

How Small Can Java Microservices Be?

Eberhard Wolff
Fellow, innoQ
@ewolff

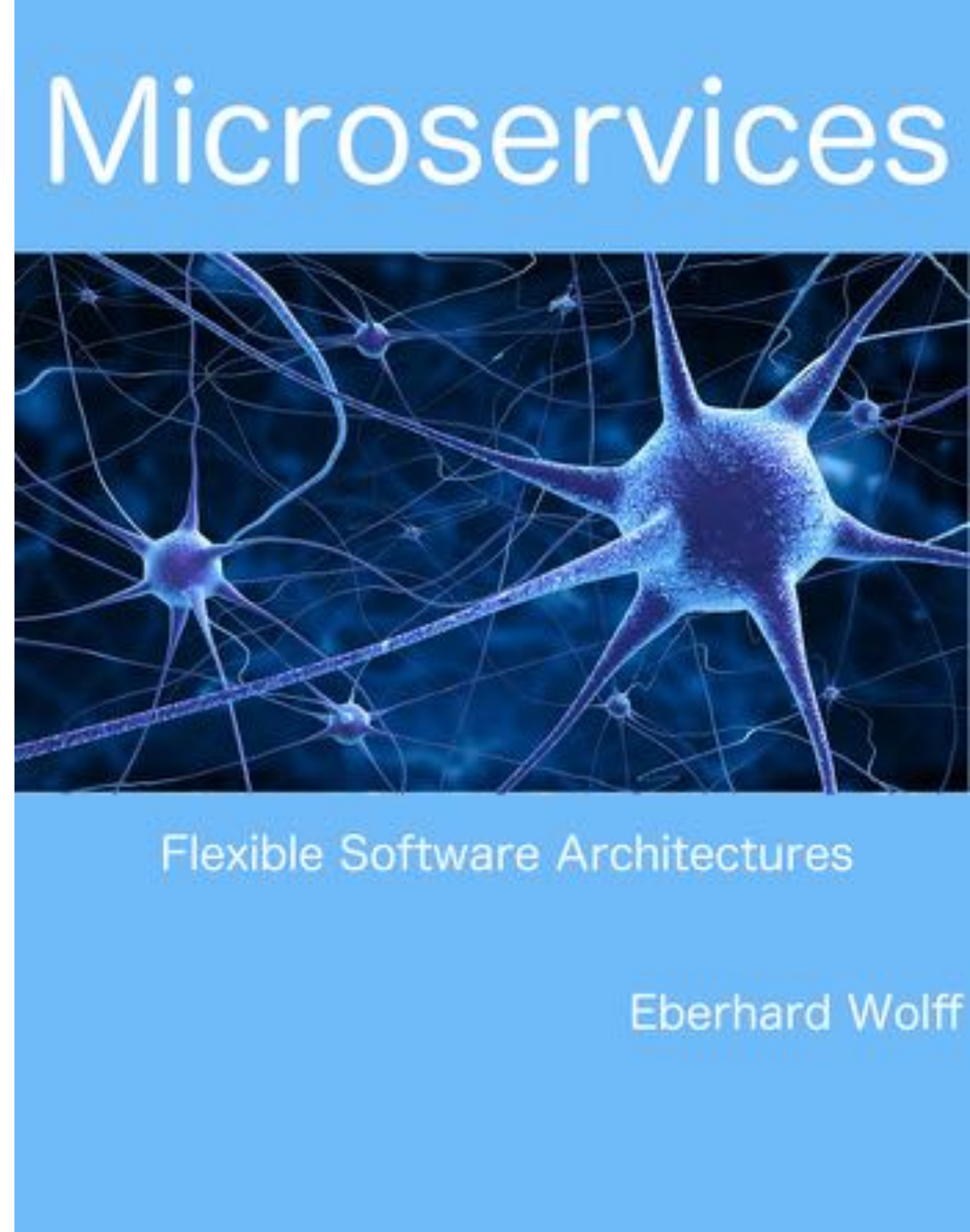




<http://continuous-delivery-buch.de/>



<http://microservices-buch.de/>



<http://microservices-book.com/>



Eberhard Wolff

Microservices Primer

A Short Overview

FREE!!!!

innoQ

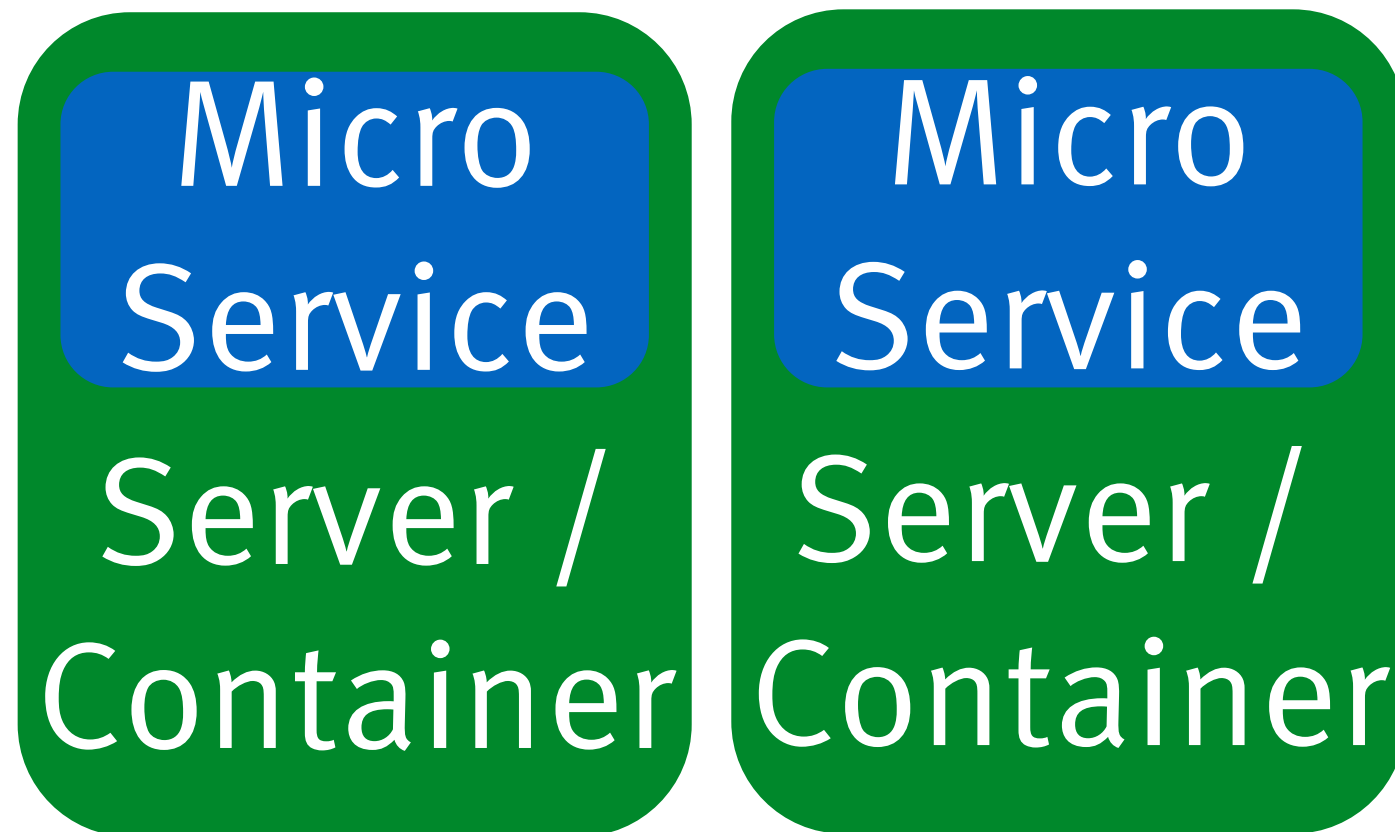
<http://microservices-book.com/primer.html>

Microservice Definition

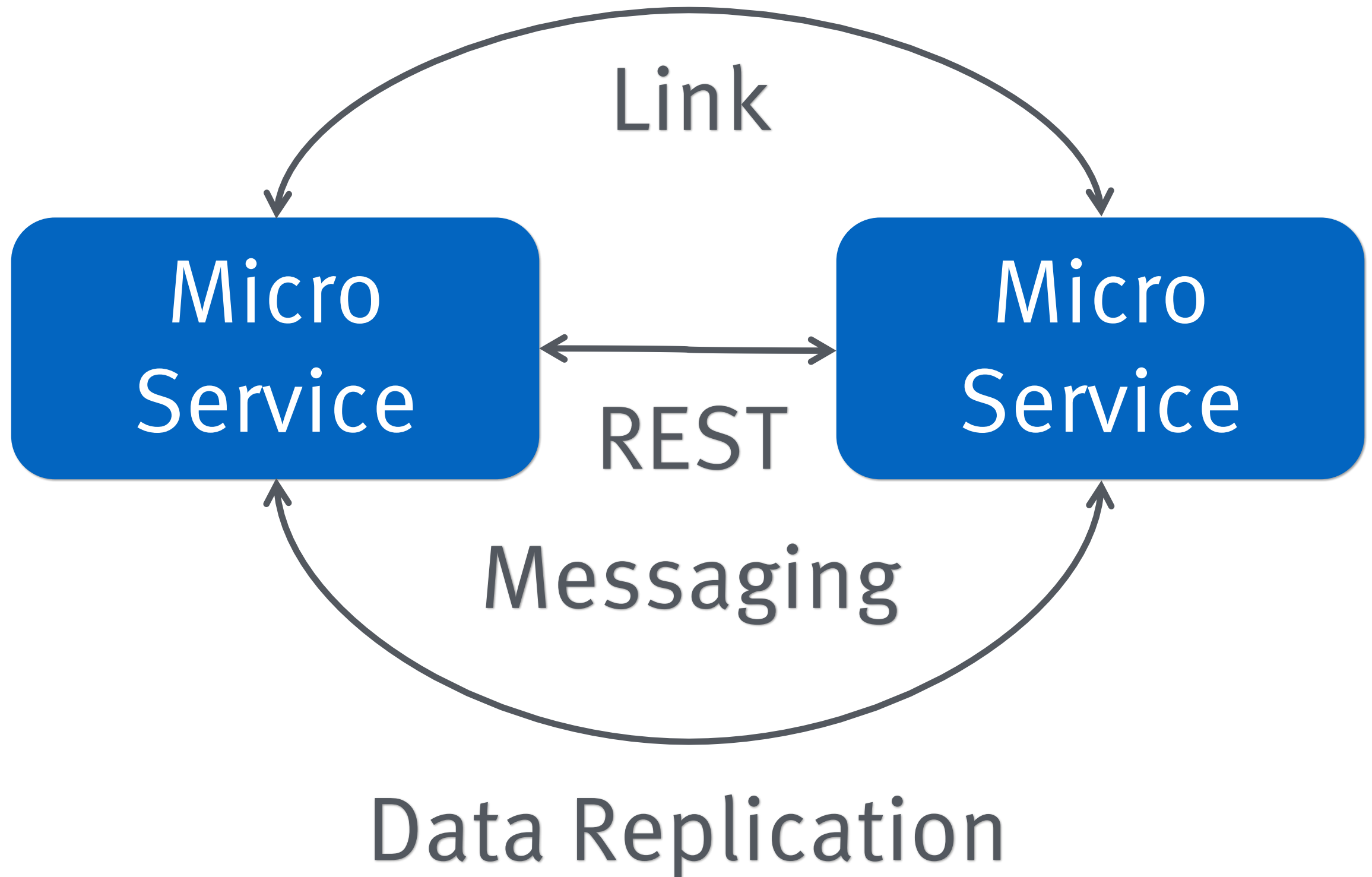
Microservices: Definition

- › No consistent definition
- › Microservices are modules
- › Independent deployment units
- › E.g. processes, Docker container
- › Microservice owned by one team

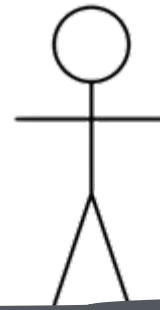
Microservices: Definition



Components Collaborate



Online Shop



Customer

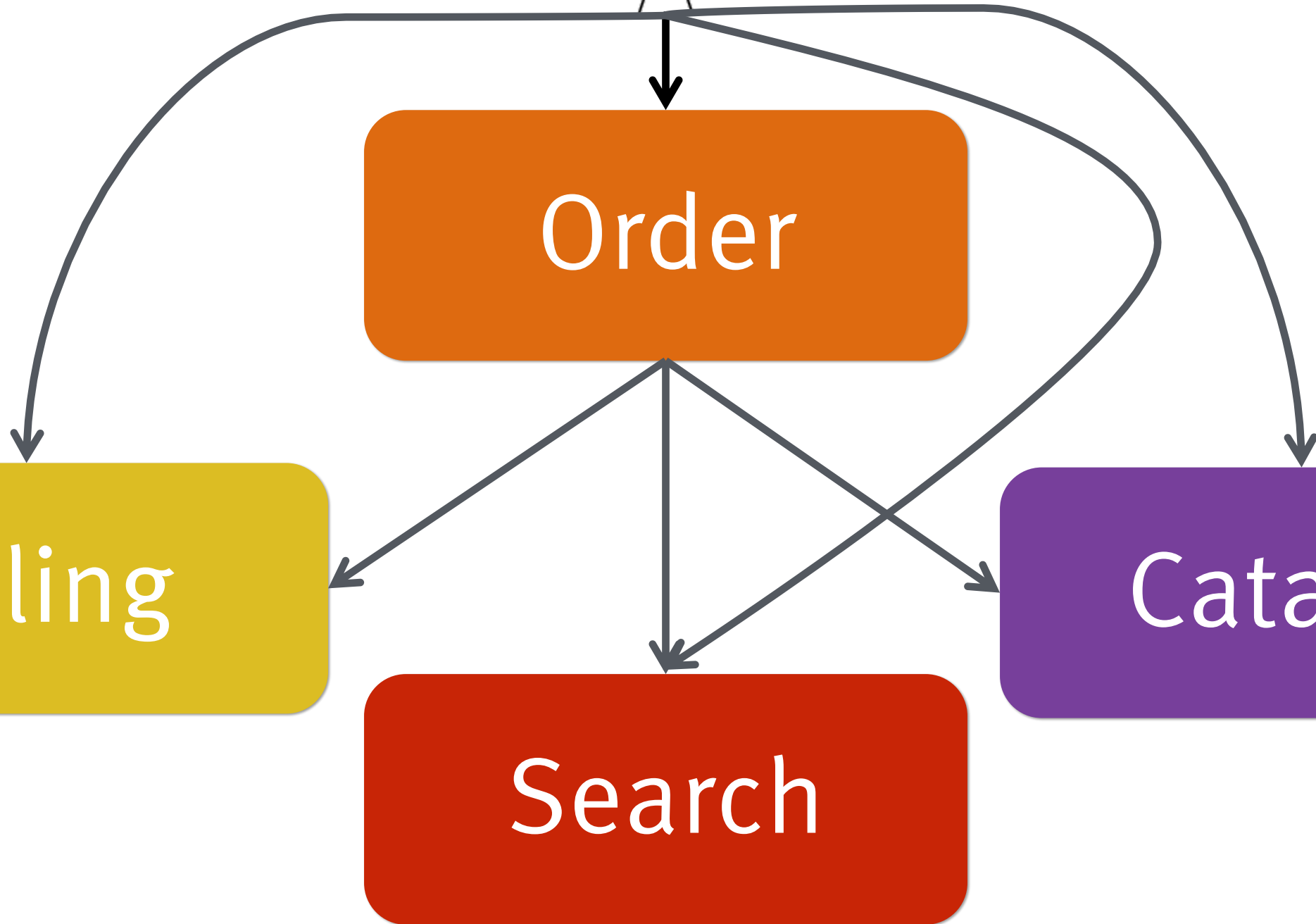
HTML /
HTTP

Order

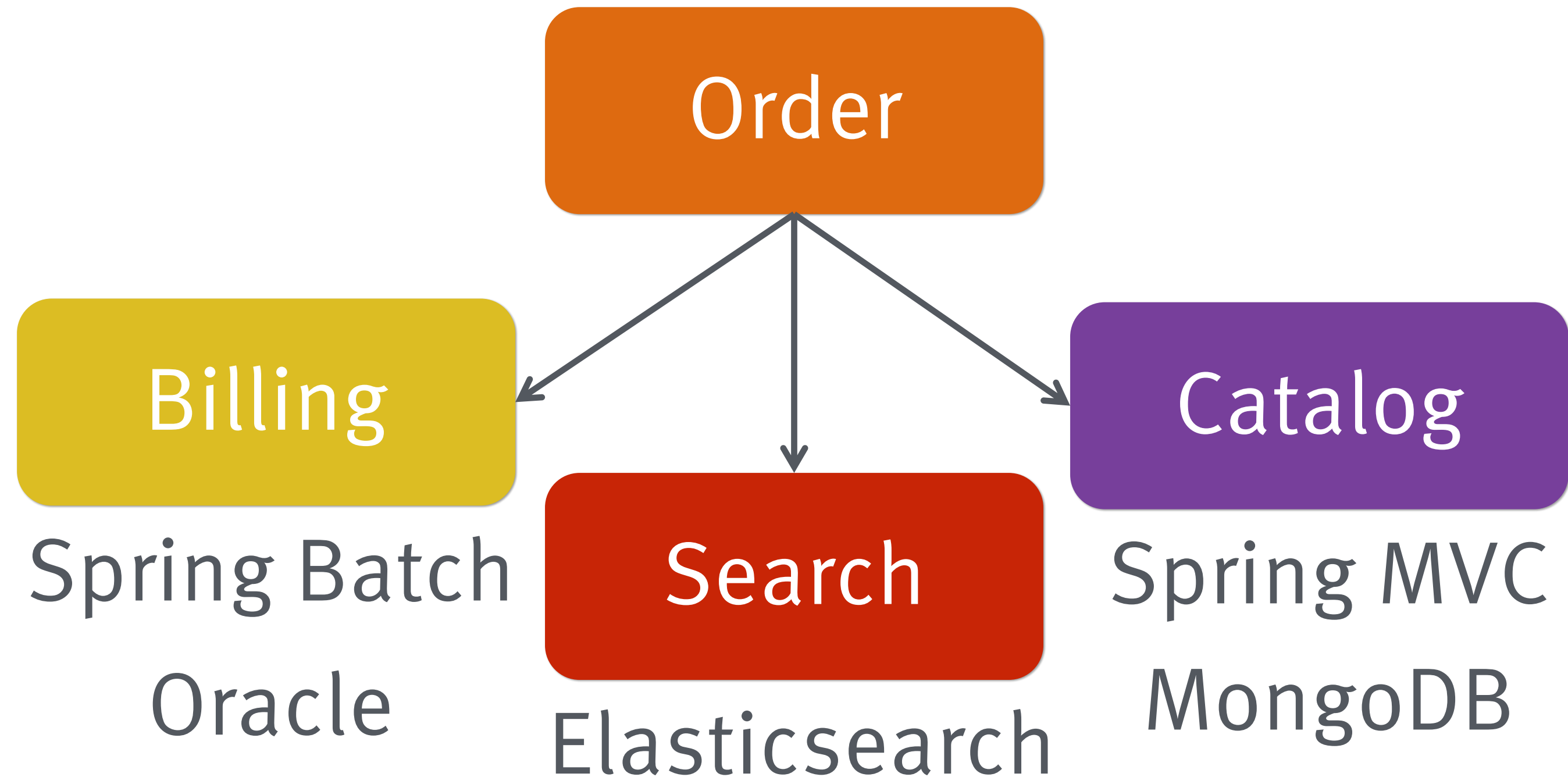
Billing

Search

Catalog



Online Shop



Microservices Stack

Docker

- › Linux (soon Windows)
- › Shared kernel
- › Filesystem just stores diffs
- › Extremely light weight
- › Can run in cluster (e.g. Kubernetes, Mesosphere etc)

Spring Boot

- › One build file
- › One Java class
- › = REST service + embedded Tomcat
- › Deploy: One Java Fat JARs
- › Alternatives: Dropwizard, Wildfly Swarm...
- › <https://github.com/ewolff/spring-boot-demos>

Microservice = Fat JAR +
Java EE *programming model*
possible!

Microservice Size



———— Team size

———— Modularization

———— Replaceability

No absolute value!

———— Infrastructure

———— Distributed Communication

Ideal size of a
Microservice

Nanoservices

Nanoservices?

#SRSLY?



#SRSLY?

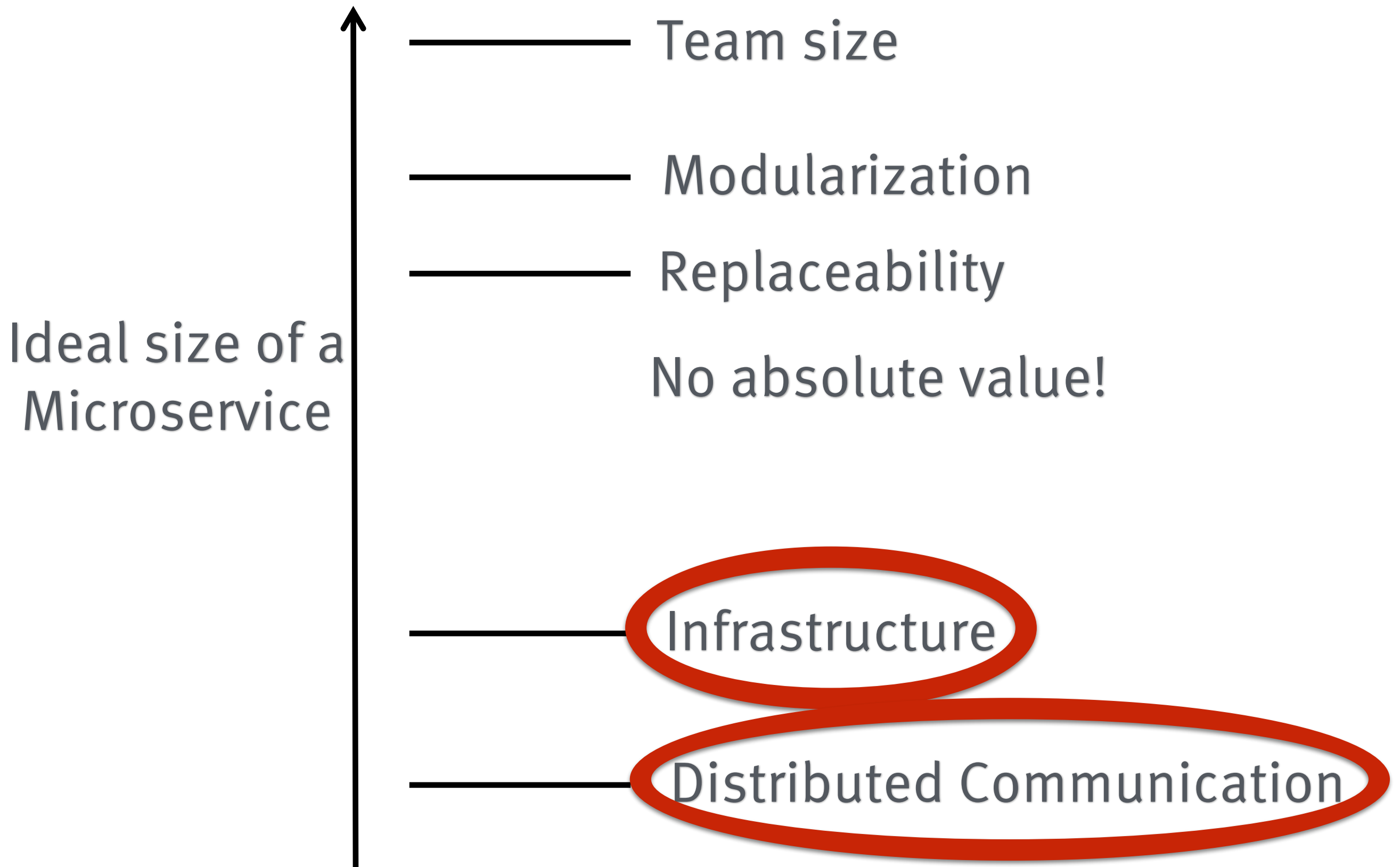
YES!



Nanoservices!=
Microservices

Nanoservices <
Microservices

Microservice Size



Microservices=
Independent
Deployment Units

Nanoservices=
Independent
Deployment Units

Nanoservices use
more light weight
technologies

Serverless

Serverless

- › Deploy a function
 - › Pay per call
 - › Example: Amazon Lambda
 - › Similar: Google Cloud Functions
 - › Azure Functions
 - › IBM OpenWhisk (Open Source)
- 

Serverless vs. PaaS



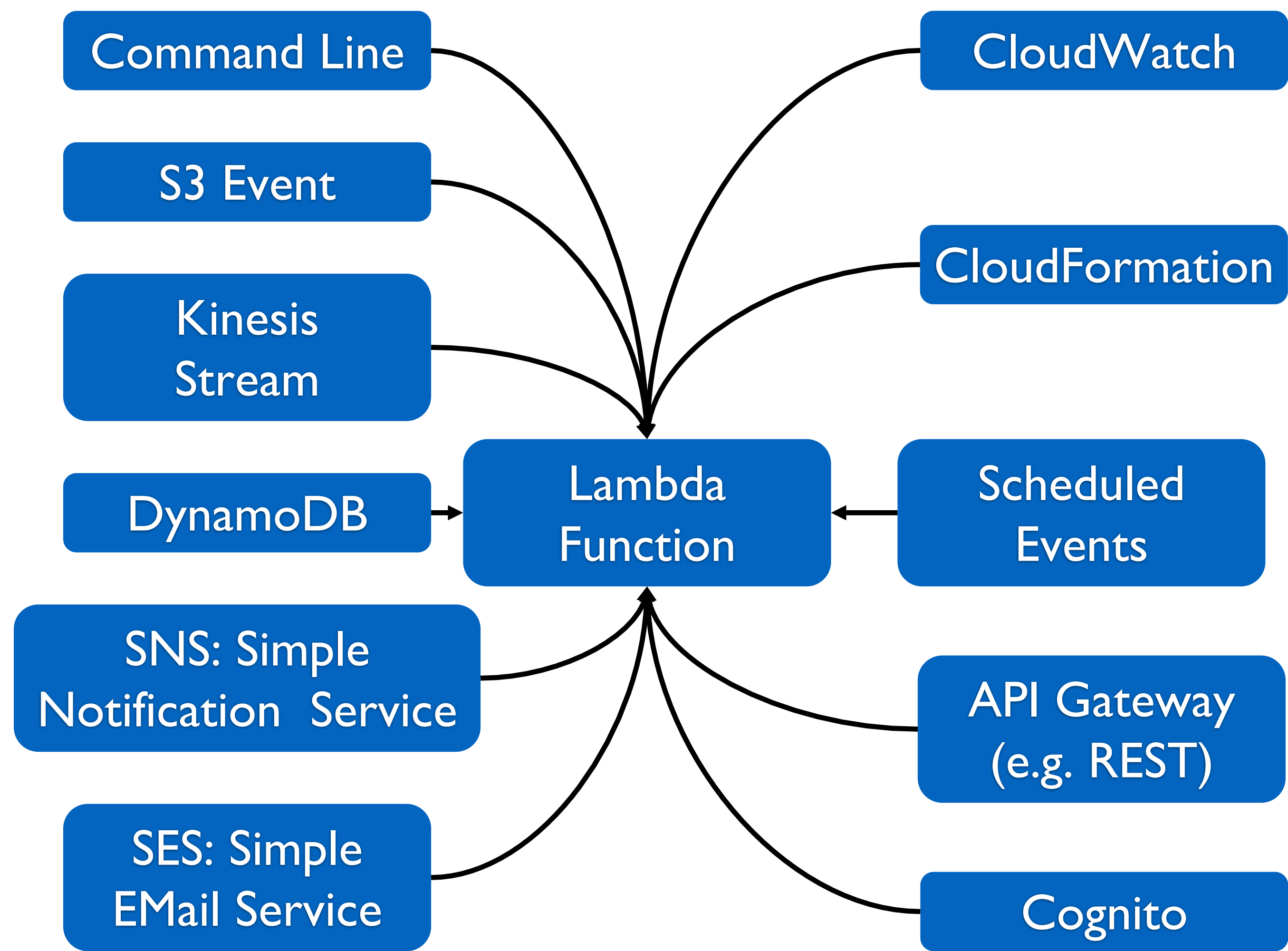
If your PaaS can efficiently start instances in 20ms that run for half a second, then call it serverless.

Amazon Lambda

- › Service in the Amazon Cloud
- › Allows you to install individual *functions*
- › Java, JavaScript, Python

Amazon Lambda vs. PaaS

- › Commercial model:
Pay-per-call / RAM / processing time
- › Fine grained
(e.g. database trigger, HTTP GET)
- › First million requests /
400.000 DB seconds FREE!
- › Can edit Python / JavaScript functions in the
browser



Amazon Lambda adds
e.g. trigger to
DynamoDB.

Amazon Lambda
+ API Gateway
= REST services.

```
public class Main {  
  
    public String myHandler(int myCount,  
        Context context) {  
        LambdaLogger logger = context.getLogger();  
        logger.log("received : " + myCount);  
        return "Hi"+myCount;  
    }  
}
```

Deploy

```
***  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\  
> aws lambda create-function \
```

Deploy

```
***  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\  
> aws lambda create-function \  
> --function-name hello_java \  
...
```

Deploy

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\n> aws lambda create-function \n> --function-name hello_java \n> --role arn:aws:iam::142613406848:role/executionrole \
```

Deploy

```
***
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\
> aws lambda create-function \
> --function-name hello_java \
> --role arn:aws:iam::142613406848:role/executionrole \
> --runtime java8 \
```

Deploy

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\n> aws lambda create-function \n> --function-name hello_java \n> --role arn:aws:iam::142613406848:role/executionrole \n> --runtime java8 \n> --handler com.ewolff.aws_demo.Main::myHandler \
```

Deploy

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]\n> aws lambda create-function \n> --function-name hello_java \n> --role arn:aws:iam::142613406848:role/executionrole \n> --runtime java8 \n> --handler com.ewolff.aws_demo.Main::myHandler \n> --zip-file fileb://target/original-java_lambda_demo-1.0-SNAPSHOT.jar
```

Deploy

```
> --runtime java8 \  
> --handler com.ewolff.aws_demo.Main::myHandler \  
> --zip-file fileb://target/original-java_lambda_demo-1.0-SNAPSHOT.jar  
{  
  "CodeSha256": "n3Wek1RpCE5//o9IpbHC3/LWo0lZyBzpS0/Zj6w+0yE=",  
  "FunctionName": "hello_java",  
  "CodeSize": 2647,  
  "MemorySize": 128,  
  "FunctionArn": "arn:aws:lambda:eu-west-1:142613406848:function:hello_java",  
  "Version": "$LATEST",  
  "Role": "arn:aws:iam::142613406848:role/executionrole",  
  "Timeout": 3,  
  "LastModified": "2016-03-04T15:42:54.126+0000",  
  "Handler": "com.ewolff.aws_demo.Main::myHandler",  
  "Runtime": "java8",  
  "Description": ""  
}  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]
```

Invoke

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \  
> --function-name hello_java \  
^C
```

Invoke

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \  
> --function-name hello_java \  
> --payload '42' \  

```

Invoke

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \  
> --function-name hello_java \  
> --payload '42' \  
> outputfile.txt
```

Invoke

```
***  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \  
> --function-name hello_java \  
> --payload '42' \  
> outputfile.txt  
{  
  "StatusCode": 200  
}
```

Invoke

```
***
[wolfff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \
> --function-name hello_java \
> --payload '42' \
> outputfile.txt
{
  "StatusCode": 200
}
[wolfff@TeraMacBook:~/aws-lambda/java-lambda-demo]more outputfile.txt
```

Invoke

```
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]aws lambda invoke \  
> --function-name hello_java \  
> --payload '42' \  
> outputfile.txt  
{  
  "StatusCode": 200  
}  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]more outputfile.txt  
"Hi42"  
[wolff@TeraMacBook:~/aws-lambda/java-lambda-demo]
```

Lambda Management Console

Eberhard

← → ↻ 🏠

https://eu-west-1.console.aws.amazon.com/lambda/home?region=eu-west-1#/functions/hello_java...

🌟 ⓘ 📺 🔍 ☰

📦 AWS ▾

Services ▾

Edit ▾

Eberhard Wolff ▾

Ireland ▾

Support ▾

Lambda > Functions > hello_java

ARN - arn:aws:lambda:eu-west-1:142613405848:function:hello_java

Test

Actions ▾

Code

Configuration

Event sources

API endpoints

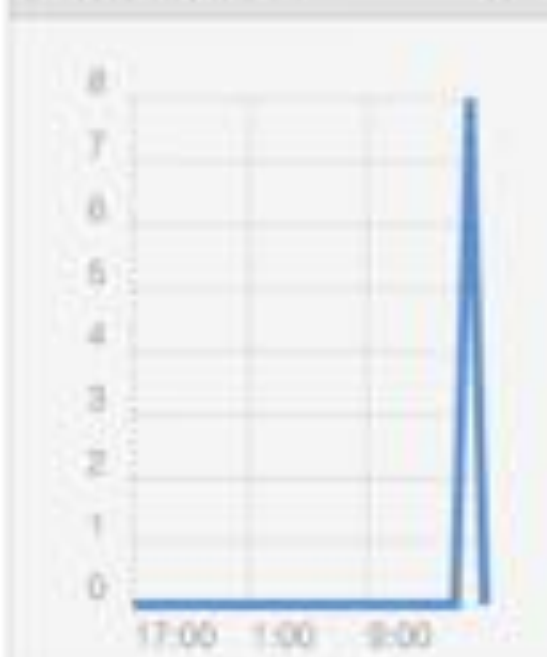
Monitoring

?

CloudWatch metrics at a glance (last 24 hours)

[View logs in CloudWatch](#)

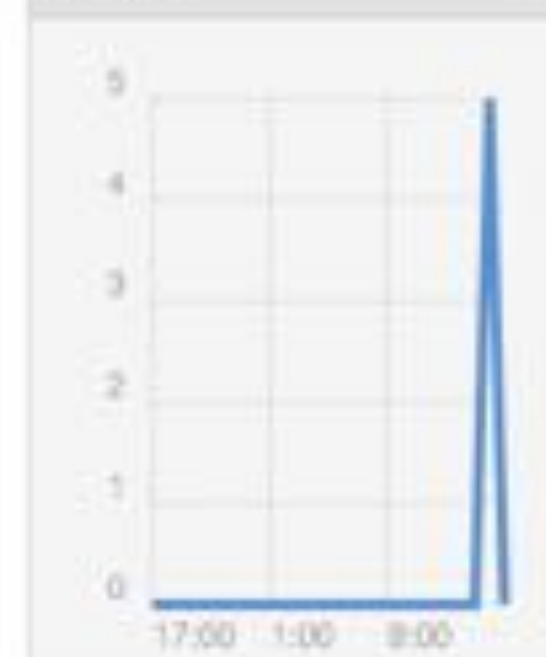
Invocations



Duration



Errors



Throttles



Other monitoring information

Last modified date

2016-03-04T15:30:03.351+0000



AWS

Services

Edit

Eberhard Wo

CloudWatch

Dashboards NEW

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events NEW

Rules

Logs

Metrics

Selected Metrics

Lambda

[Log Groups](#) > [Streams for /aws/lambda/hello_java](#) > Events for

2016/03/04[\$LATEST]ac34651...

Filter: Search for events

Date/Time:

2016/03/04

15

:

30

:

55

UTC (GMT)

Event Data

▼ START RequestId: 161291fa-e21e-11e5-b188-9f9119a08dce
Version: \$LATEST

▼ received : 42

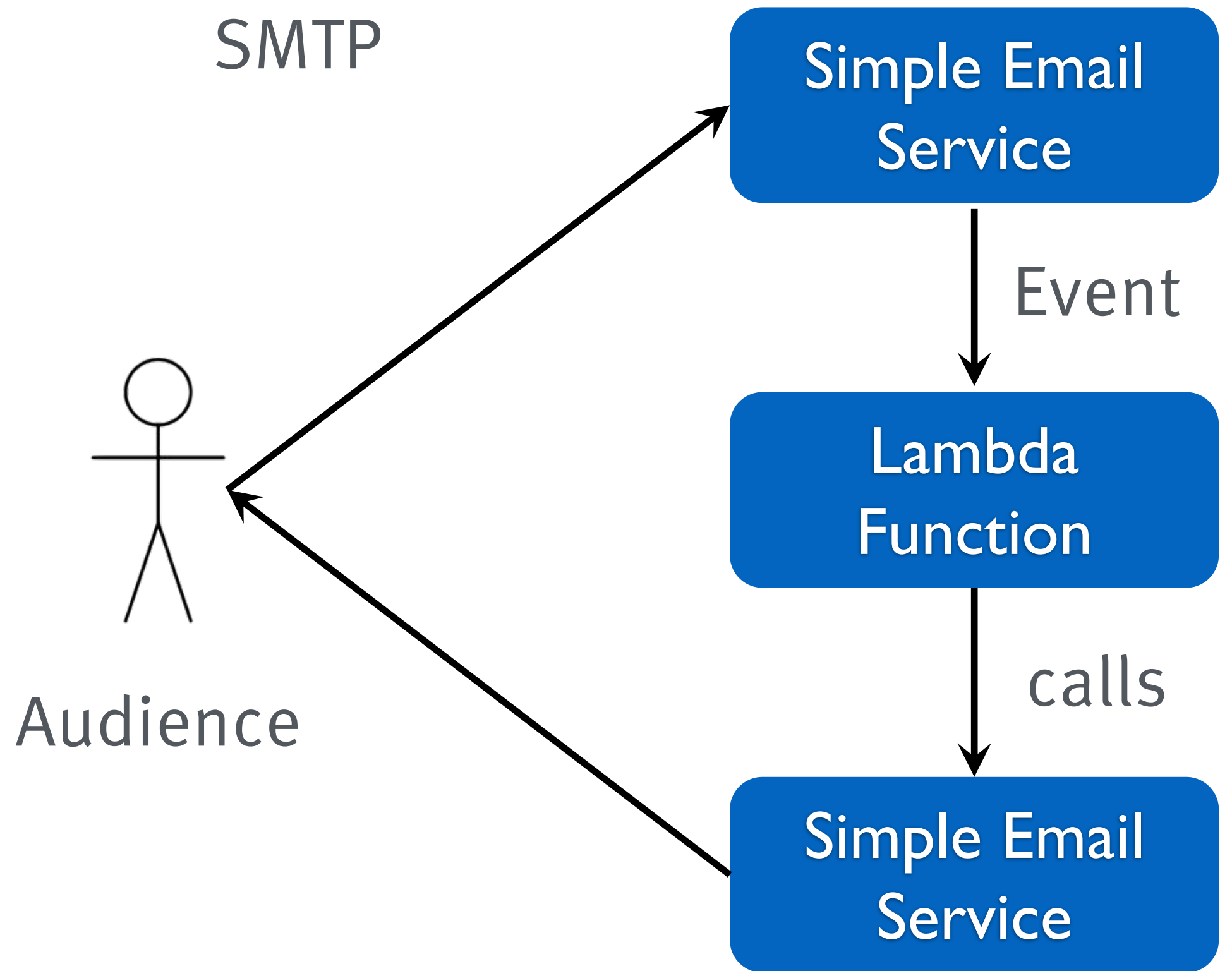
▼ END RequestId: 161291fa-e21e-11e5-b188-9f9119a08dce

▼ REPORT RequestId: 161291fa-e21e-11e5-b188-9f9119a08dce
Duration: 238.81 ms Billed Duration: 300 ms Memory Size: 128 MB
Max Memory Used: 28 MB

Amazon Lambda

- › Can add any other Amazon Service to complete the system
- › i.e. Beanstalk PaaS, EC2 IaaS ...
- › Or databased (DynamoDB, RDS...)

Send out slides
via email!



Cost: SES 0,10\$ per 1000 EMails

Lambda free

- › Developed in Python
- › In the browser
- › "Save & test"
- › Includes monitoring & alarm
- › Setting up SES was the hardest part...

Lambda Management Console

Eberhard

← → ↻ 🏠

https://eu-west-1.console.aws.amazon.com/lambda/...

☆ 🔒 ⓘ 📄 🗑️ ⋮

 AWS ▾

Services ▾

Edit ▾

Eb

AWS Lambda

Dashboard BETA

Functions

Lambda > Functions > SlideEmail

ARN - arn:aws:lambda:eu-

Qualifiers ▾

Test

Actions ▾

Code

Configuration

Triggers

Monitoring

Code entry type

Edit code inline ▾

```
1 from __future__ import print_function
2 import json
3 import smtplib
4 from email.mime.text import MIMEText
5 import boto3
6
7 def lambda_handler(event, context):
8     mail = event['Records'][0]['ses']['mail']
9     sender = mail['commonHeaders']['from'][0]
10    print("Received message from %s " % (sender))
11    client = boto3.client('ses')
```

 Feedback

 English

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

OSGi

OSGi: Bundles

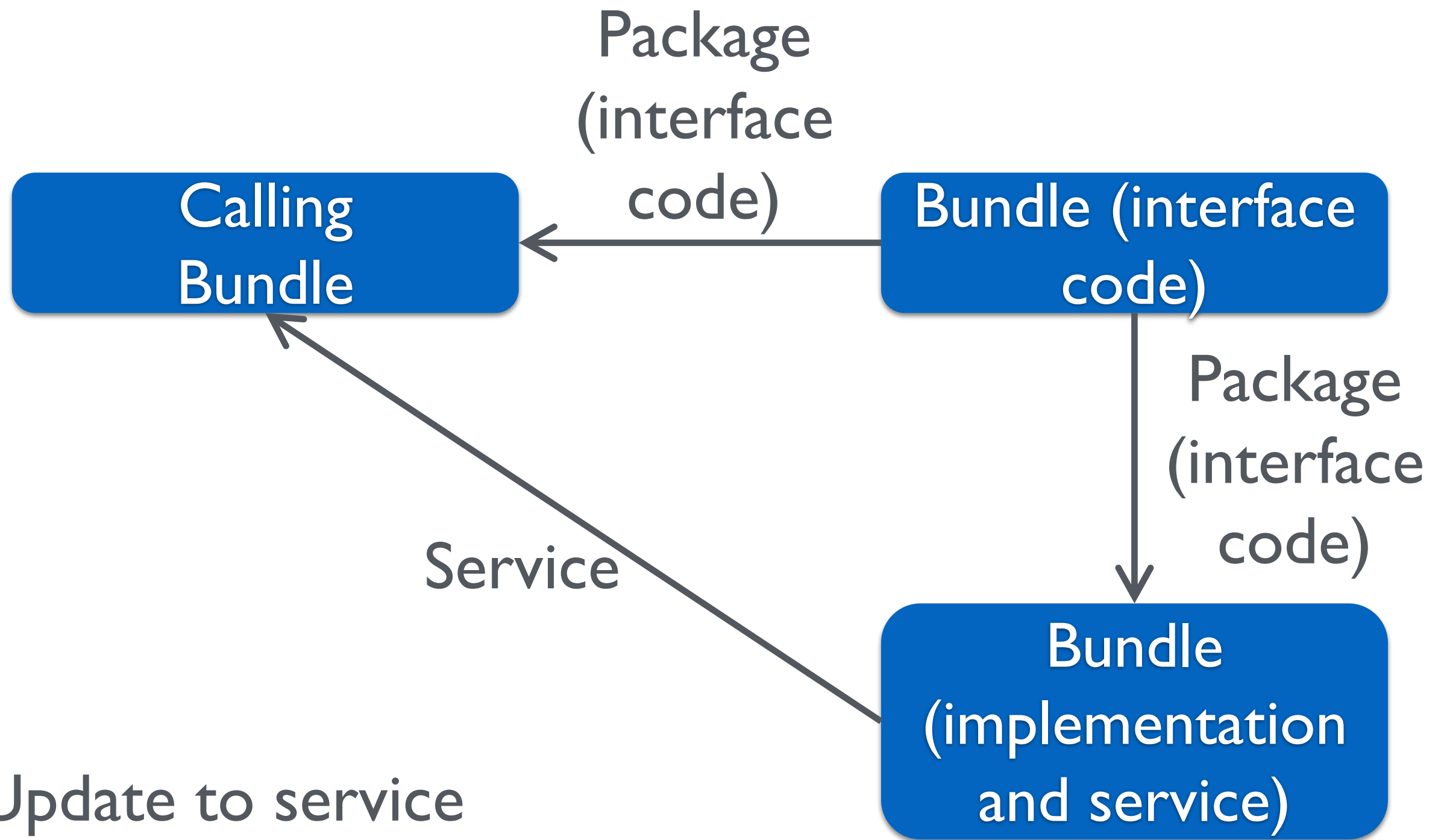
- › Partition system into – "bundles"
- › Bundles can be installed, started, stopped, uninstalled and updated
- › ...at runtime

OSGi: Code Sharing

- › Bundles can export and import packages
- › Updating code requires other bundles to start fresh

OSGi: Services

- › Bundles can **publish** services... *dynamically!*
- › Service = Java Object
- › **Service Registry** allows other bundles to **consume** services
- › Services come and go at runtime
- › Quite easy with OSGi Blueprints or OSGi Declarative Services



Update to service
in running application

(Almost)
Independent
Deployment Units

Java EE

Java EE

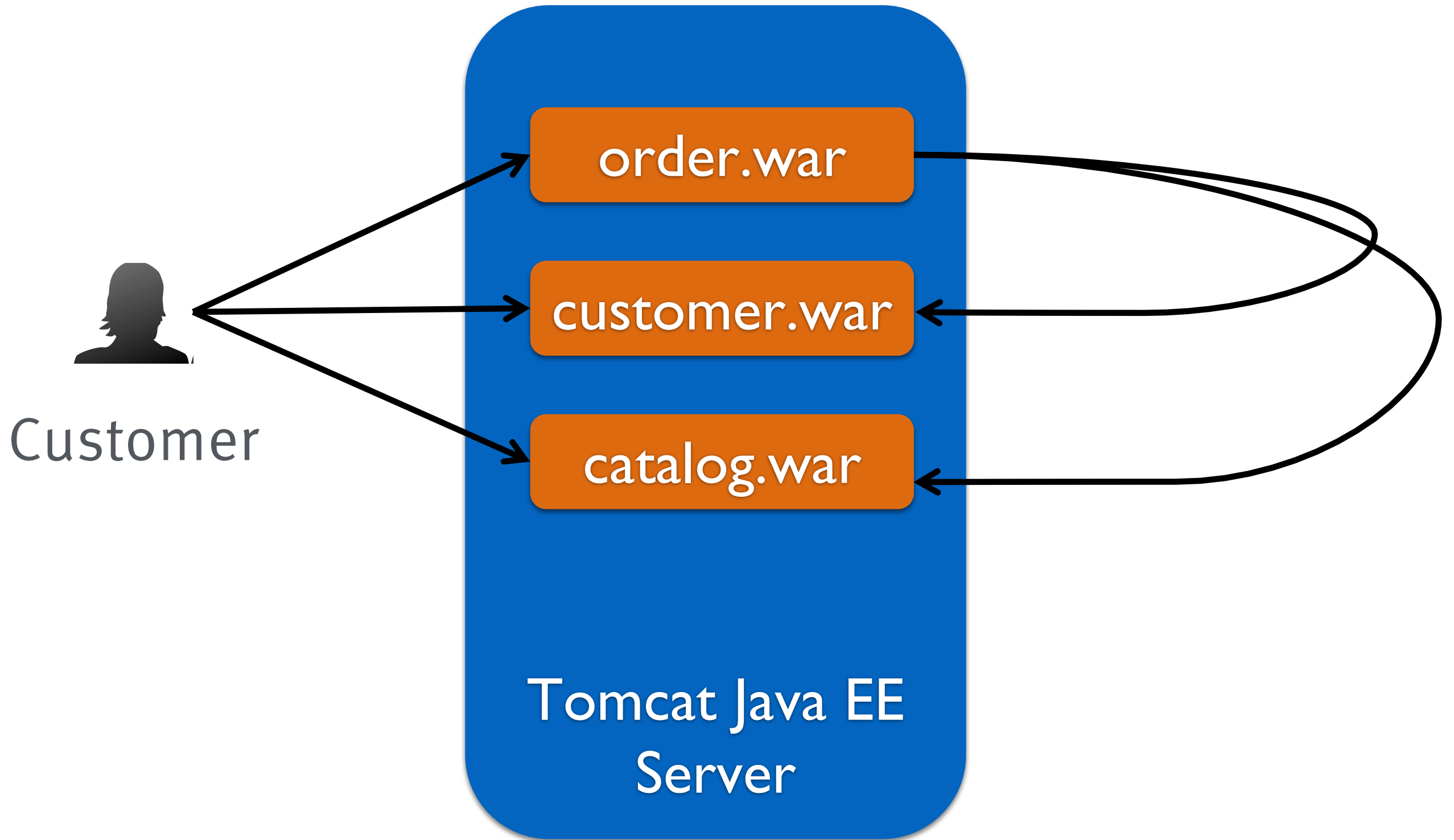
- › JARs e.g. for EJBs
- › WARs e.g. for web application
- › EARs for JARs and WARs
- › Completely separated code
- › ...unlike OSGi

Java EE

- › Relevant: Deployment model
- › i.e. application server
- › Not programming model

Java EE Nanoservices?

- › Local calls not possible
- › No shared code e.g. for interfaces



(Almost)
Independent
Deployment Units

Comparison

Independent Deployment

- › THE feature of Micro-/Nanoservices
- › Simplifies to put new features in production
- › Docker Fat JAR: ++
- › Amazon Lambda: ++
- › OSGi: + Restart the whole JVM?
- › Java EE: + Restart the whole JVM?

Benefits of Nanoservices



Effort per Service

- › How hard is it to create another service?
- › All: Create another project/JAR/WAR
- › Amazon Lambda: ++
- › OSGi: ++
- › Java EE: ++
- › Docker Fat JAR : - (Docker container etc)

Cost per Service

- › Hardware, software cost
- › Amazon Lambda: ++
- › OSGi: ++
- › Java EE: ++
- › Docker Fat JAR: -- (need Docker container etc, separat JVM Heap)

Local Communication

- › Call without network stack
- › Docker Fat JAR : --
- › Amazon Lambda: --
- › OSGi: ++
- › Java EE: --

Challenges of Nanoservices



Independent Scaling

- › Start more instances of a single service
- › Docker Fat JAR: ++
- › Amazon Lambda: ++
- › OSGi: ?
- › Java EE: --

Isolation

- › CPU / Memory consumption/crash influences other services
- › Docker Fat JAR: ++
- › Amazon Lambda: ++
- › OSGi: --
- › Java EE: --

Resilience & Isolation

- › Resilience: System can handle crash of other services
- › Needs isolation
- › Problem for OSGi/Java EE

Technology Freedom

- › Using different technologies
- › Docker Fat JAR: ++ (whatever)
- › Amazon Lambda: +
(Java, JavaScript, Python)
- › OSGi: -- (Java)
- › Java EE: -- (Java)

Conclusion

Nanoservices!=
Microservices

Nanoservices <
Microservices

Conclusion

- › Nanoservice technologies compromise
- › ...to allow smaller services
- › ...to allow local communication
- › OSGi and Java EE deployment model don't fully support independent deployment
- › ...might therefore not be “true”
Nanoservices

Conclusion

- › OSGi and Java EE weak concerning
- › Independent scaling
- › Isolation
- › Technology freedom
- › Amazon Lambda a lot stronger

Might also consider...

- › Vert.x:
polyglot, distributed, module concept
- › Erlang:
support update to running system and
processes in other languages

Thank You!

@ewolff

<http://microservices-buch.de/>

<http://microservices-book.com/primer.html>

<http://aws.amazon.com/lambda/getting-started/>

[http://www.heise.de/developer/artikel/
Nanoservices-kleiner-als-Microservices-3038541.html](http://www.heise.de/developer/artikel/Nanoservices-kleiner-als-Microservices-3038541.html)