Spring Cloud Data Flow

Cape Town JUG, July 2017

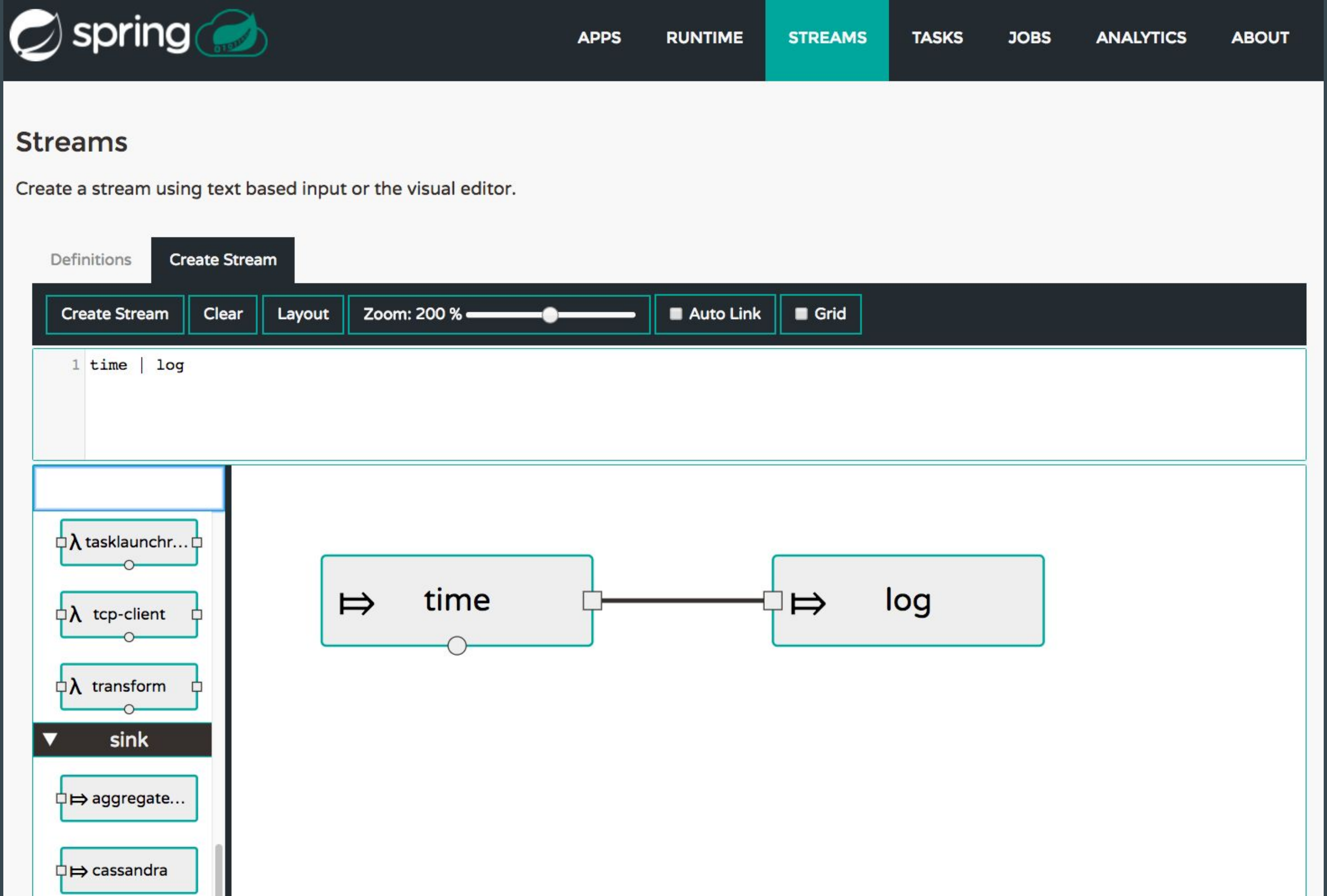
•••

@aminggs

The Vision

Stream-based microservices with the simplicity of Unix pipes

```
$ source | processor | sink
```



Streams

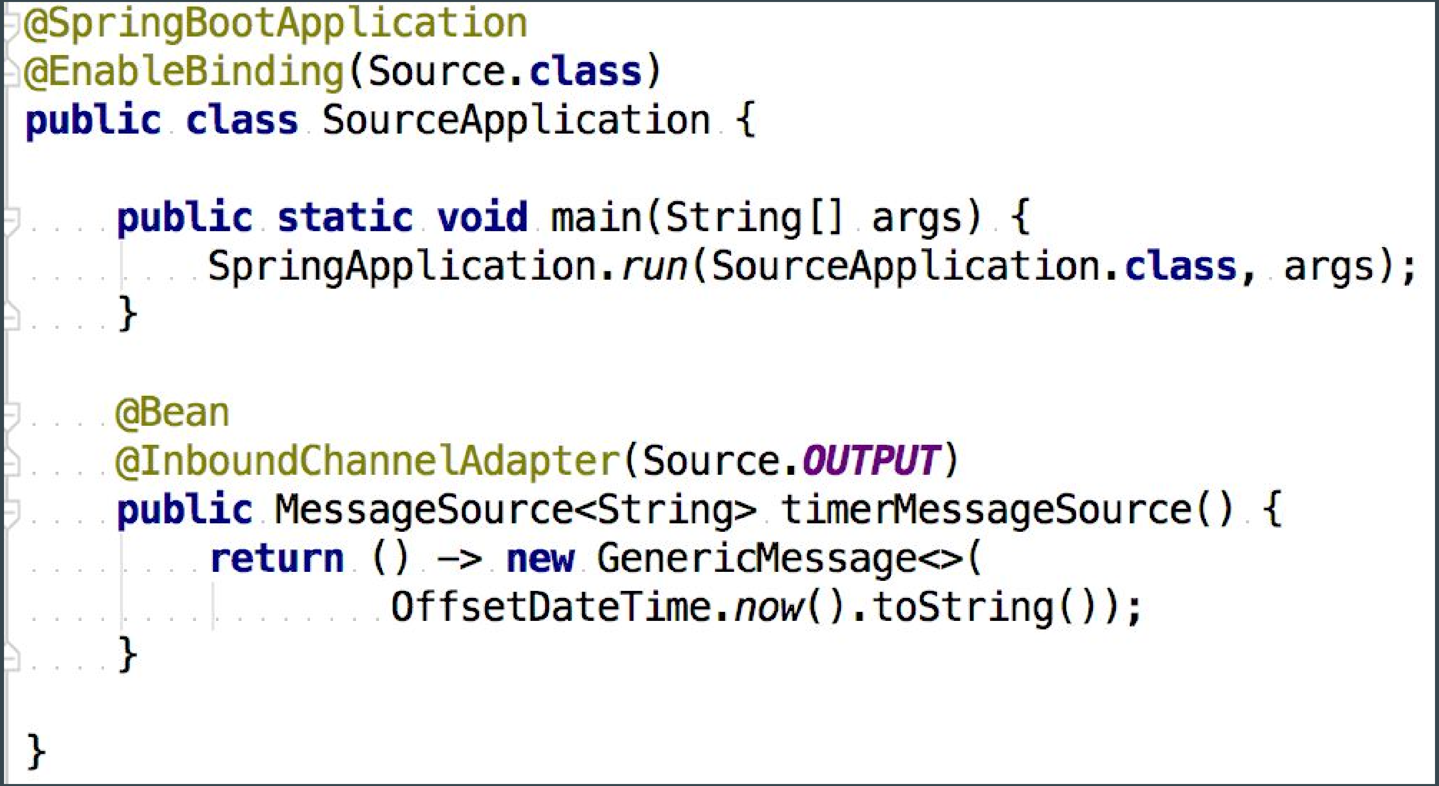
Create a stream using text based input or the visual editor.

```
time | log
```

Building an SCDF stream

- Source
- Processor
- Sink

Source



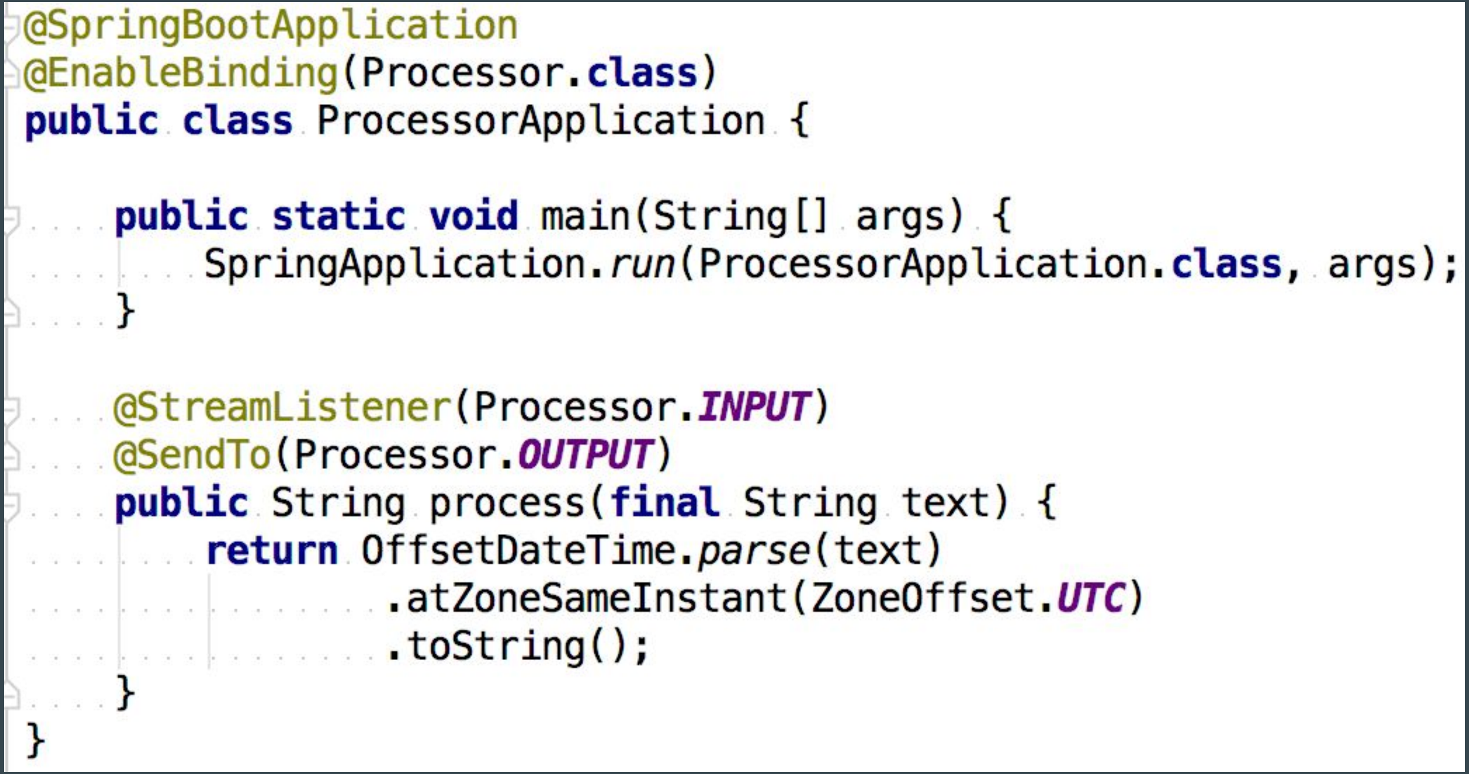
```java
@SpringBootApplication
@EnableBinding(Source.class)
public class SourceApplication {

    public static void main(String[] args) {
        SpringApplication.run(SourceApplication.class, args);
    }

    @Bean
    @InboundChannelAdapter(Source.OUTPUT)
    public MessageSource<String> timerMessageSource() {
        return () -> new GenericMessage<>(
                OffsetDateTime.now().toString());
    }

}
```

Processor

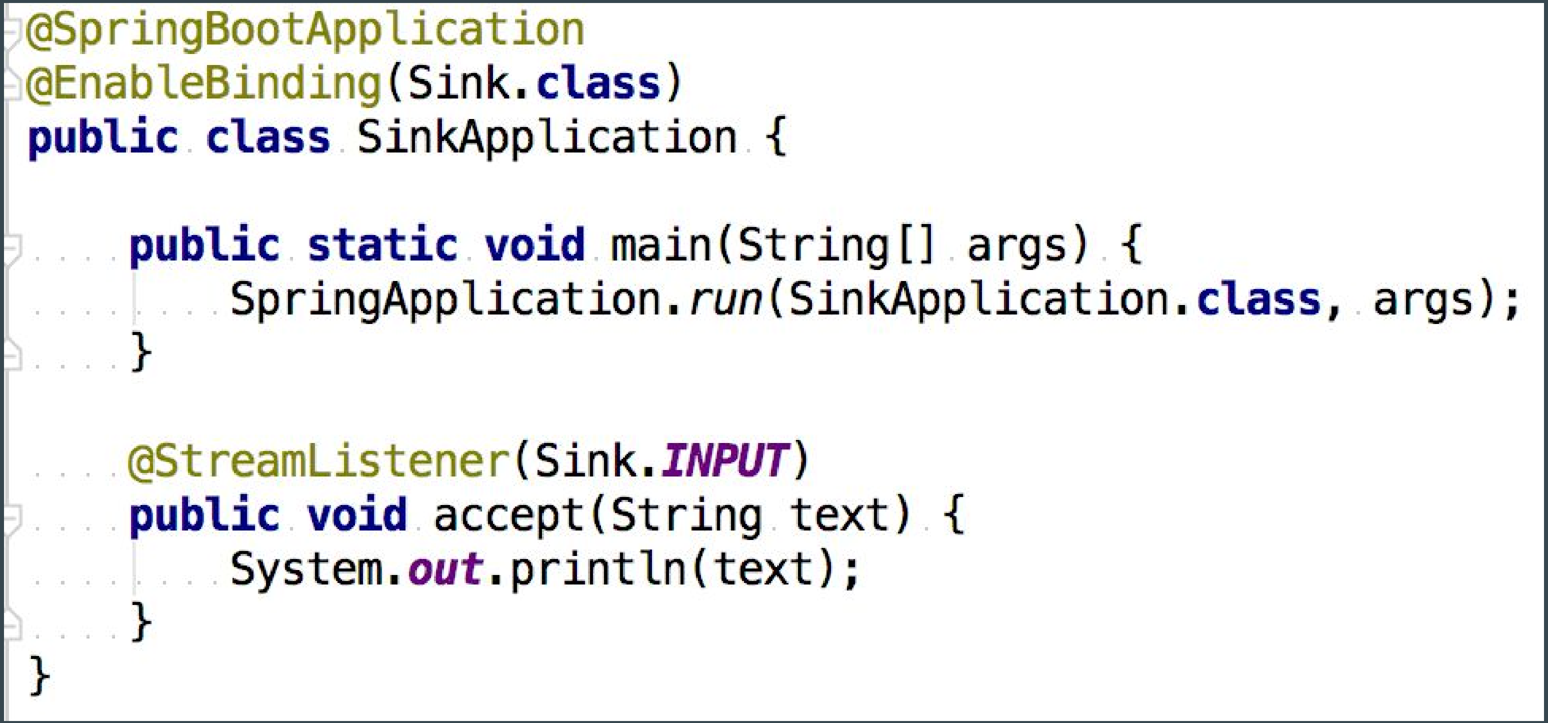


```java
@SpringBootApplication
@EnableBinding(Processor.class)
public class ProcessorApplication {

    public static void main(String[] args) {
        SpringApplication.run(ProcessorApplication.class, args);
    }

    @StreamListener(Processor.INPUT)
    @SendTo(Processor.OUTPUT)
    public String process(final String text) {
        return OffsetDateTime.parse(text)
                .atZoneSameInstant(ZoneOffset.UTC)
                .toString();
    }
}
```

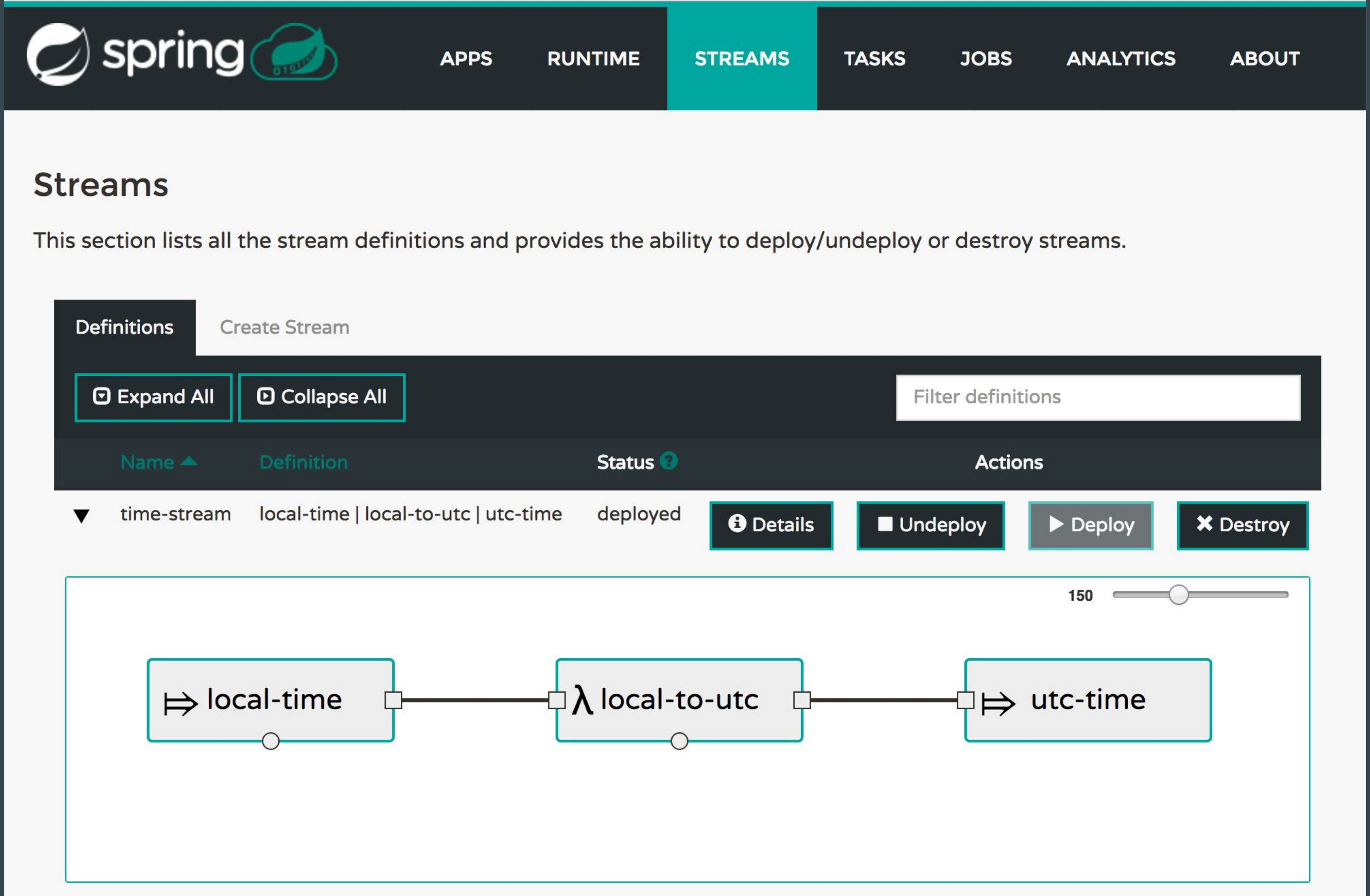
Sink



```java
@SpringBootApplication
@EnableBinding(Sink.class)
public class SinkApplication {

    public static void main(String[] args) {
        SpringApplication.run(SinkApplication.class, args);
    }

    @StreamListener(Sink.INPUT)
    public void accept(String text) {
        System.out.println(text);
    }
}
```



Streams

This section lists all the stream definitions and provides the ability to deploy/undeploy or destroy streams.

Name	Definition	Status	Actions
▼ time-stream	local-time | local-to-utc | utc-time	deployed	Details Undeploy Deploy Destroy

What makes up the stack?

- Spring Cloud Data Flow Server
- REST APIs / Shell / DSL / Dashboard UI
- Spring Cloud Data Flow – Core
- Spring Cloud Stream (Binders)
- Spring Cloud Deployer (One per platform)

Binders (Spring Cloud Stream)

Official

- Kafka
- RabbitMQ

Community

- IBM MQ

Deployers (Spring Cloud Data Flow)

Official

- CloudFoundry
- Mesos
- Kubernetes

Community

- Nomad
- OpenShift

Getting to Production

- Microservices
- CI/CD

Microservices

- Config server (Spring Cloud)
- Discovery server (Eureka)
- Log aggregation (Graylog)
- Distributed tracing (Sleuth)
- Monitoring (New Relic)

CI/CD

- Bitbucket : Merge PR → Git master
- Bamboo : Build → Nexus (Maven Repo)
- Bamboo : Build → Docker Registry
- Bamboo : Deploy → SCDF Server (via Maven plugin)
- SCDF Server → Nomad → Docker Registry

Gotchas

The simplicity of a processor method taking an input and returning an output breaks down when

- There is no output for the given input
- Exceptions are thrown
- The choice of output channel depends on the input

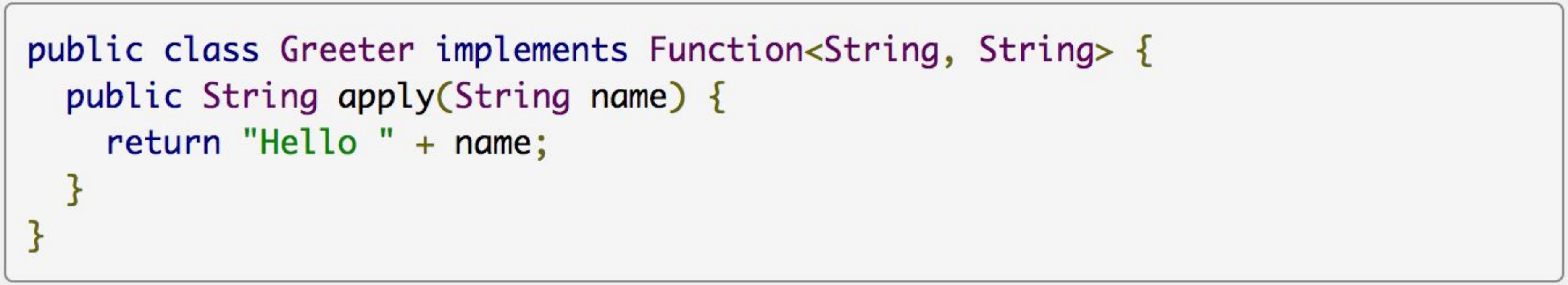
Other considerations

- Automatic retries are not always what you want (default: 3)
- Topic/Queue names

Spring Cloud Function

Classes from java.lang.function

- Function
- Consumer
- Supplier



```java
public class Greeter implements Function<String, String> {
  public String apply(String name) {
    return "Hello " + name;
  }
}
```

Decoupling Business Logic from Deployment Profiles

- The business logic in this case refers to functions, while the deployment profile could be a REST app, stream processing app, or finite task
- Spring Cloud Function provides a JAR for each of those types

Moving Up the Abstraction Ladder

Just as Spring Cloud Stream provides a Binder abstraction that eliminates the need to define Channel Adapters, Spring Cloud Function eliminates the need to declare components like Service Activators, Transformers, or even the @StreamListener-annotated methods to which Spring Cloud Stream delegates.

Demo

```
$ cd ~/src/ctjug2017/bin

$ java -jar spring-cloud-dataflow-server-local-1.2.2.RELEASE.jar

$ cd /usr/local/Cellar/rabbitmq/3.6.9_1/sbin

$ ./rabbitmq-server
```

http://localhost:9393/dashboard/index.html

Resources

- Spring Cloud Data Flow http://cloud.spring.io/spring-cloud-dataflow/
- Spring Cloud Stream http://cloud.spring.io/spring-cloud-stream/
- Introducing Spring Cloud Function https://spring.io/blog/2017/07/05/introducing-spring-cloud-function