SEGES
space for solutions

SPRING BOOT & MICROSERVICES

@ladislavGazo
gazo@seges.sk

PROBLEM
PROBLEM

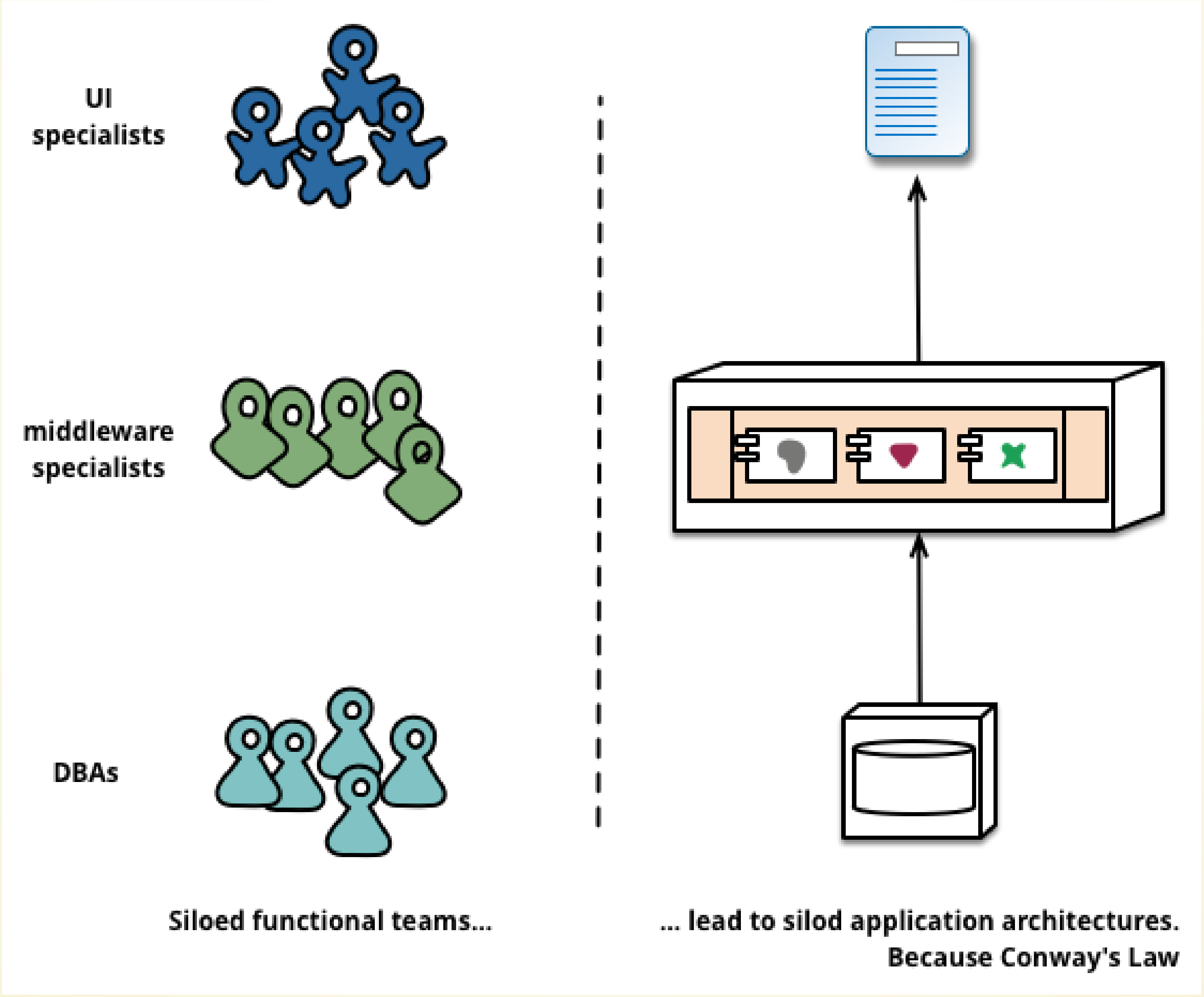
huge application servers

monolithic architecture

long restarts

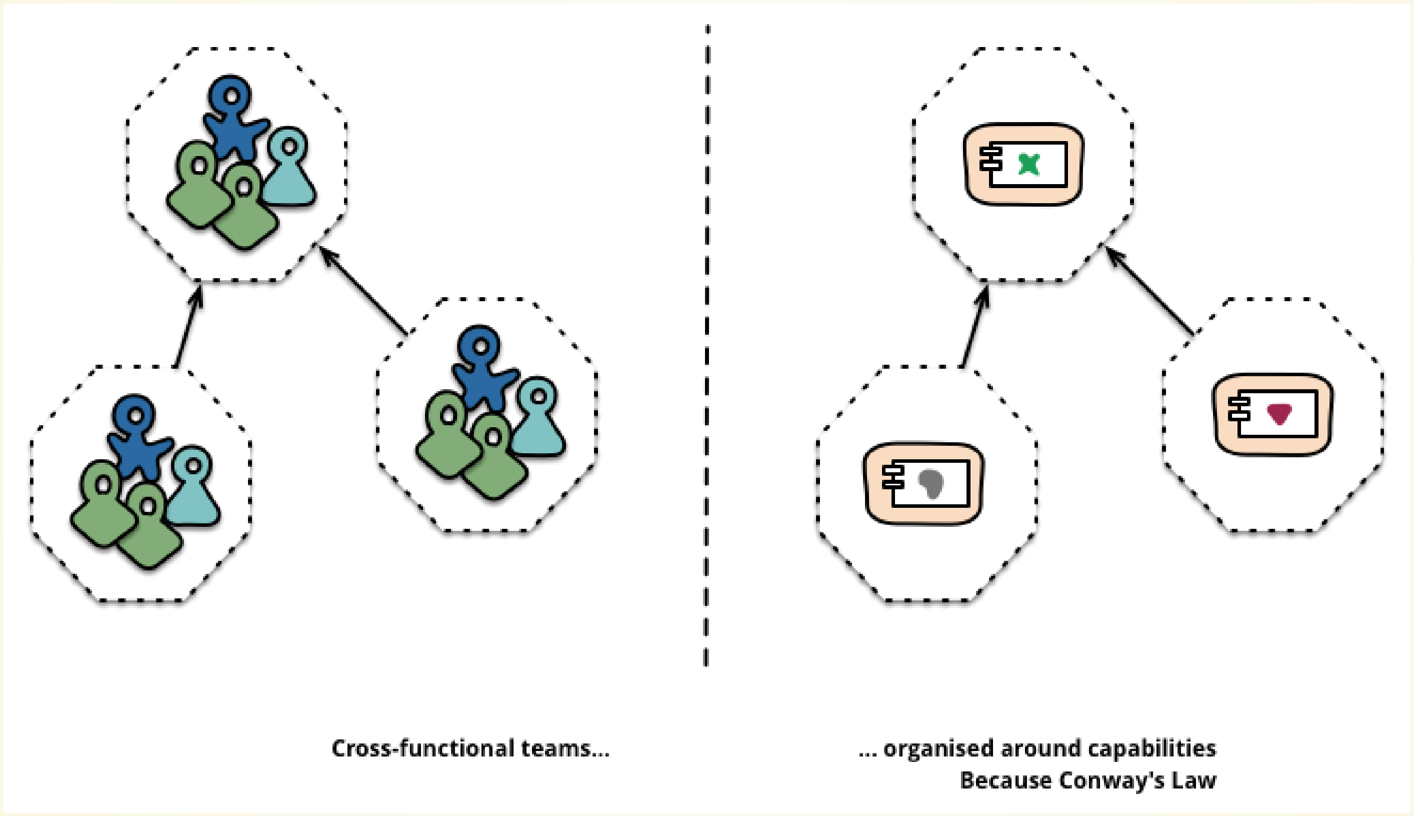
no save-reload cycle

wired to a set of dependencies


UI specialists
middleware specialists
DBAs
Siloed functional teams...
... lead to silod application architectures.
Because Conway's Law

KEEP
CALM
AND
SOLVE THE
PROBLEM

MICROSERVICES



PROS

Each microservice is relatively small

Easier for a developer to understand

- The IDE is faster making developers more productive
- The web container starts faster

Easier to scale development.

- Each team can develop, deploy and scale their service independently of all of the other teams.

PROS 2

Improved fault isolation

- In comparison, one misbehaving component of a monolithic architecture can bring down the entire system.

Each service can be developed and deployed independently

Eliminates any long-term commitment to a technology stack

BUT...

(there is always at least one...)

Developers must deal with the additional complexity of creating a distributed system

- testing
- transactions
- use-cases span multiple microservices

Deployment complexity In production

Increased memory consumption

... TACKLE WITH

when to split to multiple microservices

what should be inside one microservice

single responsibility principle

...

it is **the art**

SPRING BOOT

http://projects.spring.io/spring-boot/

opinionated way how to do modern applications

VERY EASY TO START

```
<parent>
    <groupid>org.springframework.boot</groupid>
    <artifactid>spring-boot-starter-parent</artifactid>
    <version>1.1.8.RELEASE</version>
</parent>
<dependencies>
    <dependency>
        <groupid>org.springframework.boot</groupid>
        <artifactid>spring-boot-starter-web</artifactid>
    </dependency>
</dependencies>
```

ta-daaaa .... web app built :)

APP CODE

```
package hello;

import org.springframework.boot.*;
import org.springframework.boot.autoconfigure.*;
import org.springframework.stereotype.*;
import org.springframework.web.bind.annotation.*;

@Controller
@EnableAutoConfiguration
public class SampleController {

    @RequestMapping("/")
    @ResponseBody
    String home() {
        return "Hello World!";
    }
```

... the beauty of autoconfiguration

RELOAD

TRANSFORMATIVE APPLICATIONS

3 OPTIONS

JRebel

jHipster Reloader

Spring Loaded

JHIPSTER RELOADER

-javaagent:/home/user/.m2/repository/io/github/jhipster/loaded/agent/0.12/agent-0.12.jar

```
<profiles>
  <profile>
    <id>reloaded</id>
      <dependencies>
        <dependency>
          <groupId>io.github.jhipster.loaded</groupId>
          <artifactId>agent</artifactId>
          <version>0.12</version>
          <exclusions>
            <exclusion>
              <groupId>org.springframework</groupId>
              <artifactId>springloaded</artifactId>
            </exclusion>
          </exclusions>
        </dependency>

        <dependency>
```

INTERESTING DEPENDENCIES

benefit of Spring Boot

- MVC
- data access
- templating
- security
- ...

... anything from the Spring stack

REST

@RestController

based on Spring MVC

TEMPLATING

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

using Thymeleaf

SPRING SECURITY

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

@EnableWebMvcSecurity

@EnableGlobalMethodSecurity

```
http.authorizeRequests().antMatchers("/identity/login", "/page/**", "/client/*

http.csrf().disable();

        http.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.S
http.addFilterBefore(new AuthenticationTokenFilter(identityService), Anonymous
http.exceptionHandling().authenticationEntryPoint(new UnauthorizedEntryPoint()
```

MIGRATIONS

```
<dependency>
        <groupId>org.flywaydb</groupId>
        <artifactId>flyway-core</artifactId>
</dependency>
```

migration files

V1.1__init.sql
V1.2__data.sql
...

DATA ACCESS

```
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
</dependency>
```

```
<dependency>
        <groupId>org.postgresql</groupId>
        <artifactId>postgresql</artifactId>
</dependency>
```

autodetection of the driver

DATA ACCESS II.

```
spring.datasource.url: jdbc:postgresql://localhost/mydb
spring.datasource.username: myuser
spring.datasource.password: mypass
spring.datasource.driverClassName: org.postgresql.Driver
```

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

CONFIGURATION

driven mainly by:

Spring Java Configuration

Spring Annotations

Spring Profiles

simple YAML / properties file

CHERRY

```java
@Component
@ConfigurationProperties(prefix="connection")
public class ConnectionSettings {

    private String username;

    private InetAddress remoteAddress;

    // ... getters and setters

}
```

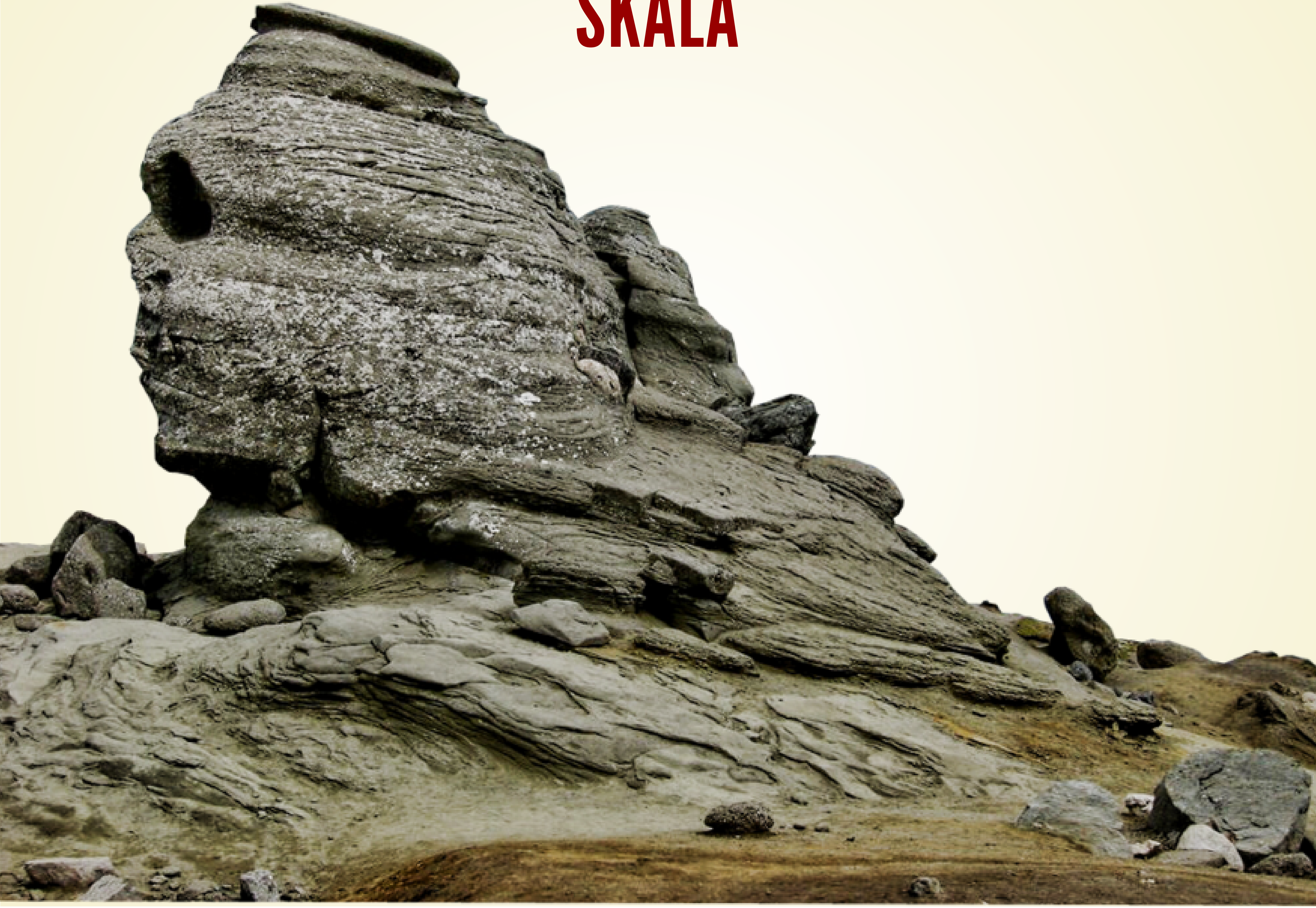
```yaml
# application.yml

connection:
    username: admin
    remoteAddress: 192.168.1.1
```

KONTANJERE

premier box
BSIU 940247 7
45G1
M.G.W.
TARE
NET
CU.CAP.
CAUTION HIGH TRAILER
PBXU 222926 0
22G1

SKALA



GITER8 TEMPLATE

https://github.com/lgazo/scala-boot.g8

```
curl https://raw.githubusercontent.com/n8han/conscript/master/setup.sh | sh
# add to your path ~/bin
cs n8han/giter8

g8 lgazo/scala-boot
```

QUESTIONS?

@ladislavGazo
gazo@seges.sk

THANK YOU... FOR...

ATTENTION