# Spring 4.0 - Evolution or Revolution

Dominique Bartholdi, Raffael Schmid
Trivadis AG

"work with Java 8 and other JVM languages (e.g. support Lambdas, ...)"

"responding to, and setting trends in (Developer Productivity), Big Data, Cloud, REST and Micro Service Architecture"

-Adrian Colyer, CTO Application Fabric at Pivotal

# Agenda

- ☐ INTRODUCTION
- ☐ MODERNISATION
- ☐ BIG DATA
- ☐ MICRO SERVICE ARCHITECTURE
- ☐ CONCLUSION

# Introduction

- INTRODUCTION
- MODERNISATION
- BIG DATA
- MICRO SERVICE ARCHITECTURE
- CONCLUSION

# a bunch of history

Spring 1.0:

- "lightweight" container (DI)
- AOP interception
- Jdbc abstraction, Transaction support
- JPetStore, Petclinic

Spring 2.0:

- bean configuration dialects (make common tasks easier)

Spring 2.5:

- reduce XML based configuration
- custom namespaces

Spring 3.0:

- task scheduling, ... with annotation support
- REST web apps
- JEE features

Spring 4.0 / Spring IO (Yummy Noodle Bar)

2004 2006 2007 2009 2013

# Platform Overview

## Execution

Spring XD

Spring Boot

Grails

## Workload Types

Integration

Batch

Web

Big Data

## Data

Relational

Non-Relational

## Core

Framework

Spring 4.0

Security


Groovy

Reactor

Modernisation

Big Data

Micro Services

(Developer Productivity)

# Spring 4.0

- [x] INTRODUCTION
- [ ] MODERNISATION
- [ ] BIG DATA
- [ ] MICRO SERVICE ARCHITECTURE
- [ ] CONCLUSION

# Java 8 Lambdas

## with Spring's JdbcTemplate

```
JdbcTemplate jt = new JdbcTemplate();

jt.query(QUERY, new PreparedStatementSetter() {

      @Override
      public void setValues(PreparedStatement ps) throws SQLException {
        ps.setString(1, "Sales");
      }

    }, new RowMapper<Person>() {

    @Override
    public Person mapRow(ResultSet rs, int rowNum) throws SQLException {
       return new Person(rs.getString(1), rs.getInt(2));
    }
});
```

# Java 8 Lambdas

## with Spring's JdbcTemplate

```
JdbcTemplate jt = new JdbcTemplate(dataSource);

jt.query( QUERY,
        ps -> ps.setString(1, "Sales"),
        (rs, row) -> new Person(rs.getString(1), rs.getInt(2))
);
```

# JSR 310:
# Date & Time

```
public class Customer {

    @DateTimeFormat(iso=ISO.DATE) //default
    private LocalDate birthDate;

    @DateTimeFormat(pattern="M/d/yy h:mm")
    private LocalDateTime lastContact;


    ...
}
```

# Generics-based Injection Matching

```
@Service
public class BookService {

    @Autowired
    public BookService(Repository<Book> repo) {

        ...

    }

}
```

# Generics-based Injection Matching

```
@Bean
public Repository<Book> bookRepository() {

  return new BookRepositoryImpl();

}
```

# Web Sockets

```
@Configuration
@EnableWebSocket
public class MyWebSocketConfig implements WebSocketConfigurer {

  public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {

      WebSocketHandler echoHandler = new EchoHandler();
      registry.addHandler(echoHandler, "/echo").withSockJS();

  }

}

public interface WebSocketHandler {

  void handleMessage(WebSocketSession s, WebSocketMessage<?> m);

}
```

# and many more...

- Conditional Bean definitions
  @Conditional
- @Autowired @Lazy on injection points
- @Order injection of arrays and lists
- Groovy style Spring configs

# BIG DATA

- [x] INTRODUCTION
- [x] MODERNISATION
- [ ] BIG DATA
- [ ] MICRO SERVICE ARCHITECTURE
- [ ] CONCLUSION

# Spring XD

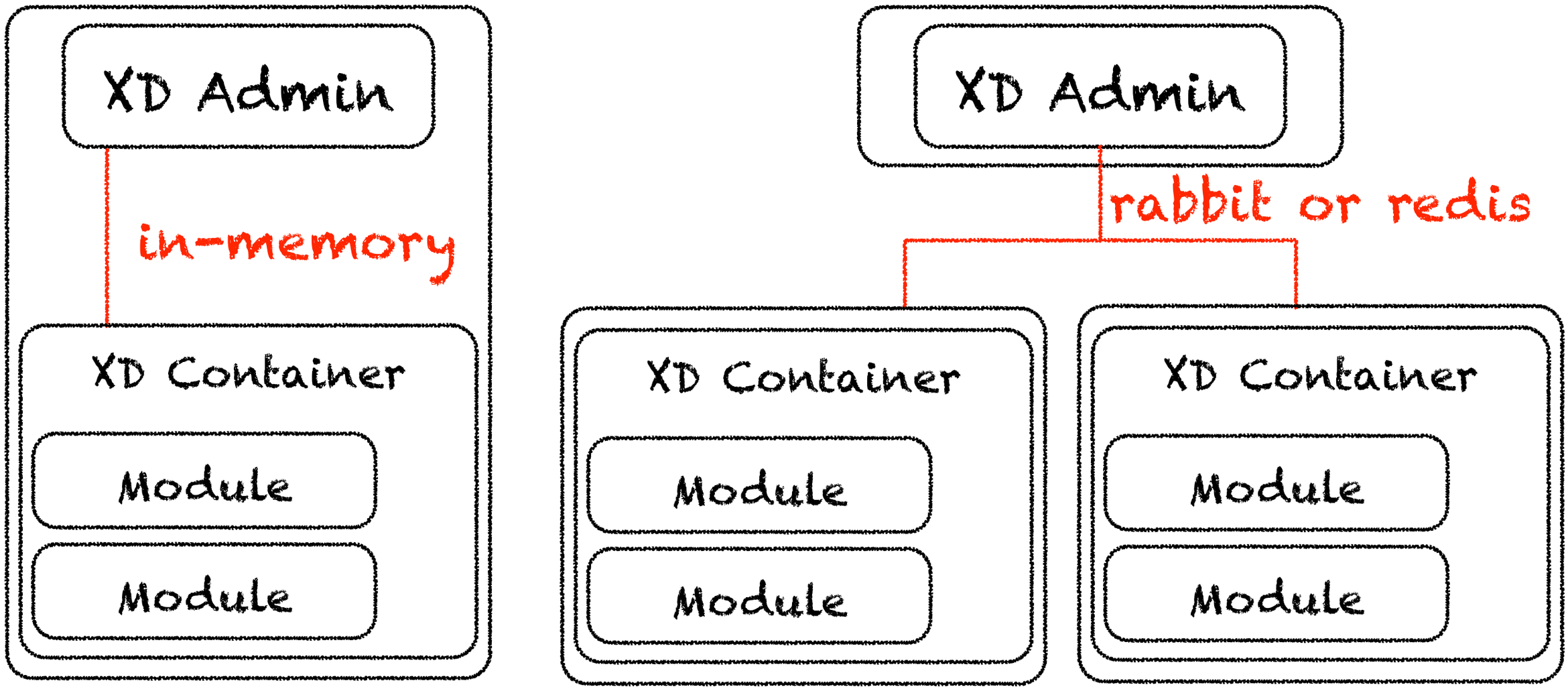
"Spring XD (Extreme Data) is a unified, distributed, and extensible service for data ingestion, real time analytics, batch processing, and data export."

-Pivotal

# Spring XD

- Combining established Spring Technology: Spring Integration, Spring Batch and Spring Data
- Providing a scalable container architecture
- Domain Specific Language (DSL) for configuration

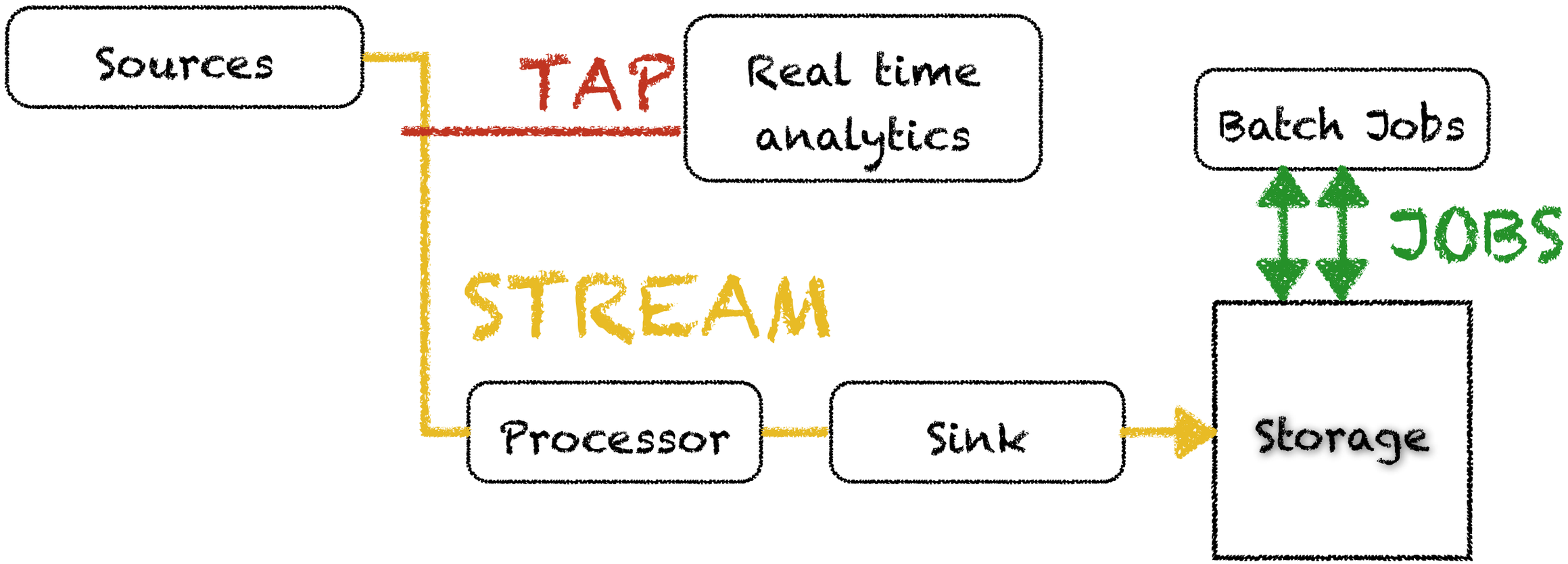
# Single vs Distributed




single node

distributed nodes

# from source to sink



# Sources

- HTTP
- Tail
- File
- Mail
- Twitter Search
- Twitter Stream
- Gemfire
- Gemfire CQ
- Syslog
- TCP
- TCP Client
- JMS
- RabbitMQ
- Time
- MQTT

# Stream Config

- Create the stream

  xd:> stream create --definition "HTTP | file"
  --name mystream

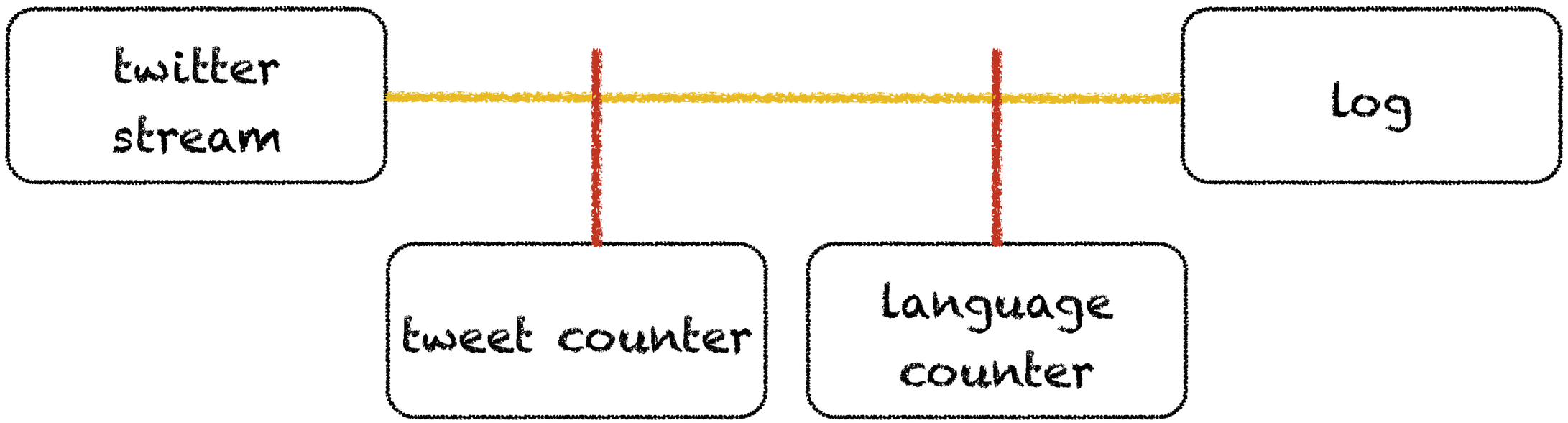
  - use pipe (|) to connect source "HTTP" to sink "file"

  - set name of stream to "mystream"

- Remove the stream

  xd:> stream destroy –name mystream

# Spring XD Demo




```
stream create tweets --definition
      "twitterstream | log"

stream create tweetcount --definition
      "tap:stream:tweets > aggregatecounter"

stream create tweetlang  --definition
      "tap:stream:tweets > field-value-counter --fieldName=lang"
```

# MICRO SERVICE ARCHITECTURE

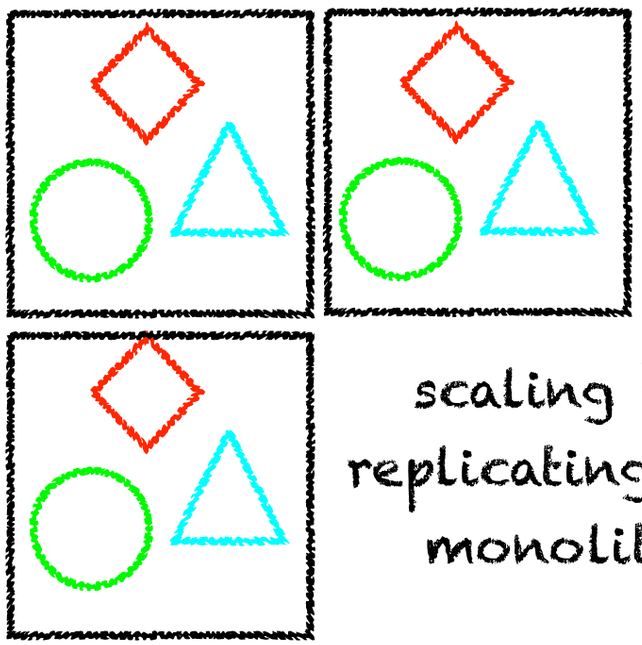
- [x] INTRODUCTION
- [x] MODERNISATION
- [x] BIG DATA
- [ ] MICRO SERVICE ARCHITECTURE
- [ ] CONCLUSION

# What's exactly a MICRO SERVICE ARCHITECTURE?

- Develop a single application as suite of small services
- Organise services around business capabilities (not technologies)
- Smart endpoints, dumb pipes (less BPEL, more REST)
- Services do one thing (small enough to throw away, fit into the head)

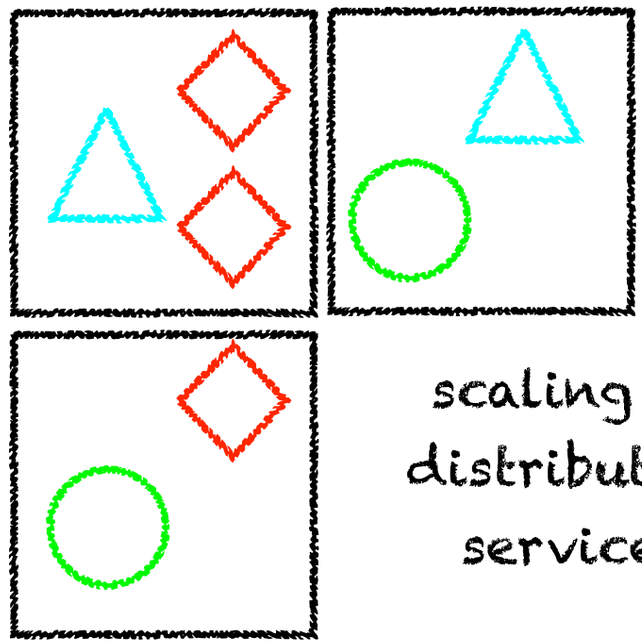
# MICRO SERVICE vs. monolithic architecture

functionality in one single process



scaling by replicating the monolith

a micro architecture puts each functionality into a service



scaling by distributing services

http://martinfowler.com/articles/microservices.html

# Objectives for MICRO SERVICES

- simple packaging (executable jar) -> embedded web containers
- monitoring, expose insights
- fast instantiation
- polyglot persistence (SQL, NoSQL)
- ...

# Spring Boot comes into play

Spring Boot makes it easy to create stand-alone (Spring based) applications that you can "just run"

- Radically faster getting started experience
  - Well defined set of dependencies
  - Auto-configuration
- "Non-functional" features out of the box
  - Metrics, Counters, Monitoring with Actuator
  - SSH access through remote shell

# Spring Boot by Example

# Spring Boot Conclusion

- Not production-ready yet (version 1.0.0 still RC 5), but surprisingly stable
- Packaging not only JAR
- Will influence a bunch of other Spring Projects (e.g. Spring XD, Roo, etc.)
- Perfect tool for quickly building Spring based applications

# CONCLUSION

- [x] INTRODUCTION
- [x] MODERNISATION
- [x] BIG DATA
- [x] MICRO SERVICE ARCHITECTURE
- [ ] CONCLUSION

# Overall

- Spring 4.0 evolved over the last years
- Spring IO most probably a revolutionary approach (not only in terms of marketing)

# Questions

Dominique Bartholdi
email: dominique.bartholdi@trivadis.com

Raffael Schmid
email: raffael.schmid@trivadis.com