



OBJECT  
COMPUTING

WEBINAR

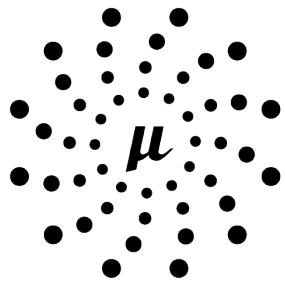
# Micronaut & GraalVM

Iván Lopez - @ilopmar

# Iván López (@ilopmar)



OBJECT  
COMPUTING



M I C R O N A U T



Groovy Users Group





**GraalVM™**

# GraalVM



- Universal Polyglot VM from Oracle
- JVM languages + Ruby, Python, JS, R
- Graal Compiler / JVMCI
- Truffle
- Substrate VM

# GraalVM™

# GraalVM



Automatic transform of interpreters to compiler

# GraalVM™

Engine integration native and managed



OpenJDK™



ORACLE®  
DATABASE



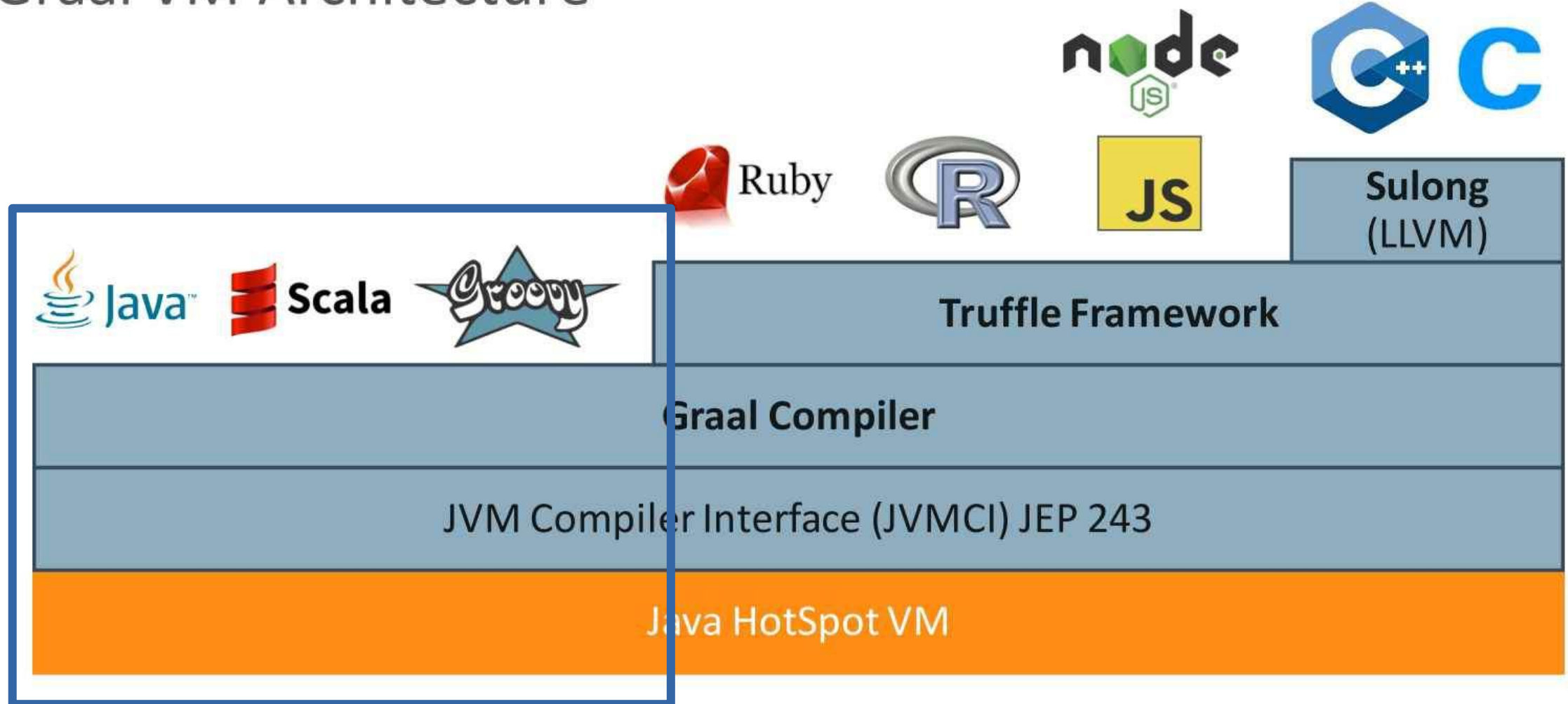
standalone



# GraalVM



## Graal VM Architecture



# GraalVM on production?



## USER CPU TIME - RATIO



■ C2 ■ Graal ■ C2 JDK9 ■ Graal JDK9

@CHRISTHALINGER | #TWITTERVMTEAM



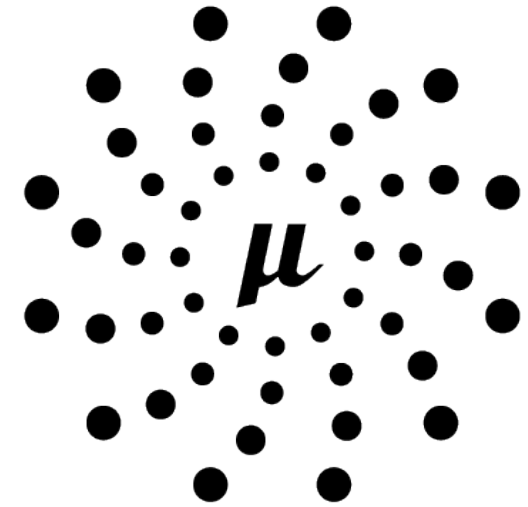
M I C R O N A U T



# Micronaut



- Framework for microservices in the JVM
- Ultra-lightweight & reactive (Netty-based)
- Java, Groovy & Kotlin
- Ahead of Time compilation (AoT)
- No reflection, no runtime proxies
- Fast startup
- Low memory footprint
- Natively Cloud Native
- Support for GraalVM



M I C R O N A U T

# Creating app with GraalVM support



```
Terminal
File Edit View Search Terminal Help

$ mn create-app basic-app
  --features=graal-native-image
```

# Micronaut support for GraalVM



## - Dependencies

```
annotationProcessor "io.micronaut:micronaut-graal"  
compileOnly "com.oracle.substratevm:svm"
```

## - Dockerfile

```
FROM oracle/graalvm-ce:1.0.0-rc15 as graalvm  
COPY . /home/app/basic-app  
WORKDIR /home/app/basic-app  
RUN native-image --no-server -cp build/libs/basic-app-*.jar
```

```
FROM frolov/alpine-glibc  
EXPOSE 8080  
COPY --from=graalvm /home/app/basic-app .  
ENTRYPOINT ["/basic-app"]
```

# Micronaut support for GraalVM (II)



- native-image.properties

```
Args = -H:IncludeResources=logback.xml|application.yml \  
      -H:Name=basic-app \  
      -H:Class=example.micronaut.Application
```

# Native image



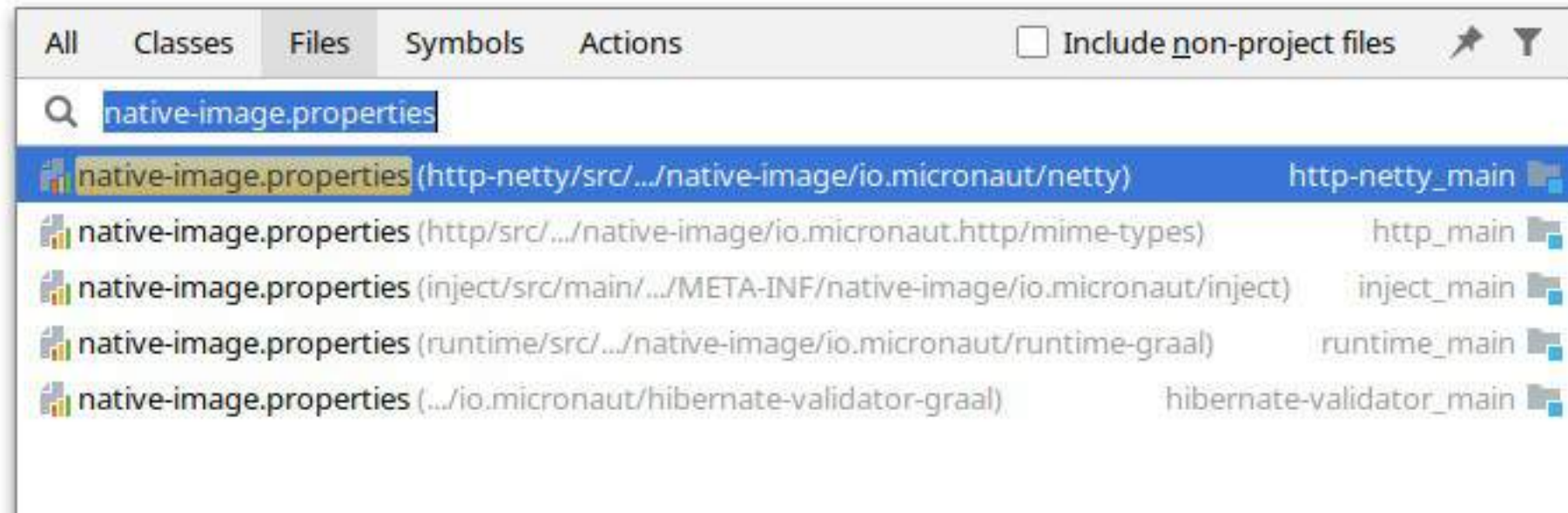
```
native-image --no-server -cp build/libs/basic-app-*.jar
```

```
# build-native-image.sh
native-image --no-server
  --class-path build/libs/basic-app-0.1-all.jar \
  -H:ReflectionConfigurationFiles=reflect.json \
  -H:EnableURLProtocols=http \
  -H:IncludeResources="logback.xml,application.properties" \
  -H:Name=basic-app \
  -H:Class=example.micronaut.Application \
  -H:+ReportUnsupportedElementsAtRuntime \
  -H:+AllowVMInspection \
  --allow-incomplete-classpath \
  --rerun-class-initialization-at-runtime=sun.security.jca.JCAUtil$CachedSecureRandomHolder,
  javax.net.ssl.SSLContext \
  --delay-class-initialization-to-runtime=io.netty.handler.codec.http.HttpObjectEncoder,
  io.netty.handler.codec.http.websocketx.WebSocket00FrameEncoder,
  io.netty.handler.ssl.util.ThreadLocalInsecureRandom,
  com.sun.jndi.dns.DnsClient
```

# How does it work?



- Composable `native-image.properties`
- `src/main/resources/META-INF/native-image/example.micronaut/app`



# Micronaut packages everything



```
# inject/native-image.properties
```

```
Args = -H:+ReportUnsupportedElementsAtRuntime \  
      --allow-incomplete-classpath
```

```
# http/native-image.properties
```

```
Args = -H:IncludeResources=META-INF/http/mime.types
```

```
# http-netty/native-image.properties
```

```
Args = --rerun-class-initialization-at-runtime=  
      sun.security.jca.JCAUtil$CachedSecureRandomHolder, javax.net.ssl.SSLContext \  
      --delay-class-initialization-to-runtime=  
io.netty.handler.codec.http.HttpObjectEncoder, io.netty.handler.codec.http.websocketx.WebSocket  
00FrameEncoder, io.netty.handler.ssl.util.ThreadLocalInsecureRandom, com.sun.jndi.dns.DnsClient,  
io.netty.handler.ssl.ConscryptAlpnSslEngine, io.netty.handler.ssl.JettyNpnSslEngine, io.netty.ha  
ndler.ssl.ReferenceCountedOpenSslEngine, io.netty.handler.ssl.JdkNpnApplicationProtocolNegotiat  
or, io.netty.handler.ssl.ReferenceCountedOpenSslServerContext, io.netty.handler.ssl.ReferenceCou  
ntedOpenSslClientContext, io.netty.handler.ssl.util.BouncyCastleSelfSignedCertGenerator \  
      -H:EnableURLProtocols=http,https
```

# Micronaut support



- @Introspected
- @TypeHint
- @ReflectiveAccess
- Custom src/main/graal/reflect.json
- Graal TypeElementVisitor



# GraalTypeElementVisitor



- `micronaut-core/graal/src/main/java/io/micronaut/graal/reflect/GraalTypeElementVisitor.java`
- Generates `reflection-config.json` and `native-image.properties`

# GraalTypeElementVisitor (II)



```
...
:compileJava
Note: Writing native-image.properties file to destination:
      META-INF/native-image/io.micronaut/http/native-image.properties
Note: Writing reflection-config.json file to destination:
      META-INF/native-image/io.micronaut/http/reflection-config.json

:http-netty:compileJava
Note: Writing native-image.properties file to destination:
      META-INF/native-image/io.micronaut/http-netty-channel/native-image.properties
Note: Writing reflection-config.json file to destination:
      META-INF/native-image/io.micronaut/http-netty-channel/reflection-config.json

:runtime:compileJava
Note: Writing native-image.properties file to destination:
      META-INF/native-image/io.micronaut/health/native-image.properties
Note: Writing reflection-config.json file to destination:
      META-INF/native-image/io.micronaut/health/reflection-config.json
...
```

# DEMO

# Demo



```
ivan@nobita: ~/micronaut-basic-app
File Edit View Search Terminal Tabs Help

ivan@nobita: ~/micronaut-basic-app
ivan@nobita:~/micronaut-basic-app $ sdk use java 1.0.0-rc-15-grl
Using java version 1.0.0-rc-15-grl in this shell.
ivan@nobita:~/micronaut-basic-app $ java -version
openjdk version "1.8.0_202"
OpenJDK GraalVM CE 1.0.0-rc15 (build 25.202-b08-jvnci-0.58, mixed mode)
ivan@nobita:~/micronaut-basic-app $ ./gradlew assemble
> Task :compileJava
Note: Writing native-image.properties file to destination: META-INF/native-image/example/micronaut/native-image.properties
Note: Writing reflection-config.json file to destination: META-INF/native-image/example/micronaut/reflection-config.json
Note: Creating bean classes for 2 type elements
> Task :compileGroovy NO-SOURCE
> Task :processResources
> Task :classes
> Task :jar
> Task :startScripts
> Task :distTar
> Task :distZip
> Task :shadowJar
> Task :startShadowScripts
> Task :shadowDistTar
> Task :shadowDistZip
> Task :assemble

BUILD SUCCESSFUL in 7s
10 actionable tasks: 10 executed
ivan@nobita:~/micronaut-basic-app $ native-image --no-server --class-path build/libs/basic-app-0.1.jar
```



# How do we test this?



- Creating a native-image takes a lot of time
- And needs a lot of RAM
- GraalVM changes so fast (and they break things)...
- ...and we also do it ;-)
- We use Travis for Micronaut, but...
- ...Travis has limits for RAM
- We needed another way



# GitLab

# Tests using Gitlab CI



- Scheduled job every hour
- Only if new commits in Micronaut or Graal
- Compile Graal from source code
- Create native image for Micronaut test applications
- Run functional tests
- <https://gitlab.com/micronaut-projects/micronaut-graal-tests>

# Micronaut Test applications



- Basic application: DI, POJO serialization, reactive type
- Micronaut function: AWS Function
- Service discovery: Consul support & HTTP Client
- Static resources & views
- Management & metrics endpoints
- Distributed tracing: Zipkin
- RabbitMQ: Fire-and-forget & RPC



# Gitlab CI



The screenshot shows the GitLab CI Pipelines page for the 'micronaut-graal-tests' project. The browser address bar shows the URL: <https://gitlab.com/micronaut-projects/micronaut-graal-tests/pipelines>. The page header includes the GitLab logo, navigation links (Projects, Groups, Snippets, Help), a search bar, and a 'Sign in / Register' button. The left sidebar contains a navigation menu with options like Project, Repository, Issues, Merge Requests, CI / CD (selected), Pipelines, Jobs, Schedules, Charts, Registry, Packages, Wiki, Snippets, and Members. The main content area displays a list of pipeline runs. At the top, it shows 'All 850' pipelines, with filters for Pending (0), Running (0), and Finished (825). The table below lists individual pipeline runs with columns for Status, Pipeline ID, Commit, Stages, and Duration. All listed pipelines are in a 'passed' state.

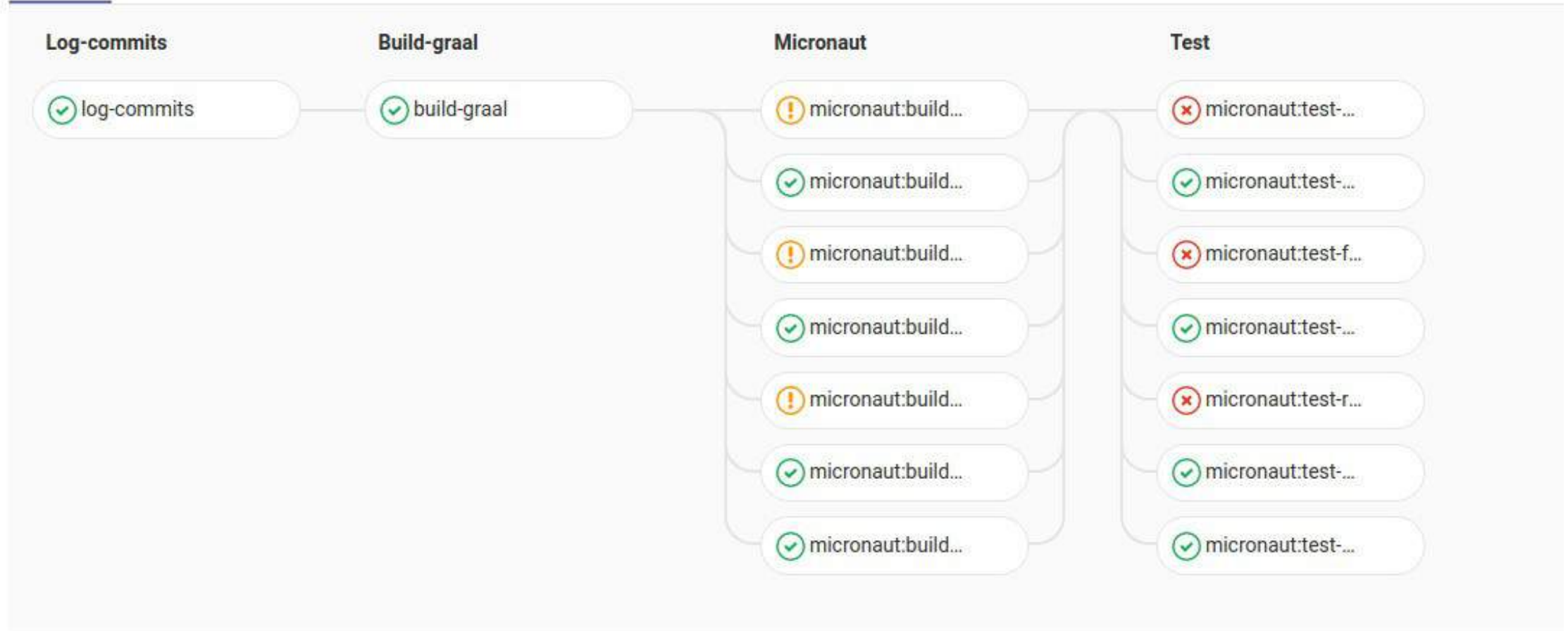
Status	Pipeline	Commit	Stages	Duration	Time Ago
passed	#56197764 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:28:54	2 hours ago
passed	#56190346 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:27:32	3 hours ago
passed	#56182536 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:27:40	4 hours ago
passed	#56174077 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:29:36	5 hours ago
passed	#56159321 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	01:02:54	6 hours ago
passed	#56146798 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:54:57	7 hours ago
passed	#56132741 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	01:13:26	8 hours ago
passed	#56120516 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:35:46	9 hours ago
passed	#56109004 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:27:49	10 hours ago
passed	#56085593 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:40:26	12 hours ago
passed	#56074202 by  latest	Y master -> 26a91c6a Use openjdk1.8.0_202-jvmci-0...	✓ ✓ ✓ ✓	00:37:32	13 hours ago

# Gitlab CI



The screenshot shows the GitLab CI pipeline interface for a project named "micronaut-graal-tests". The browser address bar indicates the URL: <https://gitlab.com/micronaut-projects/micronaut-graal-tests/pipelines/56197764>. The pipeline is titled "Use openjdk1.8.0\_202-jvmci-0.58" and is marked as "passed". It was triggered 3 hours ago by Iván López. The pipeline summary shows 16 jobs from master in 28 minutes and 54 seconds (queued for 1 second). The pipeline jobs are organized into four stages: "Log-commits", "Build-graal", "Micronaut", and "Test". Each stage contains multiple jobs, all of which are marked as successful with green checkmarks. The "Log-commits" stage has 1 job. The "Build-graal" stage has 1 job. The "Micronaut" stage has 7 jobs. The "Test" stage has 7 jobs. The left sidebar shows the project navigation menu, including "Project", "Repository", "Issues", "Merge Requests", "CI / CD", "Pipelines", "Jobs", "Schedules", "Charts", "Registry", "Packages", "Wiki", "Snippets", and "Members".

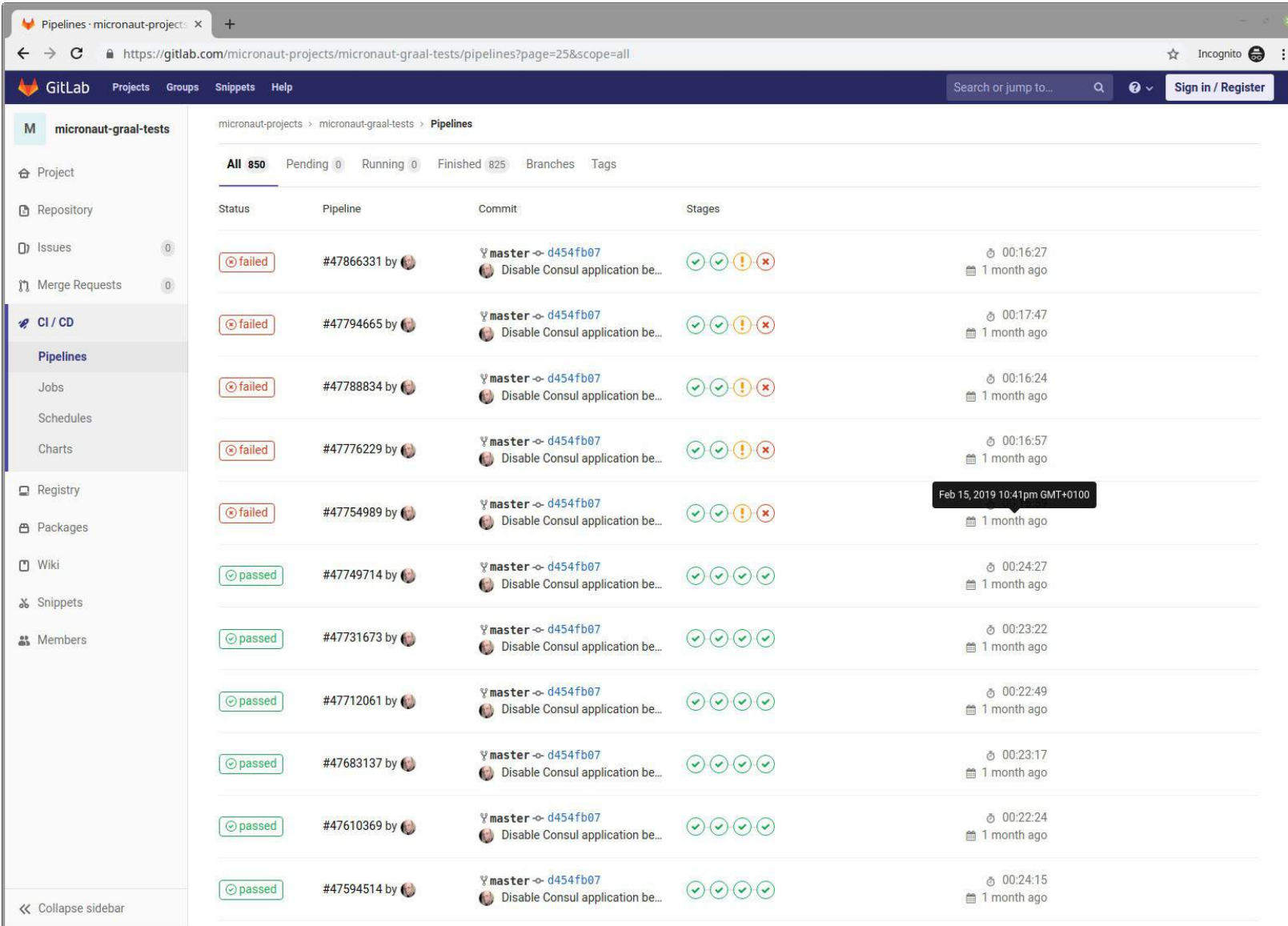
Pipeline Jobs 16 Failed Jobs 6



```
> Task :compileJava
Note: Writing native-image.properties file to destination: META-INF/native-image/example/micronaut/native-image.properties
Note: Writing reflection-config.json file to destination: META-INF/native-image/example/micronaut/reflection-config.json
Note: Creating bean classes for 1 type elements
/builds/micronaut-projects/micronaut-graal-tests/micronaut-basic-app/src/main/java/example/micronaut/GreetingService.java:5: error:
package javax.inject does not exist
import javax.inject.Singleton;
                    ^

/builds/micronaut-projects/micronaut-graal-tests/micronaut-basic-app/src/main/java/example/micronaut/GreetingService.java:7: error:
cannot find symbol
@Singleton
^
  symbol: class Singleton
2 errors

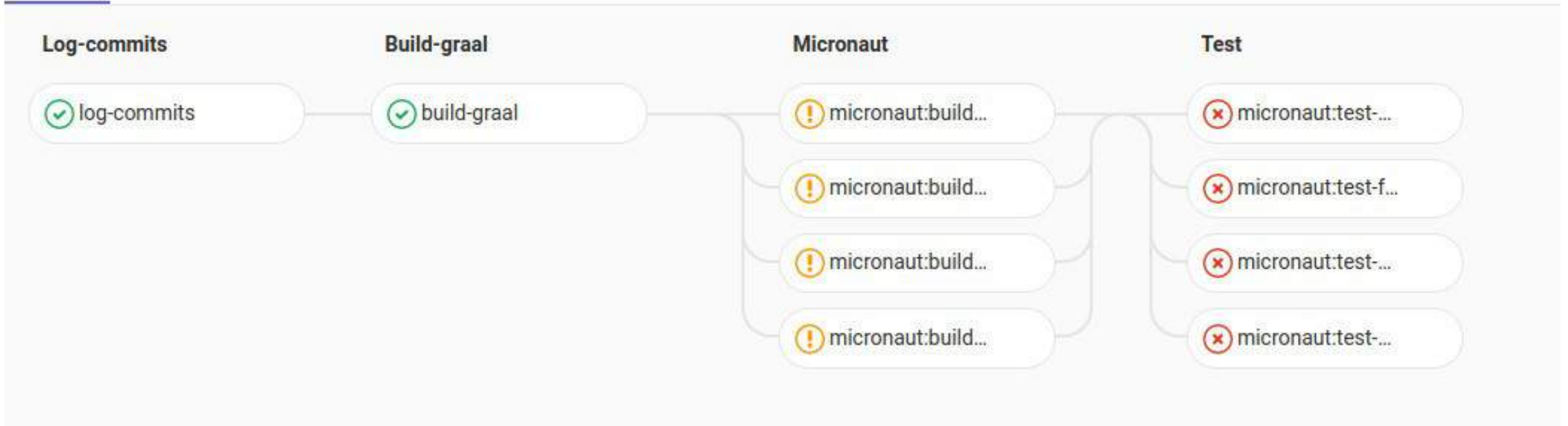
> Task :compileJava FAILED
```



The screenshot shows the GitLab CI Pipelines page for the project `micronaut-projects/micronaut-graal-tests`. The page displays a list of pipeline runs with their status, ID, commit, and stages. The status of each pipeline is indicated by a colored box: red for failed and green for passed. The stages are represented by icons: a checkmark for successful stages, an exclamation mark for failed stages, and an 'x' for canceled stages. The commit for all pipelines is `master -> d454fb07`. The stages for each pipeline are: `Disable Consul application be...`. The pipeline runs are sorted by time, with the most recent at the top. A tooltip is visible over the pipeline run #47754989, showing the date and time: `Feb 15, 2019 10:41pm GMT+0100`.

Status	Pipeline	Commit	Stages	Duration	Created
failed	#47866331 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ! ✗	00:16:27	1 month ago
failed	#47794665 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ! ✗	00:17:47	1 month ago
failed	#47788834 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ! ✗	00:16:24	1 month ago
failed	#47776229 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ! ✗	00:16:57	1 month ago
failed	#47754989 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ! ✗	00:16:24	1 month ago
passed	#47749714 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:24:27	1 month ago
passed	#47731673 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:23:22	1 month ago
passed	#47712061 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:22:49	1 month ago
passed	#47683137 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:23:17	1 month ago
passed	#47610369 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:22:24	1 month ago
passed	#47594514 by [user]	master -> d454fb07 Disable Consul application be...	✓ ✓ ✓ ✓	00:24:15	1 month ago

Pipeline Jobs 10 Failed Jobs 8





```
Graal Class Loading Analysis Enabled.
Graal Class Loading Analysis Enabled.
Writing reflect.json file to destination: build/reflect.json
[basic-app:96]   classlist:  9,785.03 ms
[basic-app:96]   (cap):    3,001.95 ms
[basic-app:96]   setup:    6,848.65 ms
[basic-app:96]   analysis: 47,532.28 ms
Fatal error: java.lang.IndexOutOfBoundsException: index 2
    at java.util.concurrent.atomic.AtomicReferenceArray.checkedByteOffset(AtomicReferenceArray.java:78)
    at java.util.concurrent.atomic.AtomicReferenceArray.get(AtomicReferenceArray.java:125)
    at com.oracle.graal.pointsto.flow.context.object.AnalysisObject.getInstanceFieldTypeStore(AnalysisObject.java:213)
    at com.oracle.graal.pointsto.flow.context.object.AnalysisObject.getInstanceFieldFlow(AnalysisObject.java:199)
    at com.oracle.graal.pointsto.flow.LoadFieldTypeFlow$LoadInstanceFieldTypeFlow.onObservedUpdate(LoadFieldTypeFlow.java:159)
    at com.oracle.graal.pointsto.flow.TypeFlow.notifyObservers(TypeFlow.java:347)
    at com.oracle.graal.pointsto.flow.TypeFlow.update(TypeFlow.java:389)
    at com.oracle.graal.pointsto.BigBang$2.run(BigBang.java:508)
    at com.oracle.graal.pointsto.util.CompletionExecutor.lambda$execute$0(CompletionExecutor.java:169)
    at java.util.concurrent.ForkJoinTask$RunnableExecuteAction.exec(ForkJoinTask.java:1402)
    at java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:289)
    at java.util.concurrent.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1056)
    at java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1692)
    at java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:157)
Error: Image building with exit status 1
ERROR: Job failed: exit code 1
```

## Regression introduced: IndexOutOfBoundsException creating Micronaut native-image apps #985

New Issue

**Closed** ilopmar opened this issue on Feb 16 · 3 comments



ilopmar commented on Feb 16

Our CI has detected a regression introduced in this commit: [d4558fb](#)

All of our Micronaut Graal test applications fail when building the native image with the following error:

```
[basic-app:96] classlist: 9,785.03 ms
[basic-app:96] (cap): 3,001.95 ms
[basic-app:96] setup: 6,848.65 ms
[basic-app:96] analysis: 47,532.28 ms
Fatal error: java.lang.IndexOutOfBoundsException: index 2
  at java.util.concurrent.atomic.AtomicReferenceArray.checkedByteOffset(AtomicReferenceArray.java:11)
  at java.util.concurrent.atomic.AtomicReferenceArray.get(AtomicReferenceArray.java:11)
  at com.oracle.graal.pointsto.flow.context.object.AnalysisObject.getInstanceFieldTypeFlow(AnalysisObject.java:11)
  at com.oracle.graal.pointsto.flow.context.object.AnalysisObject.getInstanceFieldTypeFlow(AnalysisObject.java:11)
  at com.oracle.graal.pointsto.flow.LoadFieldTypeFlow$LoadInstanceFieldTypeFlow.onObserve(LoadFieldTypeFlow.java:11)
  at com.oracle.graal.pointsto.flow.TypeFlow.notifyObservers(TypeFlow.java:347)
  at com.oracle.graal.pointsto.flow.TypeFlow.update(TypeFlow.java:389)
  at com.oracle.graal.pointsto.BigBang$2.run(BigBang.java:508)
  at com.oracle.graal.pointsto.util.CompletionExecutor.lambda$execute$0(CompletionExecutor.java:11)
  at java.util.concurrent.ForkJoinTask$RunnableExecuteAction.exec(ForkJoinTask.java:14)
  at java.util.concurrent.ForkJoinTask.doExec(ForkJoinTask.java:289)
  at java.util.concurrent.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1056)
  at java.util.concurrent.ForkJoinPool.runWorker(ForkJoinPool.java:1692)
  at java.util.concurrent.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:157)
Error: Image building with exit status 1
```

In this CI execution <https://gitlab.com/micronaut-projects/micronaut-graal-tests/pipelines/47754989> you can check that all the applications fail in the "Micronaut" stage. This is the stage we use to build native images with latest GraalVM built from master in the "Build-graal" pipeline stage.

To reproduce the issue you can use any of our test applications used in CI, for example:

- `git clone https://github.com/micronaut-graal-tests/micronaut-basic-app`
- `cd micronaut-basic-app/`
- Use GraalVM built from master
- `./build-native-image.sh`

👍 1

### Assignees

- vjovanov
- cstanacu

### Labels

- bug
- substrate

### Milestone

No milestone

### 4 participants



cstanacu commented on Feb 16

Member ...

Thanks for the quick report! I caught this on our own integration tests too and will have a fix soon.

👍 2

thomaswue assigned **cstanacu** on Feb 17

thomaswue added labels on Feb 17

cstanacu referenced this issue on Feb 19

**[native-image] wrong type in AnalysisObject #995**

**Closed**

vjovanov self-assigned this on Feb 19

dougxc pushed a commit that referenced this issue on Feb 20

[GR-14006] Fix `DynamicHub.getProtectionDomain()` substitution [#985, #995] ... 9f05467



cstanacu commented on Feb 20

Member ...

The issue is fixed in [9f05467](#). The problem was with the `DynamicHub.getProtectionDomain()` substitution: it declared `Object` return type instead of `ProtectionDomain`, so the points-to analysis was not applying the expected type narrowing. I also improved the error reporting: instead of an `ArrayIndexOutOfBoundsException` now you would get `AnalysisError$FieldNotPresentError: Field java.security.ProtectionDomain.codesource is not present on type java.lang.Object. Error encountered while analysing java.security.ProtectionDomain.getCodeSource()`.

cstanacu closed this on Feb 20



ilopmar commented on Feb 20

Author ...

Thanks! I can confirm that our integration tests pass again 🙌

👍 1



# Summary



GraalVM is new and exciting technology



Micronaut AoT is the perfect fit



Micronaut support out-of-the-box



We'll keep improving the support



Fast start-up & low memory consumption



Developers are happy & productive

# Questions?



OBJECT  
COMPUTING

## CONNECT WITH US

---

 1+ (314) 579-0066

 @objectcomputing

 [objectcomputing.com](http://objectcomputing.com)