สยามชำนาญกิจ
SIAM CHAMNAN KIT

Week 05

Topics

Application metric with Actuator

Working with Prometheus and Grafana

Domain-Driven Design

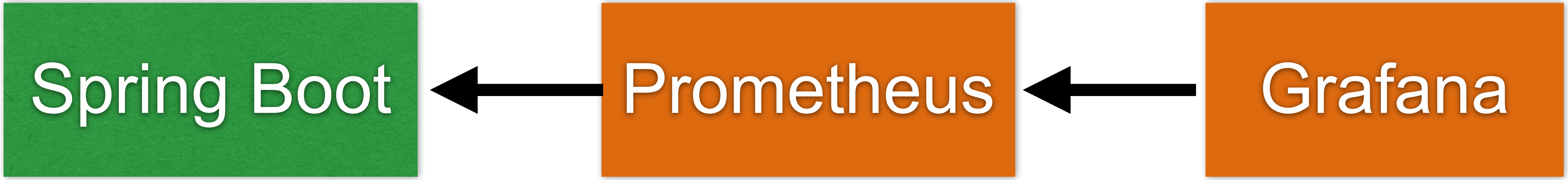
Event storming workshop

Q/A

Application metric with Actuator

https://docs.spring.io/spring-boot/docs/current/reference/html/actuator.html

Architecture

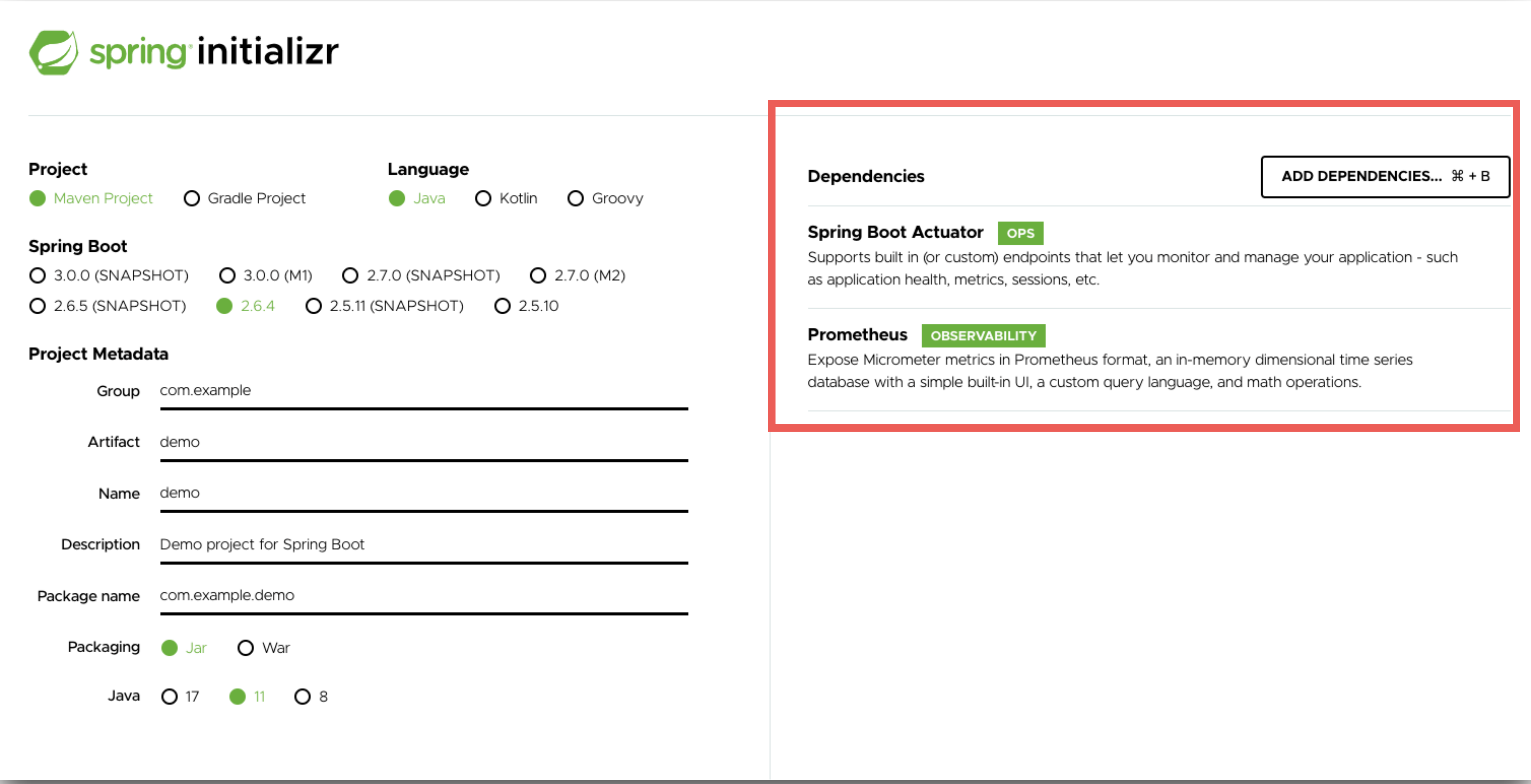


Spring Boot

Prometheus

Grafana

Add actuator and prometheus



spring initializr

Project

● Maven Project ○ Gradle Project

Language

● Java ○ Kotlin ○ Groovy

Spring Boot

○ 3.0.0 (SNAPSHOT) ○ 3.0.0 (M1) ○ 2.7.0 (SNAPSHOT) ○ 2.7.0 (M2)
○ 2.6.5 (SNAPSHOT) ● 2.6.4 ○ 2.5.11 (SNAPSHOT) ○ 2.5.10

Project Metadata

Group: com.example

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package name: com.example.demo

Packaging: ● Jar ○ War

Java: ○ 17 ● 11 ○ 8

Dependencies

Spring Boot Actuator OPS

Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

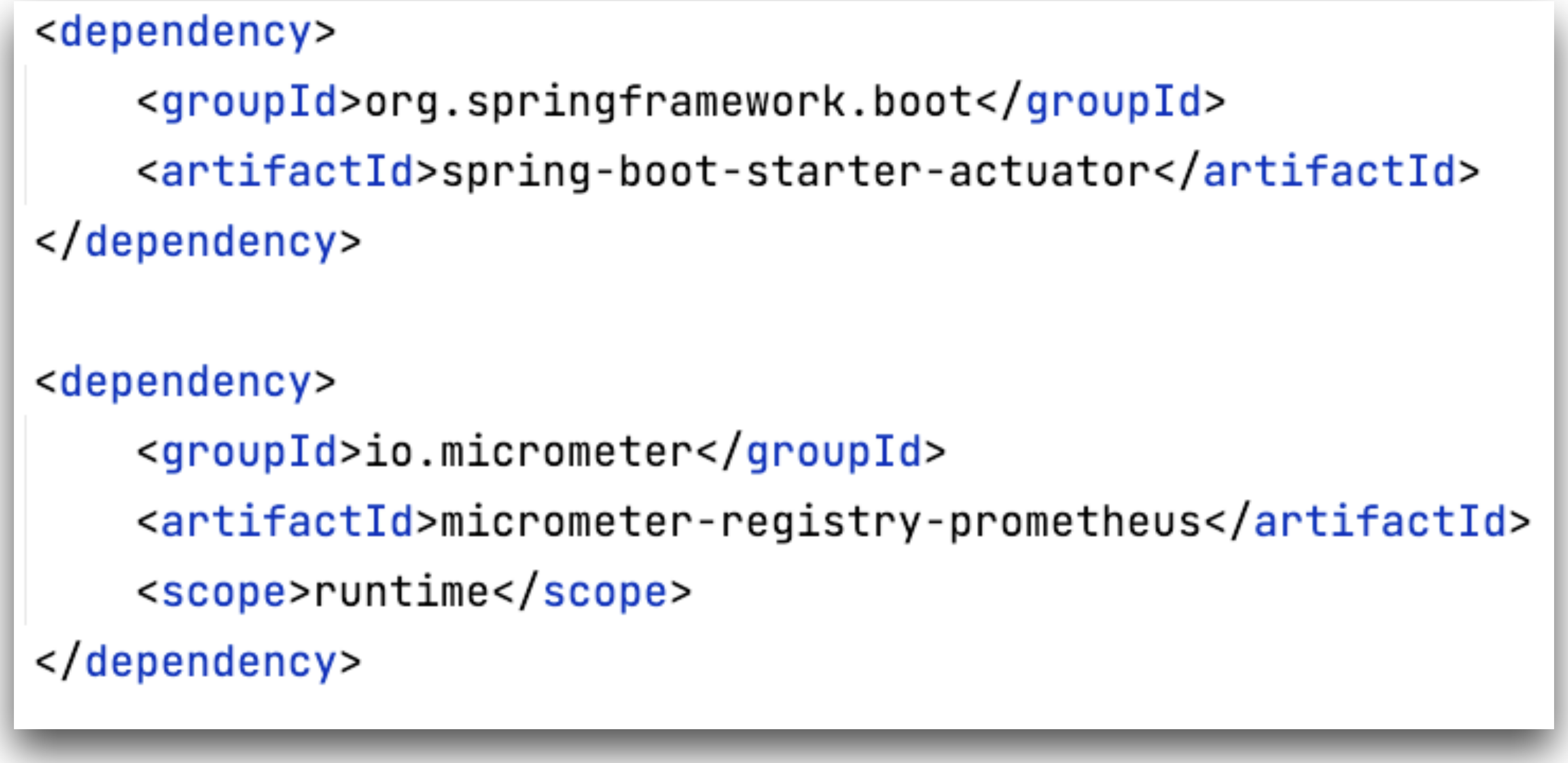
Prometheus OBSERVABILITY

Expose Micrometer metrics in Prometheus format, an in-memory dimensional time series database with a simple built-in UI, a custom query language, and math operations.

https://start.spring.io/

Add actuator and prometheus

Config in file pox.xml



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

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

https://start.spring.io/

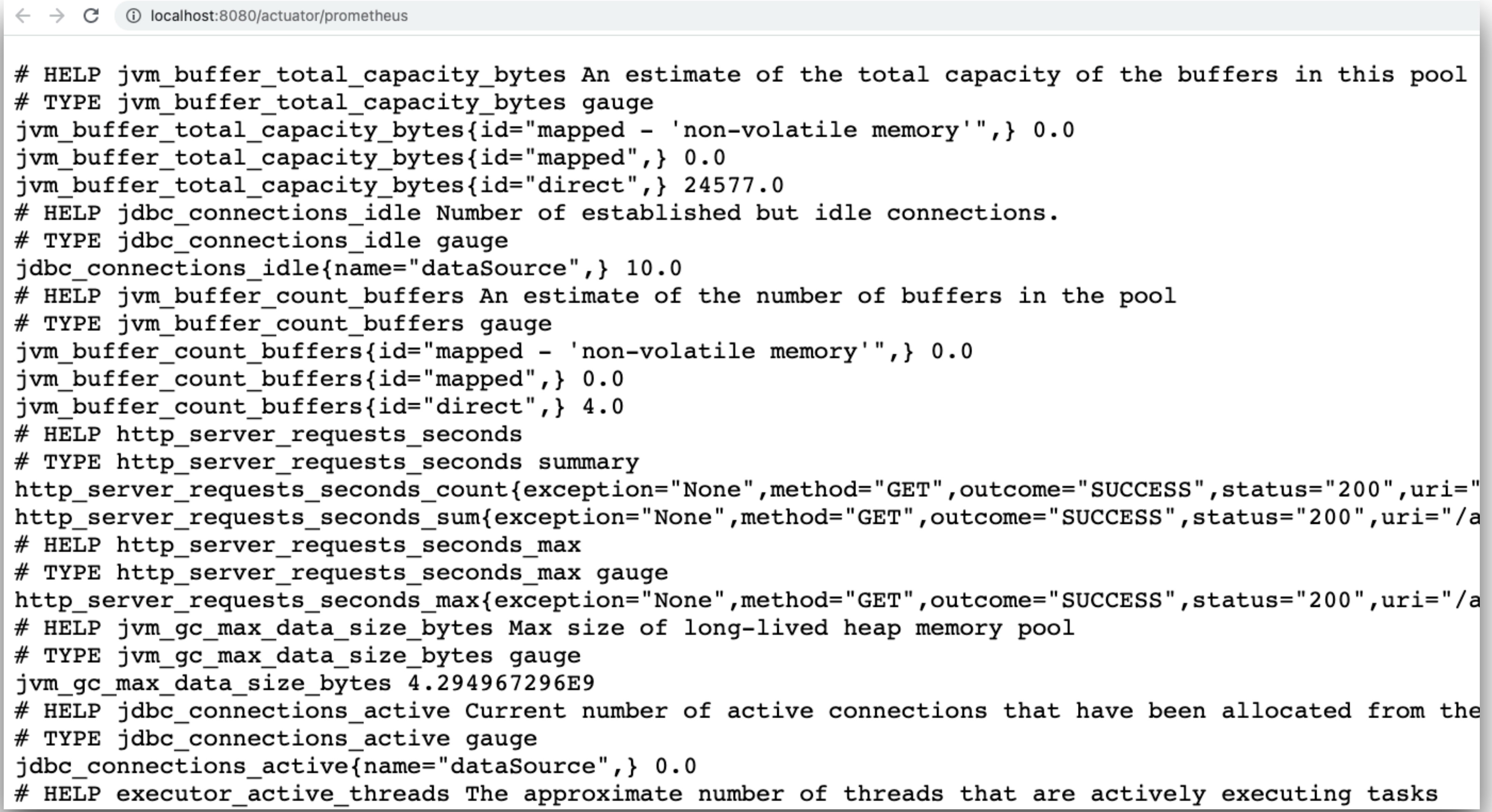
Enable prometheus

Enabled endpoint in application.properties

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

Endpoint of prometheus

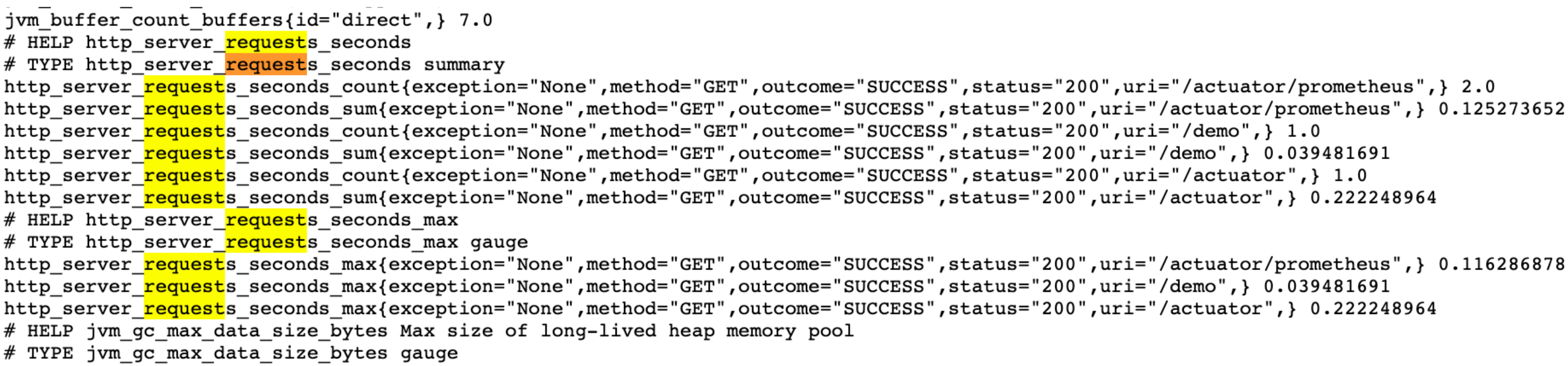
GET /actuator/prometheus



```
# HELP jvm_buffer_total_capacity_bytes An estimate of the total capacity of the buffers in this pool
# TYPE jvm_buffer_total_capacity_bytes gauge
jvm_buffer_total_capacity_bytes{id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_total_capacity_bytes{id="mapped",} 0.0
jvm_buffer_total_capacity_bytes{id="direct",} 24577.0
# HELP jdbc_connections_idle Number of established but idle connections.
# TYPE jdbc_connections_idle gauge
jdbc_connections_idle{name="dataSource",} 10.0
# HELP jvm_buffer_count_buffers An estimate of the number of buffers in the pool
# TYPE jvm_buffer_count_buffers gauge
jvm_buffer_count_buffers{id="mapped - 'non-volatile memory'",} 0.0
jvm_buffer_count_buffers{id="mapped",} 0.0
jvm_buffer_count_buffers{id="direct",} 4.0
# HELP http_server_requests_seconds
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/a
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/a
# HELP jvm_gc_max_data_size_bytes Max size of long-lived heap memory pool
# TYPE jvm_gc_max_data_size_bytes gauge
jvm_gc_max_data_size_bytes 4.294967296E9
# HELP jdbc_connections_active Current number of active connections that have been allocated from the
# TYPE jdbc_connections_active gauge
jdbc_connections_active{name="dataSource",} 0.0
# HELP executor_active_threads The approximate number of threads that are actively executing tasks
```

Endpoint of prometheus

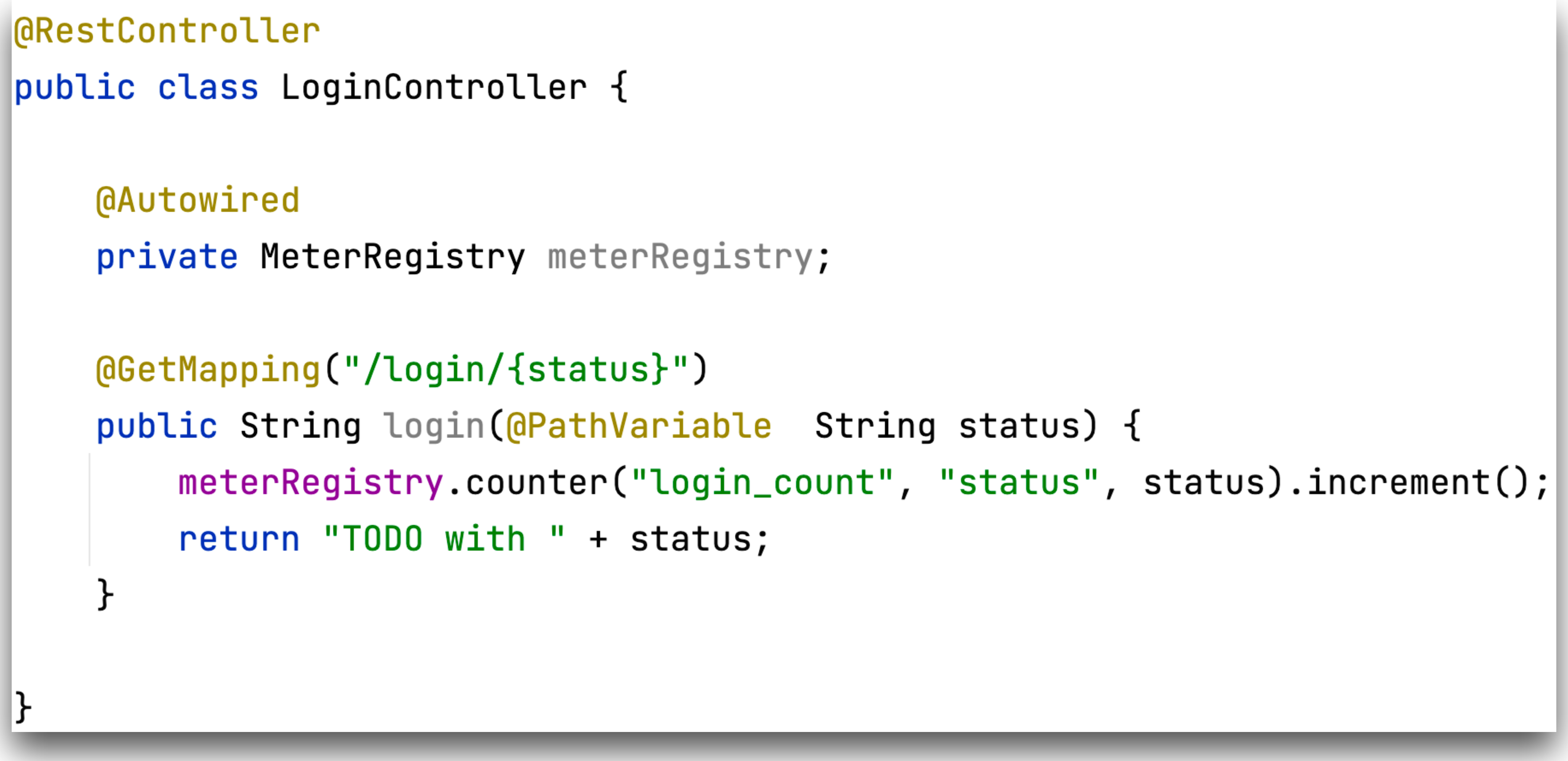
Default metrics



```
jvm_buffer_count_buffers{id="direct",} 7.0
# HELP http_server_requests_seconds
# TYPE http_server_requests_seconds summary
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 2.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 0.125273652
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/demo",} 1.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/demo",} 0.039481691
http_server_requests_seconds_count{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator",} 1.0
http_server_requests_seconds_sum{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator",} 0.222248964
# HELP http_server_requests_seconds_max
# TYPE http_server_requests_seconds_max gauge
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator/prometheus",} 0.116286878
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/demo",} 0.039481691
http_server_requests_seconds_max{exception="None",method="GET",outcome="SUCCESS",status="200",uri="/actuator",} 0.222248964
# HELP jvm_gc_max_data_size_bytes Max size of long-lived heap memory pool
# TYPE jvm_gc_max_data_size_bytes gauge
```

Custom metric

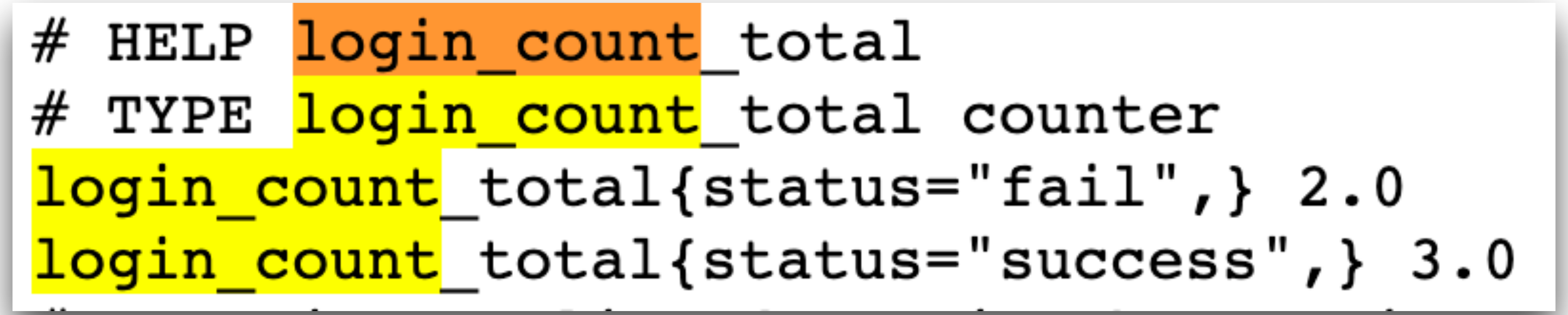
Working with MicroMeter



```java
@RestController
public class LoginController {

    @Autowired
    private MeterRegistry meterRegistry;

    @GetMapping("/login/{status}")
    public String login(@PathVariable  String status) {
        meterRegistry.counter("login_count", "status", status).increment();
        return "TODO with " + status;
    }

}
```

Endpoint of prometheus

Metric name = login_count

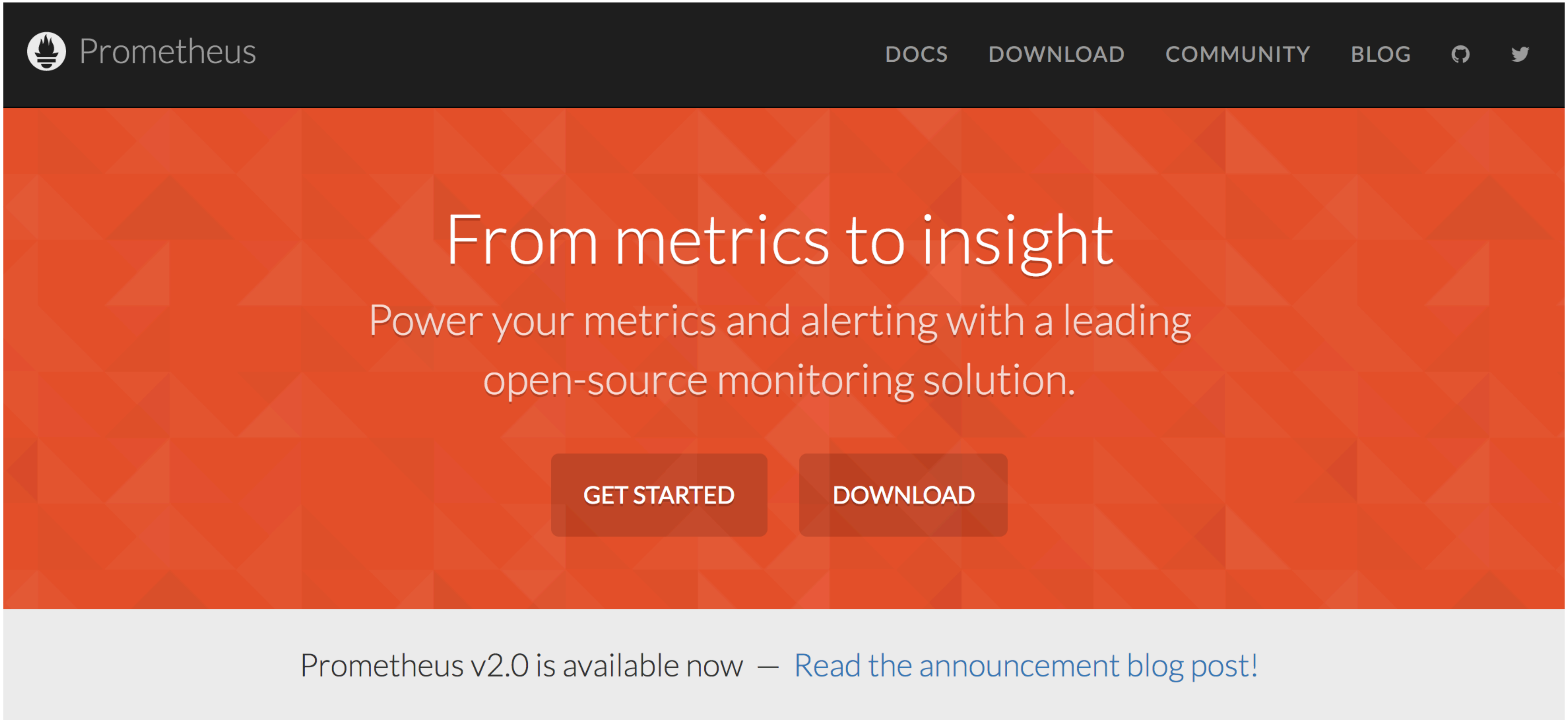


```
# HELP login_count_total
# TYPE login_count_total counter
login_count_total{status="fail",} 2.0
login_count_total{status="success",} 3.0
```

Keep data in Prometheus

https://prometheus.io/

Prometheus



From metrics to insight

Power your metrics and alerting with a leading open-source monitoring solution.

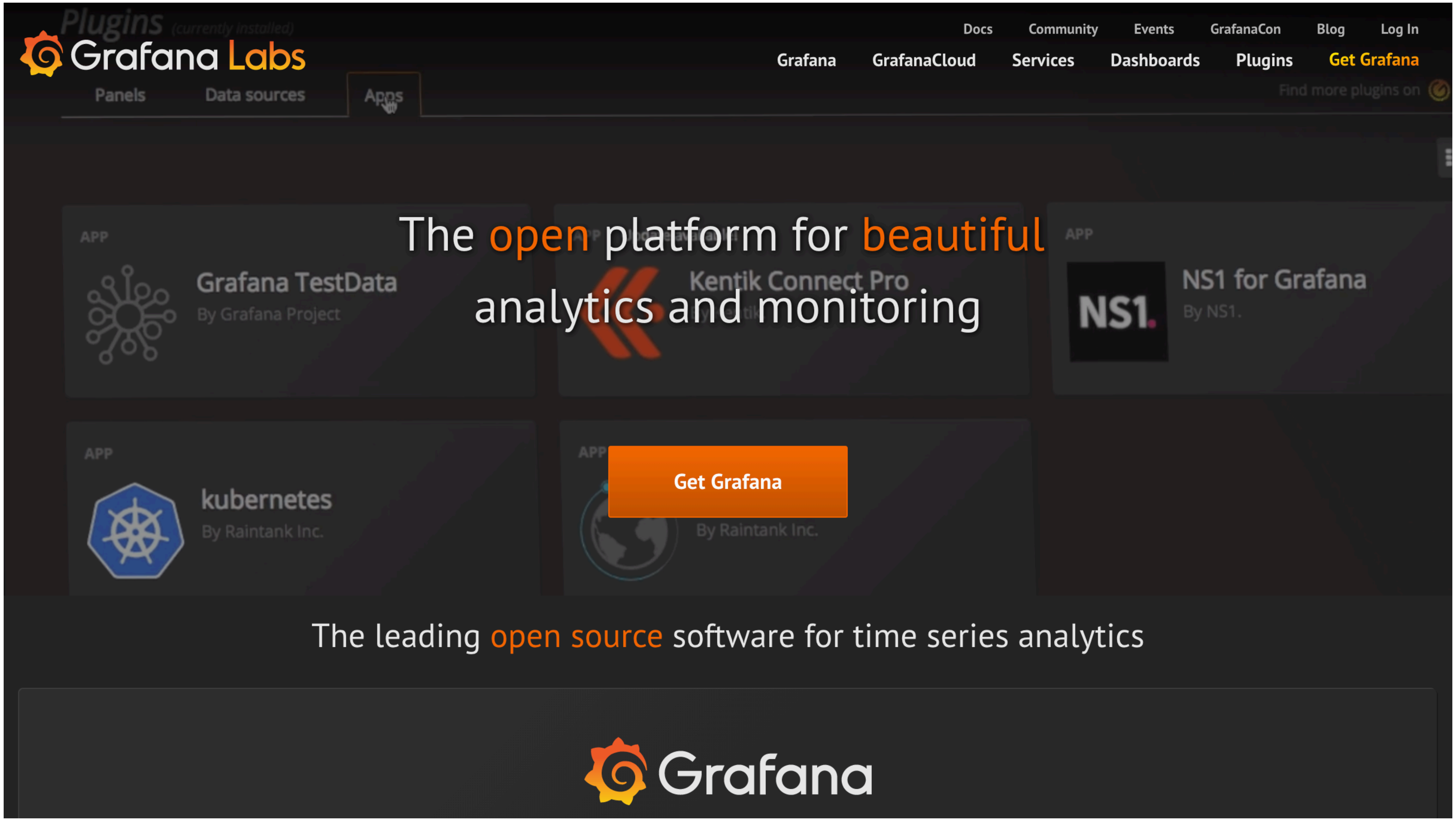
Prometheus v2.0 is available now — Read the announcement blog post!

https://prometheus.io/

Show data in Grafana

https://grafana.com/

Grafana



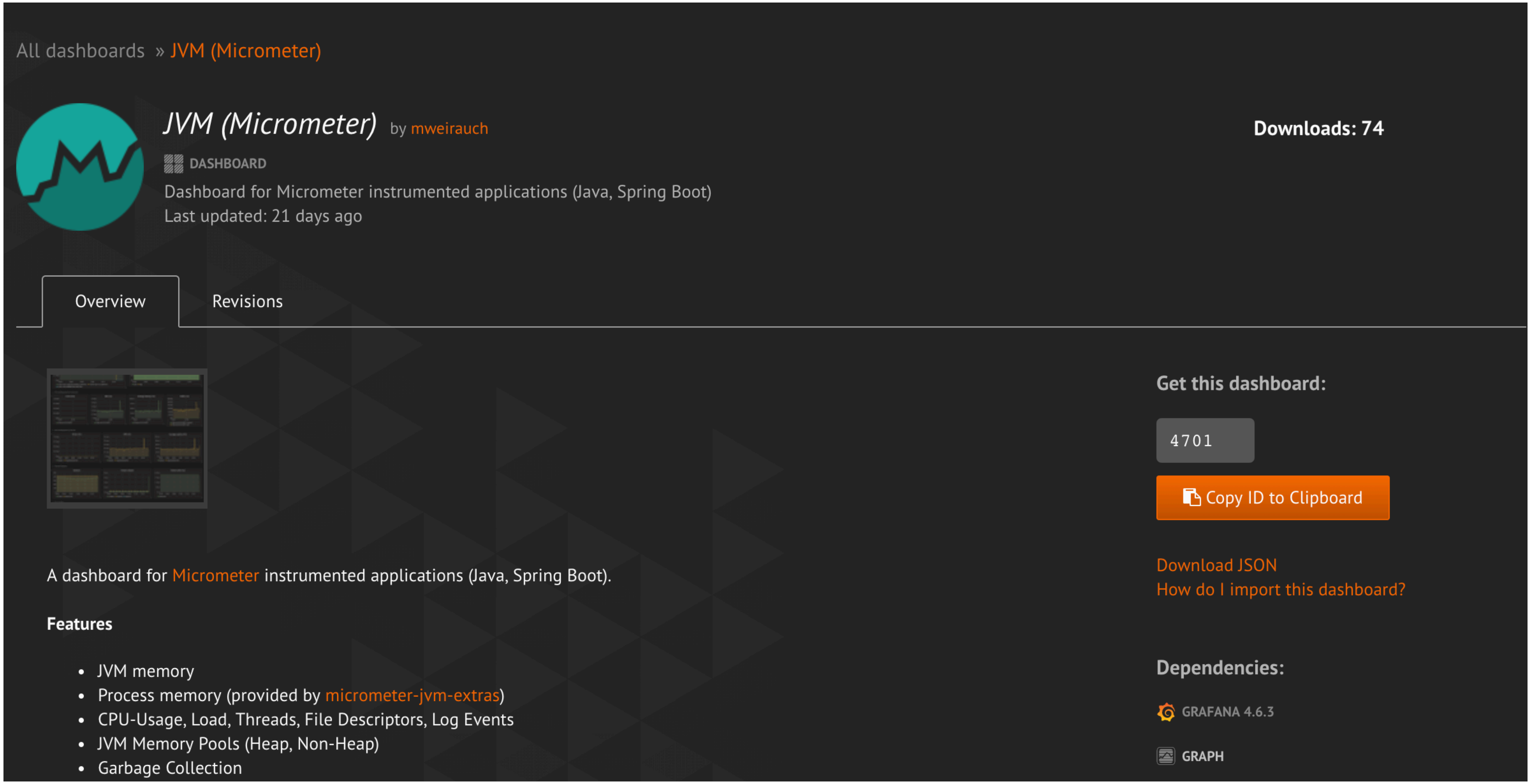
The open platform for beautiful analytics and monitoring

The leading open source software for time series analytics

https://grafana.com/

Grafana Dashboard

https://grafana.com/dashboards/4701



All dashboards » JVM (Micrometer)

JVM (Micrometer) by mweirauch

DASHBOARD

Dashboard for Micrometer instrumented applications (Java, Spring Boot)

Last updated: 21 days ago

Downloads: 74

A dashboard for Micrometer instrumented applications (Java, Spring Boot).

Features

- JVM memory
- Process memory (provided by micrometer-jvm-extras)
- CPU-Usage, Load, Threads, File Descriptors, Log Events
- JVM Memory Pools (Heap, Non-Heap)
- Garbage Collection

Get this dashboard:

4701

Dependencies:

GRAFANA 4.6.3

GRAPH

Domain-Driven Design

Domain-Driven Design

Problem space
Solution space

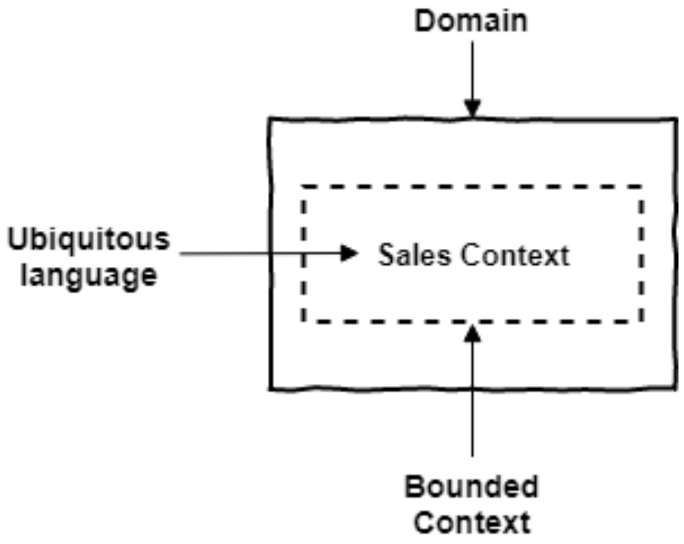
Problem space

Usages of customers

Words used by people and their meaning (Domain language)

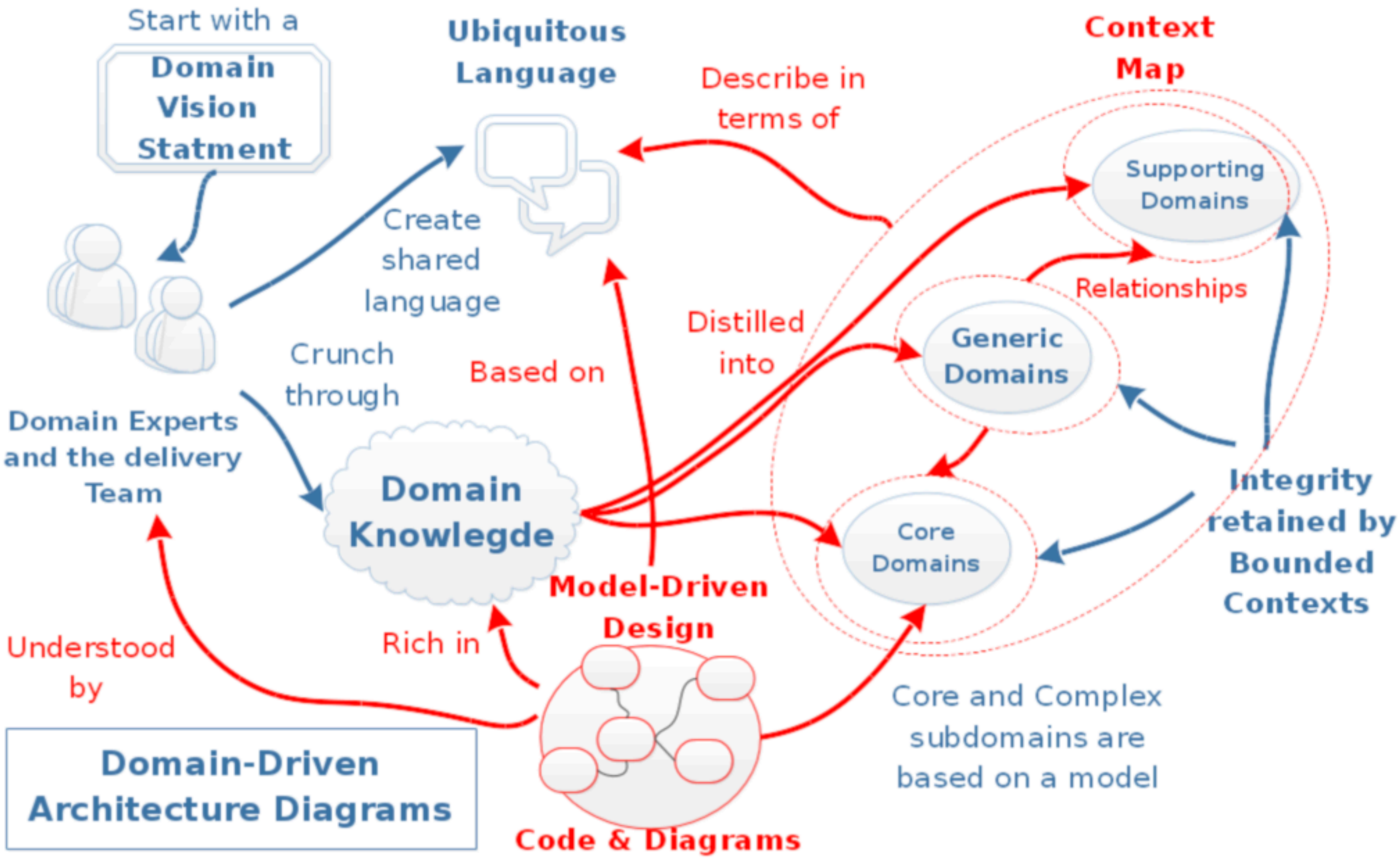
Requirements and **constraints** of business

People who operate business



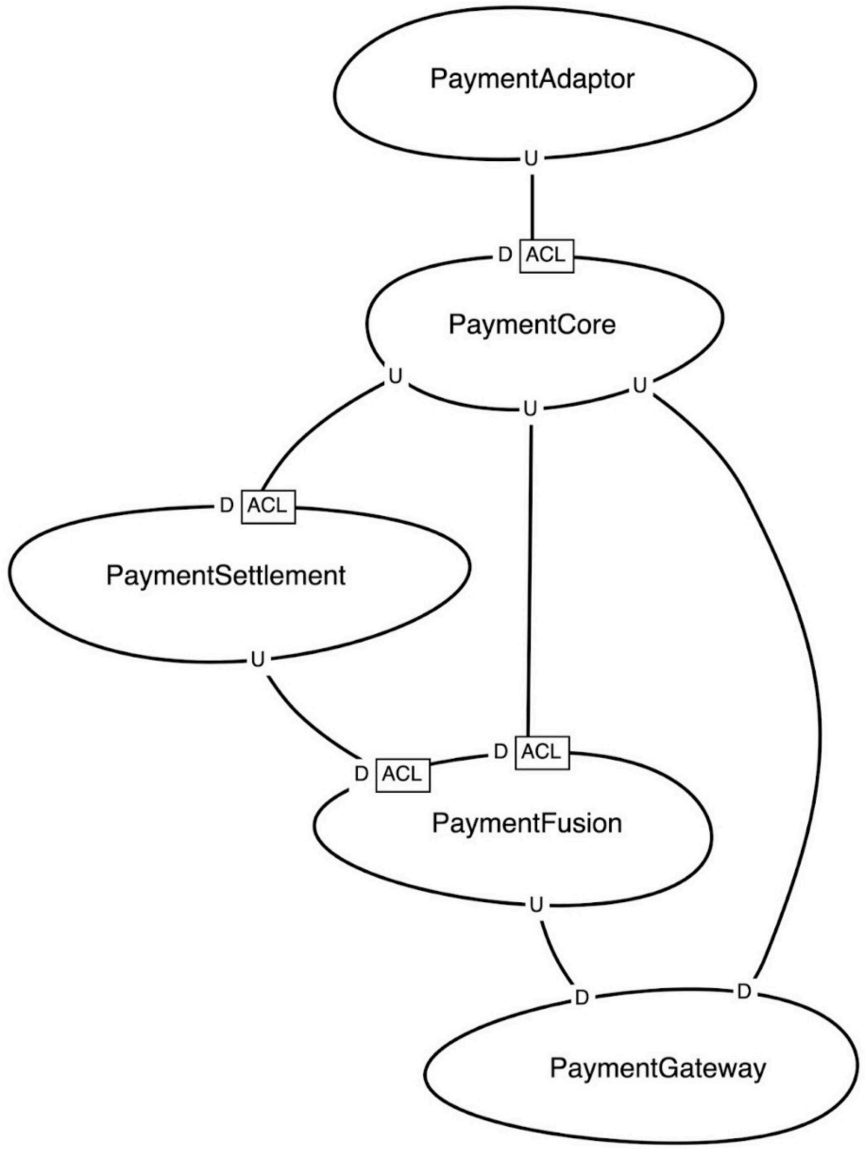
Problem space

Define structure of system
With ***strategic patterns***



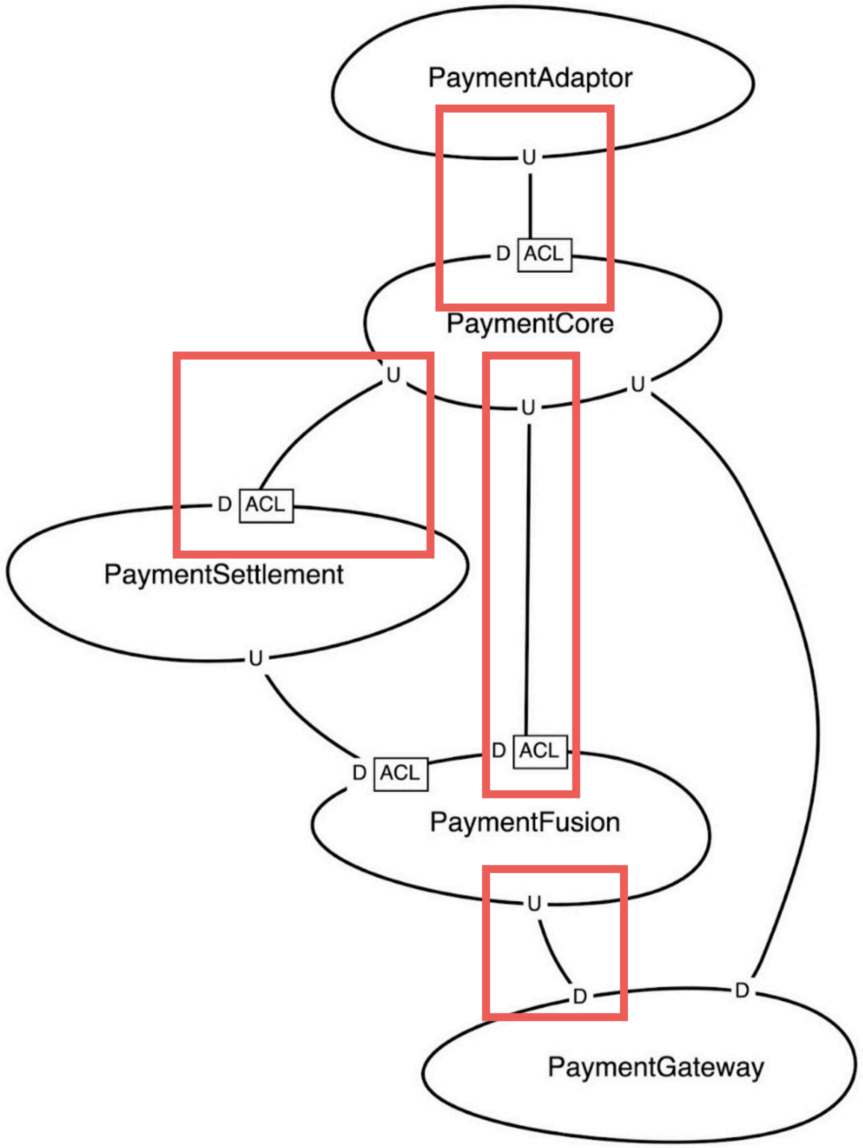
Solution space

Boundary Context (BC) and Context Map

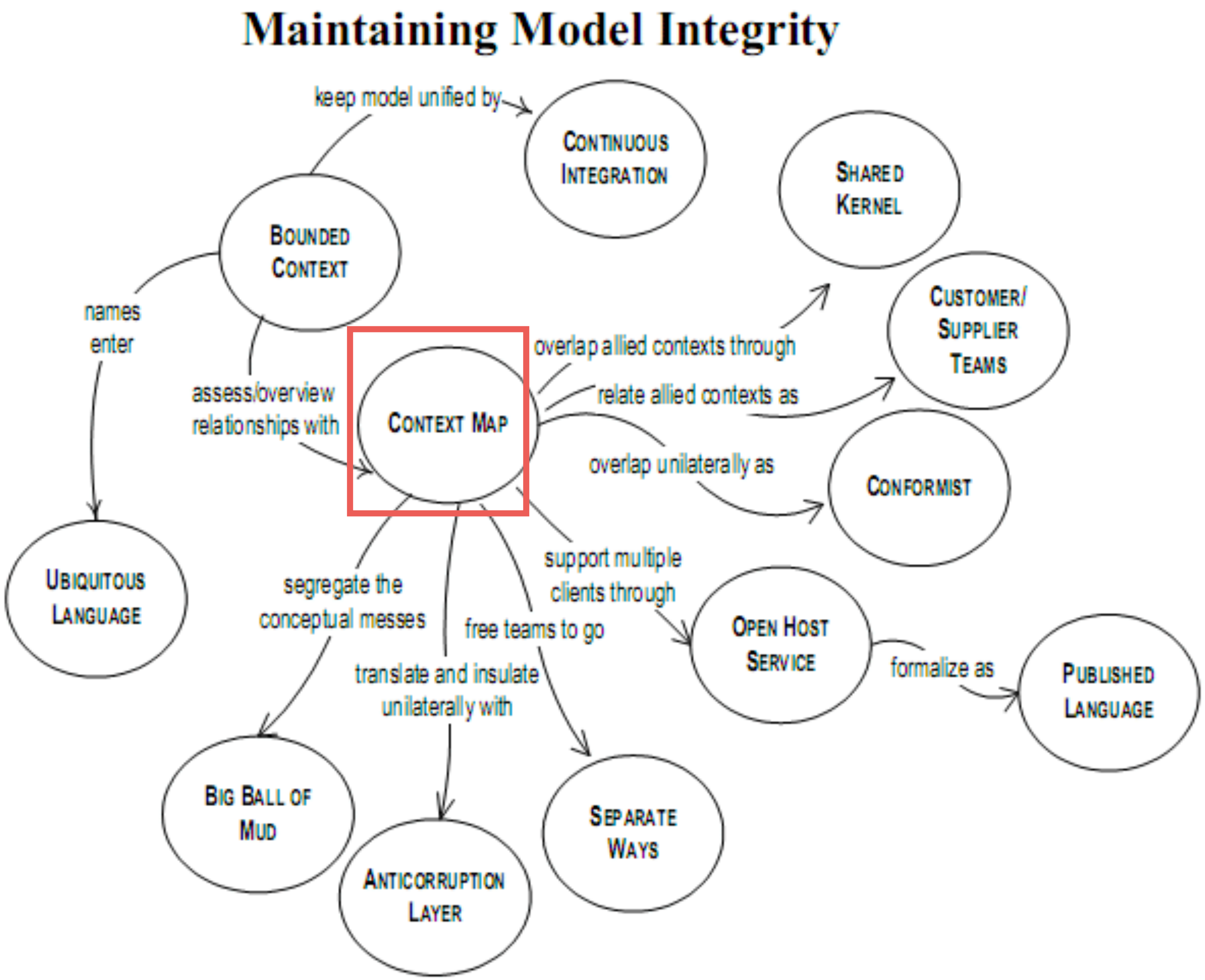


Context Map

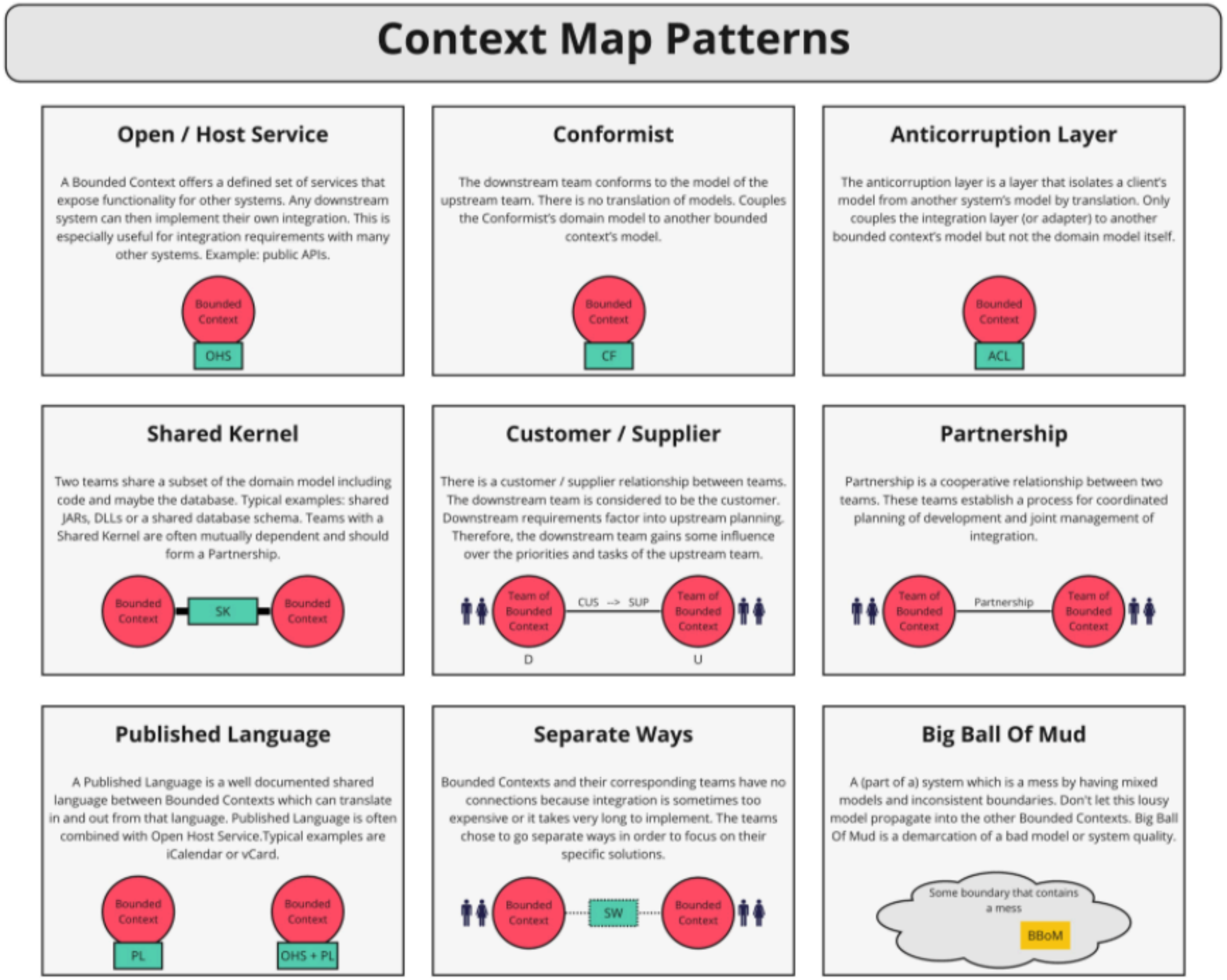
Flow of models between contexts



Context Map


Maintaining Model Integrity
keep model unified by
Continuous Integration
Shared Kernel
Bounded Context
names enter
Customer/ Supplier Teams
overlap allied contexts through
assess/overview relationships with
Context Map
relate allied contexts as
overlap unilaterally as
Conformist
support multiple clients through
Ubiquitous Language
segregate the conceptual messes
free teams to go
Open Host Service
formalize as
Published Language
translate and insulate unilaterally with
Big Ball of Mud
Separate Ways
Anticorruption Layer

Context Map



Context Map Patterns

Open / Host Service

A Bounded Context offers a defined set of services that expose functionality for other systems. Any downstream system can then implement their own integration. This is especially useful for integration requirements with many other systems. Example: public APIs.

Conformist

The downstream team conforms to the model of the upstream team. There is no translation of models. Couples the Conformist's domain model to another bounded context's model.

Anticorruption Layer

The anticorruption layer is a layer that isolates a client's model from another system's model by translation. Only couples the integration layer (or adapter) to another bounded context's model but not the domain model itself.

Shared Kernel

Two teams share a subset of the domain model including code and maybe the database. Typical examples: shared JARs, DLLs or a shared database schema. Teams with a Shared Kernel are often mutually dependent and should form a Partnership.

Customer / Supplier

There is a customer / supplier relationship between teams. The downstream team is considered to be the customer. Downstream requirements factor into upstream planning. Therefore, the downstream team gains some influence over the priorities and tasks of the upstream team.

Partnership

Partnership is a cooperative relationship between two teams. These teams establish a process for coordinated planning of development and joint management of integration.

Published Language

A Published Language is a well documented shared language between Bounded Contexts which can translate in and out from that language. Published Language is often combined with Open Host Service.Typical examples are iCalendar or vCard.

Separate Ways

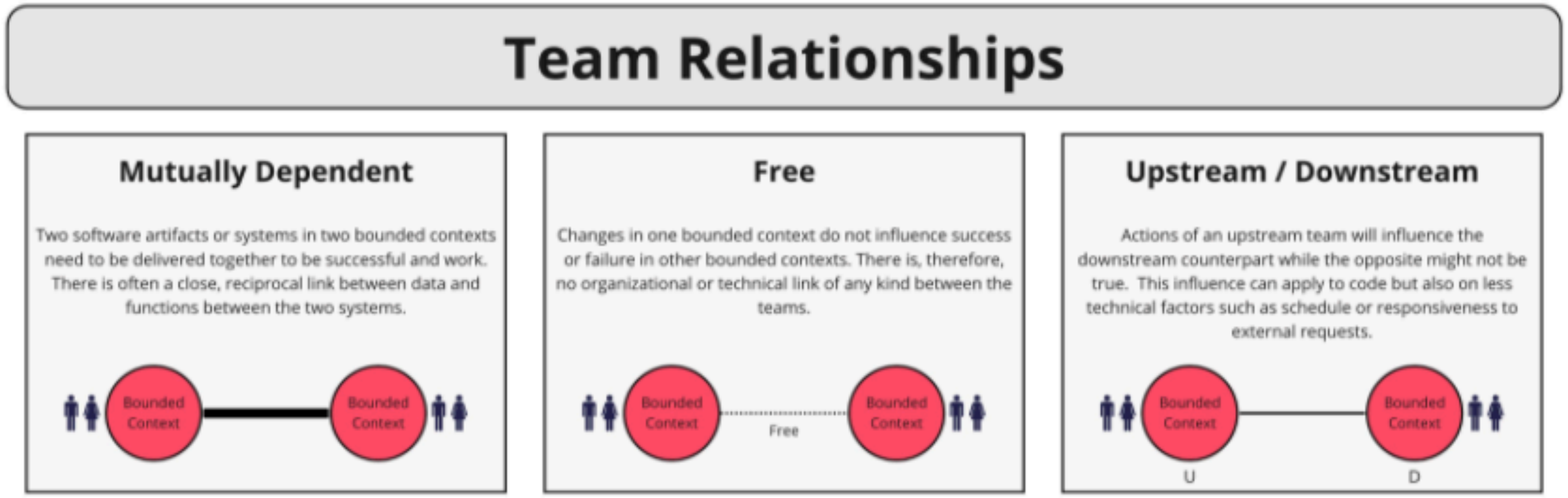
Bounded Contexts and their corresponding teams have no connections because integration is sometimes too expensive or it takes very long to implement. The teams chose to go separate ways in order to focus on their specific solutions.

Big Ball Of Mud

A (part of a) system which is a mess by having mixed models and inconsistent boundaries. Don't let this lousy model propagate into the other Bounded Contexts. Big Ball Of Mud is a demarcation of a bad model or system quality.

https://github.com/ddd-crew/context-mapping

Team Relationships



Team Relationships

Mutually Dependent

Two software artifacts or systems in two bounded contexts need to be delivered together to be successful and work. There is often a close, reciprocal link between data and functions between the two systems.

Free

Changes in one bounded context do not influence success or failure in other bounded contexts. There is, therefore, no organizational or technical link of any kind between the teams.

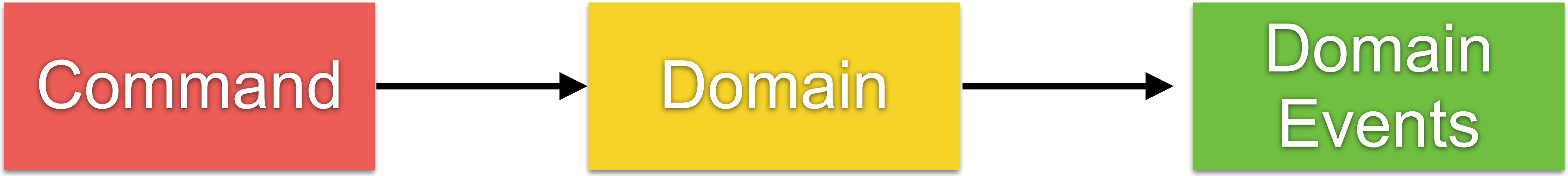
Upstream / Downstream

Actions of an upstream team will influence the downstream counterpart while the opposite might not be true. This influence can apply to code but also on less technical factors such as schedule or responsiveness to external requests.

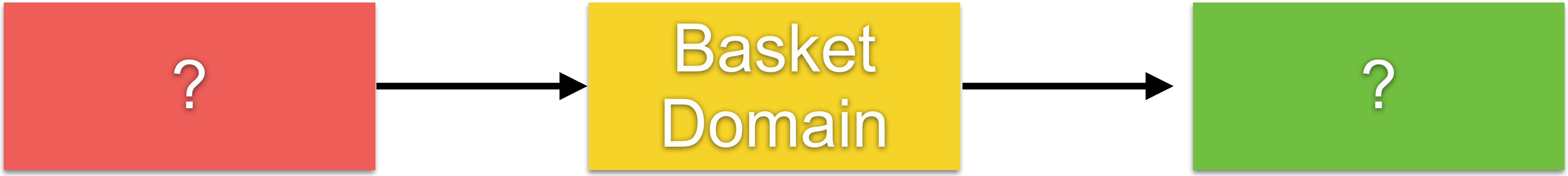
https://github.com/ddd-crew/context-mapping

Problem space with Event Storming workshop

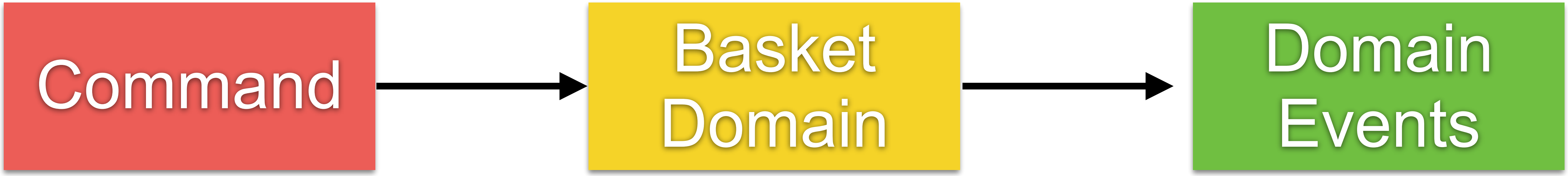
Event Storming



Event Storming



Event Storming



AddProductToBasket

Add product

AddProductToBasketEvent

Q/A