

# How a Java 8 backport wreaked havoc

Georgios Andrianakis, Red Hat



Backport:

Introduction of a feature from an newer version to an old supported version

# WHAT COULD

# POSSIBLY GO WRONG?



**Clement Escoffier**

10:16

Hello,

Pretty sure I've seen @Stephane Epardaud mentioned it yesterday. The nightly build of gRPC on Java 8 is now failing with:

```
java.lang.ClassNotFoundException: org.eclipse.jetty.alpn.ALPN$Provider
```

It's not really an issue, because I don't believe we should recommend ALPN on Java 8 anyway, but still something to investigate.

**Georgios Andrianakis**

10:22

I'm seeing it fail a lot of PRs

if we shouldn't be using it, should we disable that test on Java 8?

10:22

**Clement Escoffier**

10:53

surprisingly, I can't reproduce locally...

neither in quarkus, nor grpc.

10:53

can it be a corrupted artifact somewhere?

10:54

**Georgios Andrianakis**

11:12

I can reproduce it either :(

or maybe something with opendk version used in CI?

11:19

saying that because I am reading:

11:19

```
For Java 8 versions up to 1.8.0_242 included, you also need the Jetty's ALPN boot library to provide the AL
```

here: <https://www.eclipse.org/jetty/documentation/current/alpn-chapter.html#alpn-openjdk8>

11:19

that's it

11:21

(EDITED)

I can reproduce the issue on Java 1.8.0\_252, but not on 1.8.0\_242

11:22

@Clement Escoffier ^


11:22




# Caused by: java.lang.ClassNotFoundException: org.eclipse.jetty.alpn.ALPN\$Provider

```
2020-04-17T14:25:47.1875779Z [INFO] Running io.quarkus.vertx.http.ssl.SslServerWithP12Test
2020-04-17T14:25:47.7709276Z 2020-04-17 14:25:47,231 WARN [io.qua.dep.QuarkusAugmentor] (main) Using Java versions older than 11 to build Quarkus applications is deprecated and will be disallowed in a future release!
2020-04-17T14:25:49.1660983Z 2020-04-17 14:25:49,165 INFO [io.quarkus] (main) Quarkus 999-SNAPSHOT started in 1.583s. Listening on: http://0.0.0.0:8081 and https://0.0.0.0:8444
2020-04-17T14:25:49.1669968Z 2020-04-17 14:25:49,165 INFO [io.quarkus] (main) Profile test activated.
2020-04-17T14:25:49.1671764Z 2020-04-17 14:25:49,166 INFO [io.quarkus] (main) Installed features: [cdi, security]
2020-04-17T14:25:49.8596563Z 2020-04-17 14:25:49,855 WARN [io.net.cha.ChannelInitializer] (vert.x-eventloop-thread-3) Failed to initialize a channel. Closing: [id: 0x5f0b0486, L:/10.1.0.4:8444 - R:/10.1.0.4:54694]: java.lang.NoClassDefFoundError:
org/eclipse/jetty/alpn/ALPN$Provider
2020-04-17T14:25:49.8597123Z at io.netty.handler.ssl.JettyAlpnSslEngine.newServerEngine(JettyAlpnSslEngine.java:60)
2020-04-17T14:25:49.8597557Z at io.netty.handler.ssl.JdkAlpnApplicationProtocolNegotiator$AlpnWrapper.wrapSslEngine(JdkAlpnApplicationProtocolNegotiator.java:141)
2020-04-17T14:25:49.8597802Z at io.netty.handler.ssl.JdkSslContext.configureAndWrapEngine(JdkSslContext.java:360)
2020-04-17T14:25:49.8598373Z at io.netty.handler.ssl.JdkSslContext.newEngine(JdkSslContext.java:330)
2020-04-17T14:25:49.8598637Z at io.vertx.core.net.impl.SSLHelper.createEngine(SSLHelper.java:538)
2020-04-17T14:25:49.8598917Z at io.vertx.core.http.impl.HttpServerChannelInitializer.initChannel(HttpServerChannelInitializer.java:105)
2020-04-17T14:25:49.8599177Z at io.vertx.core.http.impl.HttpServerImpl$1.initChannel(HttpServerImpl.java:215)
2020-04-17T14:25:49.8599425Z at io.netty.channel.ChannelInitializer.initChannel(ChannelInitializer.java:129)
2020-04-17T14:25:49.8599658Z at io.netty.channel.ChannelInitializer.handlerAdded(ChannelInitializer.java:112)
2020-04-17T14:25:49.8599908Z at io.netty.channel.AbstractChannelHandlerContext.callHandlerAdded(AbstractChannelHandlerContext.java:956)
2020-04-17T14:25:49.8600156Z at io.netty.channel.DefaultChannelPipeline.callHandlerAdded0(DefaultChannelPipeline.java:609)
2020-04-17T14:25:49.8600401Z at io.netty.channel.DefaultChannelPipeline.access$100(DefaultChannelPipeline.java:46)
2020-04-17T14:25:49.8600662Z at io.netty.channel.DefaultChannelPipeline$PendingHandlerAddedTask.execute(DefaultChannelPipeline.java:1463)
2020-04-17T14:25:49.8600934Z at io.netty.channel.DefaultChannelPipeline.callHandlerAddedForAllHandlers(DefaultChannelPipeline.java:1115)
2020-04-17T14:25:49.8601184Z at io.netty.channel.DefaultChannelPipeline.invokeHandlerAddedIfNeeded(DefaultChannelPipeline.java:650)
2020-04-17T14:25:49.8601490Z at io.netty.channel.AbstractChannel$AbstractUnsafe.register0(AbstractChannel.java:502)
2020-04-17T14:25:49.8601648Z at io.netty.channel.AbstractChannel$AbstractUnsafe.access$200(AbstractChannel.java:417)
2020-04-17T14:25:49.8601886Z at io.netty.channel.AbstractChannel$AbstractUnsafe$1.run(AbstractChannel.java:474)
2020-04-17T14:25:49.8602126Z at io.netty.util.concurrent.AbstractEventExecutor.safeExecute(AbstractEventExecutor.java:164)
2020-04-17T14:25:49.8602375Z at io.netty.util.concurrent.SingleThreadEventExecutor.runAllTasks(SingleThreadEventExecutor.java:472)
2020-04-17T14:25:49.8602728Z at io.netty.channel.nio.NioEventLoop.run(NioEventLoop.java:508)
2020-04-17T14:25:49.8602972Z at io.netty.util.concurrent.SingleThreadEventExecutor$4.run(SingleThreadEventExecutor.java:989)
2020-04-17T14:25:49.8603199Z at io.netty.util.internal.ThreadExecutorMap$2.run(ThreadExecutorMap.java:74)
2020-04-17T14:25:49.8603439Z at io.netty.util.concurrent.FastThreadLocalRunnable.run(FastThreadLocalRunnable.java:30)
2020-04-17T14:25:49.8603671Z at java.lang.Thread.run(Thread.java:748)
2020-04-17T14:25:49.8603900Z Caused by: java.lang.ClassNotFoundException: org.eclipse.jetty.alpn.ALPN$Provider
2020-04-17T14:25:49.8604136Z at java.net.URLClassLoader.findClass(URLClassLoader.java:382)
2020-04-17T14:25:49.8604350Z at java.lang.ClassLoader.loadClass(ClassLoader.java:418)
2020-04-17T14:25:49.8604580Z at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:352)
2020-04-17T14:25:49.8604809Z at java.lang.ClassLoader.loadClass(ClassLoader.java:351)
2020-04-17T14:25:49.8605057Z at io.quarkus.bootstrap.classloading.QuarkusClassLoader.loadClass(QuarkusClassLoader.java:341)
2020-04-17T14:25:49.8605302Z at io.quarkus.bootstrap.classloading.QuarkusClassLoader.loadClass(QuarkusClassLoader.java:294)
2020-04-17T14:25:49.8605546Z at io.quarkus.bootstrap.classloading.QuarkusClassLoader.loadClass(QuarkusClassLoader.java:341)
2020-04-17T14:25:49.8605781Z at io.quarkus.bootstrap.classloading.QuarkusClassLoader.loadClass(QuarkusClassLoader.java:294)
2020-04-17T14:25:49.8605993Z ... 25 more
2020-04-17T14:25:49.8606174Z
```

# What happened?

 **Java Bug Database**

Search 

**Versions (Unresolved/Resolved/Fixed) 0**

JDK 8	Other
8u251 <span>Fixed</span>	openjdk8u252 <span>Fixed</span>

**Related Reports**

CSR : [JDK-8233417 - JEP 244: TLS Application-Layer Protocol Negotiation Extension \(Java SE 8\)](#)  
Relates : [JDK-8051498 - JEP 244: TLS Application-Layer Protocol Negotiation Extension](#)

**Sub Tasks**

JDK-8231729 : [Documentation JEP 244: TLS Application-Layer Protocol Negotiation Extension \(Java SE 8\) - Resolved](#)

**Description**

APIs to support a Transport Layer Security (TLS) feature called Application Layer Protocol Negotiation ([ALPN](#)) [1] were introduced in Java SE 9 under JEP 244 [JDK-8051498](#). Customers have requested ALPN in Java SE 8 to support features like HTTP/2. A couple of diverging approaches (JavaEE/Jetty and Azul's OpenJSSE) demonstrate that a standard zed ALPN API is needed to prevent platform API fragmentation.

To preserve compatibility with newer platform releases, the Java SE 9 ALPN APIs will be backported to Java SE 8.

There are five changesets that should be brought back to provide this feature: 2 API and 3 minor bug fixes:

APIs [JDK-8144093](#): JEP 244/8051498 - TLS Application-Layer Protocol Negotiation Extension [JDK-8170282](#): Enable ALPN parameters to be supplied during the TLS handshake

Bug Fixes [JDK-8145849](#): ALPN: getHandshakeApplicationProtocol() always return null [JDK-8158978](#): ALPN not working when values are set directly on a SSLServerSocket [JDK-8171443](#): (spec) An ALPN callback function may also ignore ALPN

[1] <https://tools.ietf.org/html/rfc7301>

# The result

**netty / netty** Used by 29.4k Watch 1.8k Unstar 23.5k Fork 11.2k

Code Issues 371 Pull requests 48 Actions Projects 1 Wiki Security 0 Insights

## Ensure we support ALPN when using java 8u251 #10196

Merged normanmaurer merged 6 commits into `alpn-java8` 22 days ago

Conversation 26 Commits 6 Checks 0 Files changed 7

+44 -30

normanmaurer commented 23 days ago

Motivation:

ALPN support was backported to java 8 lately. Ensure we support it if the user uses the latest java 8 release

Modifications:

- Update logic to be able to detect if ALPN is supported out of the box when using Java8
- Update jetty alpn version

Result:

Be able to use ALPN out of the box when using java 8u251

Assignees: No one assigned

Labels: None yet

Projects: None yet

**square / okhttp** Used by 26.3k Watch 1.7k Unstar 37k Fork 8k

Code Issues 62 Pull requests 5 Actions Wiki Security 0 Insights

## UnsupportedOperationException from JDK8 252 onwards [3.14.x] #5973

Merged swankjesse merged 1 commit into `square/okhttp-3.14.x` from `yschmike/unsupported_operation_314` 19 days ago

Conversation 3 Commits 1 Checks 0 Files changed 1

+10 -2

yschmike commented 22 days ago · edited

Collaborator

Fix for #5970

This will fix the exception when throw in Azul Zulu JVM. There is likely a fix to come from them to not throw, but this is defined in public API, so should be handled.

Will need to be backported to 3.12.x

Assignees: No one assigned

Reviewers: swankjesse, monkey-mas

**fabric8io / kubernetes-client** Used by 4 Watch 116 Unstar 1.4k Fork 718

Code Issues 93 Pull requests 12 Actions Projects 1 Wiki Security 0 Insights

## After updating to Azul OpenJDK 8.46 (~=8u252) the client fails with java.net.SocketException: Software caused connection abort: socket write error #2145

Closed jgoeres opened this issue 23 days ago · 5 comments

New issue

jgoeres commented 23 days ago

Hi,

we recently updated our Azul Java 8 JRE from 8.44.0.12 (which corresponds to regular 8u242) to 8.46.0.20 (which corresponds to regular 8u252). We now consistently get exceptions when doing any Kubernetes API operations through the kubernetes client, both when running stuff inside the cluster and when running it outside (in the IDE), see below.

We updated from 4.7.1 to 4.9.1 but that didn't change anything.

Regards

Assignees: mamasua

Labels: bug

Projects: None yet

Milestone

**akka / akka-grpc** Watch 26 Star 318 Fork 69

Code Issues 313 Pull requests 8 Actions Projects 8 Security 0 Insights

## Multiple test failures using AdoptOpenJDK 8u252 (hs) #946

Open ignas125 opened this issue 12 days ago · 7 comments

ignas125 commented 12 days ago

Assignees: No one assigned

Labels: None yet

Projects: None yet

Milestone: No milestone

Linked pull requests: Successfully merging a pull request may close this issue. Keep the Trade JDK on 342

Notifications: Subscribe

Customize

You're not watching notifications from this thread.

Versions used

master: on commit 52f996269f779b7a1e3b92a3295c9489bb

Using SDKman I installed and setup the 8u252 JDK ( `sdk use java 8.8.252.hs-adopt` ).

Note that I can't reproduce any of the failures when rolling back to 8u242.

Expected Behavior

All tests pass when running `sbt test`:

Actual Behavior

There are multiple test failures (consistently):

```
[error] Failed tests:
[error] akka.grpc.interop.GrpcInteropAkkaJava817MkKafkaSpec
[error] akka.grpc.interop.GrpcInterop10G17MkKafkaJavaSpec
[error] akka.grpc.interop.GrpcInteropAkkaJava817MkKafkaJavaSpec
[error] akka.grpc.interop.GrpcInteropAkkaJava817MkKafkaJavaSpec
[error] akka.grpc.interop.GrpcInterop10G17MkKafkaJavaSpec
[error] akka.grpc.interop.GrpcInterop10G17MkKafkaJavaSpec
```

# Takeaways





# Don't depend on "latest" in CI

```
- name: Set up JDK 8
  # Uses sha for added security since tags can be updated
  uses: joschi/setup-jdk@b9cc6eabf7e7e3889766b5cee486f874c9e1bd2d
  with:
    java-version: 8
```

# Don't depend on "latest" in CI

The docker community has known this for years

  <https://developers.redhat.com/blog/2016/02/24/10-things-to-avoid-in-docker-containers/>

6) Don't use only the "latest" tag – The latest tag is just like the "SNAPSHOT" for Maven

# Use feature detection instead of version detection

The JavaScript community  
has known this for years

## The concept of feature detection

The idea behind feature detection is that you can run a test to determine whether a feature is supported in the current browser, and then conditionally run code to provide an acceptable experience both in browsers that *do* support the feature, and browsers that *don't*. If you don't do this, browsers that don't support the features you are using in your code won't display your sites properly and will just fail, creating a bad user experience.

Let's recap and look at the example we touched on in our [Handling common JavaScript problems](#) — the [Geolocation API](#) (which exposes available location data for the device the web browser is running on) has the main entry point for its use, a `geolocation` property available on the global [Navigator](#) object. Therefore, you can detect whether the browser supports geolocation or not by using something like the following:

```
1 if ("geolocation" in navigator) {  
2   navigator.geolocation.getCurrentPosition(function(position) {  
3     // show the location on a map, perhaps using the Google Maps API  
4   });  
5 } else {  
6   // Give the user a choice of static maps instead perhaps  
7 }
```

Q & A

