instance="host.docker.internal:8080",job="todo-app"}

Config

```
global:
  scrape_interval:     5s
  evaluation_interval: 5s

scrape_configs:
- job_name: 'todo-app'

  metrics_path: '/actuator/prometheus'
  scrape_interval: 5s
  static_configs:
  - targets: ['host.docker.internal:8080']
```

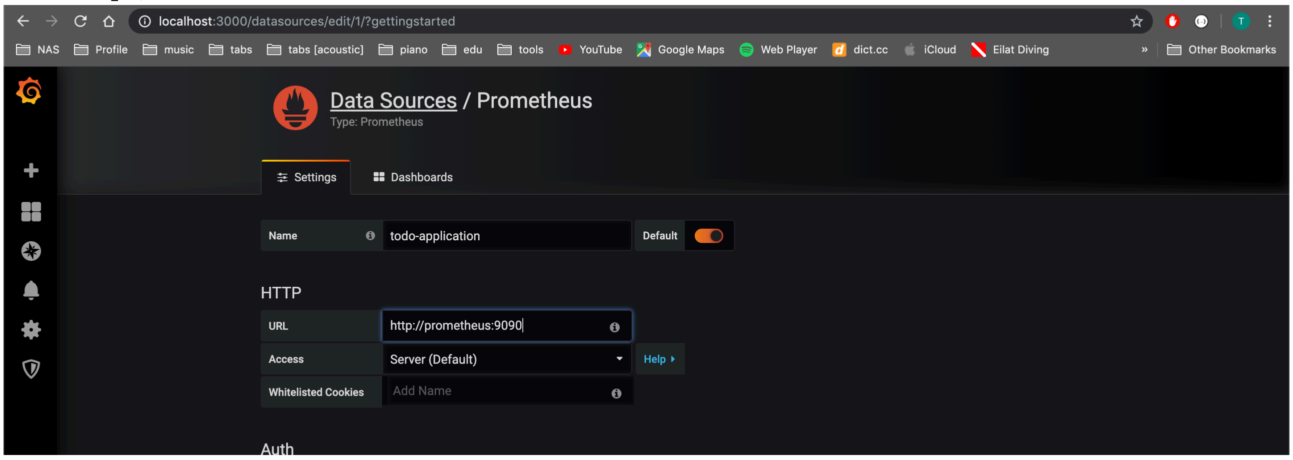
docker/prometheus/prometheus.yml

Grafana - dashboards

Grafana

- Receives Data from Prometheus
- Browser: http://localhost:3000
- Login: admin/admin

Setup, add Prometheus Datasource



localhost:3000/datasources/edit/1/?gettingstarted

Data Sources / Prometheus
Type: Prometheus

Settings | Dashboards

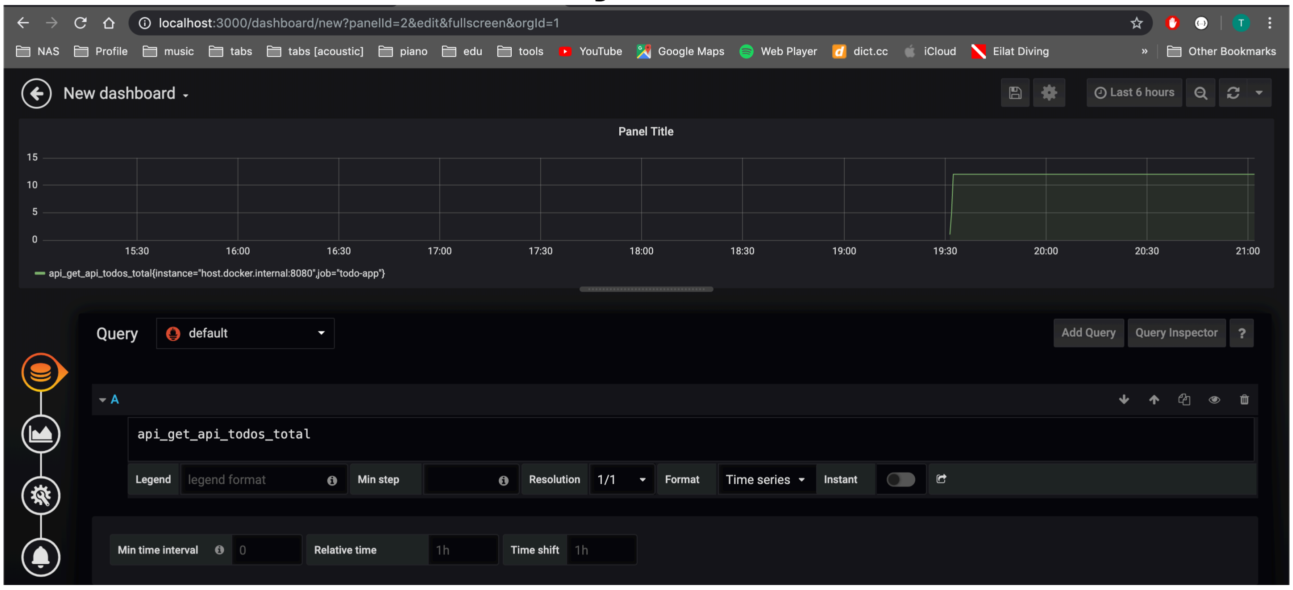
Name: todo-application — Default: on

HTTP

URL: http://prometheus:9090
Access: Server (Default)
Whitelisted Cookies: Add Name

Auth

Create Dashboard & Query



localhost:3000/dashboard/new?panelId=2&edit&fullscreen&orgId=1

New dashboard — Last 6 hours

Panel Title

api_get_api_todos_total{instance="host.docker.internal:8080",job="todo-app"}

Query: default — Add Query | Query Inspector

A

```
api_get_api_todos_total
```

Legend: legend format | Min step | Resolution: 1/1 | Format: Time series | Instant

Min time interval: 0 | Relative time: 1h | Time shift: 1h

Spring Boot Dashboard

- A dashboard of every metric that Spring Boot Actuator exposes
- import spring-boot-statistics_rev2.json in Grafana

NOTE: use http://prometheus:9090, because of internal docker DNS

Monitoring Systems

Prometheus, Netflix Atlas, CloudWatch, Datadog, Graphite, Ganglia, JMX, InfluxDB/Telegraf, New Relic, StatsD, SignalFX, and WaveFront