

CS544

LESSON 13

MONITORING

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
November 28 Lesson 1 Introduction Spring framework Dependency injection	November 29 Lesson 2 Spring Boot AOP	November 30 Lesson 3 JDBC JPA	December 1 Lesson 4 JPA mapping 1	December 2 Lesson 5 JPA mapping 2	December 3 Lesson 6 JPA queries	December 4
December 5 Lesson 7 Transactions	December 6 Lesson 8 MongoDB	December 7 Midterm Review	December 8 Midterm exam	December 9 Lesson 9 REST webservices	December 10 Lesson 10 SOAP webservices	December 11
December 12 Lesson 11 Messaging	December 13 Lesson 12 Scheduling Events Configuration	December 14 Lesson 13 Monitoring	December 15 Lesson 14 Testing your application	December 16 Final review	December 17 Final exam	December 18
December 19 Project	December 20 Project	December 21 Project	December 22 Presentations			

SPRING BOOT LOGGING

Zero configuration logging

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

- When you add this dependency, Spring boot automatically uses Log4J for logging.

Using a Logger

@Component

```
public class CustomerService {  
    Logger logger = LoggerFactory.getLogger(CustomerService.class);  
  
    public void addCustomer(){  
        logger.trace("A TRACE Message");  
    }  
  
    public void updateCustomer(){  
        logger.debug("A DEBUG Message");  
    }  
  
    public void removeCustomer(){  
        logger.info("An INFO Message");  
    }  
  
    public void findCustomerById(){  
        logger.warn("A WARN Message");  
    }  
  
    public void findCustomersByName(){  
        logger.error("An ERROR Message");  
    }  
}
```

The application

@SpringBootApplication

```
public class CustomerApplication implements CommandLineRunner {
```

@Autowired

```
private CustomerService customerService;
```

```
public static void main(String[] args) {  
    SpringApplication.run(CustomerApplication.class, args);  
}
```

@Override

```
public void run(String... args) throws Exception {  
    customerService.addCustomer();  
    customerService.updateCustomer();  
    customerService.removeCustomer();  
    customerService.findCustomerById();  
    customerService.findCustomersByName();  
}
```

```
2022-07-05 13:32:45.706 INFO 1948 --- [main] app.CustomerApplication : Starting CustomerApplication using Java
11.0.1 on DESKTOP-BVHRK6K with PID 1948 (C:\EnterpriseArchiterture\demo code\Lesson13Logging\target\classes started by vedam in
C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:32:45.708 INFO 1948 --- [main] app.CustomerApplication : No active profile set, falling back to
default profiles: default
2022-07-05 13:32:46.216 INFO 1948 --- [main] app.CustomerApplication : Started CustomerApplication in 0.936
seconds (JVM running for 1.381)
2022-07-05 13:32:46.218 INFO 1948 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:32:46.218 WARN 1948 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:32:46.218 ERROR 1948 --- [main] app.CustomerService : An ERROR Message
```

7

Logging level

- TRACE: gives detailed information about the code
- DEBUG: gives more specific diagnostic information that you need during debugging
- INFO (default level): gives high level information
- WARN: potential problems that might cause problems
- ERROR: serious issues like exceptions

ERROR ← **WARN** ← **INFO** ← **DEBUG** ← **TRACE**

Logging level

Default logging level is INFO

```
#logging.level.root=DEBUG
```

2022-07-05 12:18:51.737	INFO	29548	---	[main]	app.CustomerService	:	An INFO Message
2022-07-05 12:18:51.737	WARN	29548	---	[main]	app.CustomerService	:	A WARN Message
2022-07-05 12:18:51.737	ERROR	29548	---	[main]	app.CustomerService	:	An ERROR Message

```
logging.level.root=WARN
```

2022-07-05 12:19:52.173	WARN	2692	---	[main]	app.CustomerService	:	A WARN Message
2022-07-05 12:19:52.175	ERROR	2692	---	[main]	app.CustomerService	:	An ERROR Message

```
logging.level.root=ERROR
```

2022-07-05 12:20:57.133	ERROR	3560	---	[main]	app.CustomerService	:	An ERROR Message
-------------------------	-------	------	-----	---	-------	---------------------	---	------------------

Logging level



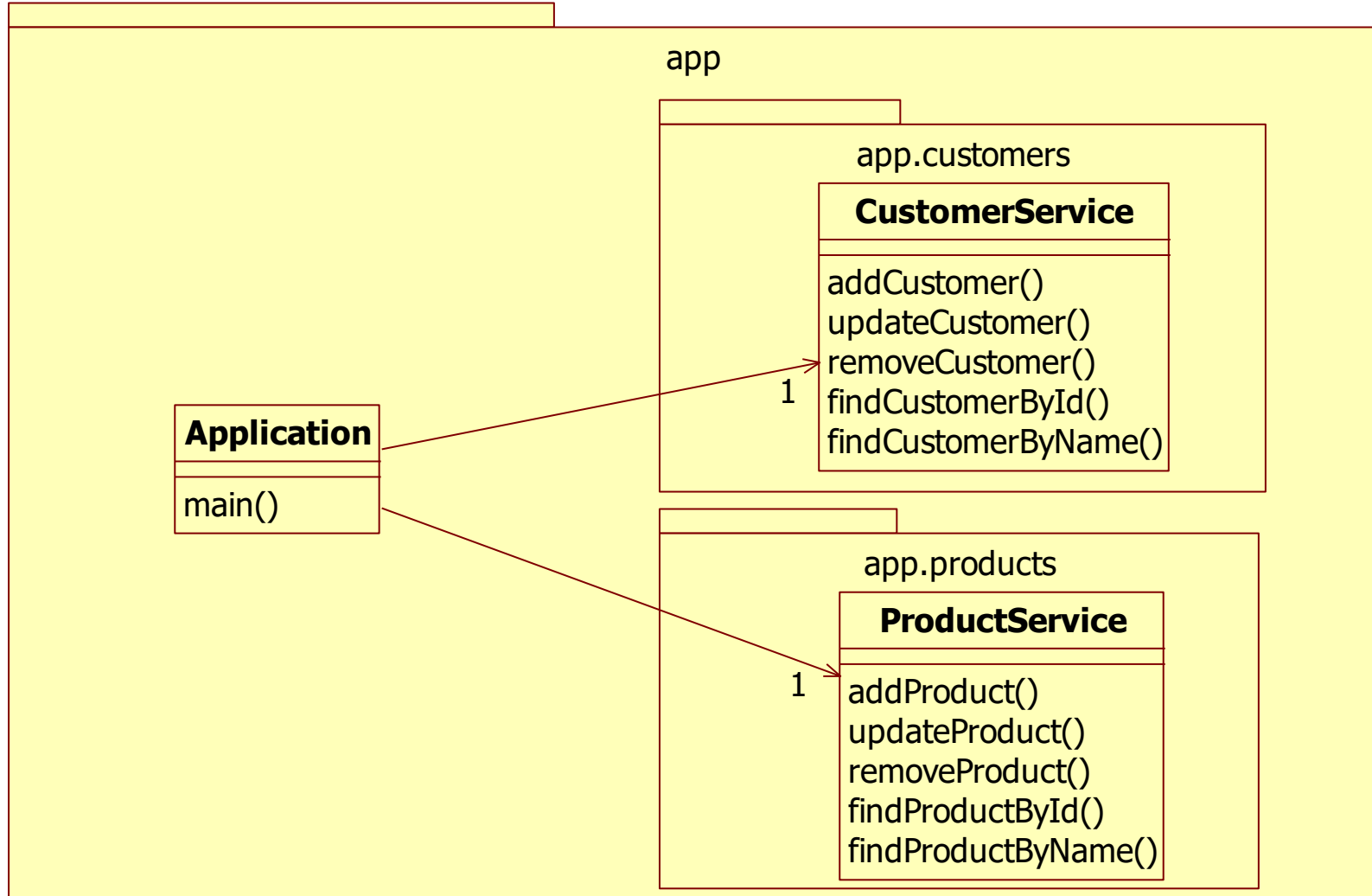
```
logging.level.root=DEBUG
```

```
2022-07-05 12:16:15.581 DEBUG 29812 --- [main] app.CustomerService : A DEBUG Message
2022-07-05 12:16:15.581 INFO 29812 --- [main] app.CustomerService : An INFO Message
2022-07-05 12:16:15.581 WARN 29812 --- [main] app.CustomerService : A WARN Message
2022-07-05 12:16:15.581 ERROR 29812 --- [main] app.CustomerService : An ERROR Message
```

```
logging.level.root=TRACE
```

```
2022-07-05 12:13:47.372 TRACE 8380 --- [main] app.CustomerService : A TRACE Message
2022-07-05 12:13:47.372 DEBUG 8380 --- [main] app.CustomerService : A DEBUG Message
2022-07-05 12:13:47.372 INFO 8380 --- [main] app.CustomerService : An INFO Message
2022-07-05 12:13:47.372 WARN 8380 --- [main] app.CustomerService : A WARN Message
2022-07-05 12:13:47.372 ERROR 8380 --- [main] app.CustomerService : An ERROR Message
```

Logging level on packages



The application

@SpringBootApplication

```
public class Application implements CommandLineRunner {
```

@Autowired

```
private CustomerService customerService;
```

@Autowired

```
private ProductService productService;
```

```
public static void main(String[] args) {
```

```
    SpringApplication.run(Application.class, args);
```

```
}
```

@Override

```
public void run(String... args) throws Exception {
```

```
    Logger logger = LoggerFactory.getLogger(Application.class);
```

```
    logger.info("An INFO Message");
```

```
    logger.error("An ERROR Message");
```

```
    customerService.addCustomer();
```

```
    customerService.updateCustomer();
```

```
    customerService.removeCustomer();
```

```
    customerService.findCustomerById();
```

```
    customerService.findCustomersByName();
```

```
    productService.addProduct();
```

```
    productService.updateProduct();
```

```
    productService.removeProduct();
```

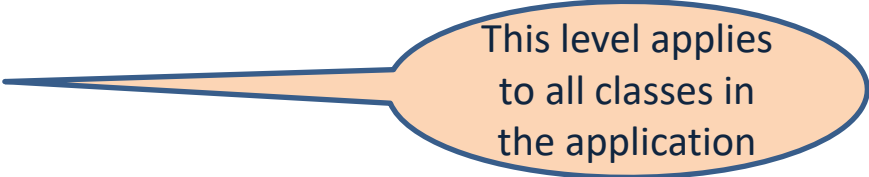
```
    productService.findProductById();
```

```
    productService.findProductByName();
```

```
}
```

Logging level on packages

```
logging.level.root=INFO
```



This level applies
to all classes in
the application

2022-07-05 19:10:59.036	INFO	29528	---	[main]	app.Application	:	An INFO Message
2022-07-05 19:10:59.036	ERROR	29528	---	[main]	app.Application	:	An ERROR Message
2022-07-05 19:10:59.036	INFO	29528	---	[main]	app.customers.CustomerService	:	An INFO Message
2022-07-05 19:10:59.036	WARN	29528	---	[main]	app.customers.CustomerService	:	A WARN Message
2022-07-05 19:10:59.036	ERROR	29528	---	[main]	app.customers.CustomerService	:	An ERROR Message
2022-07-05 19:10:59.036	INFO	29528	---	[main]	app.products.ProductService	:	An INFO Message
2022-07-05 19:10:59.036	WARN	29528	---	[main]	app.products.ProductService	:	A WARN Message
2022-07-05 19:10:59.036	ERROR	29528	---	[main]	app.products.ProductService	:	An ERROR Message

Logging level on packages

```
logging.level.root=INFO
logging.level.app.customers=ERROR
logging.level.app.products=TRACE
```

Logging level on
individual
packages

```
2022-07-05 19:21:21.497 INFO 30568 --- [main] app.Application : An INFO Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.Application : An ERROR Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.customers.CustomerService : An ERROR Message
2022-07-05 19:21:21.497 TRACE 30568 --- [main] app.products.ProductService : A TRACE Message
2022-07-05 19:21:21.497 DEBUG 30568 --- [main] app.products.ProductService : A DEBUG Message
2022-07-05 19:21:21.497 INFO 30568 --- [main] app.products.ProductService : An INFO Message
2022-07-05 19:21:21.497 WARN 30568 --- [main] app.products.ProductService : A WARN Message
2022-07-05 19:21:21.497 ERROR 30568 --- [main] app.products.ProductService : An ERROR Message
```

Log format



2022-07-05	12:13:47.372	TRACE	8380	---	[main]	app.CustomerService	:	A TRACE Message
2022-07-05	12:13:47.372	DEBUG	8380	---	[main]	app.CustomerService	:	A DEBUG Message
2022-07-05	12:13:47.372	INFO	8380	---	[main]	app.CustomerService	:	An INFO Message
2022-07-05	12:13:47.372	WARN	8380	---	[main]	app.CustomerService	:	A WARN Message
2022-07-05	12:13:47.372	ERROR	8380	---	[main]	app.CustomerService	:	An ERROR Message
1	2	3	4			5	6		7

1. Date and Time
2. Log level
3. Process ID
4. The separator ---
5. Thread name
6. Logger name source class
7. Log message

Change Log format

```
logging.level.root=INFO
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
2022-07-05 12:37:13 - app.CustomerService - An INFO Message
```

```
2022-07-05 12:37:13 - app.CustomerService - A WARN Message
```

```
2022-07-05 12:37:13 - app.CustomerService - An ERROR Message
```


Change Log format

```
logging.level.root=INFO
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} - %logger - %msg%n
```

```
2022-07-05 12:37:13 - app.CustomerService - An INFO Message
```

```
2022-07-05 12:37:13 - app.CustomerService - A WARN Message
```

```
2022-07-05 12:37:13 - app.CustomerService - An ERROR Message
```

```
logging.pattern.console= %d{yyyy-MM-dd HH:mm:ss} [%thread] %level %logger - %msg%n
```

```
logging.pattern.file= %d{yyyy-MM-dd HH:mm:ss} [%thread] %level %logger - %msg%n
```

```
2022-07-05 12:46:26 [main] INFO app.CustomerService - An INFO Message
```

```
2022-07-05 12:46:26 [main] WARN app.CustomerService - A WARN Message
```

```
2022-07-05 12:46:26 [main] ERROR app.CustomerService - An ERROR Message
```

Logging to a file

```
logging.file.name=c:/temp/application.log
```

C:\temp\application.log

```
2022-07-05 13:52:49.002 INFO 5640 --- [main] app.CustomerApplication : Starting
CustomerApplication using Java 11.0.1 on DESKTOP-BVHRK6K with PID 5640 (C:\EnterpriseArchiterture\demo
code\Lesson13Logging\target\classes started by vedam in C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:52:49.005 INFO 5640 --- [main] app.CustomerApplication : No active profile set,
falling back to default profiles: default
2022-07-05 13:52:49.628 INFO 5640 --- [main] app.CustomerApplication : Started
CustomerApplication in 1.141 seconds (JVM running for 1.569)
2022-07-05 13:52:49.634 INFO 5640 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:52:49.634 WARN 5640 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:52:49.634 ERROR 5640 --- [main] app.CustomerService : An ERROR Message
```

Logging to a file



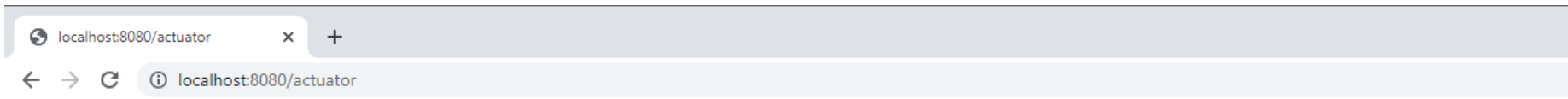
```
logging.file.path=c:/temp/logs
```

C:\temp\logs\spring.log

```
2022-07-05 13:52:49.002 INFO 5640 --- [main] app.CustomerApplication : Starting
CustomerApplication using Java 11.0.1 on DESKTOP-BVHRK6K with PID 5640 (C:\EnterpriseArchiterture\demo
code\Lesson13Logging\target\classes started by vedam in C:\EnterpriseArchiterture\demo code\Lesson13Logging)
2022-07-05 13:52:49.005 INFO 5640 --- [main] app.CustomerApplication : No active profile set,
falling back to default profiles: default
2022-07-05 13:52:49.628 INFO 5640 --- [main] app.CustomerApplication : Started
CustomerApplication in 1.141 seconds (JVM running for 1.569)
2022-07-05 13:52:49.634 INFO 5640 --- [main] app.CustomerService : An INFO Message
2022-07-05 13:52:49.634 WARN 5640 --- [main] app.CustomerService : A WARN Message
2022-07-05 13:52:49.634 ERROR 5640 --- [main] app.CustomerService : An ERROR Message
```

ACTUATORS

/actuator



```
{"_links":{"self":{"href":"http://localhost:8080/actuator","templated":false},"beans":{"href":"http://localhost:8080/actuator/beans","templated":false},"caches-cache":{"href":"http://localhost:8080/actuator/caches/{cache}","templated":true},"caches":{"href":"http://localhost:8080/actuator/caches","templated":false},"health":{"href":"http://localhost:8080/actuator/health","templated":false},"health-path":{"href":"http://localhost:8080/actuator/health/{*path}","templated":true},"info":{"href":"http://localhost:8080/actuator/info","templated":false},"conditions":{"href":"http://localhost:8080/actuator/conditions","templated":false},"shutdown":{"href":"http://localhost:8080/actuator/shutdown","templated":false},"configprops":{"href":"http://localhost:8080/actuator/configprops","templated":false},"configprops-prefix":{"href":"http://localhost:8080/actuator/configprops/{prefix}","templated":true},"env":{"href":"http://localhost:8080/actuator/env","templated":false},"env-toMatch":{"href":"http://localhost:8080/actuator/env/{toMatch}","templated":true},"loggers":{"href":"http://localhost:8080/actuator/loggers","templated":false},"loggers-name":{"href":"http://localhost:8080/actuator/loggers/{name}","templated":true},"heapdump":{"href":"http://localhost:8080/actuator/heapdump","templated":false},"threaddump":{"href":"http://localhost:8080/actuator/threaddump","templated":false},"metrics-requiredMetricName":{"href":"http://localhost:8080/actuator/metrics/{requiredMetricName}","templated":true},"metrics":{"href":"http://localhost:8080/actuator/metrics","templated":false},"scheduledtasks":{"href":"http://localhost:8080/actuator/scheduledtasks","templated":false},"mappings":{"href":"http://localhost:8080/actuator/mappings","templated":false}}}
```

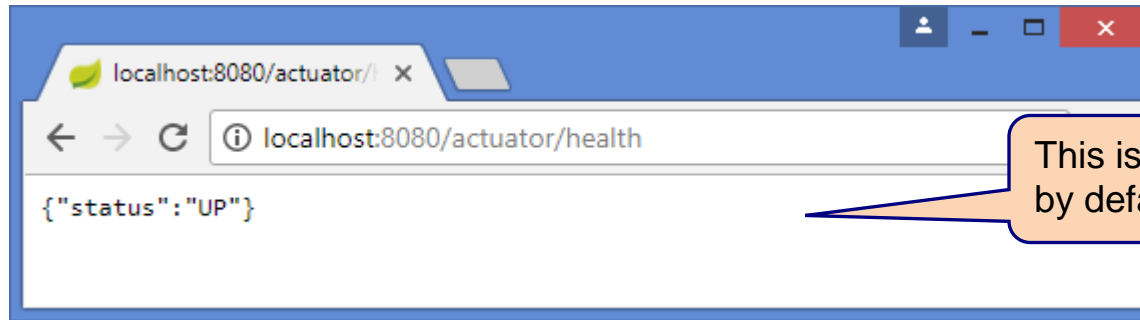
Actuator

- Actuator brings production-ready features to our application

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

- Once this dependency is on the classpath several endpoints are available for us out of the box.
- You can modify existing actuators and you can write you own actuators

/actuator/health



This is all that is shown by default



`management.endpoint.health.show-details=ALWAYS`

Show all health information

Exposing actuators

- Only the /health actuator is exposed by default

- Exposing particular actuators

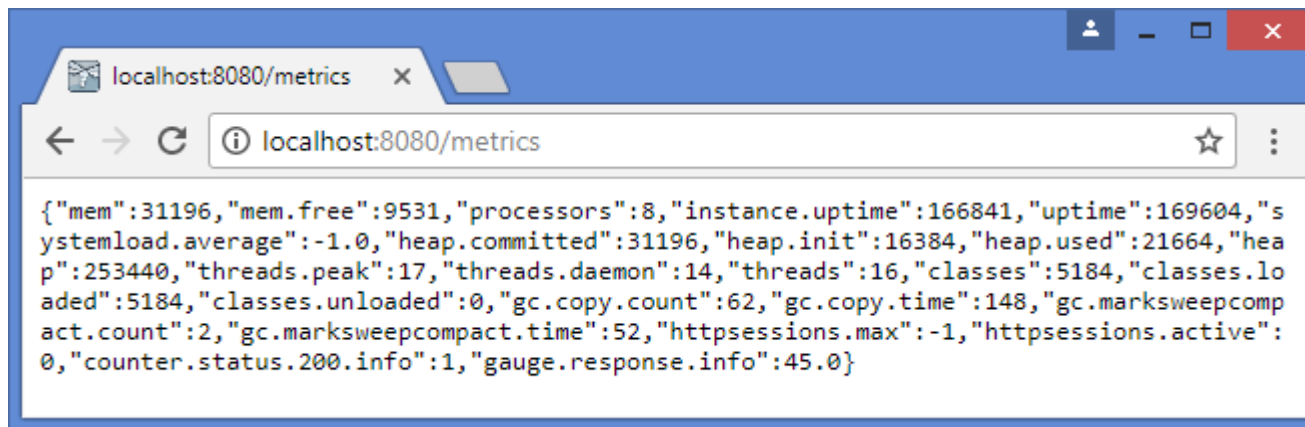
```
management.endpoints.web.exposure.include=beans,mappings
```

- Exposing all actuators

```
management.endpoints.web.exposure.include=*
```

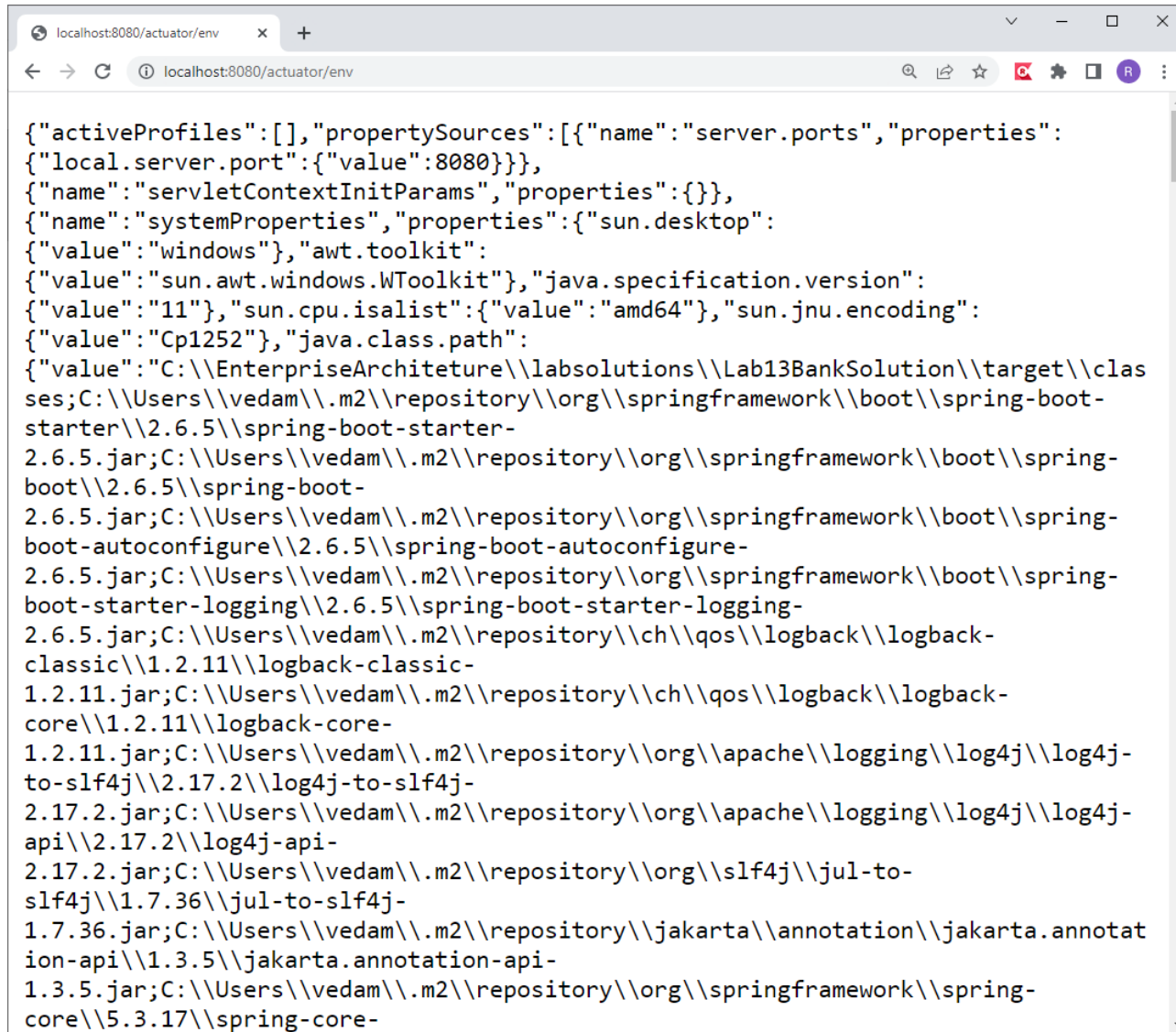

/actuator/metrics

- Gives information such as memory, heap, processors, threads, classes loaded, classes unloaded, thread pools along with some HTTP metrics as well



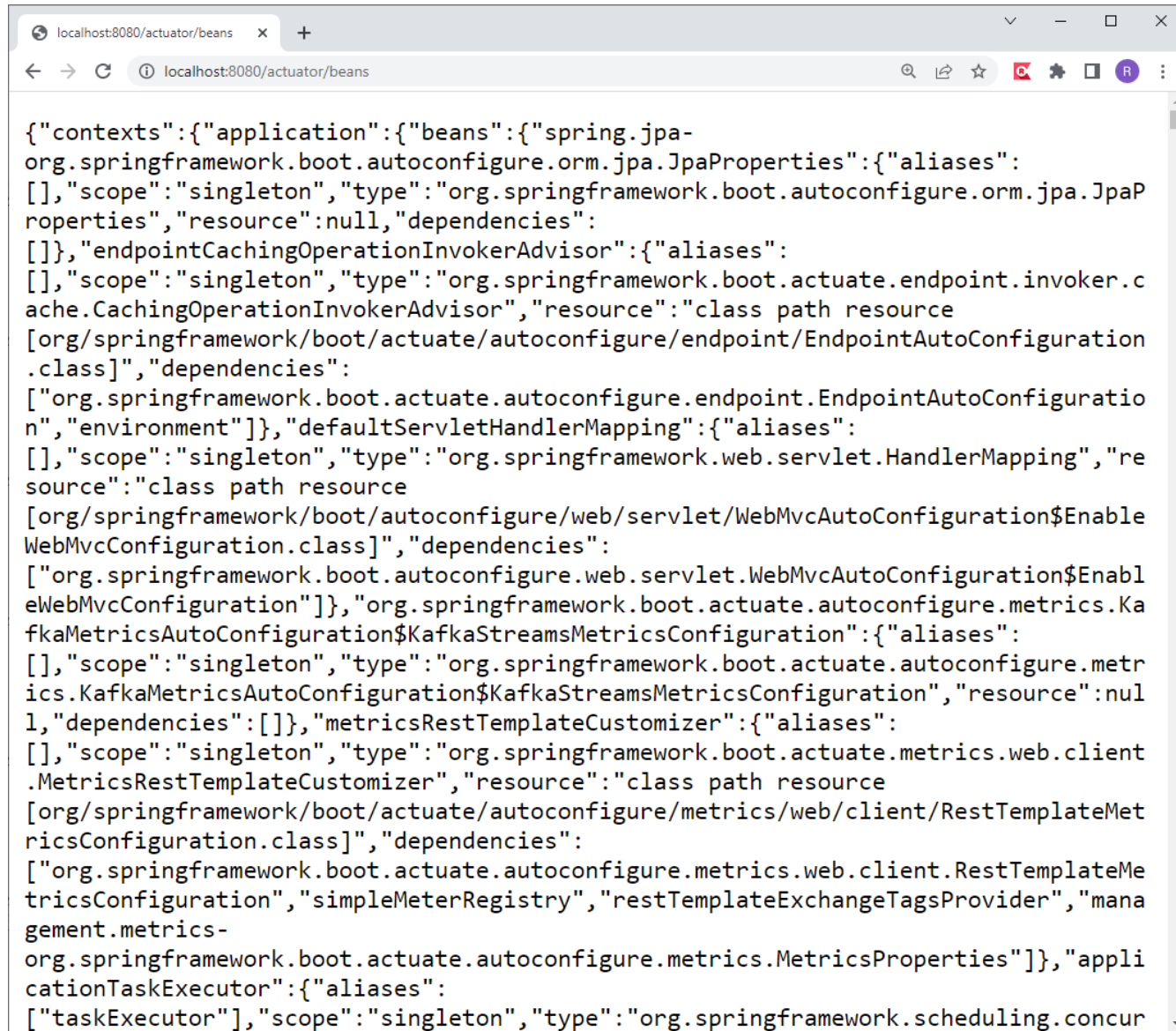
```
{ "mem": 31196, "mem.free": 9531, "processors": 8, "instance.uptime": 166841, "uptime": 169604, "systemload.average": -1.0, "heap.committed": 31196, "heap.init": 16384, "heap.used": 21664, "heap": 253440, "threads.peak": 17, "threads.daemon": 14, "threads": 16, "classes": 5184, "classes.loaded": 5184, "classes.unloaded": 0, "gc.copy.count": 62, "gc.copy.time": 148, "gc.markswEEPcompact.count": 2, "gc.markswEEPcompact.time": 52, "http.sessions.max": -1, "http.sessions.active": 0, "counter.status.200.info": 1, "gauge.response.info": 45.0 }
```

/actuator/env



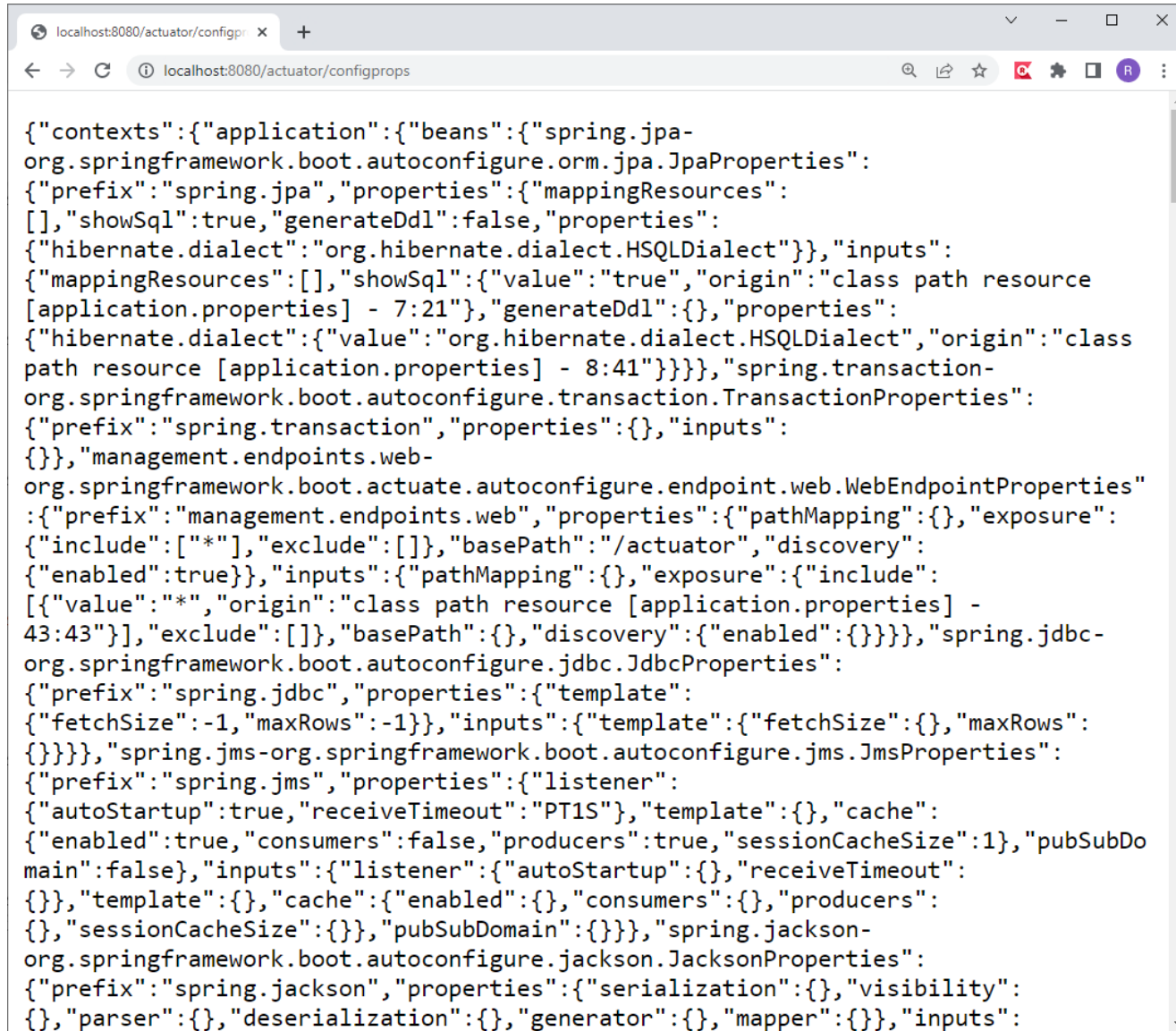
```
{
  "activeProfiles": [],
  "propertySources": [
    {
      "name": "server.ports",
      "properties": {
        "local.server.port": {
          "value": 8080
        }
      }
    },
    {
      "name": "servletContextInitParams",
      "properties": {}
    },
    {
      "name": "systemProperties",
      "properties": {
        "sun.desktop": {
          "value": "windows"
        },
        "awt.toolkit": {
          "value": "sun.awt.windows.WToolkit"
        },
        "java.specification.version": {
          "value": "11"
        },
        "sun.cpu.isalist": {
          "value": "amd64"
        },
        "sun.jnu.encoding": {
          "value": "Cp1252"
        },
        "java.class.path": {
          "value": "C:\\\\EnterpriseArchitecture\\\\labsolutions\\\\Lab13BankSolution\\\\target\\\\classes;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-starter\\\\2.6.5\\\\spring-boot-starter-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot\\\\2.6.5\\\\spring-boot-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-autoconfigure\\\\2.6.5\\\\spring-boot-autoconfigure-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\boot\\\\spring-boot-starter-logging\\\\2.6.5\\\\spring-boot-starter-logging-2.6.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\ch\\\\qos\\\\logback\\\\logback-classic\\\\1.2.11\\\\logback-classic-1.2.11.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\ch\\\\qos\\\\logback\\\\logback-core\\\\1.2.11\\\\logback-core-1.2.11.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\apache\\\\logging\\\\log4j\\\\log4j-to-slf4j\\\\2.17.2\\\\log4j-to-slf4j-2.17.2.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\apache\\\\logging\\\\log4j\\\\log4j-api\\\\2.17.2\\\\log4j-api-2.17.2.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\slf4j\\\\jul-to-slf4j\\\\1.7.36\\\\jul-to-slf4j-1.7.36.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\jakarta\\\\annotation\\\\jakarta.annotation-api\\\\1.3.5\\\\jakarta.annotation-api-1.3.5.jar;C:\\\\Users\\\\vedam\\\\.m2\\\\repository\\\\org\\\\springframework\\\\spring-core\\\\5.3.17\\\\spring-core-"
        }
      }
    }
  ]
}
```

/actuator/beans



```
{
  "contexts": {
    "application": {
      "beans": {
        "spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.autoconfigure.orm.jpa.JpaProperties",
          "resource": null,
          "dependencies": []
        },
        "endpointCachingOperationInvokerAdvisor": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.endpoint.invoker.cache.CachingOperationInvokerAdvisor",
          "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/endpoint/EndpointAutoConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.actuate.autoconfigure.endpoint.EndpointAutoConfiguration",
            "environment"
          ]
        },
        "defaultServletHandlerMapping": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.web.servlet.HandlerMapping",
          "resource": "class path resource [org/springframework/boot/autoconfigure/web/servlet/WebMvcAutoConfiguration$EnableWebMvcConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.autoconfigure.web.servlet.WebMvcAutoConfiguration$EnableWebMvcConfiguration"
          ]
        },
        "org.springframework.boot.actuate.autoconfigure.metrics.KafkaMetricsAutoConfiguration$KafkaStreamsMetricsConfiguration": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.autoconfigure.metrics.KafkaMetricsAutoConfiguration$KafkaStreamsMetricsConfiguration",
          "resource": null,
          "dependencies": []
        },
        "metricsRestTemplateCustomizer": {
          "aliases": [],
          "scope": "singleton",
          "type": "org.springframework.boot.actuate.metrics.web.client.MetricsRestTemplateCustomizer",
          "resource": "class path resource [org/springframework/boot/actuate/autoconfigure/metrics/web/client/RestTemplateMetricsConfiguration.class]",
          "dependencies": [
            "org.springframework.boot.actuate.autoconfigure.metrics.web.client.RestTemplateMetricsConfiguration",
            "simpleMeterRegistry",
            "restTemplateExchangeTagsProvider",
            "management.metrics-org.springframework.boot.actuate.autoconfigure.metrics.MetricsProperties"
          ]
        },
        "applicationTaskExecutor": {
          "aliases": [
            "taskExecutor"
          ],
          "scope": "singleton",
          "type": "org.springframework.scheduling.concurrent"
        }
      }
    }
  }
}
```

/actuator/configprops



```
{
  "contexts": {
    "application": {
      "beans": {
        "spring.jpa-
org.springframework.boot.autoconfigure.orm.jpa.JpaProperties": {
          "prefix": "spring.jpa",
          "properties": {
            "mappingResources": [
            ],
            "showSql": true,
            "generateDdl": false,
            "properties": {
              "hibernate.dialect": "org.hibernate.dialect.HSQLDialect"
            }
          },
          "inputs": {
            "mappingResources": [
            ],
            "showSql": {
              "value": "true",
              "origin": "class path resource [application.properties] - 7:21"
            },
            "generateDdl": {
            },
            "properties": {
              "hibernate.dialect": {
                "value": "org.hibernate.dialect.HSQLDialect",
                "origin": "class path resource [application.properties] - 8:41"
              }
            }
          }
        },
        "spring.transaction-
org.springframework.boot.autoconfigure.transaction.TransactionProperties": {
          "prefix": "spring.transaction",
          "properties": {
          },
          "inputs": {
          }
        },
        "management.endpoints.web-
org.springframework.boot.actuate.autoconfigure.endpoint.web.WebEndpointProperties": {
          "prefix": "management.endpoints.web",
          "properties": {
            "pathMapping": {
            },
            "exposure": {
              "include": [
                "*"
              ],
              "exclude": [
            ],
              "basePath": "/actuator",
              "discovery": {
                "enabled": true
              }
            },
            "inputs": {
              "pathMapping": {
            },
              "exposure": {
                "include": [
                  {
                    "value": "*",
                    "origin": "class path resource [application.properties] - 43:43"
                  }
                ],
                "exclude": [
            ],
                "basePath": {
            },
                "discovery": {
                  "enabled": {
            }
                }
              }
            },
            "spring.jdbc-
org.springframework.boot.autoconfigure.jdbc.JdbcProperties": {
          "prefix": "spring.jdbc",
          "properties": {
            "template": {
              "fetchSize": -1,
              "maxRows": -1
            }
          },
          "inputs": {
            "template": {
              "fetchSize": {
            },
              "maxRows": {
            }
            }
          }
        },
        "spring.jms-org.springframework.boot.autoconfigure.jms.JmsProperties": {
          "prefix": "spring.jms",
          "properties": {
            "listener": {
              "autoStartup": true,
              "receiveTimeout": "PT1S",
              "template": {
            },
              "cache": {
                "enabled": true,
                "consumers": false,
                "producers": true,
                "sessionCacheSize": 1,
                "pubSubDomain": false
              },
              "inputs": {
                "listener": {
                  "autoStartup": {
            },
                  "receiveTimeout": {
            }
                },
                "template": {
            },
                "cache": {
                  "enabled": {
            },
                  "consumers": {
            },
                  "producers": {
            },
                  "sessionCacheSize": {
            },
                  "pubSubDomain": {
            }
                }
              }
            },
            "spring.jackson-
org.springframework.boot.autoconfigure.jackson.JacksonProperties": {
          "prefix": "spring.jackson",
          "properties": {
            "serialization": {
            },
            "visibility": {
            },
            "parser": {
            },
            "deserialization": {
            },
            "generator": {
            },
            "mapper": {
            },
            "inputs": {
            }
          }
        }
      }
    }
  }
}
```

/actuator/mappings

{\"contexts\":{\"application\":{\"mappings\":{\"dispatcherServlets\":{\"dispatcherServlet\": [{\"handler\":\"Actuator web endpoint 'caches-cache'\",\"predicate\":\"{GET [\"/actuator/caches/{cache}], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}\", \"details\":{\"handlerMethod\":{\"className\":\"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler\", \"name\":\"handle\", \"descriptor\":\"(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;\", \"requestMappingConditions\":{\"consumes\": [], \"headers\": [], \"methods\": [\"GET\"], \"params\": [], \"patterns\": [\"/actuator/caches/{cache}\"], \"produces\": [{\"mediaType\":\"application/vnd.spring-boot.actuator.v3+json\", \"negated\": false}, {\"mediaType\":\"application/vnd.spring-boot.actuator.v2+json\", \"negated\": false}, {\"mediaType\":\"application/json\", \"negated\": false}]}]}], {\"handler\":\"Actuator web endpoint 'metrics-requiredMetricName'\", \"predicate\":\"{GET [\"/actuator/metrics/{requiredMetricName}], produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}\", \"details\":{\"handlerMethod\":{\"className\":\"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler\", \"name\":\"handle\", \"descriptor\":\"(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;\", \"requestMappingConditions\":{\"consumes\": [], \"headers\": [], \"methods\": [\"GET\"], \"params\": [], \"patterns\": [\"/actuator/metrics/{requiredMetricName}\"], \"produces\": [{\"mediaType\":\"application/vnd.spring-boot.actuator.v3+json\", \"negated\": false}, {\"mediaType\":\"application/vnd.spring-boot.actuator.v2+json\", \"negated\": false}, {\"mediaType\":\"application/json\", \"negated\": false}]}]}], {\"handler\":\"Actuator web endpoint 'configprops'\", \"predicate\":\"{GET [\"/actuator/configprops\", produces [application/vnd.spring-boot.actuator.v3+json || application/vnd.spring-boot.actuator.v2+json || application/json]}\", \"details\":{\"handlerMethod\":{\"className\":\"org.springframework.boot.actuate.endpoint.web.servlet.AbstractWebMvcEndpointHandlerMapping.OperationHandler\", \"name\":\"handle\", \"descriptor\":\"(Ljavax/servlet/http/HttpServletRequest;Ljava/util/Map;)Ljava/lang/Object;\", \"requestMappingConditions\":{\"consumes\": [], \"headers\": [], \"methods\": [\"GET\"], \"params\": [], \"patterns\": [\"/actuator/configprops\"], \"produces\": [{\"mediaType\":\"application/vnd.spring-boot.actuator.v3+json\", \"negated\": false}, {\"mediaType\":\"application/vnd.spring-boot.actuator.v2+json\", \"negated\": false}, {\"mediaType\":\"application/json\", \"negated\": false}]}]}]}]}]}\"}

/actuator/scheduledtasks



```
{"cron":[{"runnable":{"target":"bank.service.AccountService.printBankStatements","expression":"*/20 * * * * *"}}, {"fixedDelay":[],"fixedRate":[],"custom":[]}]}
```


Available actuators

GET	/autoconfig	Provides an auto-configuration report describing what auto-configuration conditions passed and failed.
GET	/configprops	Describes how beans have been injected with configuration properties (including default values).
GET	/beans	Describes all beans in the application context and their relationship to each other.
GET	/dump	Retrieves a snapshot dump of thread activity.
GET	/env	Retrieves all environment properties.

Available actuators

GET	/env/{name}	Retrieves a specific environment value by name.
GET	/health	Reports health metrics for the application, as provided by HealthIndicator implementations.
GET	/info	Retrieves custom information about the application, as provided by any properties prefixed with info.
GET	/mappings	Describes all URI paths and how they're mapped to controllers (including Actuator endpoints).
GET	/metrics	Reports various application metrics such as memory usage and HTTP request counters.

Available actuators

GET	/metrics/{name}	Reports an individual application metric by name.
POST	/shutdown	Shuts down the application; requires that endpoints.shutdown.enabled be set to true.
GET	/trace	Provides basic trace information (timestamp, headers, and so on) for HTTP requests.

shutdown

```
management.endpoint.shutdown.enabled=true
```

POST



http://localhost:8080/actuator/shutdown

Params

Authorization

Headers (9)

Body ●

Pre-request Script

Tests

Settings

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

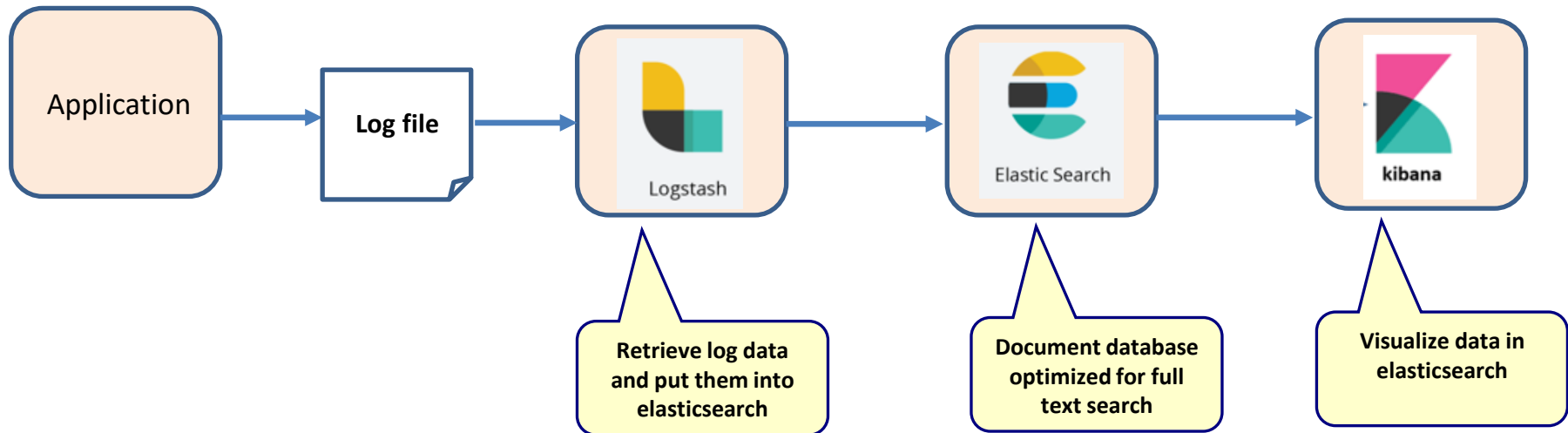
JSON



```
1 {  
2   "message": "Shutting down, bye..."  
3 }
```

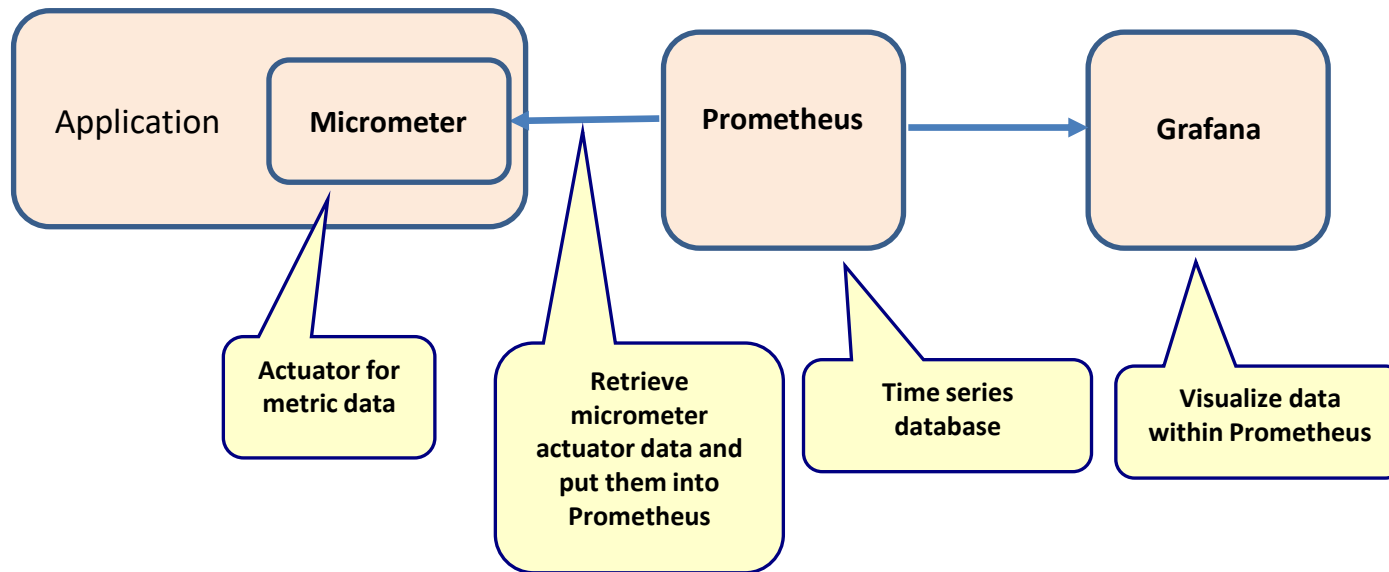
APPLICATION MONITORING

Approach 1: ELK stack



- Good for application specific log data

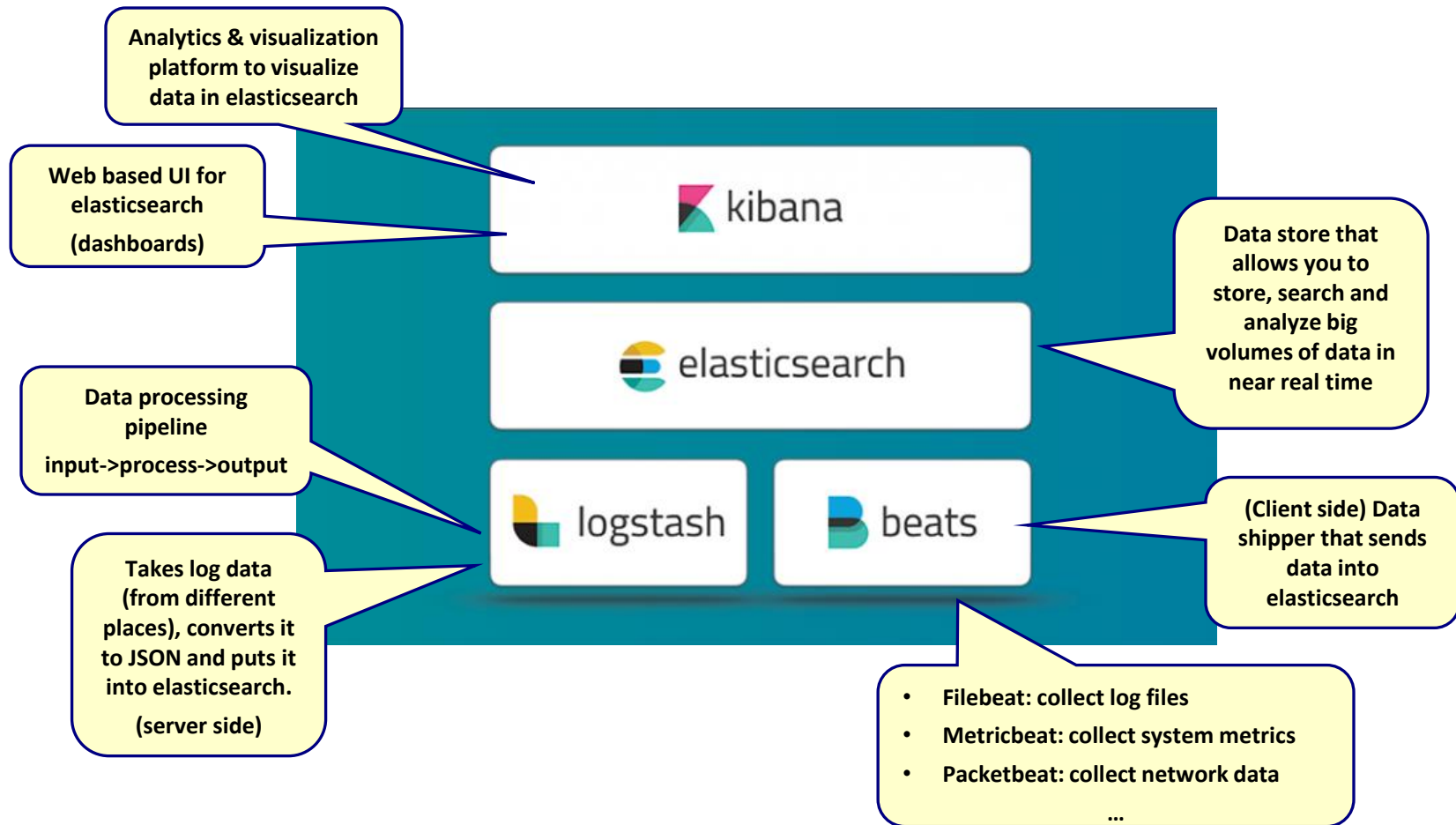
Approach 2: Prometheus/Grafana



- Good for metric data
 - Memory usage
 - CPU usage
 - JVM specific data

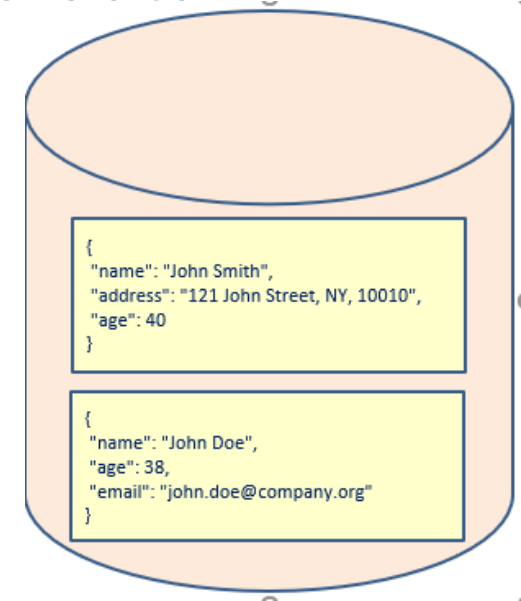
THE ELASTIC STACK

Elastic stack components

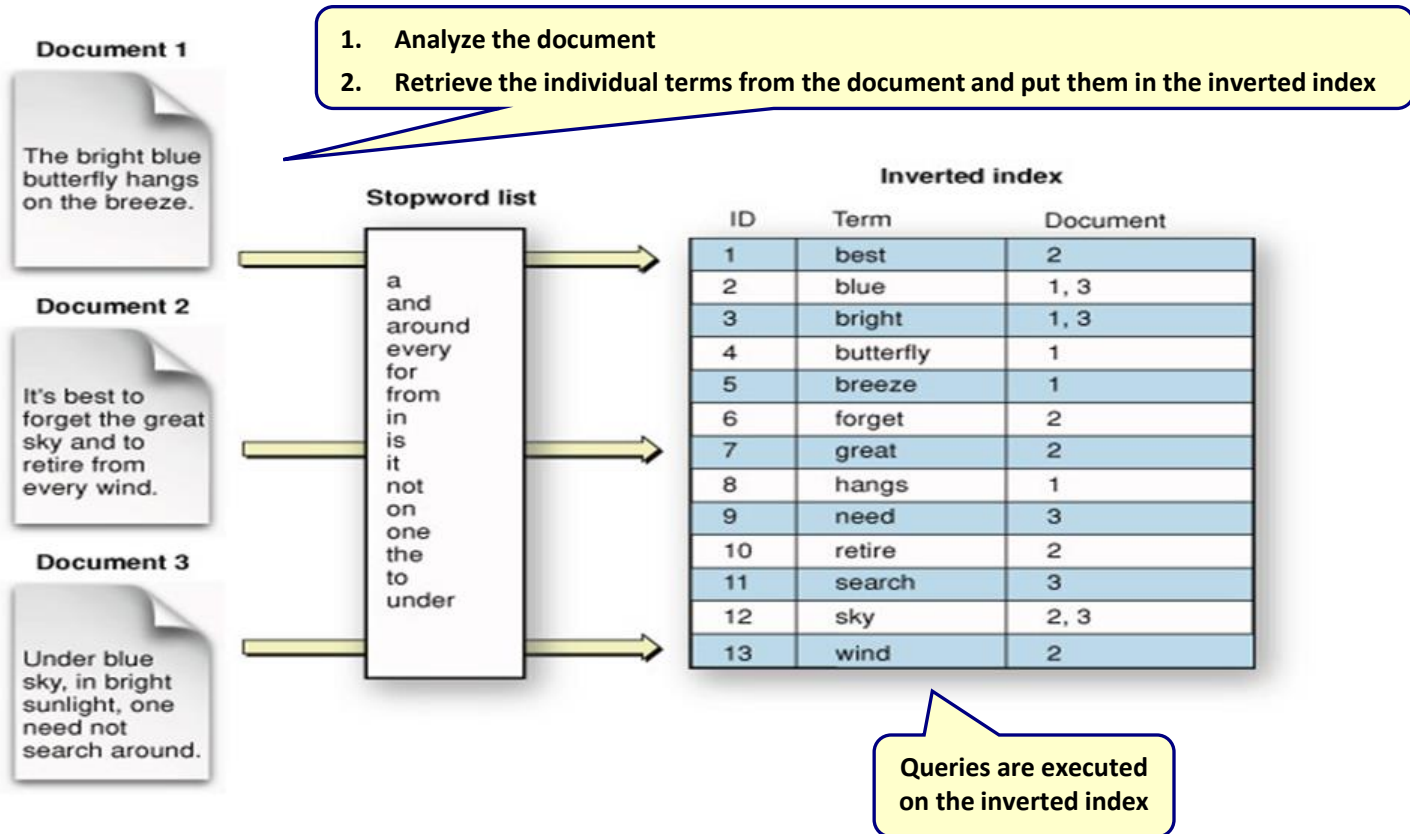


What is Elasticsearch?

- Database
 - Data is stored as documents
 - Data is structured in JSON format
- Full text search engine
- Analytics platform for structured data



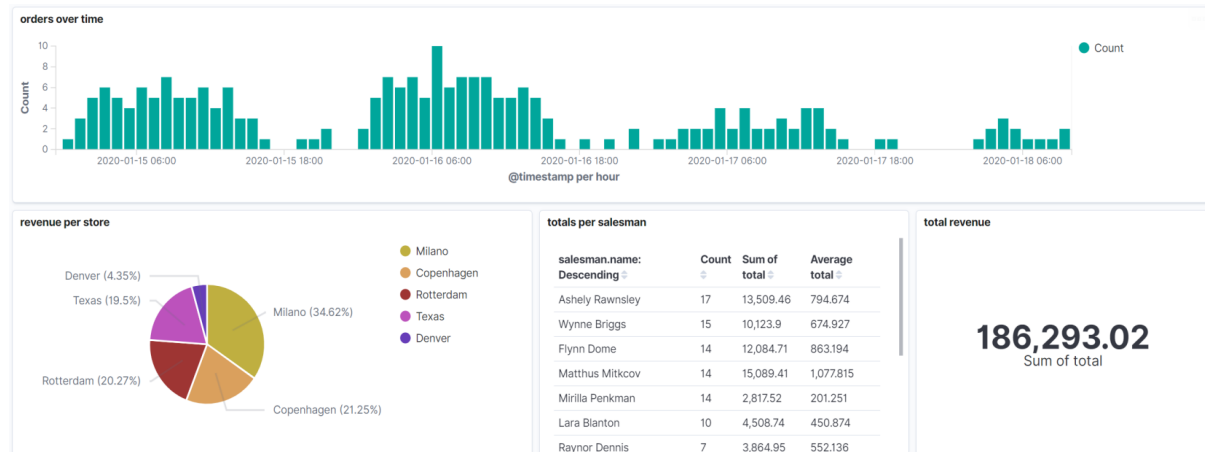
Inverted index



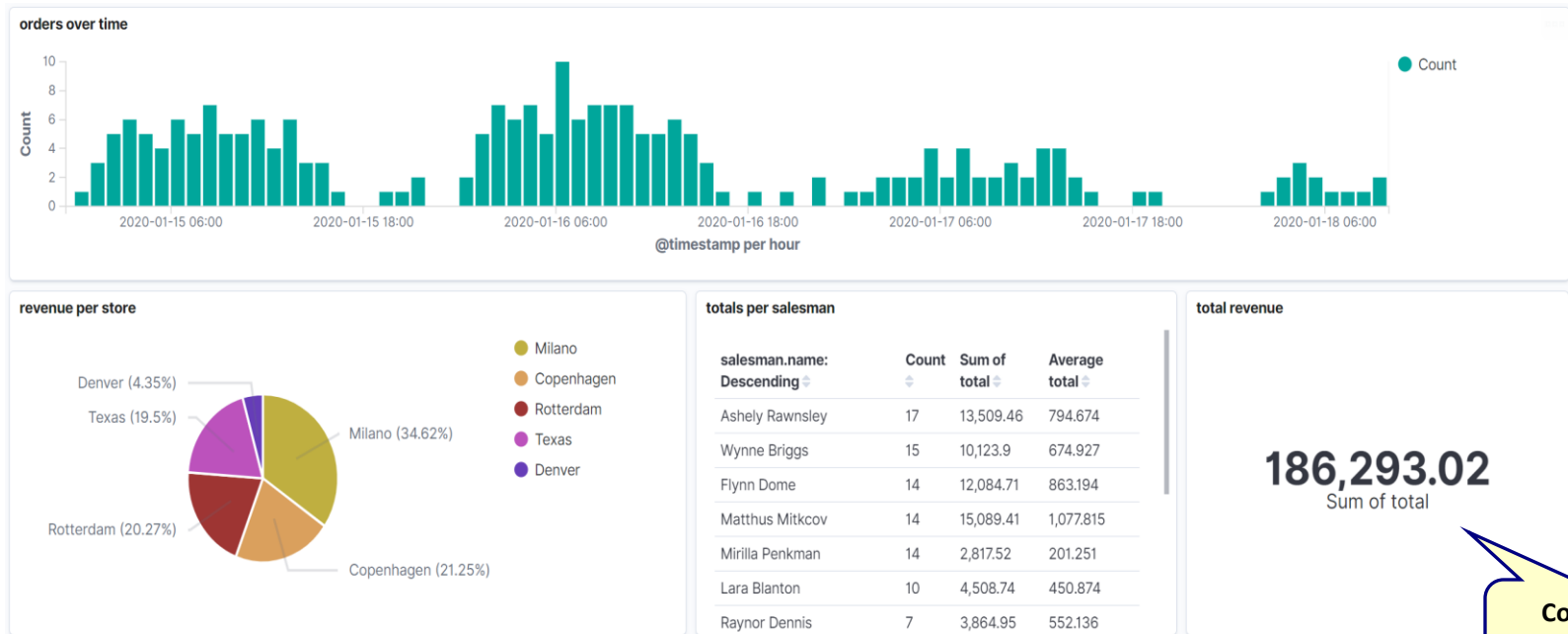
KIBANA

Kibana

- Web UI on top of elasticsearch
- Has its own Kibana query language (KQL)
- Objects (Queries, visualizations, dashboards, etc.) are saved in elasticsearch



Dashboard

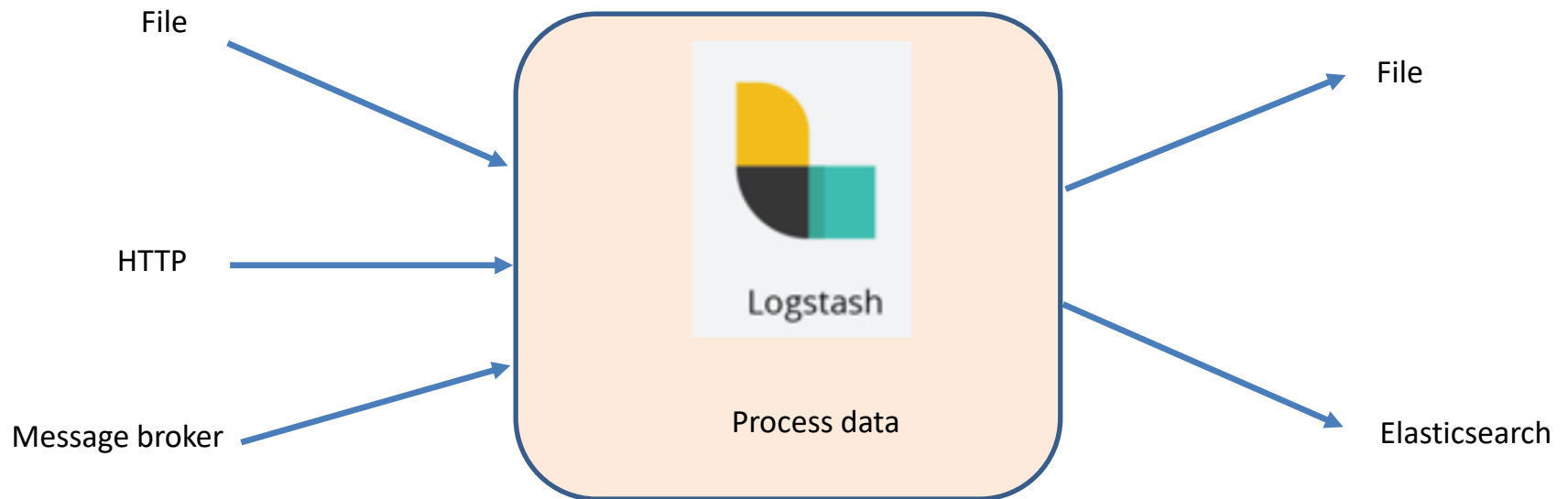


Contains
visualizations

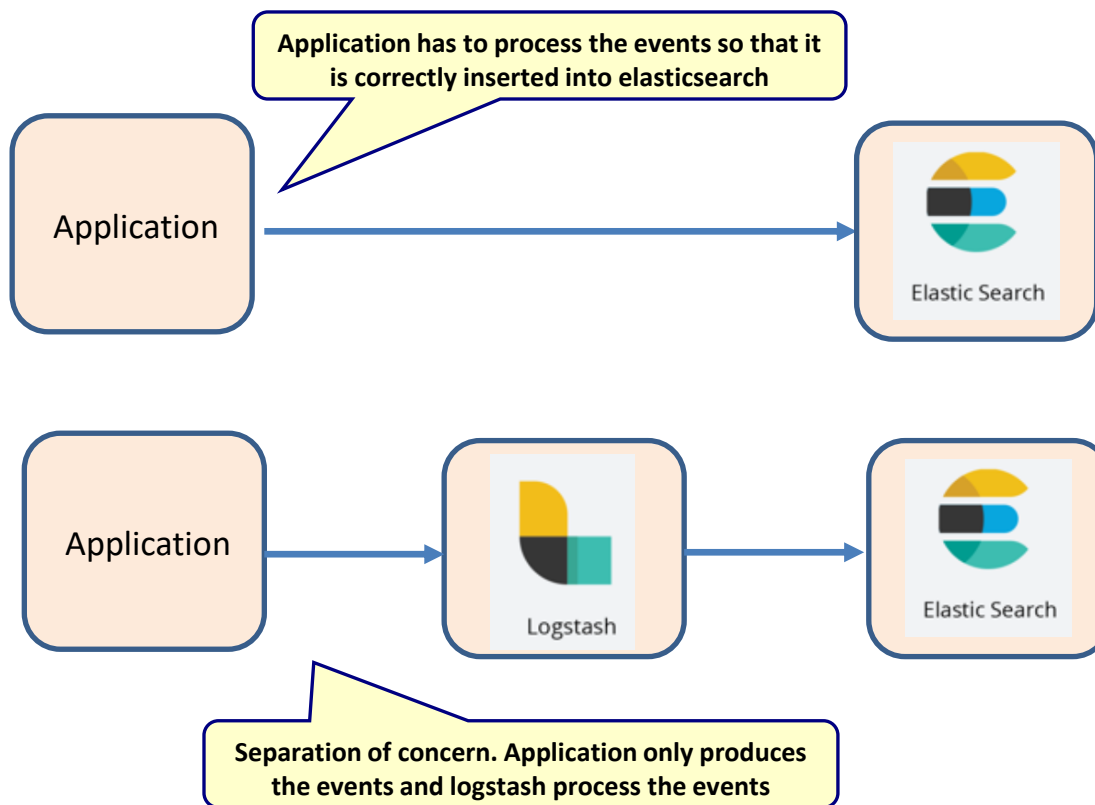
LOGSTASH

Logstash

- Event processing engine



Why logstash in ELK?



Logstash configuration

input.txt

Hello world

pipeline.conf

```
input {  
  file {  
    path => "C:/elasticsearchtraining/temp/input.txt"  
    start_position => "beginning"  
  }  
}  
  
output {  
  stdout {  
    codec => rubydebug  
  }  
  file {  
    path => "C:/elasticsearchtraining/temp/output.txt"  
  }  
}
```

Write the output to
the console

output.txt

```
{  
  "host": "DESKTOP-BVHRK6K",  
  "@version": "1",  
  "path": "C:/elasticsearchtraining/temp/input.txt",  
  "message": "Hello world\r",  
  "@timestamp": "2021-01-16T13:52:32.726Z"  
}
```

Anytime this file changes, read from
this file

Write the output to
the specified file

Logstash configuration

input.txt

Hi there

pipeline.conf

```
input {  
  file {  
    path => "C:/elasticsearchtraining/temp/input.txt"  
    start_position => "beginning"  
  }  
}  
  
filter {  
  mutate {  
    uppercase => ["message"]  
  }  
}  
  
output {  
  stdout {  
    codec => rubydebug  
  }  
  file {  
    path => "C:/elasticsearchtraining/temp/output.txt"  
  }  
}
```

output.txt

```
{  
  "path": "C:/elasticsearchtraining/temp/input.txt",  
  "message": "HI THERE\r",  
  "host": "DESKTOP-BVHRK6K",  
  "@version": "1",  
  "@timestamp": "2021-01-16T14:17:10.537Z"  
}
```

Logstash configuration

input.txt

```
get 2500 300
```

output.txt

```
{
  "bytes": "2500",
  "@timestamp": "2021-01-16T14:46:40.613Z",
  "path": "C:/elasticsearchtraining/temp/input.txt",
  "duration": "300",
  "method": "GET",
  "@version": "1",
  "message": "get 2500 300\r",
  "host": "DESKTOP-BVHRK6K"
}
```

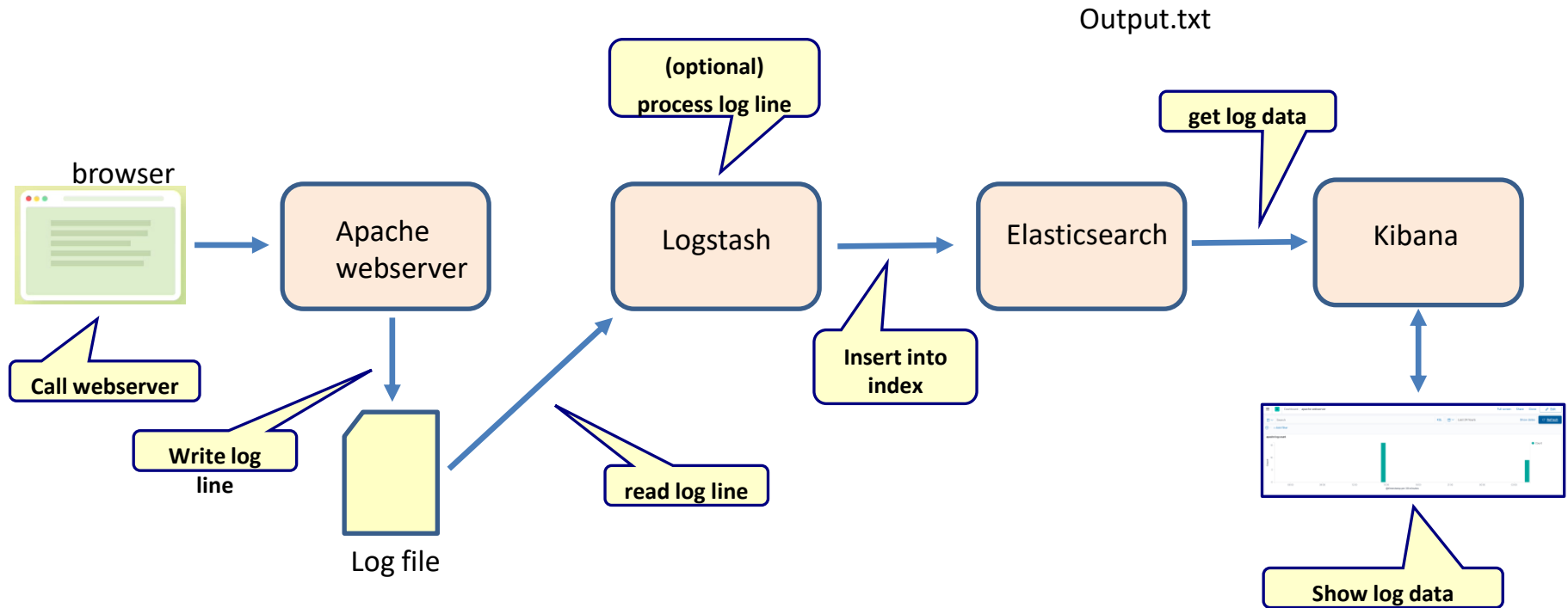
pipeline.conf

```
input {
  file {
    path => "C:/elasticsearchtraining/temp/input.txt"
    start_position => "beginning"
  }
}

filter {
  grok{
    match => {"message" => "%{WORD:method} %{NUMBER:bytes} %{NUMBER:duration}"}
  }
  mutate {
    uppercase => ["method"]
  }
}

output {
  stdout {
    codec => rubydebug
  }
  file {
    path => "C:/elasticsearchtraining/temp/output.txt"
  }
}
```

logstash example

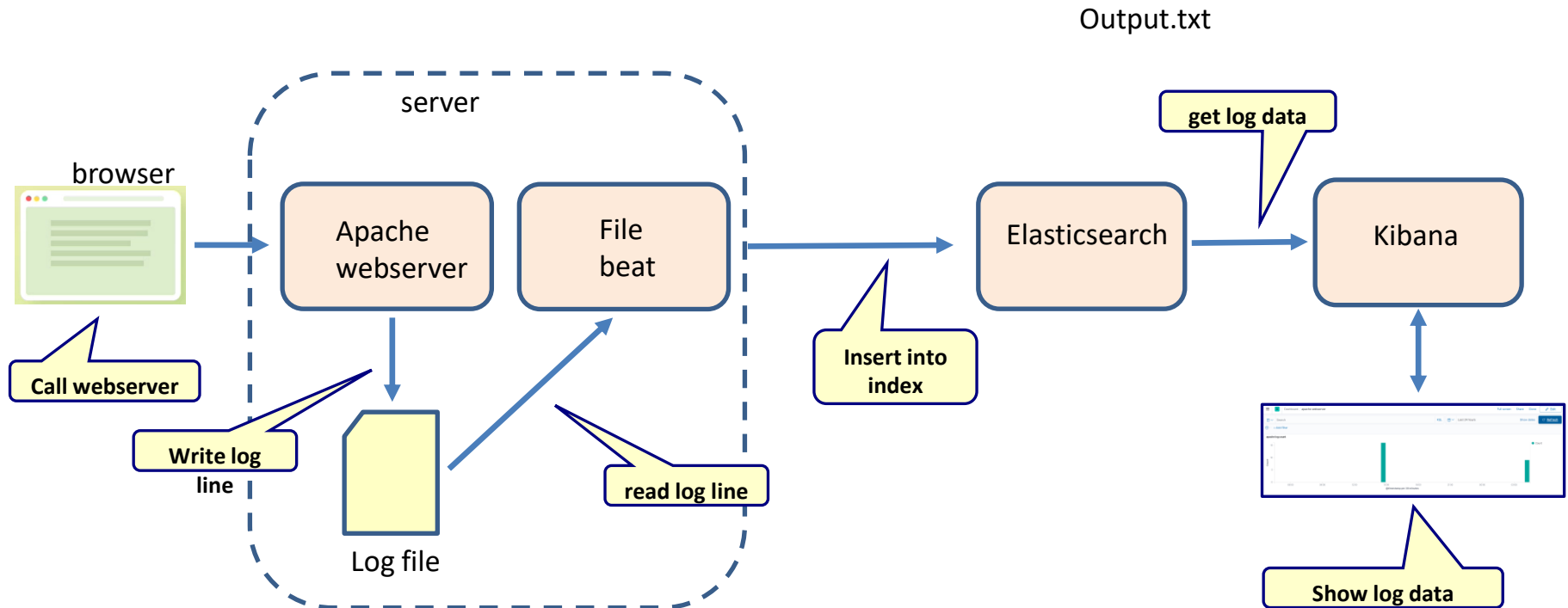


BEATS

Beats

- Data shippers that act as agents installed on the different servers in your infrastructure for collecting logs or metrics
 - log files (Filebeat)
 - network data (Packetbeat)
 - server metrics (Metricbeat)
- Once collected, the data is sent either directly into Elasticsearch or to Logstash for additional processing

filebeat example



Filebeat configuration

filebeat.yml

```
filebeat.inputs:
```

```
# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.
```

```
- type: log
```

```
# Change to true to enable this input configuration.
```

```
enabled: true
```

```
# Paths that should be crawled and fetched. Glob based paths.
```

```
paths:
```

```
- C:/elasticsearchtraining/Apache24/logs/access.log
```

read the access log file

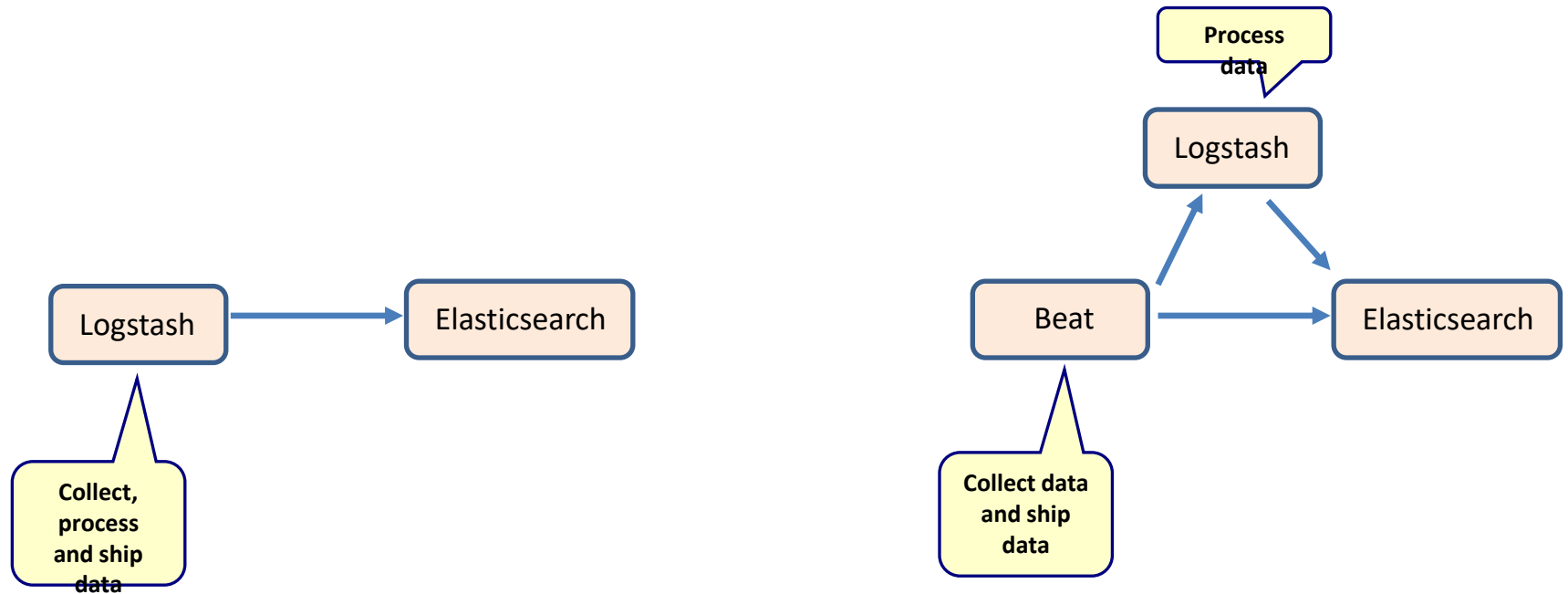
```
output.elasticsearch:
```

```
# Array of hosts to connect to.
```

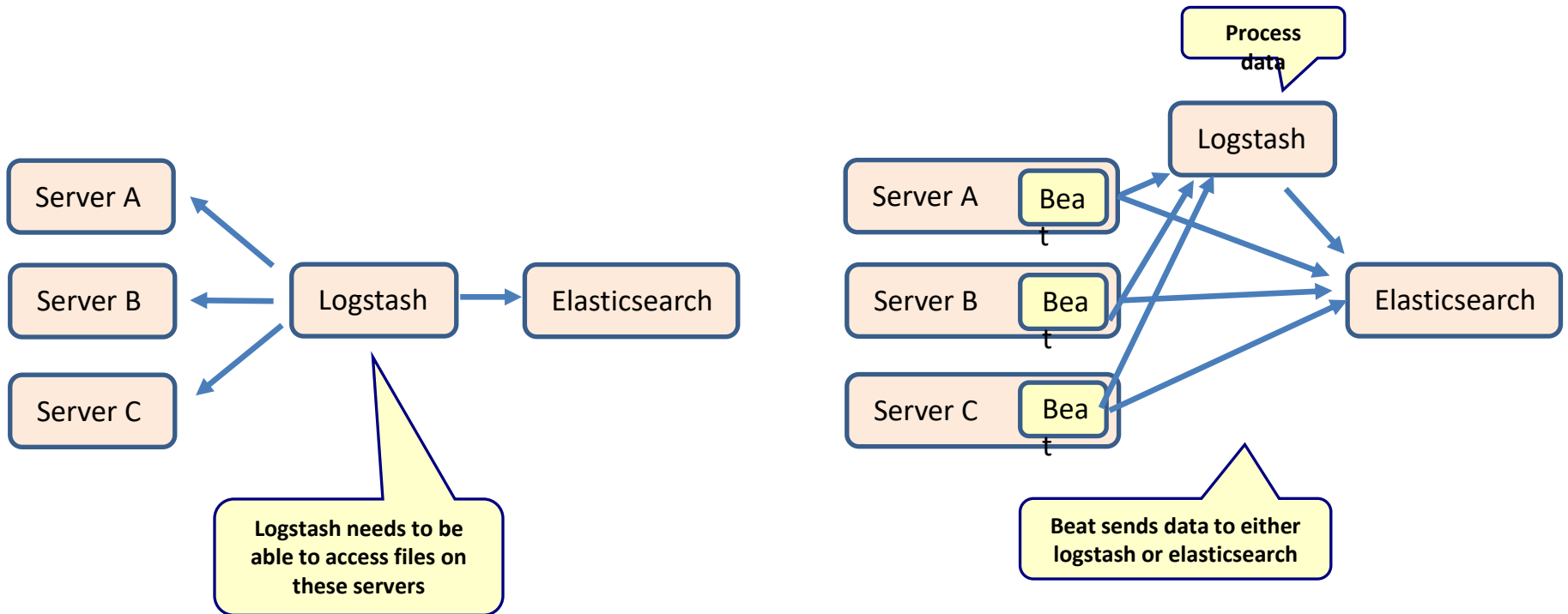
```
hosts: ["localhost:9200"]
```

Output to default index
'filebeat' in elasticsearch

Difference between beats and logstash



Difference between beats and logstash



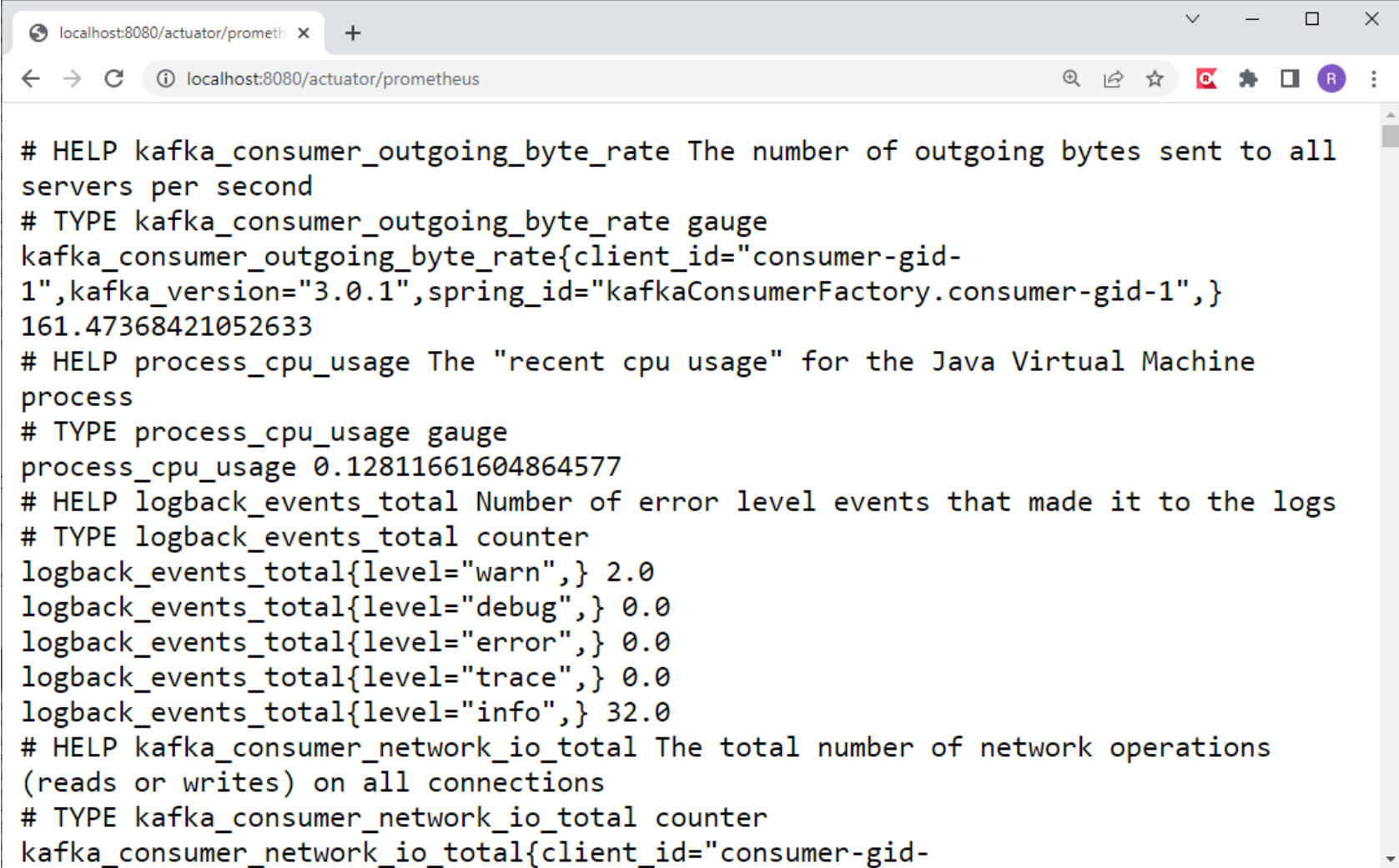
MONITOR ACTUATOR DATA

Micrometer

- Captures metric data and expose this data via an actuator endpoint

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

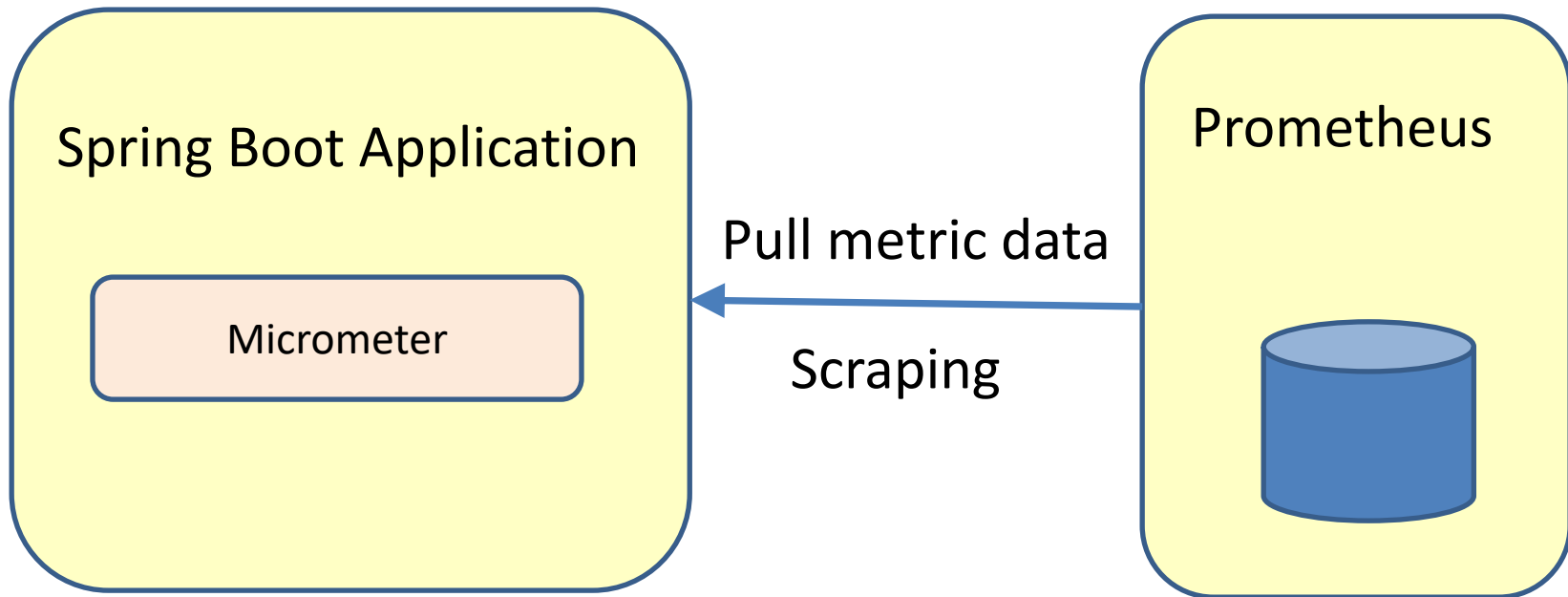
Actuator/prometheus



```
# HELP kafka_consumer_outgoing_byte_rate The number of outgoing bytes sent to all
servers per second
# TYPE kafka_consumer_outgoing_byte_rate gauge
kafka_consumer_outgoing_byte_rate{client_id="consumer-gid-
1",kafka_version="3.0.1",spring_id="kafkaConsumerFactory.consumer-gid-1",}
161.47368421052633
# HELP process_cpu_usage The "recent cpu usage" for the Java Virtual Machine
process
# TYPE process_cpu_usage gauge
process_cpu_usage 0.12811661604864577
# HELP logback_events_total Number of error level events that made it to the logs
# TYPE logback_events_total counter
logback_events_total{level="warn",} 2.0
logback_events_total{level="debug",} 0.0
logback_events_total{level="error",} 0.0
logback_events_total{level="trace",} 0.0
logback_events_total{level="info",} 32.0
# HELP kafka_consumer_network_io_total The total number of network operations
(reads or writes) on all connections
# TYPE kafka_consumer_network_io_total counter
kafka_consumer_network_io_total{client_id="consumer-gid-
```

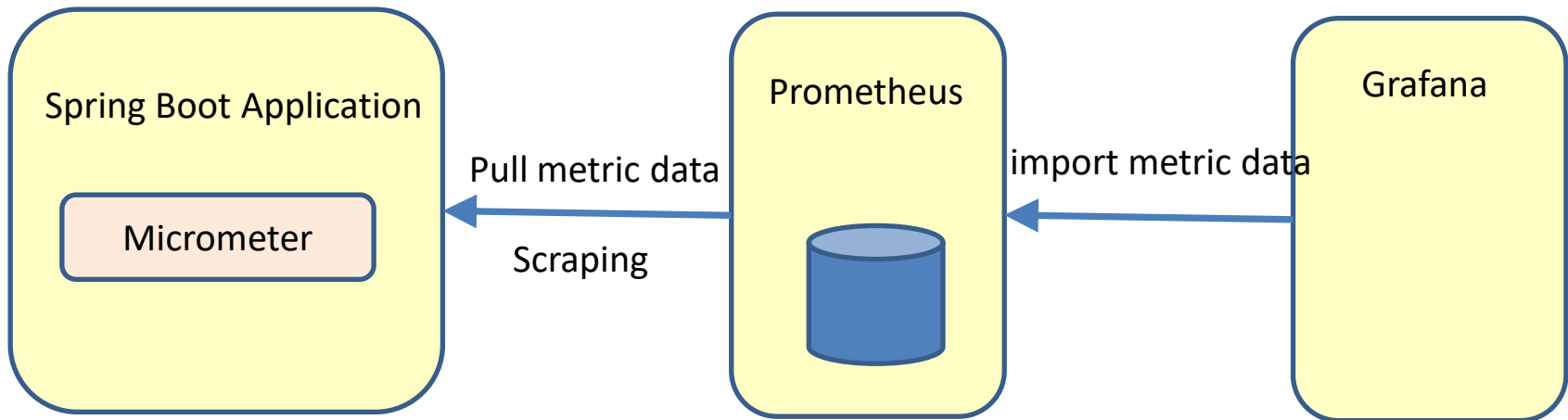
Prometheus

- Time series database
- Stores metric and performance data

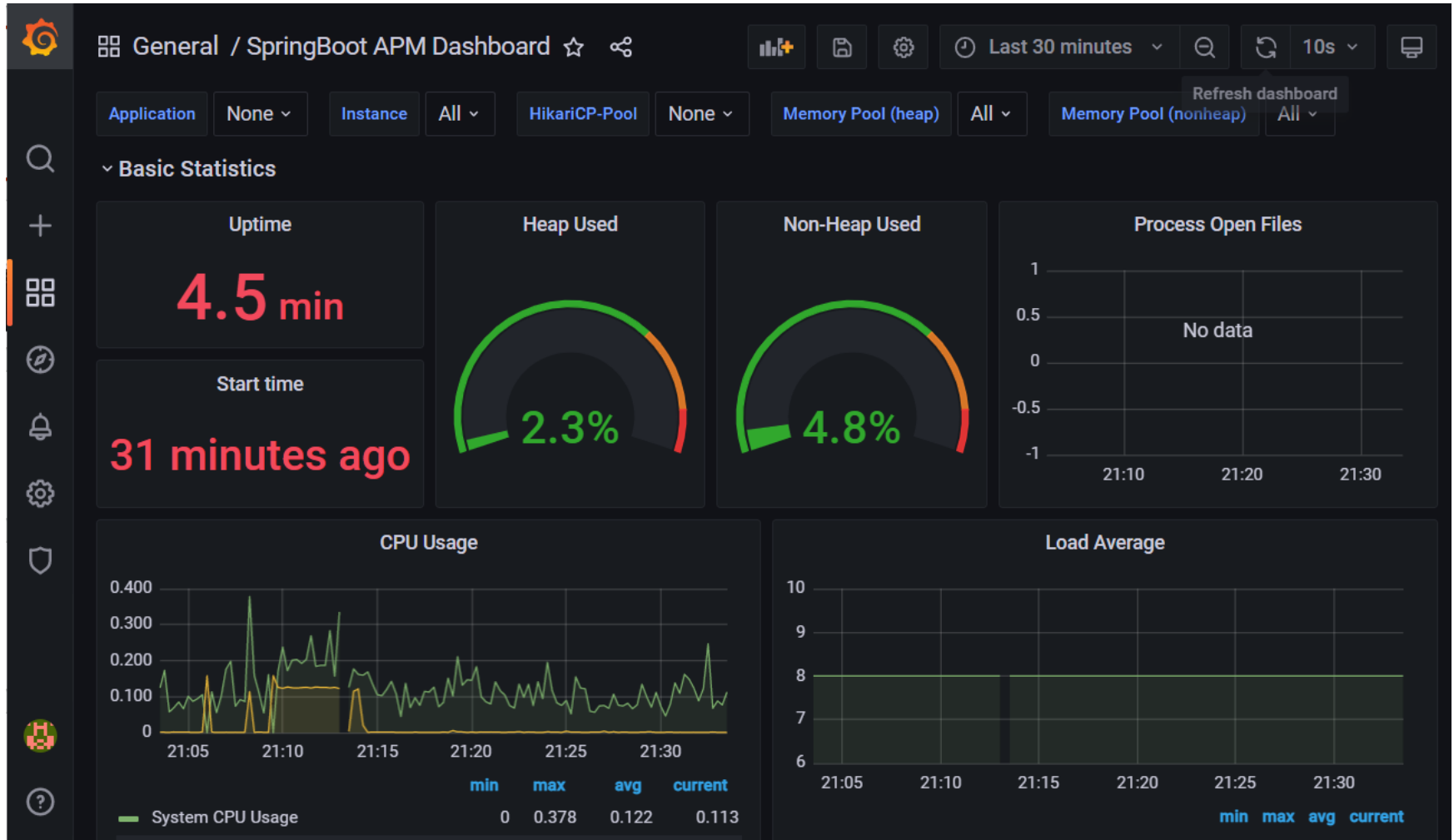


Grafana

- Dashboard to visualize metric data

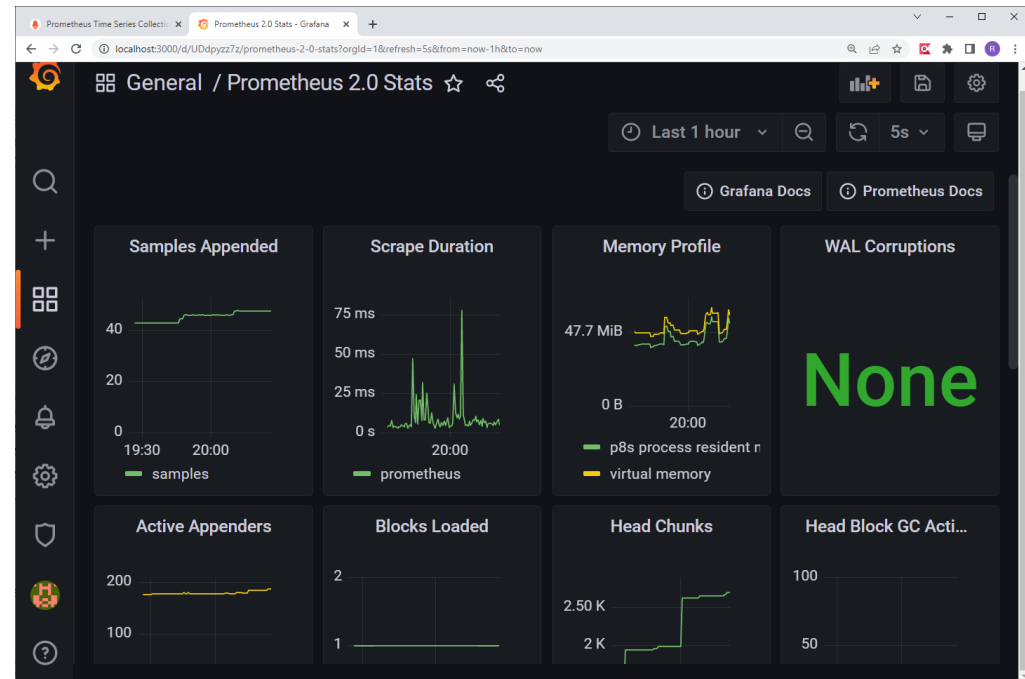


Grafana dashboard



Grafana

- Make your own dashboards
- Alerts
- Refresh interval
- Timespan



UNIT TESTING WITH JUNIT

What is unit testing?

- A unit test is a test that test one single class.
 - A test case test one single method
 - A test class test one single class
 - A test suite is a collection of test classes
- Unit tests make use of a testing framework
- A unit test
 1. Create an object
 2. Call a method
 3. Check if the result is correct

Example of unit testing

```
package count;

public class Counter {
    private int counterValue=0;

    public int increment(){
        return ++counterValue;
    }

    public int decrement(){
        return --counterValue;
    }

    public int getCounterValue() {
        return counterValue;
    }
}
```

Example of unit testing

```
public class CounterTest {  
    private Counter counter;
```

Initialization

```
@BeforeEach  
public void setUp() throws Exception {  
    counter = new Counter();  
}
```

Test method

```
@Test  
public void testIncrement() {  
    assertEquals("Counter.increment does not work correctly", 1, counter.increment());  
    assertEquals("Counter.increment does not work correctly", 2, counter.increment());  
}
```

Test method

```
@Test  
public void testDecrement() {  
    assertEquals("Counter.decrement does not work correctly", -1, counter.decrement());  
    assertEquals("Counter.decrement does not work correctly", -2, counter.decrement());  
}
```

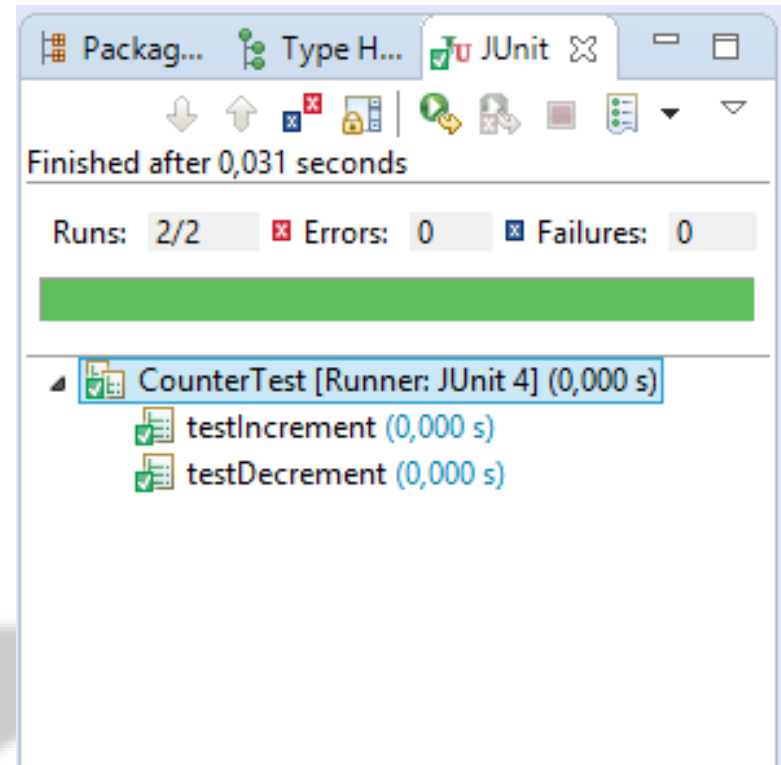
```
public class Counter {  
    private int counterValue=0;  
  
    public int increment() {  
        return ++counterValue;  
    }  
    public int decrement() {  
        return --counterValue;  
    }  
    public int getCounterValue() {  
        return counterValue;  
    }  
}
```

Running the test

```
package count;

public class Counter {
    private int counterValue=0;

    public int increment(){
        return ++counterValue;
    }
    public int decrement(){
        return --counterValue;
    }
    public int getCounterValue() {
        return counterValue;
    }
}
```



Running the test

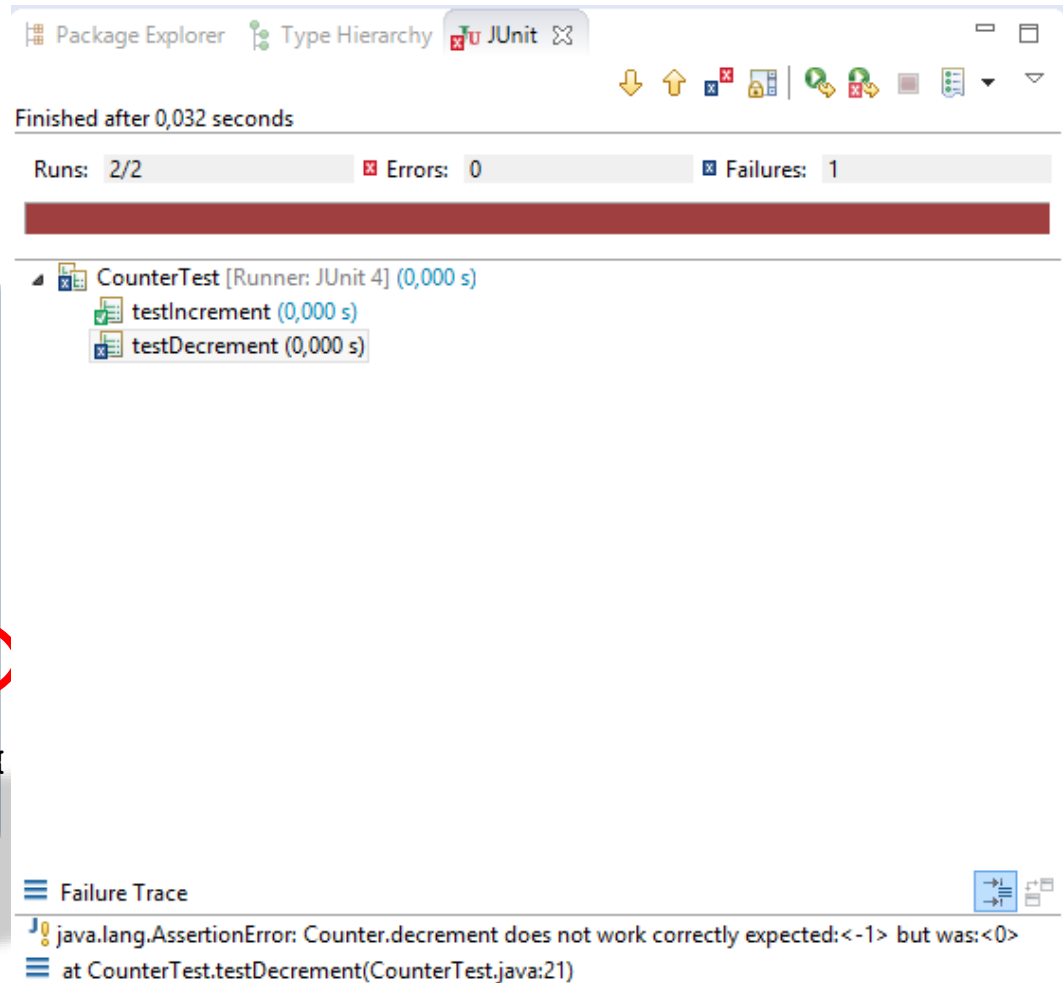
```
package count;
```

```
public class Counter {  
    private int counterValue=0;
```

```
    public int increment() {  
        return ++counterValue;  
    }
```

```
    public int decrement() {  
        return counterValue;  
    }
```

```
    public int getCounterValue() {  
        return counterValue;  
    }  
}
```



Package Explorer Type Hierarchy JUnit

Finished after 0,032 seconds

Runs: 2/2 Errors: 0 Failures: 1

CounterTest [Runner: JUnit 4] (0,000 s)

- testIncrement (0,000 s)
- testDecrement (0,000 s)

Failure Trace

java.lang.AssertionError: Counter.decrement does not work correctly expected:<-1> but was:<0>
at CounterTest.testDecrement(CounterTest.java:21)

JUnit test case

```
public class Calculator
{
    public double add( double number1, double number2 )
    {
        return number1 + number2;
    }
}
```

```
public class CalculatorTest
{
    @Test
    public void add()
    {
        Calculator calculator = new Calculator();
        double result = calculator.add( 10, 50 );
        assertEquals( 60, result, 0 );
    }
}
```

expected

Value to
assert

delta

Junit assert methods

- static void assertTrue(boolean *test*)
- static void assertTrue(String *message*, boolean *test*)
- static void assertFalse(boolean *test*)
- static void assertFalse(String *message*, boolean *test*)
- assertEquals(Object *expected*, Object *actual*)
- assertEquals(String *message*, *expected*, *actual*)
- assertSame (Object *expected*, Object *actual*)
- assertSame(String *message*, Object *expected*, Object *actual*)
- assertNotSame(Object *expected*, Object *actual*)
- assertNotSame(String *message*, Object *expected*, Object *actual*)
- assertNull(Object *object*)
- assertNull(String *message*, Object *object*)
- assertNotNull(Object *object*)
- assertNotNull(String *message*, Object *object*)
- fail()
- fail(String *message*)

@Before and @After

```
public class CounterTest {  
    private Counter counter;
```

This method is called before every testmethod

```
@BeforeEach
```

```
public void setUp() throws Exception {  
    counter = new Counter();  
}
```

This method is called after every testmethod

```
@AfterEach
```

```
public void tearDown() throws Exception {  
    counter=null;  
}
```

```
@Test
```

```
public void testConstructor() {  
    assertEquals("Counter constructor does not set counter to  
0", 0, counter.getCounterValue());  
}
```

```
}
```

@BeforeClass and @AfterClass

```
public class CounterTest {  
    private static Counter counter;  
  
    @BeforeClass  
    public static void setUpOnce() throws Exception {  
        counter = new Counter();  
    }  
  
    @AfterClass  
    public static void tearDownOnce() throws Exception {  
        counter=null;  
    }  
  
    @Test  
    public void testConstructor() {  
        assertEquals("Counter constructor does not set counter to  
                        0",0,counter.getCounterValue());  
    }  
}
```

This method is called once, before the testmethods are called

This method is called once, after the testmethods are called

Timeout tests

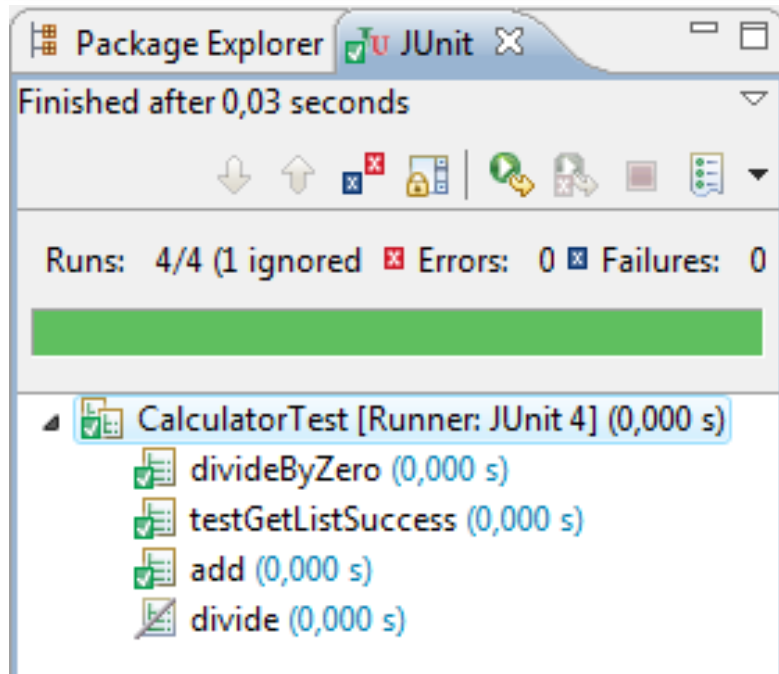
```
@Test(timeout=2000)  
public void longOperation() {  
  
}
```

Fail if the test method takes longer than 2000 milliseconds

Skip a test

```
@Test
@Ignore
public void divide(){
    assertEquals( 5, calculator.divide( 10, 2 ), 0 );
}
```

Skip this test



Test suite



```
@RunWith(value=Suite.class)
@SuiteClasses(value={CalculatorTest.class, ParameterizedTest.class})
public class CalculatorTestSuite {

}
```



Suite of 2 test classes

- You can also have a suite of suites
- Organize your tests

JUnit example: Calculator

```
public class Calculator {  
    private double value;  
  
    public Calculator() {  
        value = 0.0;  
    }  
    public void add(double number) {  
        value = value + number;  
    }  
    public void subtract (double number) {  
        value = value - number;  
    }  
    public void multiply(double number) {  
        value = value * number;  
    }  
    public void divide (double number) throws DivideByZeroException{  
        if (number == 0){  
            throw new DivideByZeroException();  
        }  
        value = value / number;  
    }  
    public double getValue() {  
        return value;  
    }  
}
```

JUnit example: CalculatorTest

```
import calculation.Calculator;

public class CalculatorTest {

    private Calculator calculator;

    @BeforeEach
    public void setup(){
        calculator = new Calculator();
    }

    @Test
    public void testInitialization() {
        assertEquals(0.0, calculator.getValue(), 0.0000001);
    }

    @Test
    public void testAddZero() {
        calculator.add(0.0);
        assertEquals(0.0, calculator.getValue(), 0.0000001);
    }
}
```

JUnit example: CalculatorTest

```
@Test
public void testAddPositive() {
    calculator.add(23.255);
    assertEquals(23.255, calculator.getValue(), 0.0000001);
}

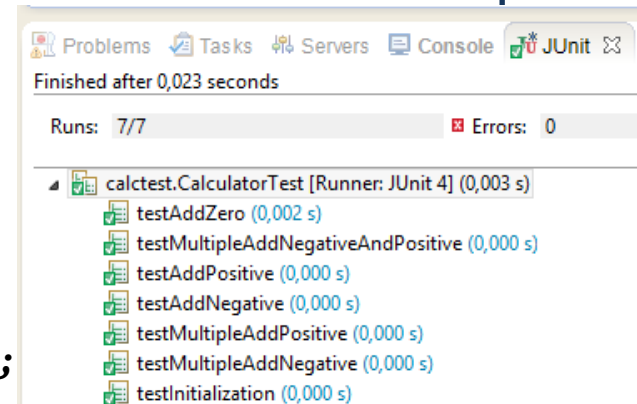
@Test
public void testAddNegative() {
    calculator.add(-23.255);
    assertEquals(-23.255, calculator.getValue(), 0.0000001);
}

@Test
public void testMultipleAddPositive() {
    calculator.add(23.255);
    calculator.add(10.255);
    assertEquals(33.510, calculator.getValue(), 0.0000001);
}

@Test
public void testMultipleAddNegative() {
    calculator.add(-23.255);
    calculator.add(-10.255);
    assertEquals(-33.510, calculator.getValue(), 0.0000001);
}

@Test
public void testMultipleAddNegativeAndPositive() {
    calculator.add(-23.255);
    calculator.add(10.250);
    assertEquals(-13.005, calculator.getValue(), 0.0000001);
}
```

Only test methods for add()



HAMCREST MATCHERS

Traditional asserts

- Parameter order is counter-intuitive
- Assert statements don't read well

`assertEquals(expected, actual)`

```
import static org.junit.Assert.*;

@Test
public void AssertEqualToRed(){
    String color = "red";
    assertEquals("red", color);
}
```

assertThat with hamcrest matchers

```
import static org.junit.Assert.*;
import static org.hamcrest.CoreMatchers.*;
import org.junit.jupiter.api.Before;
import org.junit.jupiter.api.Test;
```

Static import of matchers

```
public class CalculatorHamcrestTest{
    Calculator calculator=null;
```

```
    @BeforeEach
```

```
    public void createAcalculator(){
        calculator = new Calculator();
    }
```

```
    @Test
```

```
    public void add(){
        assertThat( calculator.add( 10, 50), equalTo (60.0));
    }
```

matcher

assertThat

```
    @Test
```

```
    public void divide(){
        assertThat(calculator.divide( 10, 2 ), equalTo (5.0));
    }
```

actual

expected

assert vs assertThat

```
@Test
public void AssertEqualToRed(){
    String color = "red";
    assertEquals("red", color);
}
```

assert

```
@Test
public void hamcrestAssertEqualToRed(){
    String color = "red";
    assertThat(color, equalTo("red"));
}
```

assertThat

assertThat equality tests

```
String color = "red";  
assertThat(color, is("red"));
```

assertThat ... is

```
String color = "red";  
assertThat(color, equalTo("red"));
```

assertThat ... equalTo

```
String color = "red";  
assertThat(color, not("blue"));
```

assertThat ... not

```
String color = "red";  
assertThat(color, isOneOf("blue", "red"));
```

assertThat ... isOneOf

```
List myList = new ArrayList();  
assertThat(myList, is(Collection.class));
```

assertThat ... is a class

assertThat testing for null values

```
String color = "red";  
assertThat(color, is(notNullValue()));  
assertNotNull(color);
```

notNullValue

```
String color = null;  
assertThat(color, is(nullValue()));  
assertNull(color);
```

nullValue

assertThat testing with collections

```
List<String> colors = new ArrayList<String>();  
colors.add("red");  
colors.add("green");  
colors.add("blue");  
assertThat(colors, hasItem("blue"));
```

hasItem

```
assertThat(colors, hasItems("red", "blue"));
```

hasItems

```
String[] colors = new String[] {"red", "green", "blue"};  
assertThat(colors, hasItemInArray("blue"));
```

hasItemInArray

```
assertThat("red", isIn(colors));
```

isIn

```
List<Integer> ages = new ArrayList<Integer>();  
ages.add(20);  
ages.add(30);  
ages.add(40);  
assertThat(ages, not(hasItem(lessThan(18))));
```

Combined matchers

Hamcrest matchers

- **Core**
 - `anything` - always matches, useful if you don't care what the object under test is
 - `describedAs` - decorator to adding custom failure description
 - `is` - decorator to improve readability
- **Logical**
 - `allOf` - matches if all matchers match, short circuits (like Java `&&`)
 - `anyOf` - matches if any matchers match, short circuits (like Java `||`)
 - `not` - matches if the wrapped matcher doesn't match and vice versa
- **Object**
 - `equalTo` - test object equality using `Object.equals`
 - `hasToString` - test `Object.toString`
 - `instanceOf`, `isCompatibleType` - test type
 - `notNullValue`, `nullValue` - test for null
 - `sameInstance` - test object identity
- **Beans**
 - `hasProperty` - test JavaBeans properties
- **Collections**
 - `array` - test an array's elements against an array of matchers
 - `hasEntry`, `hasKey`, `hasValue` - test a map contains an entry, key or value
 - `hasItem`, `hasItems` - test a collection contains elements
 - `hasItemInArray` - test an array contains an element
- **Number**
 - `closeTo` - test floating point values are close to a given value
 - `greaterThan`, `greaterThanOrEqualTo`, `lessThan`, `lessThanOrEqualTo` - test ordering
- **Text**
 - `equalToIgnoringCase` - test string equality ignoring case
 - `equalToIgnoringWhiteSpace` - test string equality ignoring differences in runs of whitespace
 - `containsString`, `endsWith`, `startsWith` - test string matching