Prometheus Actuator

1)修改项目的pom.xml文件，添加actuator及micrometer依赖；

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>

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

2)修改应用配置文件application.yml，通过actuator暴露监控端口/actuator/prometheus；

eladmin-system/src/main/resources/config/application-dev.yml

```
@@ -120,7 +120,7 @@ management:
  endpoints:
    web:
      exposure:
-       include: health,info, metrics, shutdown
+       include: health,info, metrics, shutdown, prometheus
```

3) 增加对Header "application/openmetrics-text", "text/plain"的支持，Promethuse在发送时使用"application/openmetrics-text", 但接收response时需要 "text/plain"，具体参看：
https://github.com/spring-projects/spring-boot/issues/28446
https://github.com/OpenObservability/OpenMetrics/blob/main/specification/OpenMetrics.md

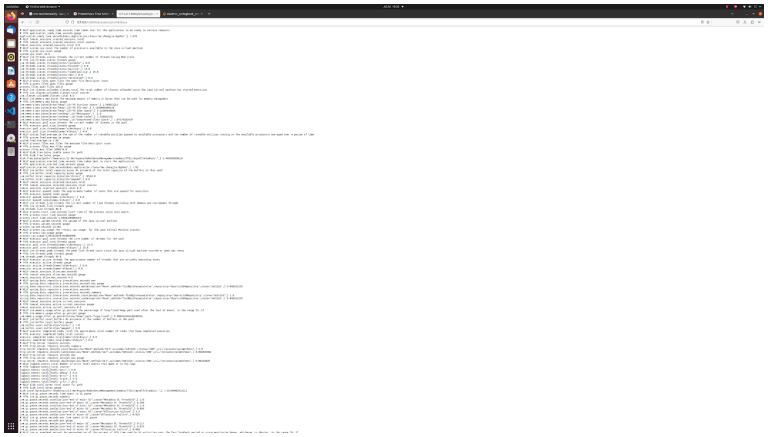
```
public FilterRegistrationBean someFilterRegistration()
{
    FilterRegistrationBean registration = new FilterRegistrationBean();
    registration.setFilter(new PromethuseResponseFilter());// 配置一个返回值过滤器
    registration.addUrlPatterns("/actuator/prometheus");
    registration.addInitParameter("paramName", "paramValue");
    registration.setName("responseFilter");
    return registration;
}
```

4)测试（需要安装Promethuse和Graphana）：https://zhuanlan.zhihu.com/p/389489129

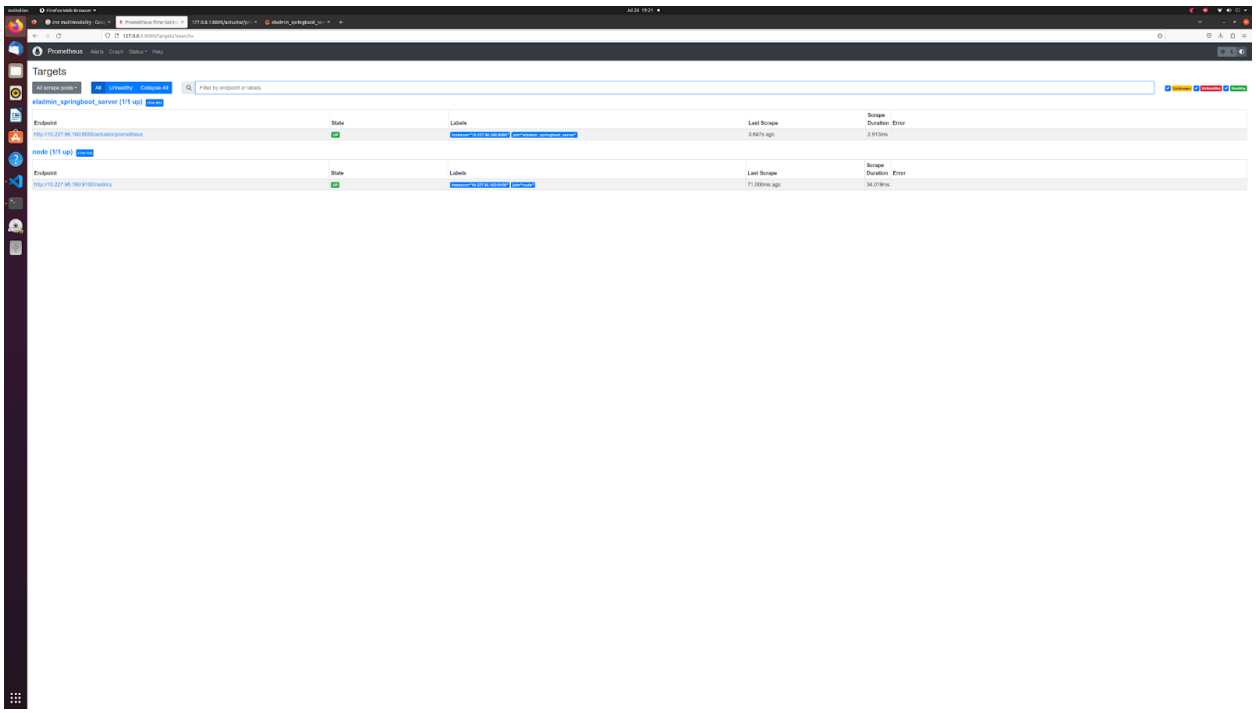
4.1）API: http://127.0.0.1:8000/actuator/prometheus
监控项示例：

```
# TYPE executor_completed_tasks counter
# HELP executor_completed_tasks The approximate total number of tasks that have completed execution
executor_completed_tasks_total{name="otherAsync"} 0.0
executor_completed_tasks_total{name="elAsync"} 0.0
# TYPE system_load_average_1m gauge
# HELP system_load_average_1m The sum of the number of runnable entities queued to available processors and the number of runnable entities running on the available processors averaged over a period of time
system_load_average_1m 0.27
# TYPE application_ready_time_seconds gauge
# HELP application_ready_time_seconds Time taken (ms) for the application to be ready to service requests
application_ready_time_seconds{main_application_class="me.zhengjie.AppRun"} 7.998
```

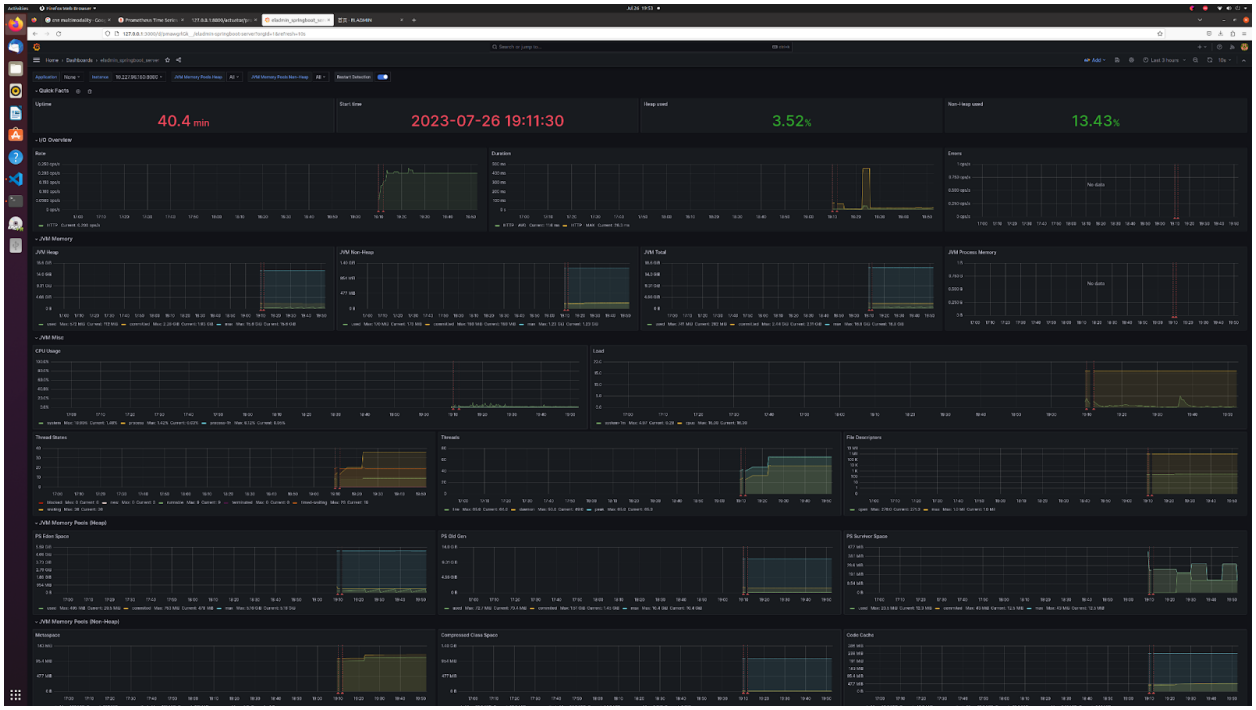
```
# TYPE system_cpu_count gauge
# HELP system_cpu_count The number of processors available to the Java virtual machine
system_cpu_count 16.0
# TYPE jvm_memory_committed_bytes gauge
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
jvm_memory_committed_bytes{area="heap",id="PS Survivor Space"} 2.5165824E7
jvm_memory_committed_bytes{area="heap",id="PS Old Gen"} 1.57810688E9
jvm_memory_committed_bytes{area="heap",id="PS Eden Space"} 5.43162368E8
jvm_memory_committed_bytes{area="nonheap",id="Metaspace"} 1.08679168E8
jvm_memory_committed_bytes{area="nonheap",id="Code Cache"} 4.4040192E7
jvm_memory_committed_bytes{area="nonheap",id="Compressed Class Space"} 1.4307328E7
```



4.2）Prometheus 界面 127.0.0.1:9090



4.3）Grafana 界面 127.0.0.1:3000


Uptime
40.4 min
Start time
2023-07-26 19:11:30
Heap used
3.52%
Non-Heap used
13.43%

4.4)界面正常：