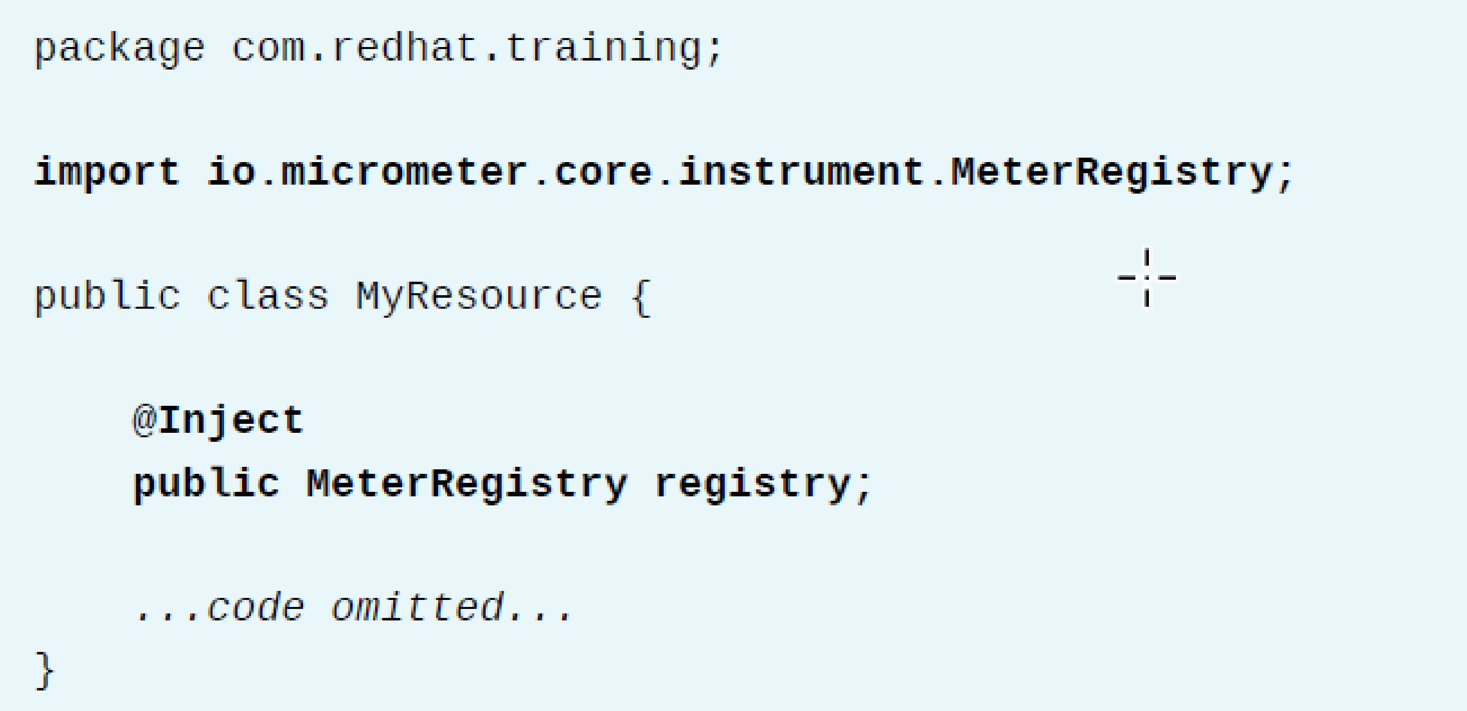
Quarkus Metrics

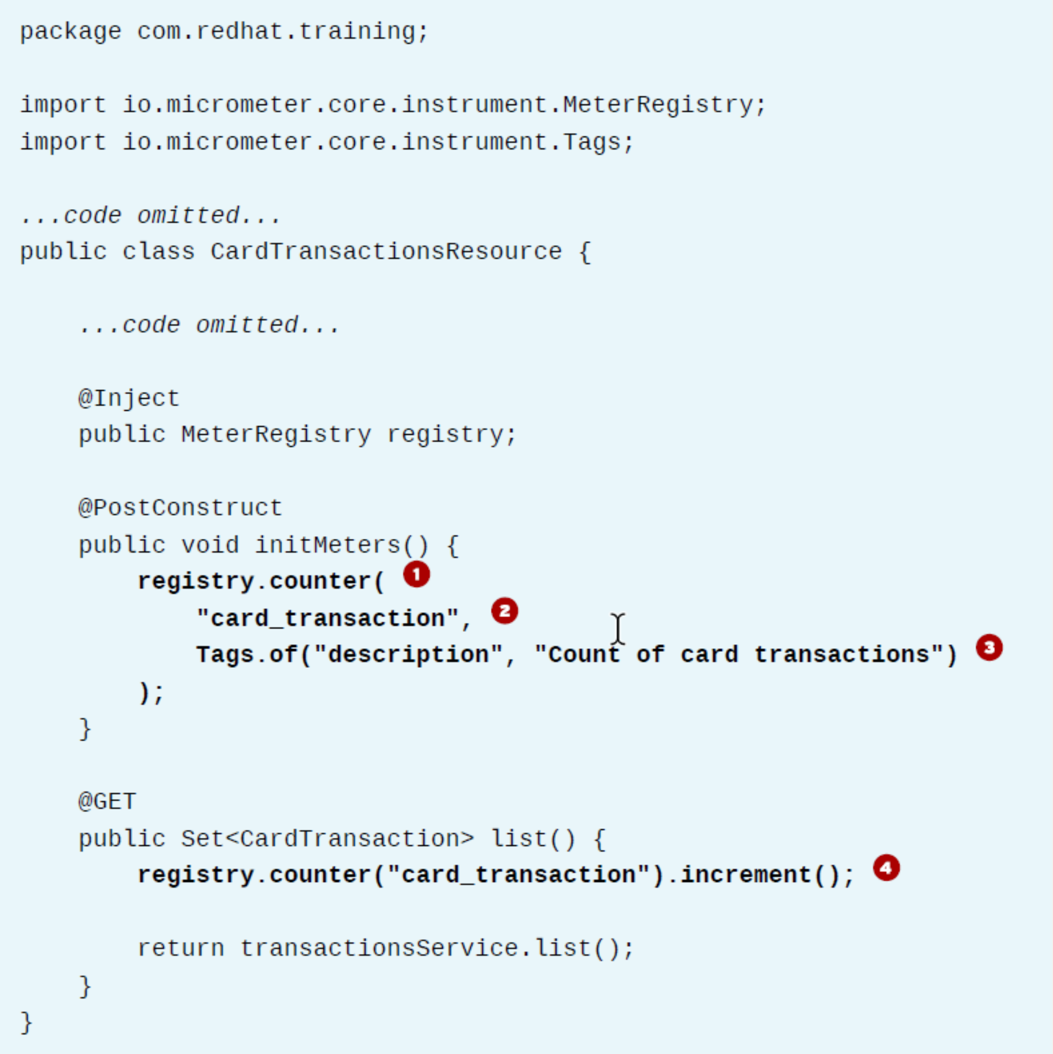
Quarkus-micrometer



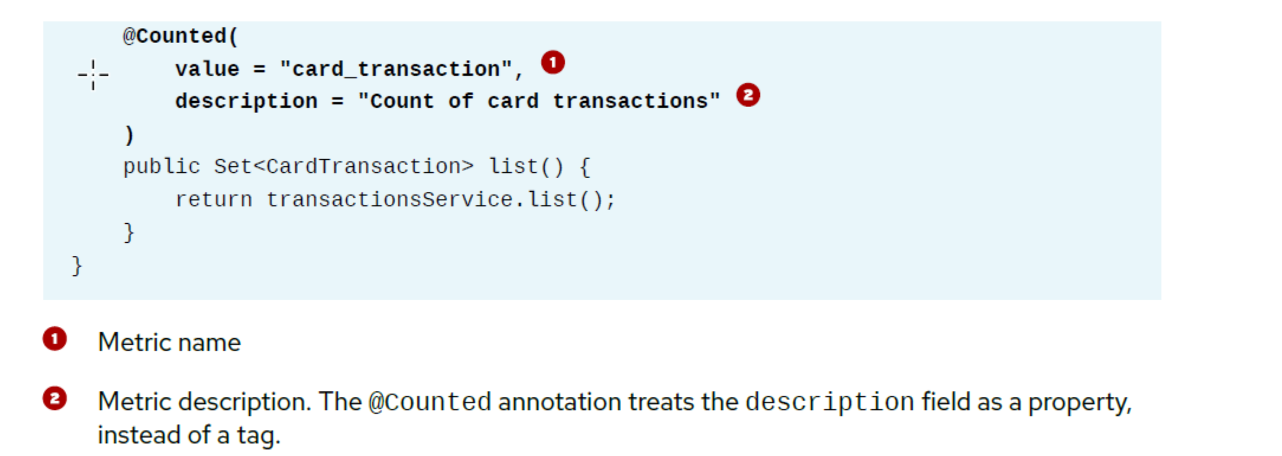
```
package com.redhat.training;

import io.micrometer.core.instrument.MeterRegistry;

public class MyResource {

    @Inject
    public MeterRegistry registry;

    ...code omitted...
}
```

/q/metrics

Counter



```
package com.redhat.training;

import io.micrometer.core.instrument.MeterRegistry;
import io.micrometer.core.instrument.Tags;

...code omitted...
public class CardTransactionsResource {

    ...code omitted...

    @Inject
    public MeterRegistry registry;

    @PostConstruct
    public void initMeters() {
        registry.counter( ❶
            "card_transaction", ❷
            Tags.of("description", "Count of card transactions") ❸
        );
    }

    @GET
    public Set<CardTransaction> list() {
        registry.counter("card_transaction").increment(); ❹

        return transactionsService.list();
    }
}
```

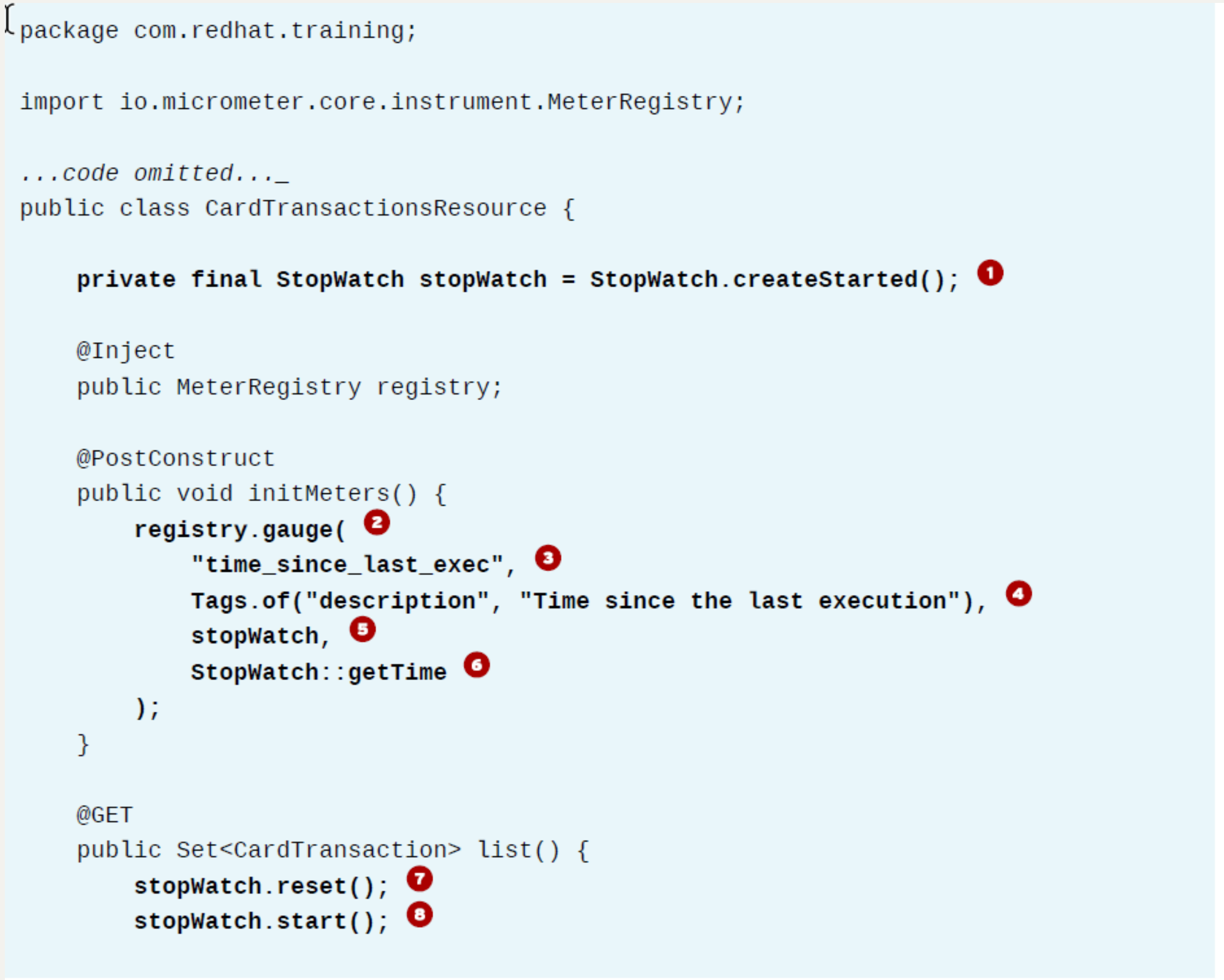


```
    @Counted(
        value = "card_transaction", ❶
        description = "Count of card transactions" ❷
    )
    public Set<CardTransaction> list() {
        return transactionsService.list();
    }
}
```

❶ Metric name

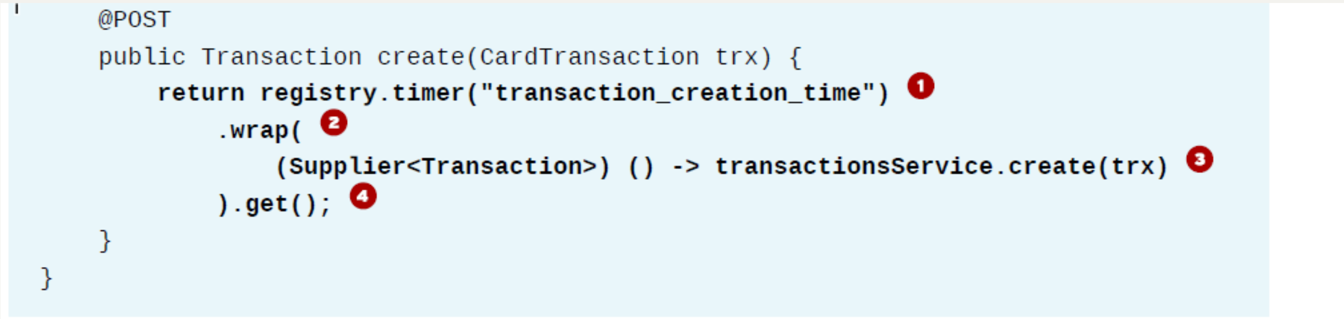
❷ Metric description. The `@Counted` annotation treats the `description` field as a property, instead of a tag.

Gauge



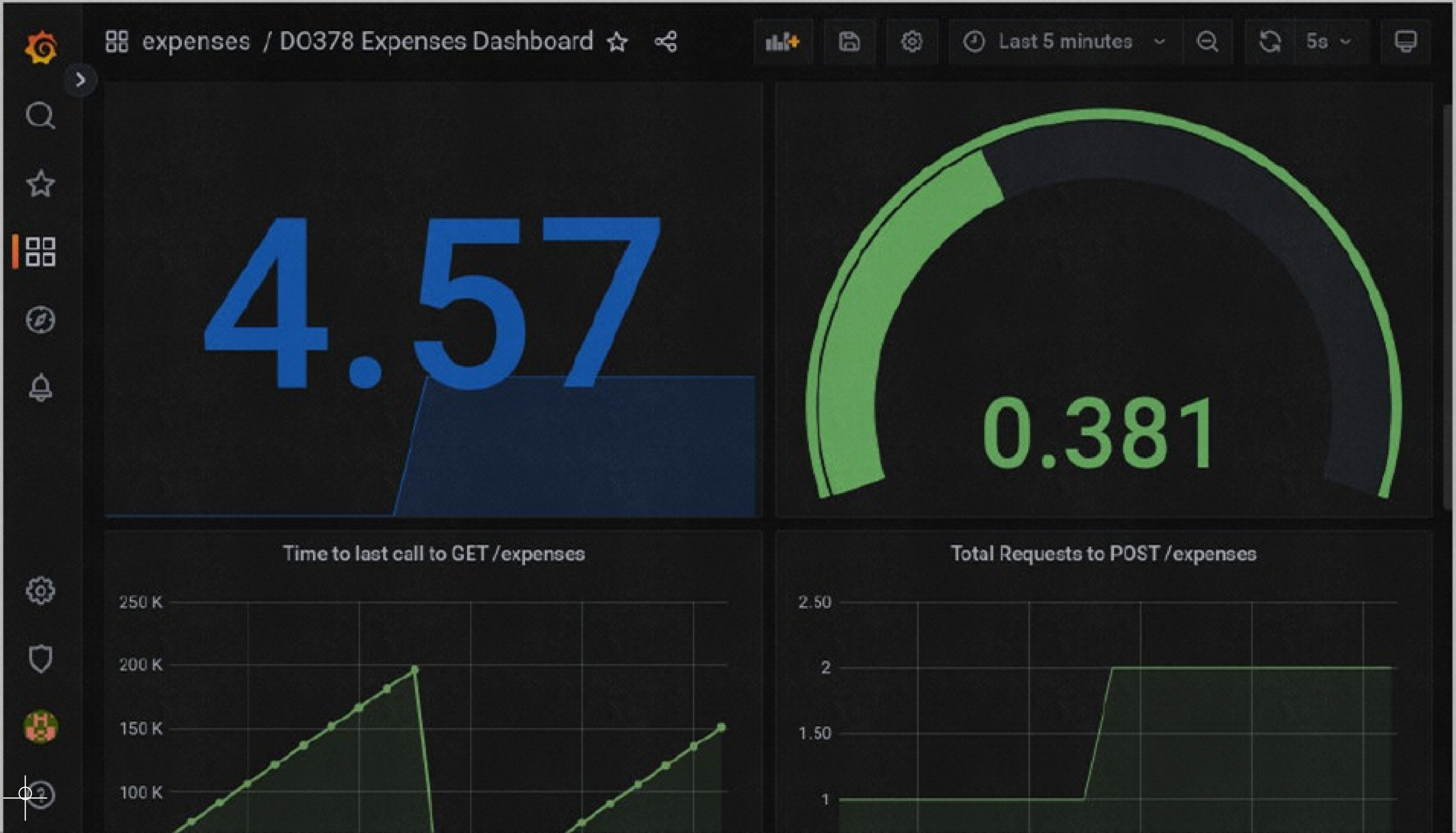
```
package com.redhat.training;

import io.micrometer.core.instrument.MeterRegistry;

...code omitted..._
public class CardTransactionsResource {

    private final StopWatch stopWatch = StopWatch.createStarted(); ❶

    @Inject
    public MeterRegistry registry;

    @PostConstruct
    public void initMeters() {
        registry.gauge( ❷
            "time_since_last_exec", ❸
            Tags.of("description", "Time since the last execution"), ❹
            stopWatch, ❺
            StopWatch::getTime ❻
        );
    }

    @GET
    public Set<CardTransaction> list() {
        stopWatch.reset(); ❼
        stopWatch.start(); ❽
```

Timers



```
    @POST
    public Transaction create(CardTransaction trx) {
        return registry.timer("transaction_creation_time") ❶
            .wrap( ❷
                (Supplier<Transaction>) () -> transactionsService.create(trx) ❸
            ).get(); ❹
    }
}
```

Grafana



expenses / DO378 Expenses Dashboard

Last 5 minutes

5s

4.57

0.381

Time to last call to GET /expenses

Total Requests to POST /expenses