Spring I/O | Barcelona | 2025-05-23

Getting your application production-ready with Actuator

INNOQ

MICHAEL VITZ
SENIOR CONSULTANT

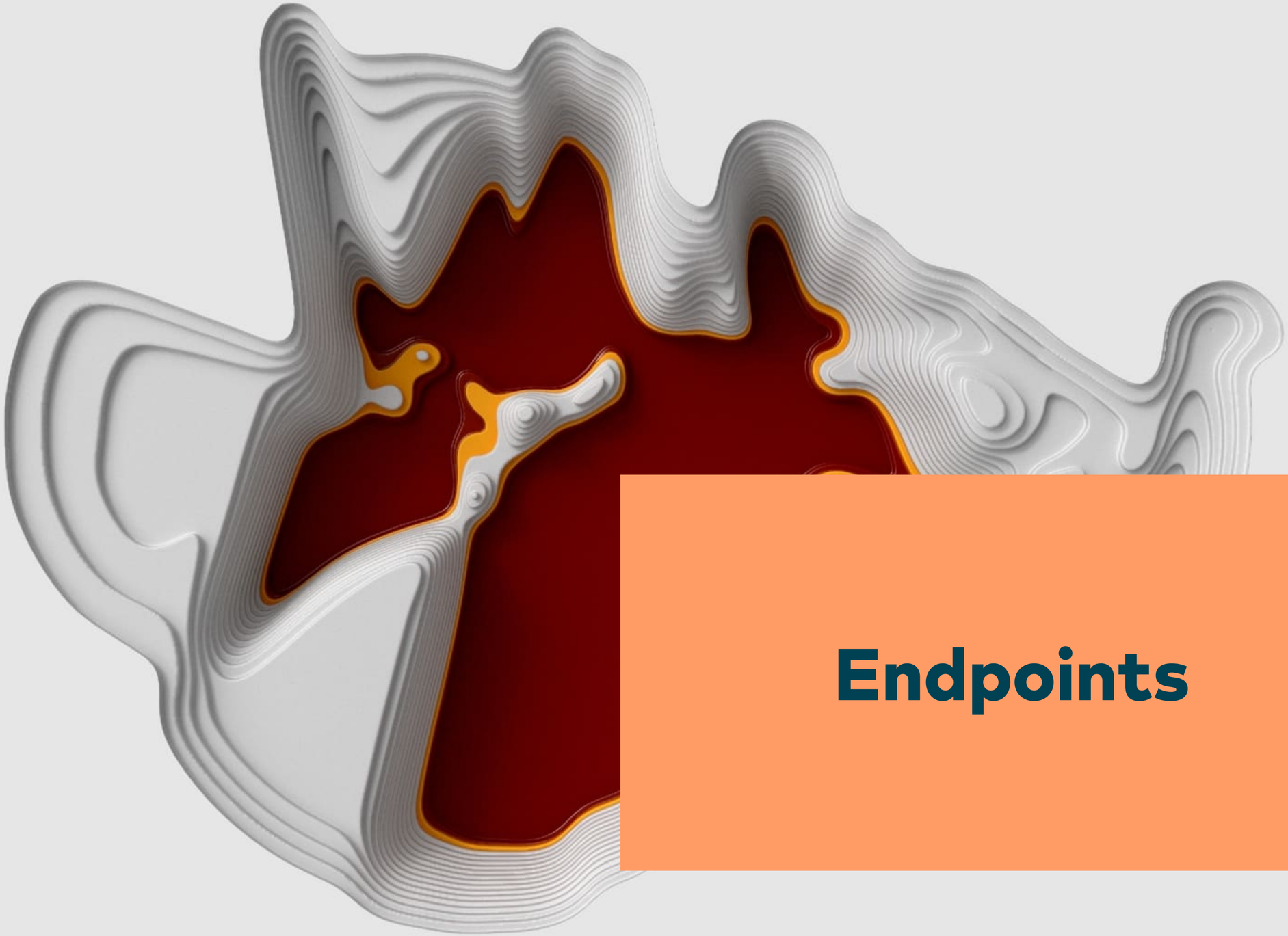
MICHAEL VITZ

Java Champion
Senior Consultant at INNOQ

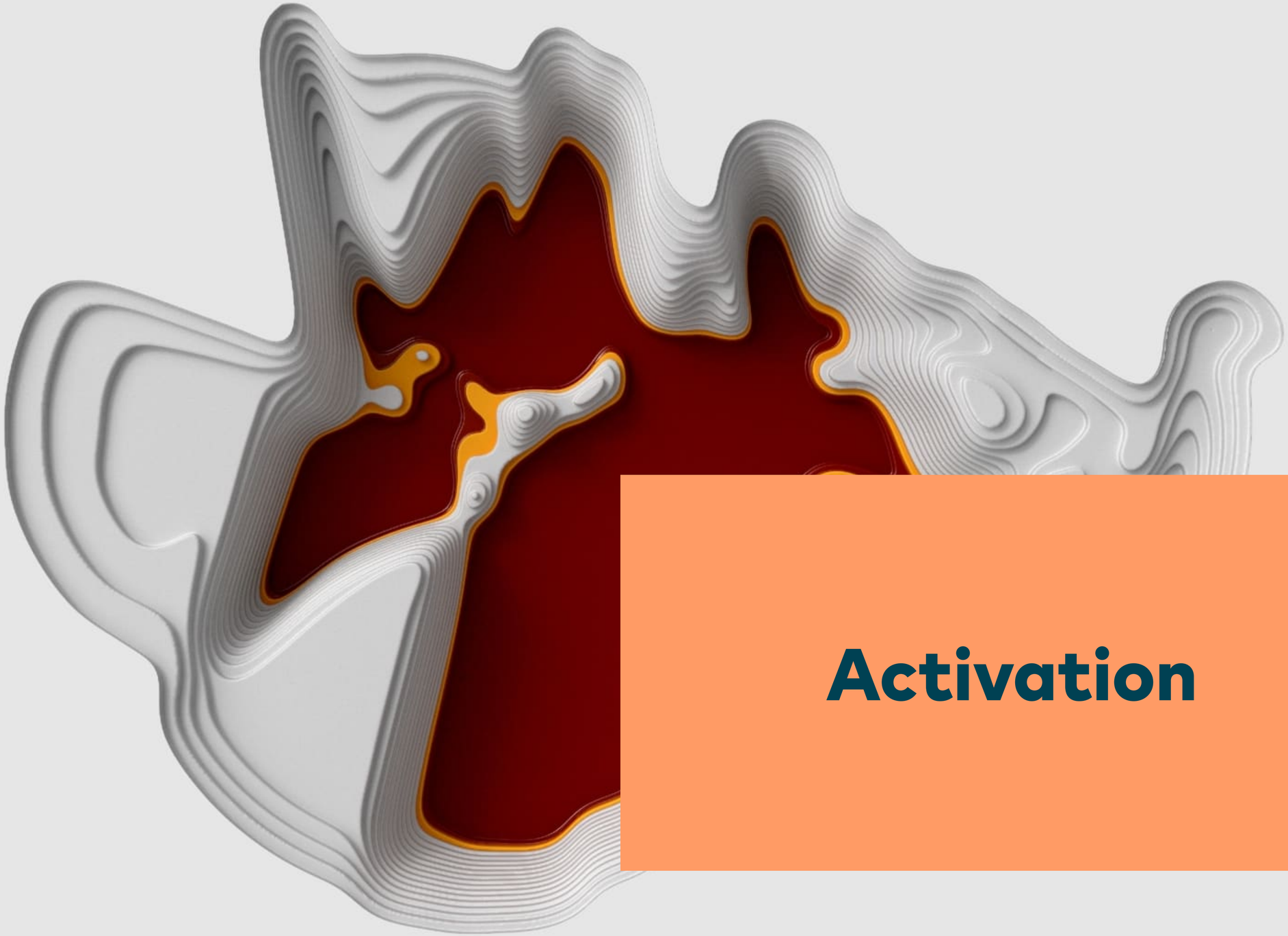
Actuator

Endpoints

Observability



Endpoints



Activation

Enable Actuator

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

Enable Actuator Web

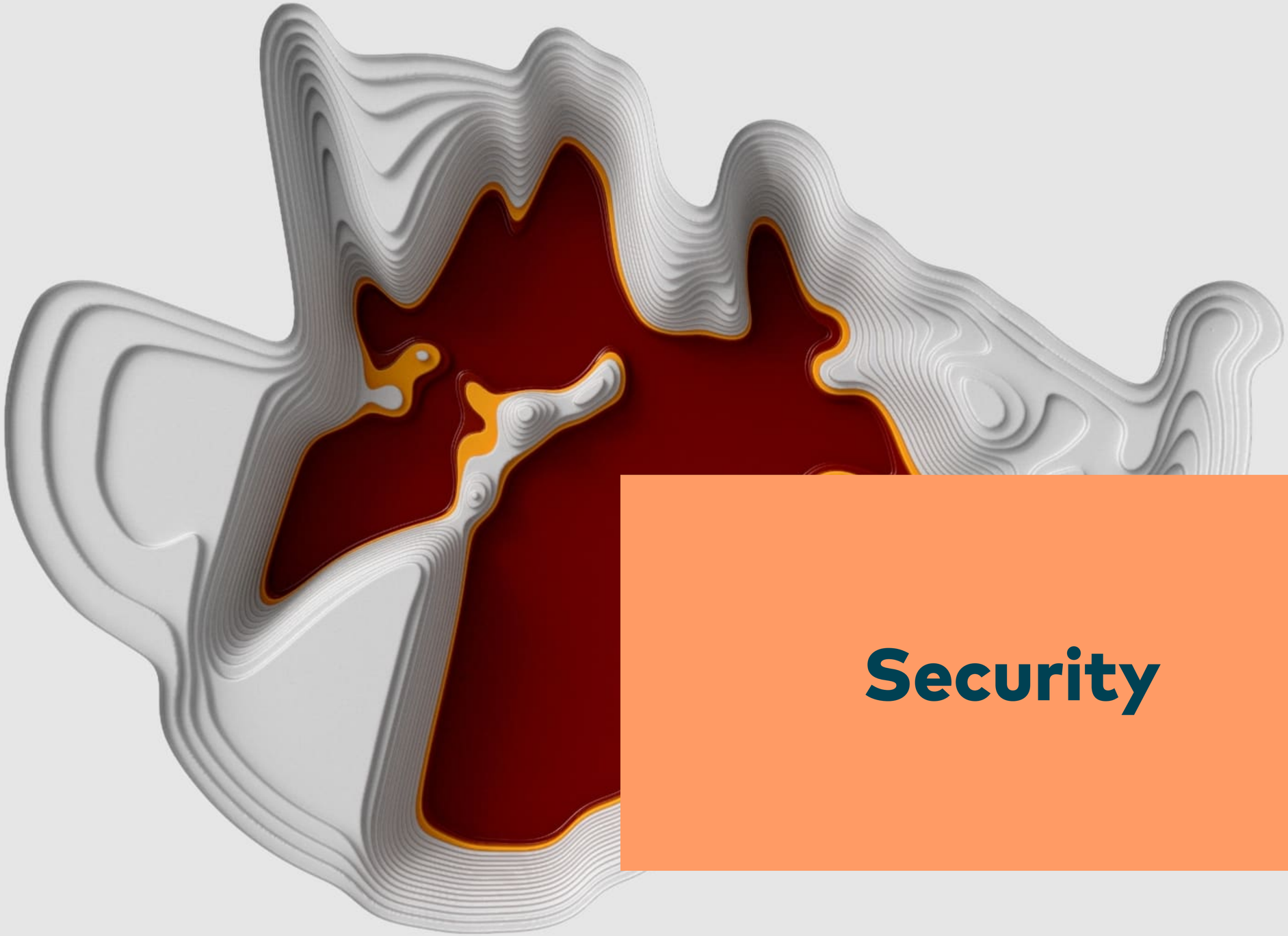
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Expose (all) endpoints

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

Only expose what you really use!

heapdump and shutdown require additional activation



Security

Expose on own port

```
management.server.port=8081
```

Add Spring Security

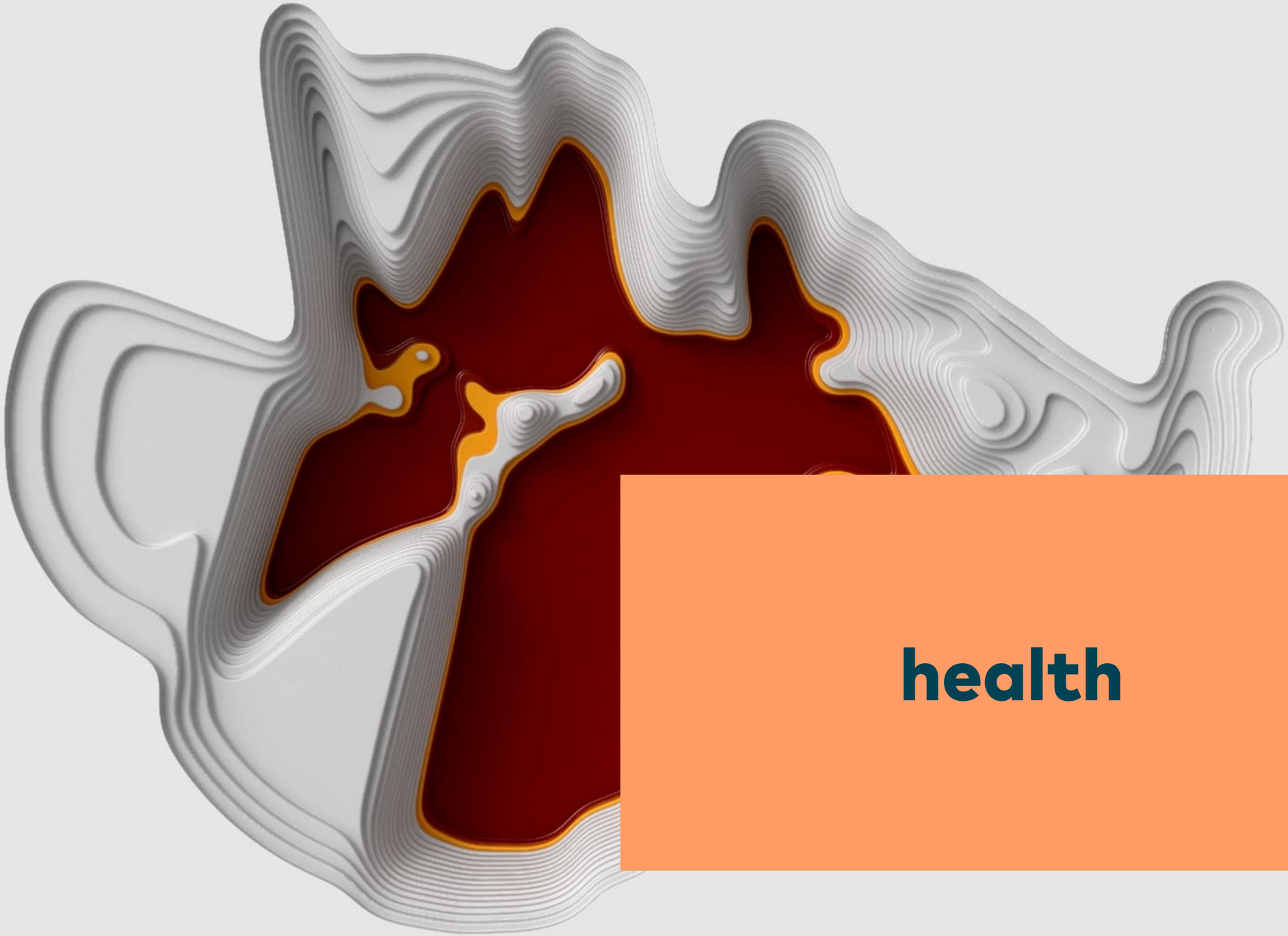
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

With custom SecurityFilterChain

```
@Bean
public SecurityFilterChain actuatorSecurityFilterChain(HttpSecurity http) throws Exception {
    return http.securityMatcher(EndpointRequest.toAnyEndpoint())
            .authorizeHttpRequests(requests -> requests
                    .requestMatchers(EndpointRequest.to("health")).permitAll()
                    .anyRequest().authenticated())
            .httpBasic(Customizer.withDefaults())
            .build();
}
```

Adding your own SecurityFilterChain disables defaults

-> you need to secure your endpoints yourself!



health

Show details

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

More builtin

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-cache</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-redis</artifactId>
</dependency>
```

Additional starters may add their own health indicators

Write your own

```
@Bean
HealthIndicator foo() {
    return new HealthIndicator() {
        @Override
        public Health health() {
            if (UP.get()) {
                return Health.up().withDetail("foo", Instant.now().toString()).build();
            }
            return Health.down().build();
        }
    };
}
```

Caching

```
management.endpoint.health.cache.time-to-live=60s
```

Status Aggregation

```
management.endpoint.health.status.order=UP, DOWN, OUT_OF_SERVICE, UNKNOWN
management.endpoint.health.status.http-mapping.UP=502
```

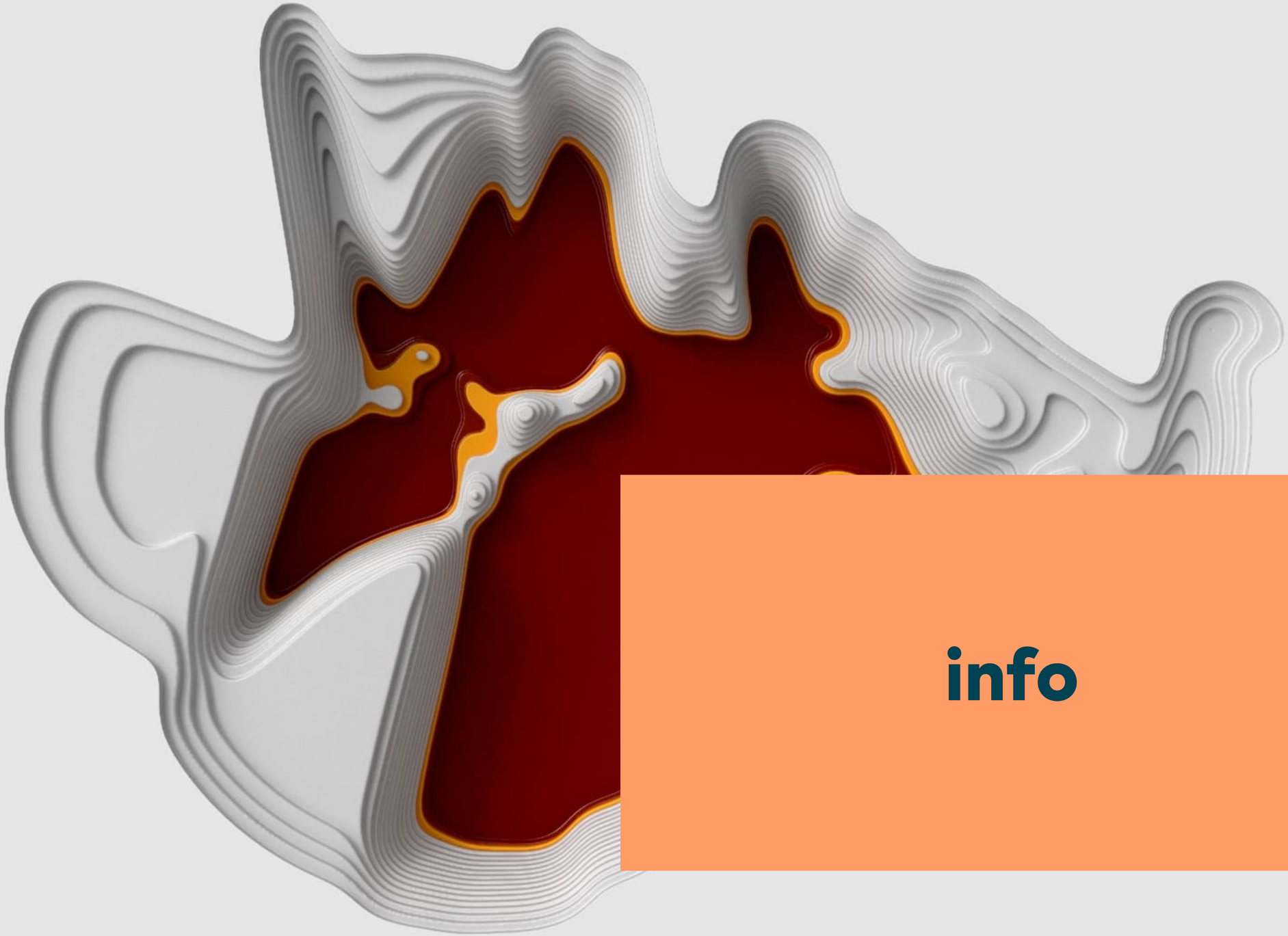
```
return new StatusAggregator() {
    @Override
    public Status getAggregateStatus(Set<Status> statuses) {
        return Status.UP;
    }
};
```

Adding your own http-mapping removes the defaults!

K8s support

```
management.endpoint.health.probes.enabled=true
management.endpoint.health.probes.add-additional-paths=true
```

App should detect K8s itself -> probes are enabled by default



info

Builtin contributors

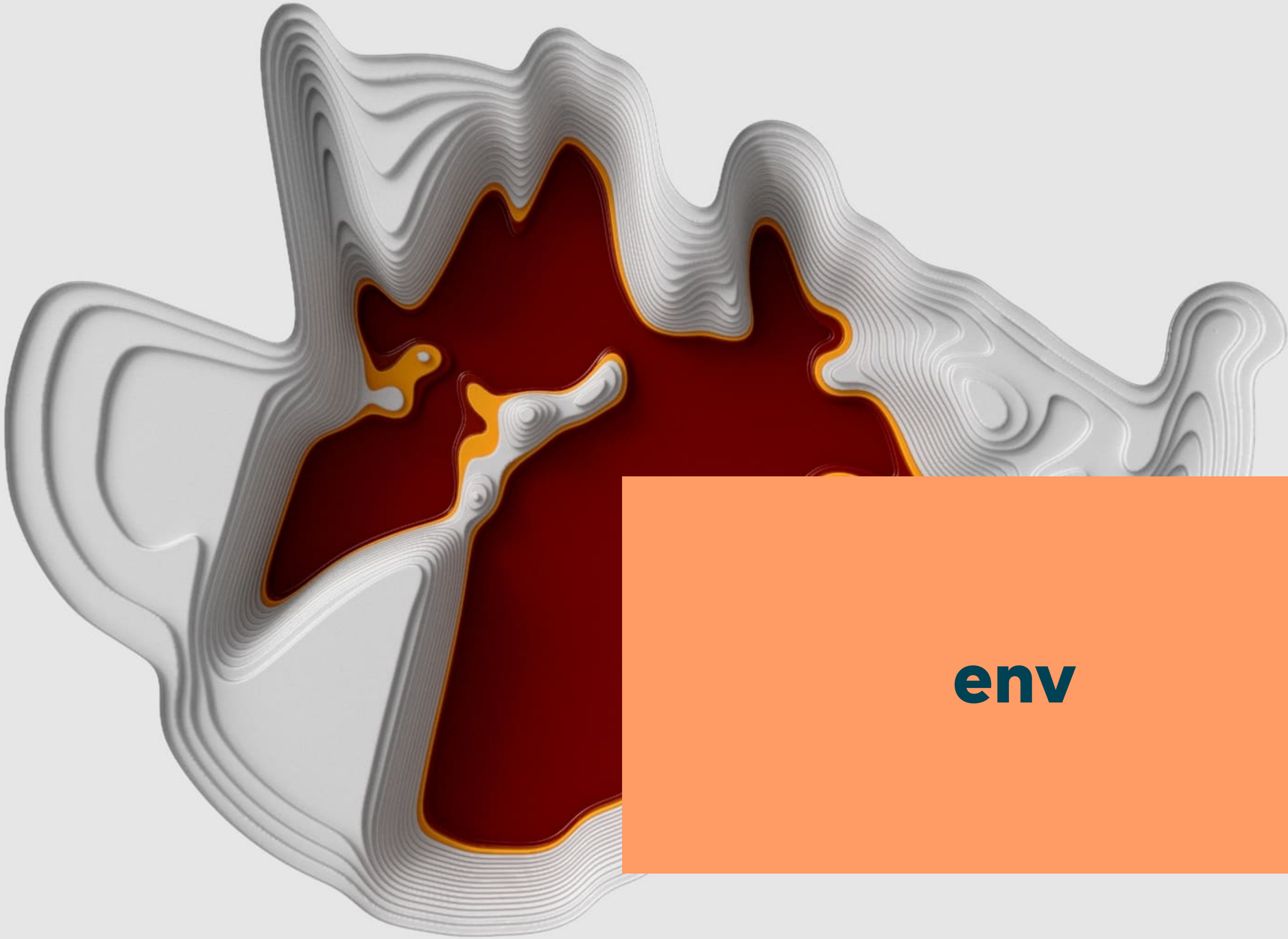
- build
- env
- git
- java
- os
- process
- ssl

Builtin contributors

```
<plugin>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-maven-plugin</artifactId>
  <goals>
    <goal>build-info</goal>
  </goals>
</plugin>
<plugin>
  <groupId>io.github.git-commit-id</groupId>
  <artifactId>git-commit-id-maven-plugin</artifactId>
</plugin>
```

Write your own

```
@Bean
InfoContributor barInfoContributor() {
    return new InfoContributor() {
        @Override
        public void contribute(Info.Builder builder) {
            builder.withDetail("bar", Map.of("now", Instant.now().toString()));
        }
    };
}
```



env

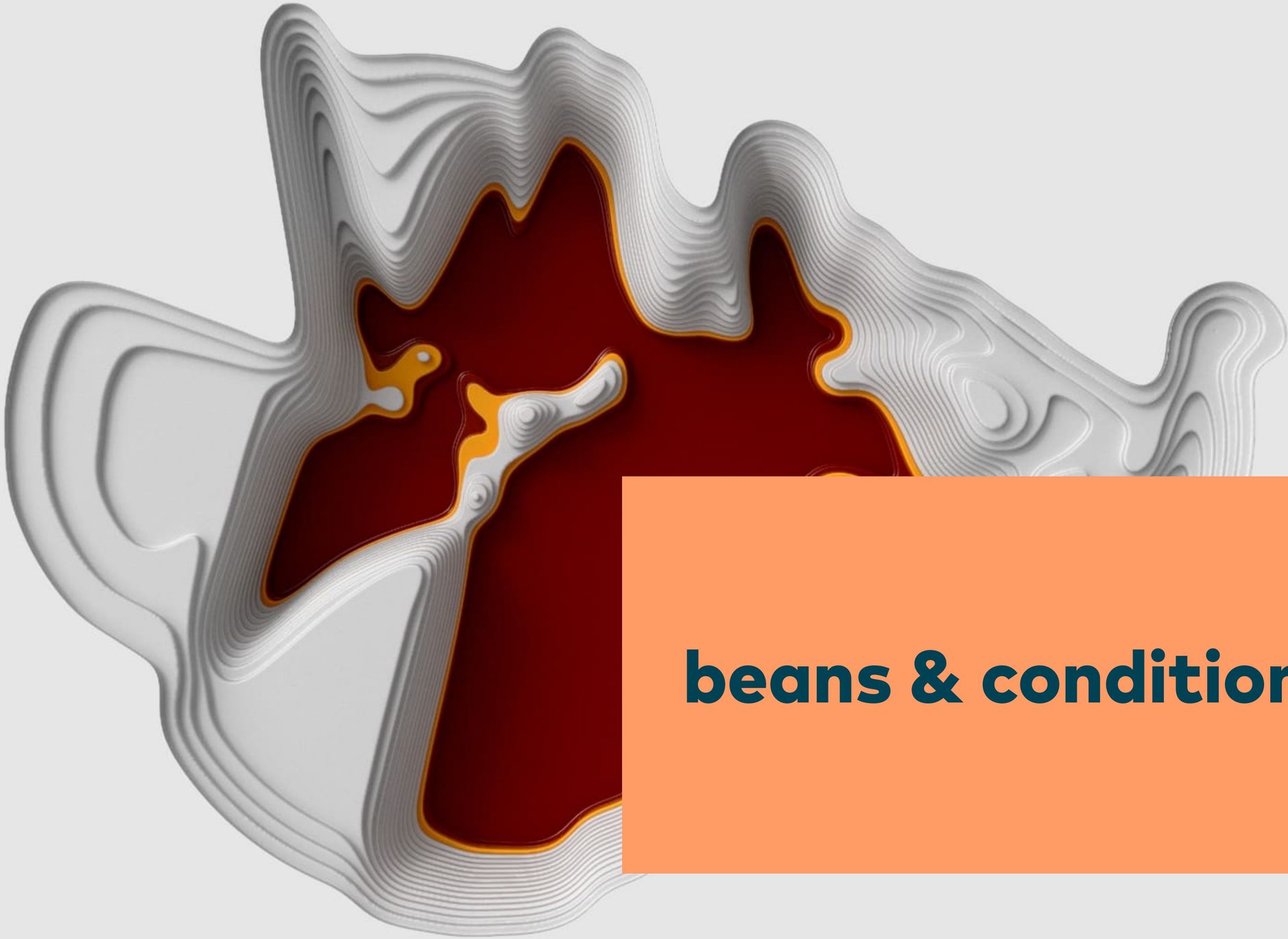
Exposes all variables

Sanitized Values

```
management.endpoint.env.show-values=always
```

```
@Bean
SanitizingFunction a() {
    return new SanitizingFunction() {
        @Override
        public SanitizableData apply(SanitizableData data) {
            if (data.getLowerCaseKey().equals("spring.application.pid")) {
                return data.withValue(42);
            }
            if (data.getLowerCaseKey().equals("spring.security.user.password")) {
                return data.withSanitizedValue();
            }

            return data;
        }
    };
}
```



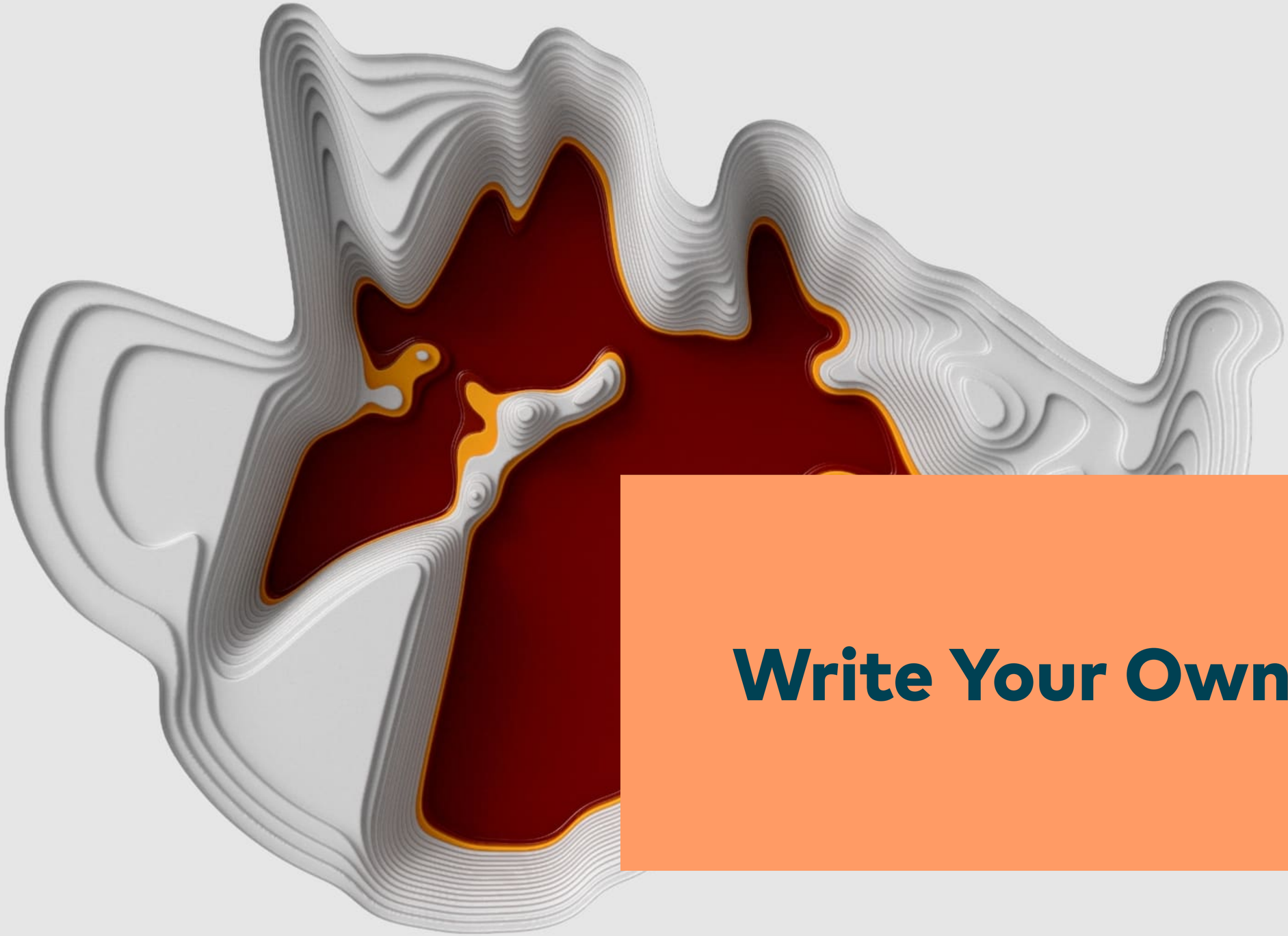
beans & conditions

beans

- Exposes all configured beans of your application
- Includes some detailed information where the bean is defined
- Useful for debugging when you wonder whether or not an expected bean is present

conditions

- Explains why AutoConfigurations are included or not
- Useful for debugging when you wonder why an AutoConfiguration is missing or when you think there is one to many



Write Your Own

Write your own

```
@Component
@Endpoint(id = "custom")
public class CustomEndpoint {

    @ReadOperation
    public String foo() {
        return "bar";
    }
}
```

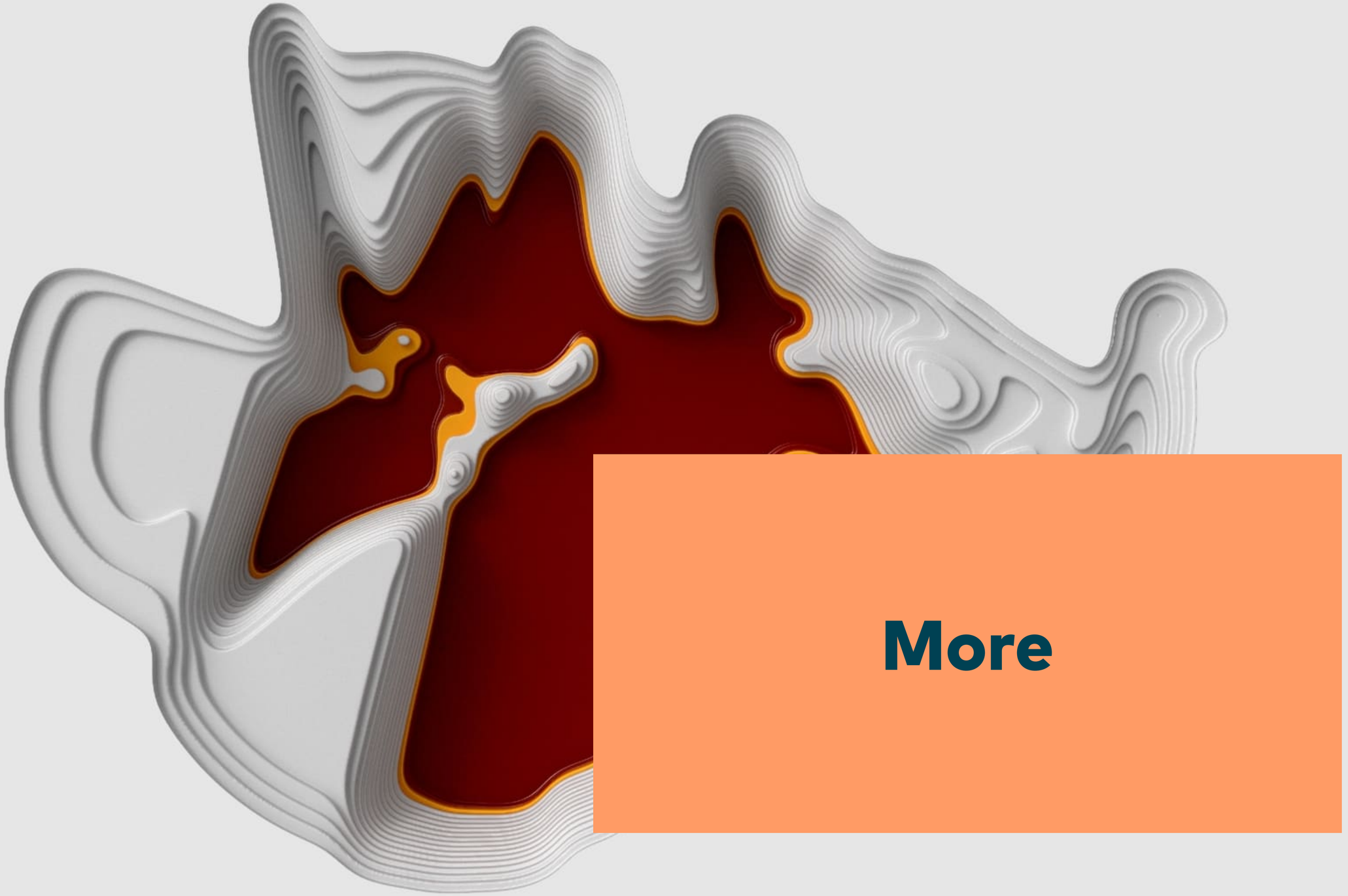
Extend with web capabilities

```
@Component
@EndpointWebExtension(endpoint = CustomEndpoint.class)
public class CustomWebEndpoint {

    private final CustomEndpoint delegatee;

    public CustomWebEndpoint(CustomEndpoint delegatee) {
        this.delegatee = delegatee;
    }

    @ReadOperation
    public Map<String, Object> bar() {
        return Map.of("value", delegatee.foo());
    }
}
```



More

Startup

```
var app = new SpringApplication(Application.class);
app.setApplicationStartup(new BufferingApplicationStartup(4096));
app.run(args);
```

Logger

```
{
  "configuredLevel": "WARN"
}
```

Audit Events

```
@Bean
AuditEventRepository auditEventRepository() {
    return new InMemoryAuditEventRepository();
}
```

Mappings

- Exposes all configured HTTP mappings of your application

SBOM

```
<plugin>
  <groupId>org.cyclonedx</groupId>
  <artifactId>cyclonedx-maven-plugin</artifactId>
</plugin>
```

HTTP Exchanges

```
@Bean
InMemoryHttpExchangeRepository httpExchangeRepository() {
    return new InMemoryHttpExchangeRepository();
}
```

Observability

Metrics

Spring Boot Actuator provides dependency management and auto-configuration for Micrometer, an application metrics facade that supports numerous monitoring systems, including:

- AppOptics
- Atlas
- Datadog
- Dynatrace
- Elastic
- Ganglia
- Graphite
- Humio
- Influx
- JMX
- KairosDB
- New Relic
- OTLP
- Prometheus
- Simple (in-memory)
- Stackdriver
- StatsD
- Wavefront

Traces

Supported Tracers

Spring Boot ships auto-configuration for the following tracers:

- OpenTelemetry with Zipkin, Wavefront, or OTLP
- OpenZipkin Brave with Zipkin or Wavefront

Thanks! Questions?

INNOQ
www.innoq.com

Michael Vitz

Mail	michael.vitz@innoq.com
Bluesky	@michael.vitz.me
Mastodon	@michaelvitz@innoq.social
LinkedIn	michaelvitz
X	@michaelvitz