aws

APACHECON

Modernize APIs to run serverless using Apache CXF

Dennis Kieselhorst

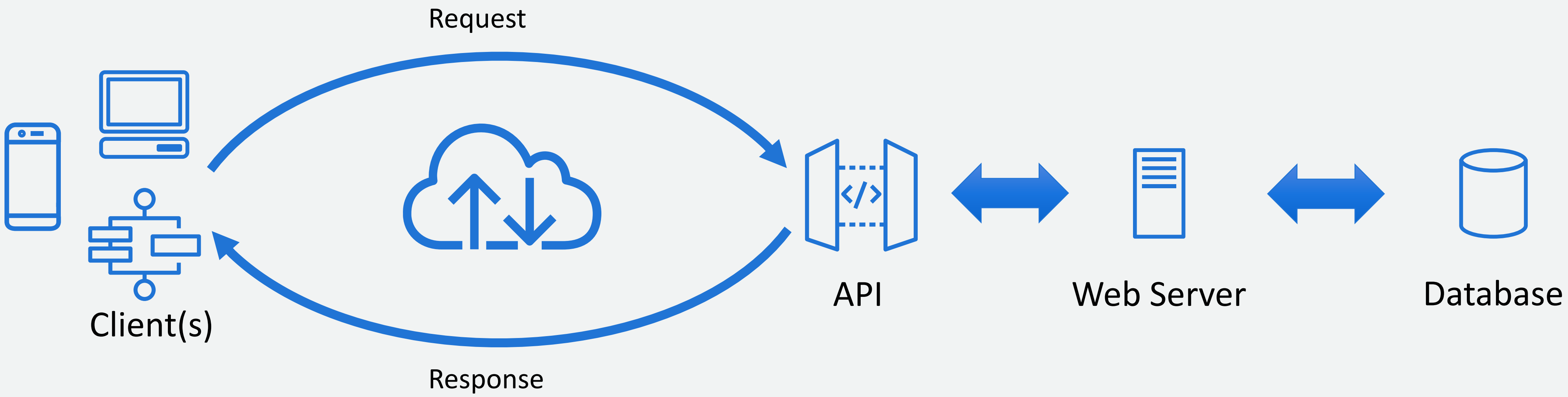
Principal Solutions Architect
Amazon Web Services

Agenda

- Intro - APIs, Apache CXF, Serverless
- Scenario
- Contract/ API-First vs. Code First approach
- Demo with Java runtime
- Lifecycle of a serverless function
- GraalVM and related frameworks
- Demo with native image
- Summary

Application Programming Interfaces (APIs)

- Simplify programming by abstracting the underlying implementation and only exposing objects or actions needed.
- APIs are the „glue“ between applications.



Apache CXF

- Apache CXF is an open source services framework.
- CXF helps you build and develop services in Java using frontend programming APIs, like JAX-WS and JAX-RS.
- These services can speak a variety of protocols such as SOAP or RESTful HTTP and work over a variety of transports such as HTTP or JMS.
- CXF supports API specifications like WSDL and the OpenAPI Specification (formerly known as Swagger).

What is Serverless?

No infrastructure provisioning, no management

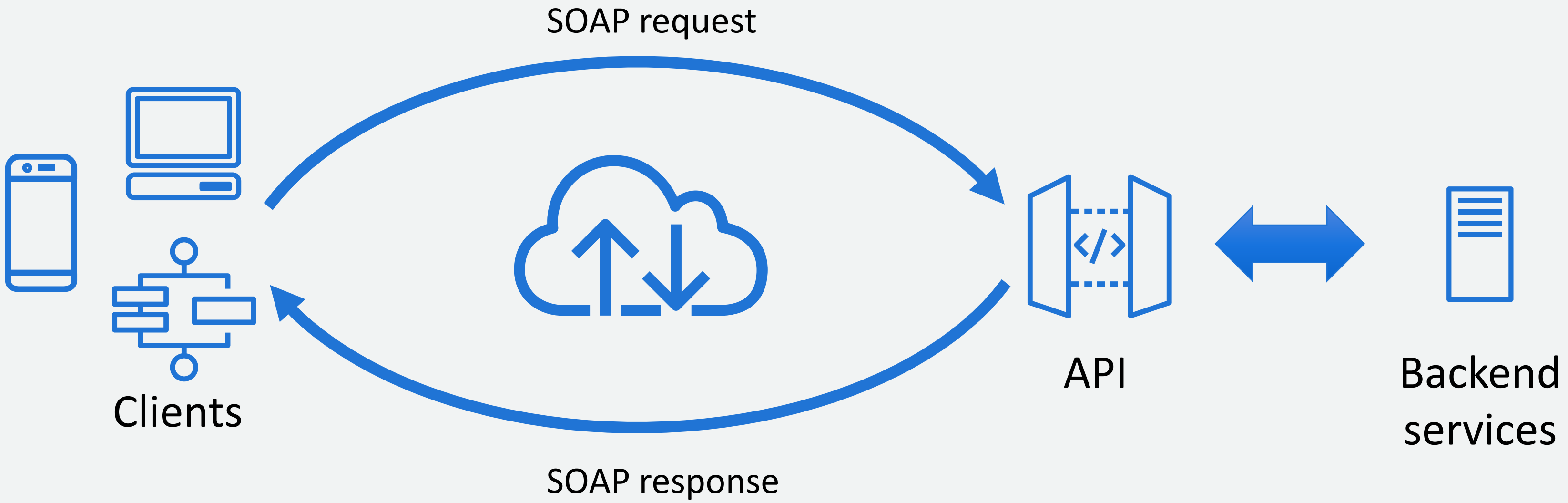
Automatic scaling

Pay for value

Highly available and secure

Scenario

- Mature API in place, no changes happened for a long time
- Outdated infrastructure causes relability and security challenges
- >20 consumers (internal and external), unable/ not willing to change their implementation
- Interface definition (WSDL) is part of signed business contracts



Contract/ API-First vs. Code First approach

Contract/ API-First

Specification is defined first and acts as service contract

- helpful for different teams on client-/ server side
- even more across different companies/ with third parties

Client- and servercode can be generated with a code generator

- ensures code is always consistent to the API
- compile errors for breaking changing (possible to automate using Continious Integration tool)
- code is not as clean as handwritten code, may look confusing
- tolerant reader pattern may be a better option over spec-based code generation (depends on the scope and change frequency of the API)

Code First

Specification is derived from API implementation

- code can be annotated
- export is either done at compile or runtime
- developers are familiar with it → fast for simple APIs

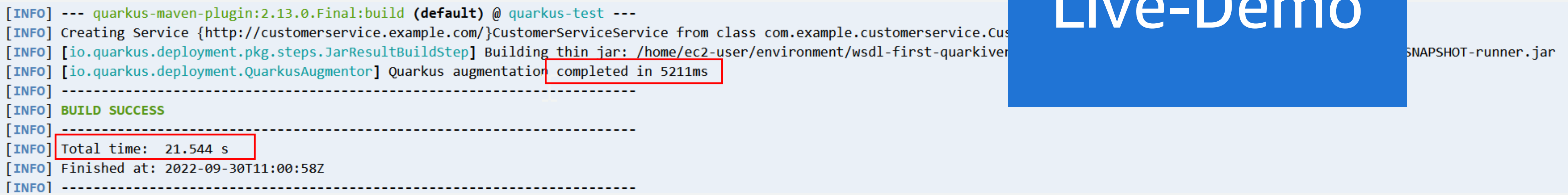
Generated specification may contain unused resources

- easily happens that something is accidently exposed
- often lack of documentation

Demo with Java runtime

Build time

Live-Demo

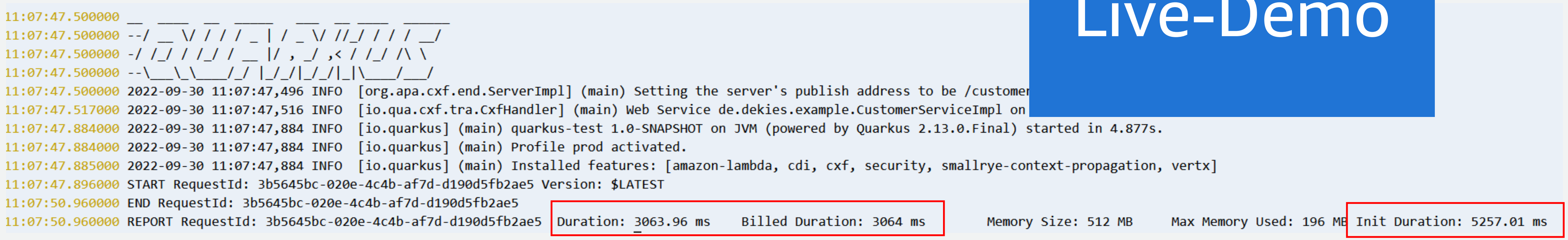


```
[INFO] --- quarkus-maven-plugin:2.13.0.Final:build (default) @ quarkus-test ---
[INFO] Creating Service {http://customerservice.example.com/}CustomerServiceService from class com.example.customerservice.Cus
[INFO] [io.quarkus.deployment.pkg.steps.JarResultBuildStep] Building thin jar: /home/ec2-user/environment/wsdl-first-quarkiver                                                 SNAPSHOT-runner.jar
[INFO] [io.quarkus.deployment.QuarkusAugmentor] Quarkus augmentation completed in 5211ms
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  21.544 s
[INFO] Finished at: 2022-09-30T11:00:58Z
[INFO] ------------------------------------------------------------------------
```

Execution time

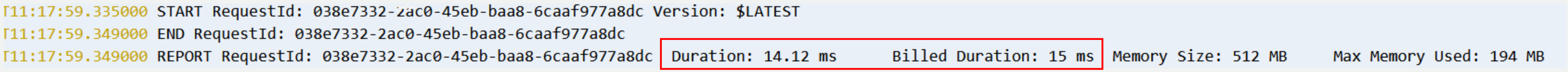
First execution:

Live-Demo



```
11:07:47.500000 __  ____  __  _____   ___  __ ____  ______
11:07:47.500000 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
11:07:47.500000 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
11:07:47.500000 --\___\_\____/_/ |_/_/|_/_/|_|\____/___/
11:07:47.500000 2022-09-30 11:07:47,496 INFO  [org.apa.cxf.end.ServerImpl] (main) Setting the server's publish address to be /customer
11:07:47.517000 2022-09-30 11:07:47,516 INFO  [io.qua.cxf.tra.CxfHandler] (main) Web Service de.dekies.example.CustomerServiceImpl on
11:07:47.884000 2022-09-30 11:07:47,884 INFO  [io.quarkus] (main) quarkus-test 1.0-SNAPSHOT on JVM (powered by Quarkus 2.13.0.Final) started in 4.877s.
11:07:47.884000 2022-09-30 11:07:47,884 INFO  [io.quarkus] (main) Profile prod activated.
11:07:47.885000 2022-09-30 11:07:47,884 INFO  [io.quarkus] (main) Installed features: [amazon-lambda, cdi, cxf, security, smallrye-context-propagation, vertx]
11:07:47.896000 START RequestId: 3b5645bc-020e-4c4b-af7d-d190d5fb2ae5 Version: $LATEST
11:07:50.960000 END RequestId: 3b5645bc-020e-4c4b-af7d-d190d5fb2ae5
11:07:50.960000 REPORT RequestId: 3b5645bc-020e-4c4b-af7d-d190d5fb2ae5  Duration: 3063.96 ms    Billed Duration: 3064 ms        Memory Size: 512 MB     Max Memory Used: 196 MB Init Duration: 5257.01 ms
```

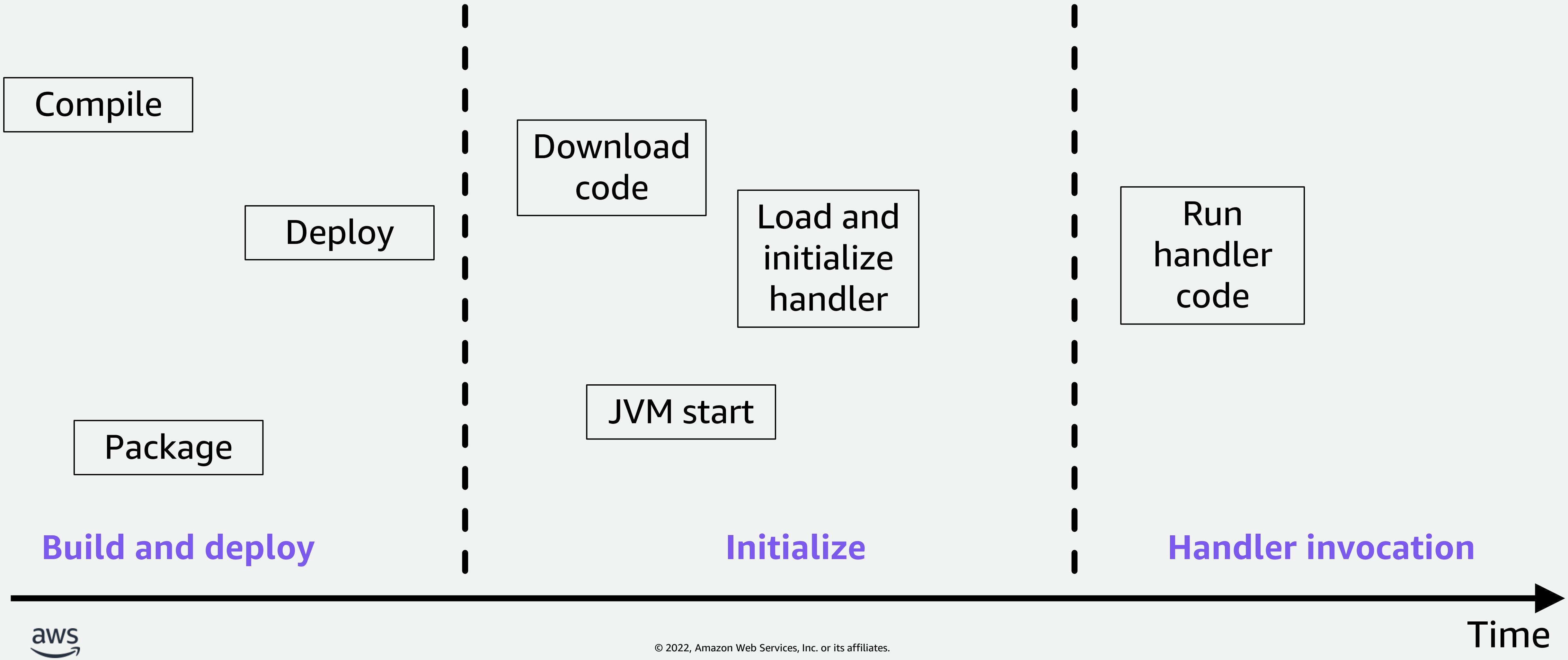
Second execution:



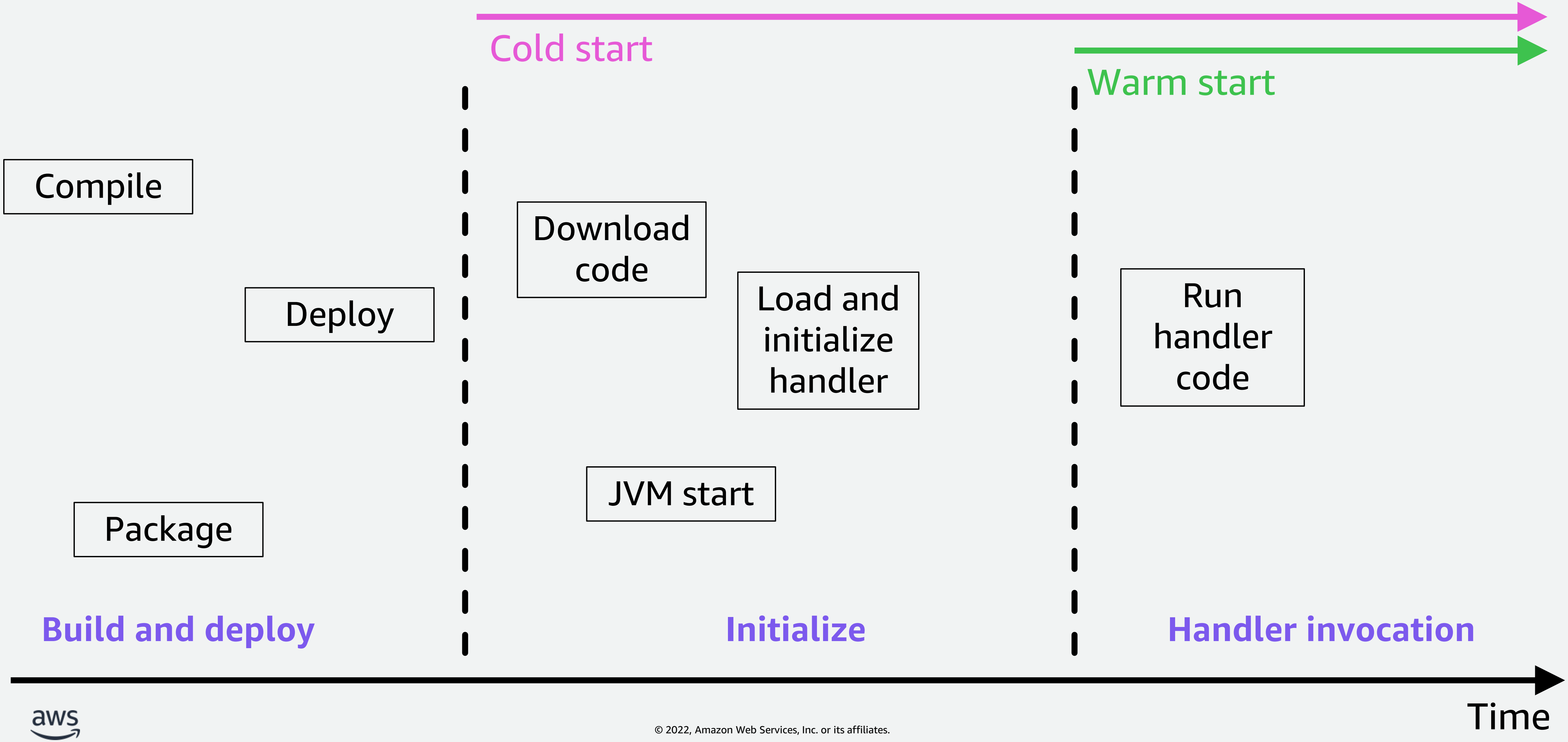
```
11:17:59.335000 START RequestId: 038e7332-2ac0-45eb-baa8-6caaf977a8dc Version: $LATEST
11:17:59.349000 END RequestId: 038e7332-2ac0-45eb-baa8-6caaf977a8dc
11:17:59.349000 REPORT RequestId: 038e7332-2ac0-45eb-baa8-6caaf977a8dc  Duration: 14.12 ms       Billed Duration: 15 ms  Memory Size: 512 MB     Max Memory Used: 194 MB
```

Lifecycle of a serverless function

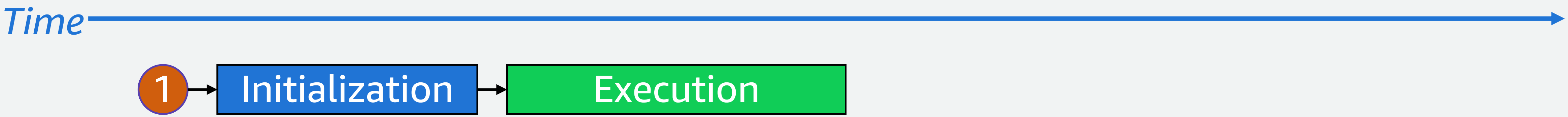
The lifecycle of an AWS Lambda function



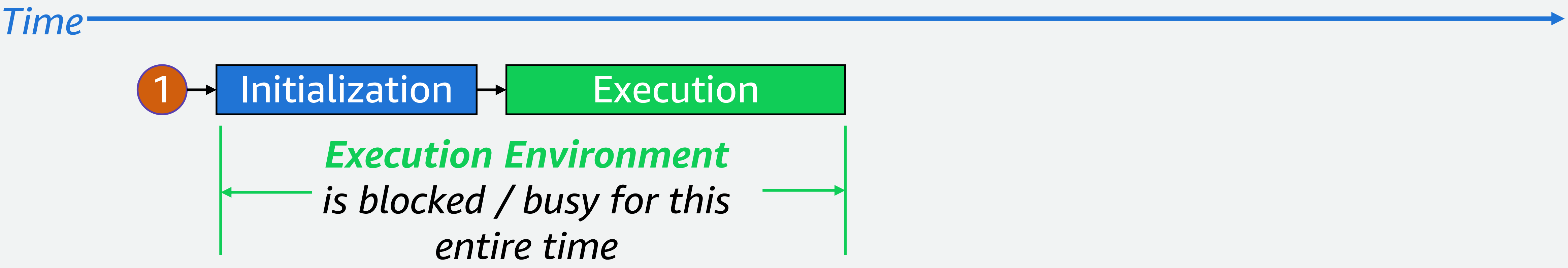
The lifecycle of an AWS Lambda function



How Lambda scales: A primer on Lambda concurrency

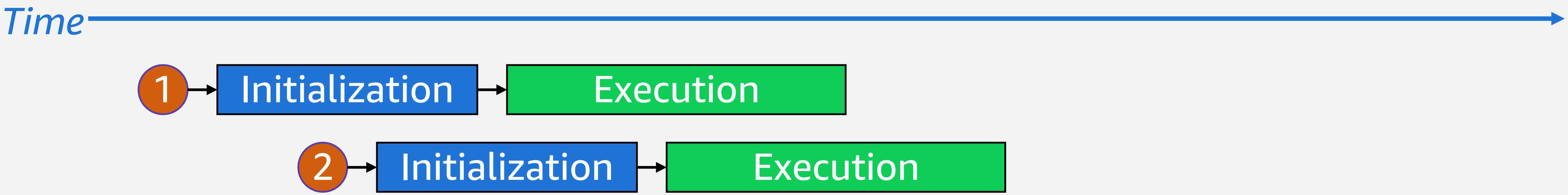


How Lambda scales: A primer on Lambda concurrency

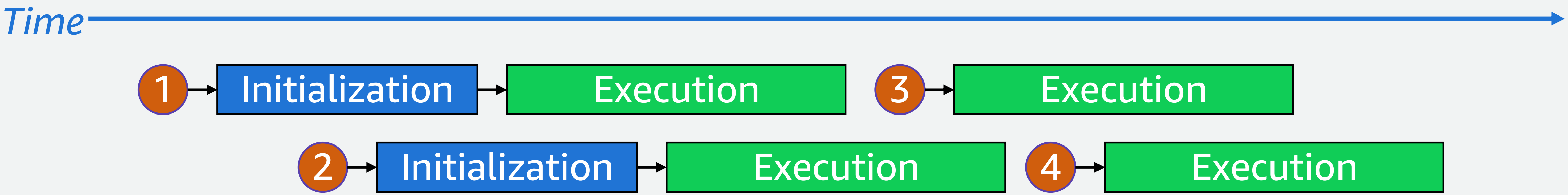


Execution Environment is blocked / busy for this entire time

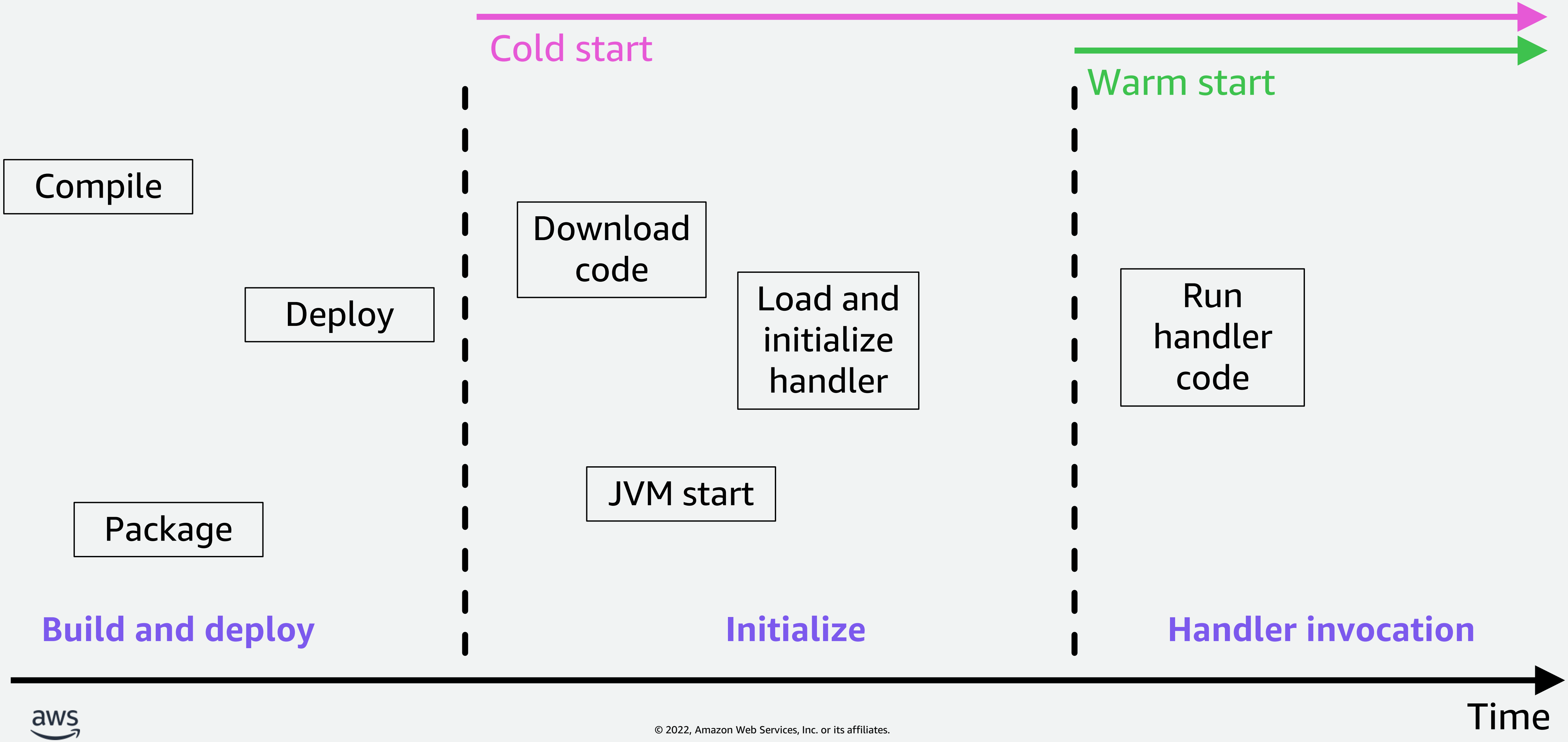
How Lambda scales: A primer on Lambda concurrency



How Lambda scales: A primer on Lambda concurrency



The lifecycle of an AWS Lambda function

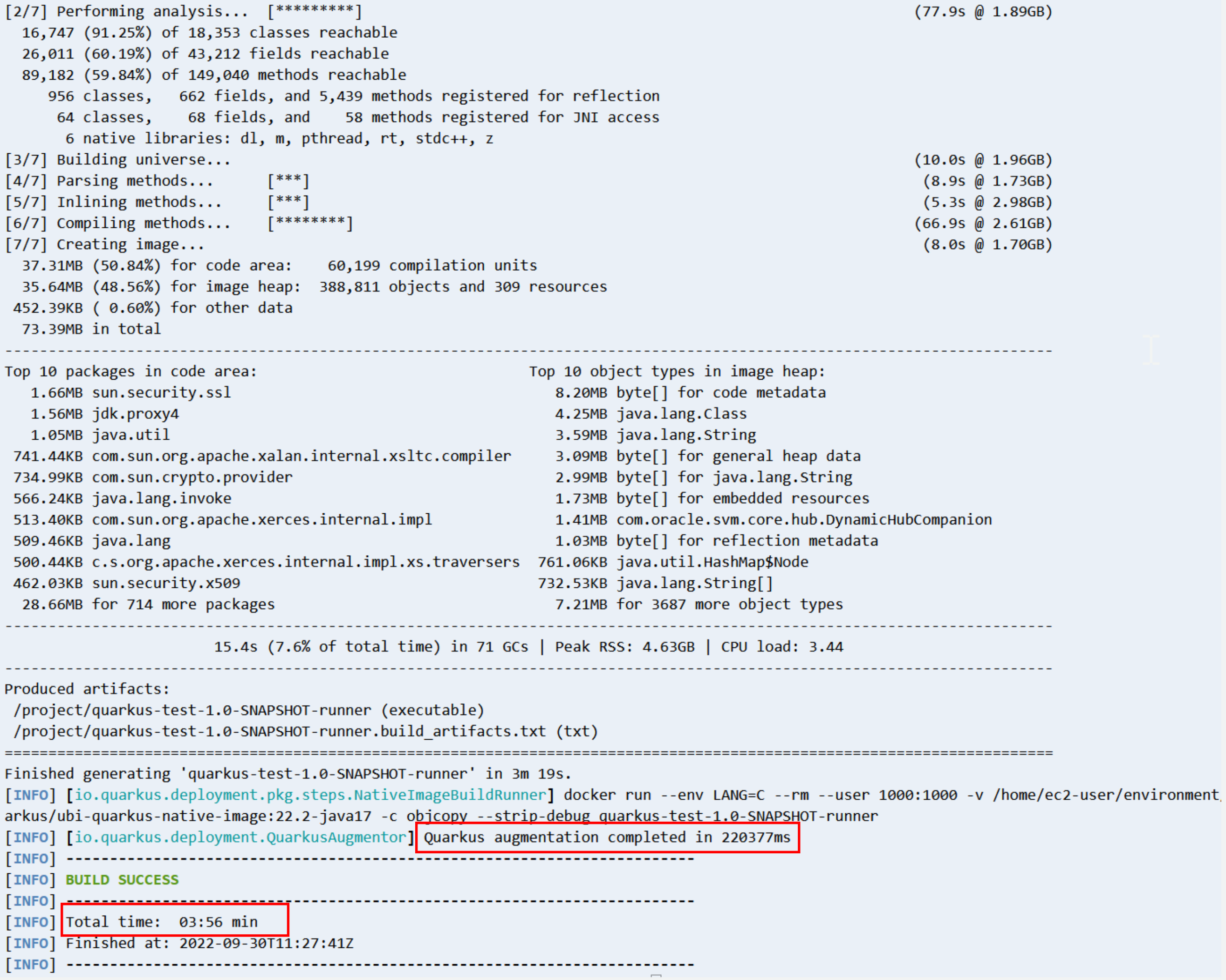


GraalVM and related frameworks

- GraalVM is a high-performance runtime that is designed to address the limitations of traditional VMs such as initialization overhead and memory consumption.
- Beyond using GraalVM as just another JVM you can also create a native executable via the native image capability. This executable already includes all necessary dependencies (e.g. Garbage collector) and therefore does not a require a JVM to run your code.
- Major frameworks like Quarkus, Micronaut and Spring Native allow to leverage GraalVM conveniently.
- Library changes may be required to make them compatible. A Quarkus extension for CXF (in the Quarkiverse project) already exists to address that.

Demo with native image

Build time



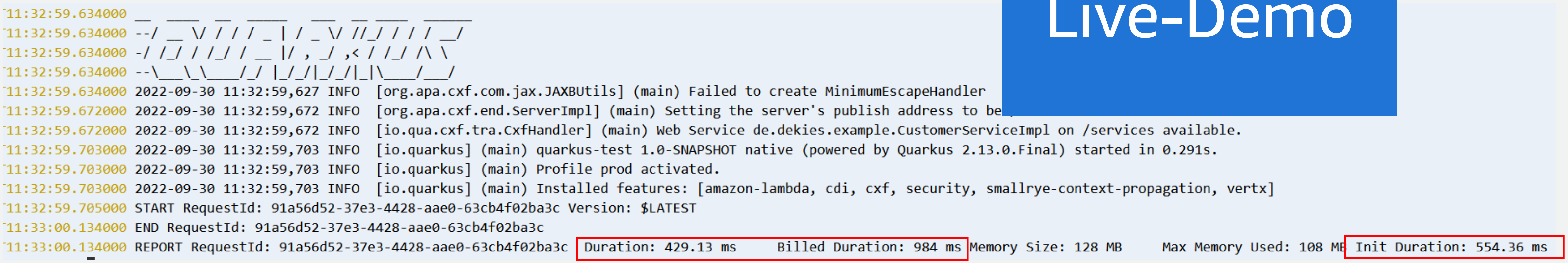
```
[2/7] Performing analysis...  [*********]                                                        (77.9s @ 1.89GB)
  16,747 (91.25%) of 18,353 classes reachable
  26,011 (60.19%) of 43,212 fields reachable
  89,182 (59.84%) of 149,040 methods reachable
     956 classes,   662 fields, and 5,439 methods registered for reflection
      64 classes,    68 fields, and    58 methods registered for JNI access
       6 native libraries: dl, m, pthread, rt, stdc++, z
[3/7] Building universe...                                                                       (10.0s @ 1.96GB)
[4/7] Parsing methods...      [***]                                                               (8.9s @ 1.73GB)
[5/7] Inlining methods...     [***]                                                               (5.3s @ 2.98GB)
[6/7] Compiling methods...    [********]                                                         (66.9s @ 2.61GB)
[7/7] Creating image...                                                                           (8.0s @ 1.70GB)
  37.31MB (50.84%) for code area:    60,199 compilation units
  35.64MB (48.56%) for image heap:  388,811 objects and 309 resources
 452.39KB ( 0.60%) for other data
  73.39MB in total
------------------------------------------------------------------------------------------------------------------------
Top 10 packages in code area:                               Top 10 object types in image heap:
   1.66MB sun.security.ssl                                     8.20MB byte[] for code metadata
   1.56MB jdk.proxy4                                           4.25MB java.lang.Class
   1.05MB java.util                                            3.59MB java.lang.String
 741.44KB com.sun.org.apache.xalan.internal.xsltc.compiler     3.09MB byte[] for general heap data
 734.99KB com.sun.crypto.provider                              2.99MB byte[] for java.lang.String
 566.24KB java.lang.invoke                                     1.73MB byte[] for embedded resources
 513.40KB com.sun.org.apache.xerces.internal.impl              1.41MB com.oracle.svm.core.hub.DynamicHubCompanion
 509.46KB java.lang                                            1.03MB byte[] for reflection metadata
 500.44KB c.s.org.apache.xerces.internal.impl.xs.traversers  761.06KB java.util.HashMap$Node
 462.03KB sun.security.x509                                  732.53KB java.lang.String[]
  28.66MB for 714 more packages                                7.21MB for 3687 more object types
------------------------------------------------------------------------------------------------------------------------
                        15.4s (7.6% of total time) in 71 GCs | Peak RSS: 4.63GB | CPU load: 3.44
------------------------------------------------------------------------------------------------------------------------
Produced artifacts:
 /project/quarkus-test-1.0-SNAPSHOT-runner (executable)
 /project/quarkus-test-1.0-SNAPSHOT-runner.build_artifacts.txt (txt)
========================================================================================================================
Finished generating 'quarkus-test-1.0-SNAPSHOT-runner' in 3m 19s.
[INFO] [io.quarkus.deployment.pkg.steps.NativeImageBuildRunner] docker run --env LANG=C --rm --user 1000:1000 -v /home/ec2-user/environment,
arkus/ubi-quarkus-native-image:22.2-java17 -c objcopy --strip-debug quarkus-test-1.0-SNAPSHOT-runner
[INFO] [io.quarkus.deployment.QuarkusAugmentor] Quarkus augmentation completed in 220377ms
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  03:56 min
[INFO] Finished at: 2022-09-30T11:27:41Z
[INFO] ------------------------------------------------------------------------
```

Live-Demo

Execution time

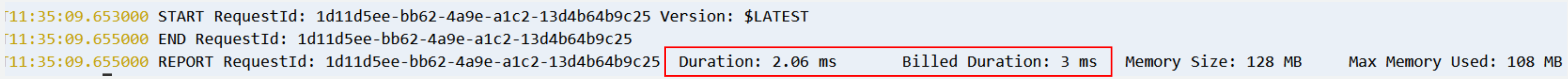
First execution

Live-Demo



```
11:32:59.634000 __  ____  __  _____   ___  __ ____  ______
11:32:59.634000 --/ __ \/ / / / _ | / _ \/ //_/ / / / __/
11:32:59.634000 -/ /_/ / /_/ / __ |/ , _/ ,< / /_/ /\ \
11:32:59.634000 --\___\_\____/_/ |_/_/|_/_/|_|\____/___/
11:32:59.634000 2022-09-30 11:32:59,627 INFO  [org.apa.cxf.com.jax.JAXBUtils] (main) Failed to create MinimumEscapeHandler
11:32:59.672000 2022-09-30 11:32:59,672 INFO  [org.apa.cxf.end.ServerImpl] (main) Setting the server's publish address to be
11:32:59.672000 2022-09-30 11:32:59,672 INFO  [io.qua.cxf.tra.CxfHandler] (main) Web Service de.dekies.example.CustomerServiceImpl on /services available.
11:32:59.703000 2022-09-30 11:32:59,703 INFO  [io.quarkus] (main) quarkus-test 1.0-SNAPSHOT native (powered by Quarkus 2.13.0.Final) started in 0.291s.
11:32:59.703000 2022-09-30 11:32:59,703 INFO  [io.quarkus] (main) Profile prod activated.
11:32:59.703000 2022-09-30 11:32:59,703 INFO  [io.quarkus] (main) Installed features: [amazon-lambda, cdi, cxf, security, smallrye-context-propagation, vertx]
11:32:59.705000 START RequestId: 91a56d52-37e3-4428-aae0-63cb4f02ba3c Version: $LATEST
11:33:00.134000 END RequestId: 91a56d52-37e3-4428-aae0-63cb4f02ba3c
11:33:00.134000 REPORT RequestId: 91a56d52-37e3-4428-aae0-63cb4f02ba3c  Duration: 429.13 ms     Billed Duration: 984 ms Memory Size: 128 MB     Max Memory Used: 108 MB Init Duration: 554.36 ms
```

Second execution



```
11:35:09.653000 START RequestId: 1d11d5ee-bb62-4a9e-a1c2-13d4b64b9c25 Version: $LATEST
11:35:09.655000 END RequestId: 1d11d5ee-bb62-4a9e-a1c2-13d4b64b9c25
11:35:09.655000 REPORT RequestId: 1d11d5ee-bb62-4a9e-a1c2-13d4b64b9c25  Duration: 2.06 ms       Billed Duration: 3 ms   Memory Size: 128 MB     Max Memory Used: 108 MB
```

Summary

Summary

- Apache CXF enables you to provide stable, mature APIs even with a long lifecycle (>10 years).
- Modernizing API infrastructure to serverless allows to lower your costs and adapt at scale while eliminating infrastructure management tasks.
- GraalVM native images significantly reduce cold-start time and memory consumption.

Thank you!

Dennis Kieselhorst

in kieselhorst