Week 4: Service-to-Service Communication

# Unit 4: Calling the User Service via Hystrix

Run Simple

open.sap.com

# Calling the User Service via Hystrix

Resilience

**Definition of resilience:**

the ability of a system to handle unexpected situations

- without the user noticing it (best case)
- with a graceful degradation of service (worst case)

**Fallback approach for graceful degradation:**

which result to use if the call fails?

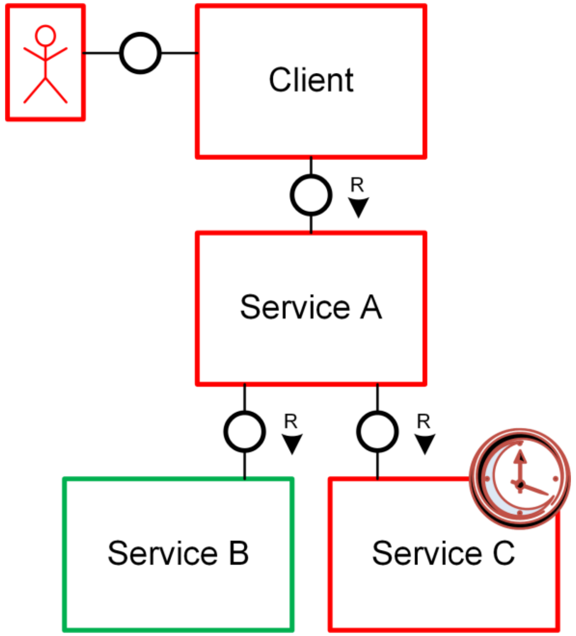
- use cached (potentially outdated) value
- use sensible default
- try second-best alternative (e.g. apply to message queue)

# Calling the User Service via Hystrix

## Challenges

Service dependencies introduce latency and other points of failure

- What if called service does not answer (in time)?
- How to avoid cascading failures / latency?
- How to avoid flooding server after restart?



### A Typical Problem Case

A customer creates an order. This triggers the order creation, and if it is successful, the warehouse system is informed for dispatching the order.

But: What should happen if the call to the warehouse system fails?

# Calling the User Service via Hystrix

Hystrix – Resilience library

## Hystrix

Easy to start with

- Many fault-tolerance patterns implementable
  - Fail fast / silent
  - Circuit breaker pattern
  - Load shedding (thread pool)
  - Advanced: request caching
- Many configuration options

**Load shedding**
requests are rejected under certain conditions

Hystrix Wiki

# Calling the User Service via Hystrix

## HystrixCommand

Wrap all potentially failing calls in a **HystrixCommand**

```
public class MyCommand extends HystrixCommand<String> {

    public MyCommand() {
        super(HystrixCommandGroupKey.Factory.asKey("ExampleGroup"));
    }

    @Override
    protected String run() {
        // creates  client, sends  request,  handles response
        return callGetUserService(id);
    }
}
```

Note: Each HystrixCommand is executed within a separate thread and is timed out automatically after 1000ms by default.

# Calling the User Service via Hystrix

## HystrixCommand – Execution patterns

### Synchronous execution

```
String string = new MyCommand().execute();
```

### Asynchronous execution

```
Future<String> future = new MyCommand().queue();
String string = future.get(); // this blocks, consider future.isDone()
```

### **Reactive execution** – inverts control flow (IoC)

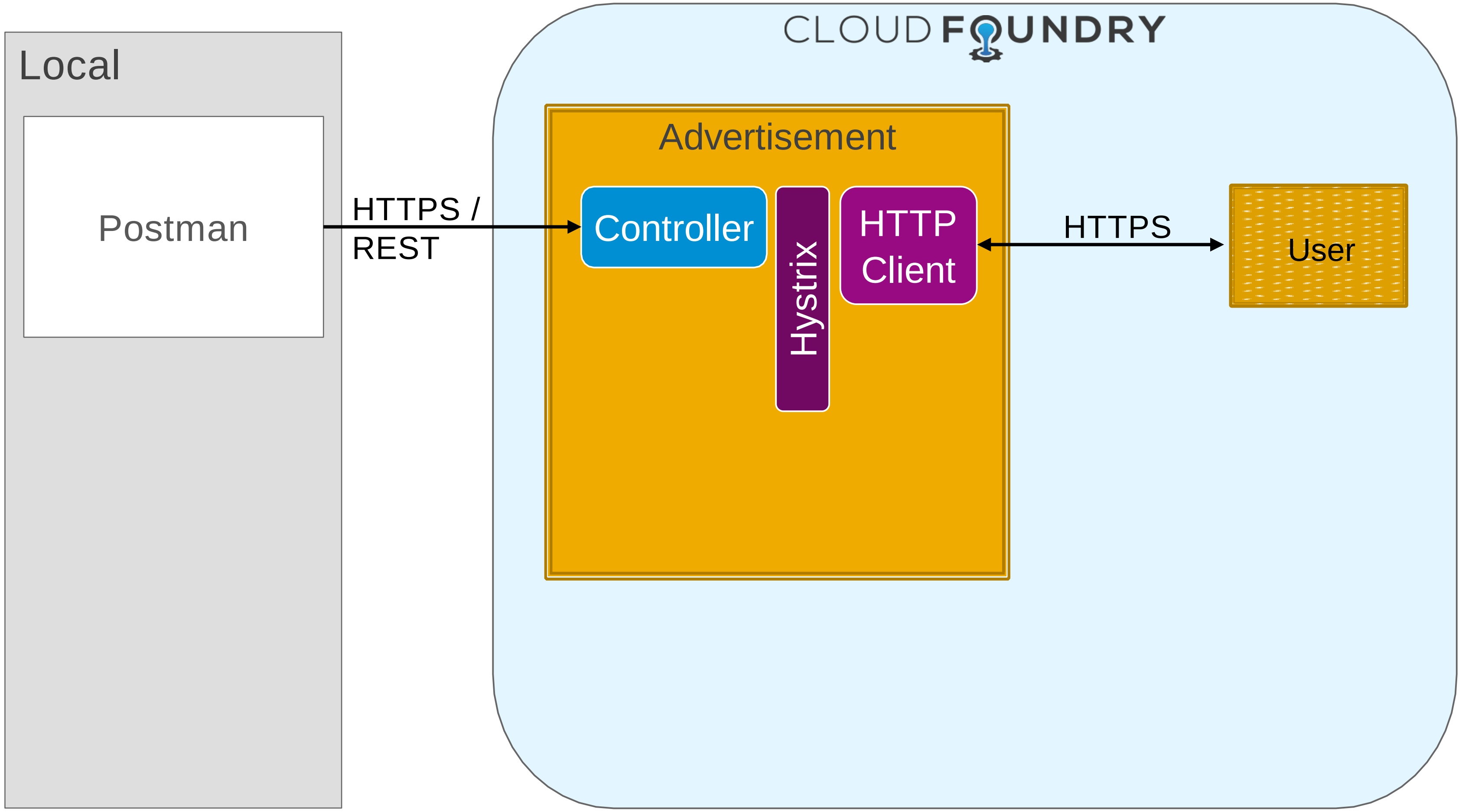
```
Observable<String> observable = new MyCommand().observe();
observable.subscribe( new Observer<String>() {
            public void onError(Throwable e) { /* observed call encounters issue */ }
            public void onNext(String v) { /* observed call emits data */ }
            public void onCompleted() { /* after the  last onNext() call */ }
});
```

mechanism
to process data streams
(out of scope for this course)

# Calling the User Service via Hystrix

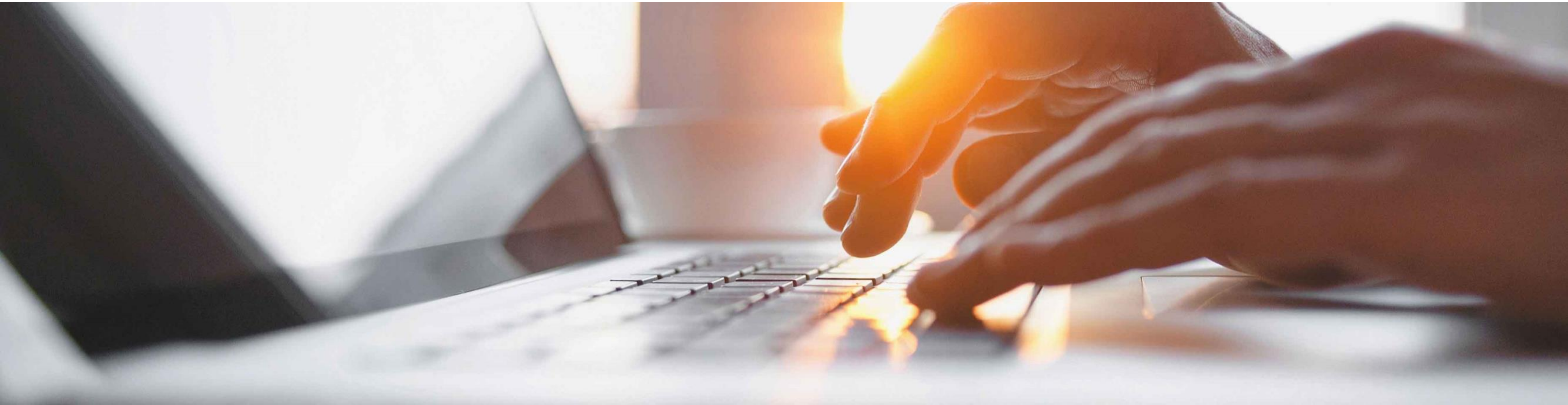
Exercise 17

[Exercise 17:
Introduce Hystrix]



# Calling the User Service via Hystrix

Review of sample solution



# Calling the User Service via Hystrix

Further reading

i Additional Material

- Hystrix Wiki

# Calling the User Service via Hystrix

What you’ve learned in this unit

- Challenges in service-to-service communication
- What resilience is
- Hystrix
  - What it is
  - Why we use it
  - How to use it
- How to call the user service via Hystrix

# Thank you.

**Contact information:**

**open@sap.com**

Run Simple

openSAP
open.sap.com

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.