**VERT.X PROJECTS**

After a very thorough research into what sort of tools or toolkit or platform for some API I have been planning for some time, vertx came up as the final and possible immediate solution or option for the level of server/application performance I was aiming for.

In my research I used the following websites for my Benchmarking checks and confirmations :

|  |
| --- |
| **Other Benchmarking websites :** |
| <https://dev.to/tuananhpham/popular-backend-frameworks-performance-benchmark-1bkh> |
| <https://www.tiobe.com/tiobe-index/> |
| <https://pypl.github.io/PYPL.html#google_vignette> |
| <https://trends.builtwith.com/framework> |
| <https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html> |
| <https://programming-language-benchmarks.vercel.app/> |
| <https://programming-language-benchmarks.vercel.app/java> |
| <https://programming-language-benchmarks.vercel.app/java-vs-go> |
| <https://www.quora.com/What-are-the-best-alternatives-to-Java-for-high-performance-backend-development> |
| <https://news.ycombinator.com/item?id=17254152> |
| <https://github.com/rwf2/Rocket/issues/710> |
| <https://just.billywhizz.io/blog/on-javascript-performance-01/> |
| [https://www.techempower.com](https://www.techempower.com/) |

Although, one could argue that the way benchmarking is done or executed, it might not be the same as what you might experience in a production environment. And for that I say : If the framework or platform or tool or toolkit or language of choice performed at this level for a very simple hello world app or sample project or in applying a. very simple hello world json structure, then , that is just it. All the other technologies are been benchmarked with same. You better believe it is just what it is. If you did pull the sample technologies and tested them your results would not be too far away nor apart from what has already been done.  
  
Below is a composite framework scoring ( source : techempower.com )

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **­­­** | | | |  | **Source** : *techempower.com* | | | | |  |  |  |  |
| *Each framework's peak performance in each test type (shown in the colored columns below) is multiplied by the weights shown above. The results are then summed to yield a weighted score. Only frameworks that implement all test types are included. 159 total frameworks ranked, 10 visible, 149 hidden by filters. See filter panel above.* | | | | | | | | | |  | **Hardware :** Citrine Dell R440 Xeon Gold + 10 GbE | | |
| Rnk | Framework | JSON | 1-query | 20-query | Fortunes | Updates | Plaintext | Weighted score | | Min | Max | Average |  |
| 5 | redkale | 1,210,086 | 457,935 | 32,272 | 413,537 | 22,900 | 6,981,831 | **7,096** | 87.80% | 22,900 | 6,981,831 | 1,519,760 |  |
| 7 | vert.x | 1,194,427 | 581,542 | 32,505 | 453,406 | 17,122 | 5,624,745 | **6,847** | 84.70% | 17,122 | 5,624,745 | 1,317,291 |  |
| 11 | jooby | 1,162,248 | 515,795 | 32,196 | 427,682 | 15,889 | 4,801,653 | **6,370** | 78.80% | 15,889 | 4,801,653 | 1,159,244 |  |
| 16 | vertx-web | 1,121,889 | 566,804 | 30,669 | 388,512 | 16,598 | 3,774,206 | **6,107** | 75.50% | 16,598 | 3,774,206 | 983,113 |  |
| 19 | inverno | 1,042,404 | 467,759 | 32,441 | 309,684 | 15,864 | 4,043,496 | **5,566** | 68.80% | 15,864 | 4,043,496 | 985,275 |  |
| 38 | quarkus | 903,185 | 318,897 | 17,610 | 214,275 | 6,697 | 2,861,479 | **3,637** | 45.00% | 6,697 | 2,861,479 | 720,357 |  |
| 50 | actframework | 964,004 | 231,641 | 16,942 | 124,422 | 1,985 | 3,273,101 | **2,911** | 36.00% | 1,985 | 3,273,101 | 768,683 |  |
| 55 | javalin | 512,495 | 211,243 | 16,582 | 161,275 | 10,405 | 897,788 | **2,755** | 34.10% | 10,405 | 897,788 | 301,631 |  |
| 57 | revenj.jvm | 527,667 | 290,147 | 14,330 | 177,853 | 5,206 | 730,158 | **2,543** | 31.40% | 5,206 | 730,158 | 290,894 |  |
| 125 | wicket | 378,043 | 24,624 | 1,426 | 25,882 | 551 | 516,369 | **679** | 8.40% | 551 | 516,369 | 157,816 |  |

I had been pretty much used to developing some very sophisticated API solutions using micronaut for some fintech backend that had processed in excess of $40million by the time it was been considered for an acquisition deal/discussion. And at the point, the final part of the negotiation(s) was left to my C.E.O to take over.

So in my quest to decide on what tools, or tool-kit or framework and platform to use for my next big fintech backend vertx had become the obvious. Meanwhile, I took some time off to consider the differences and general comparison between micronaut and vertx.

**Micronaut Framework vs Vert.x: What are the differences?**

**Introduction**

Micronaut Framework and Vert.x are both popular frameworks used for developing microservices and reactive applications. While they share some similarities, there are key differences that set them apart from each other.

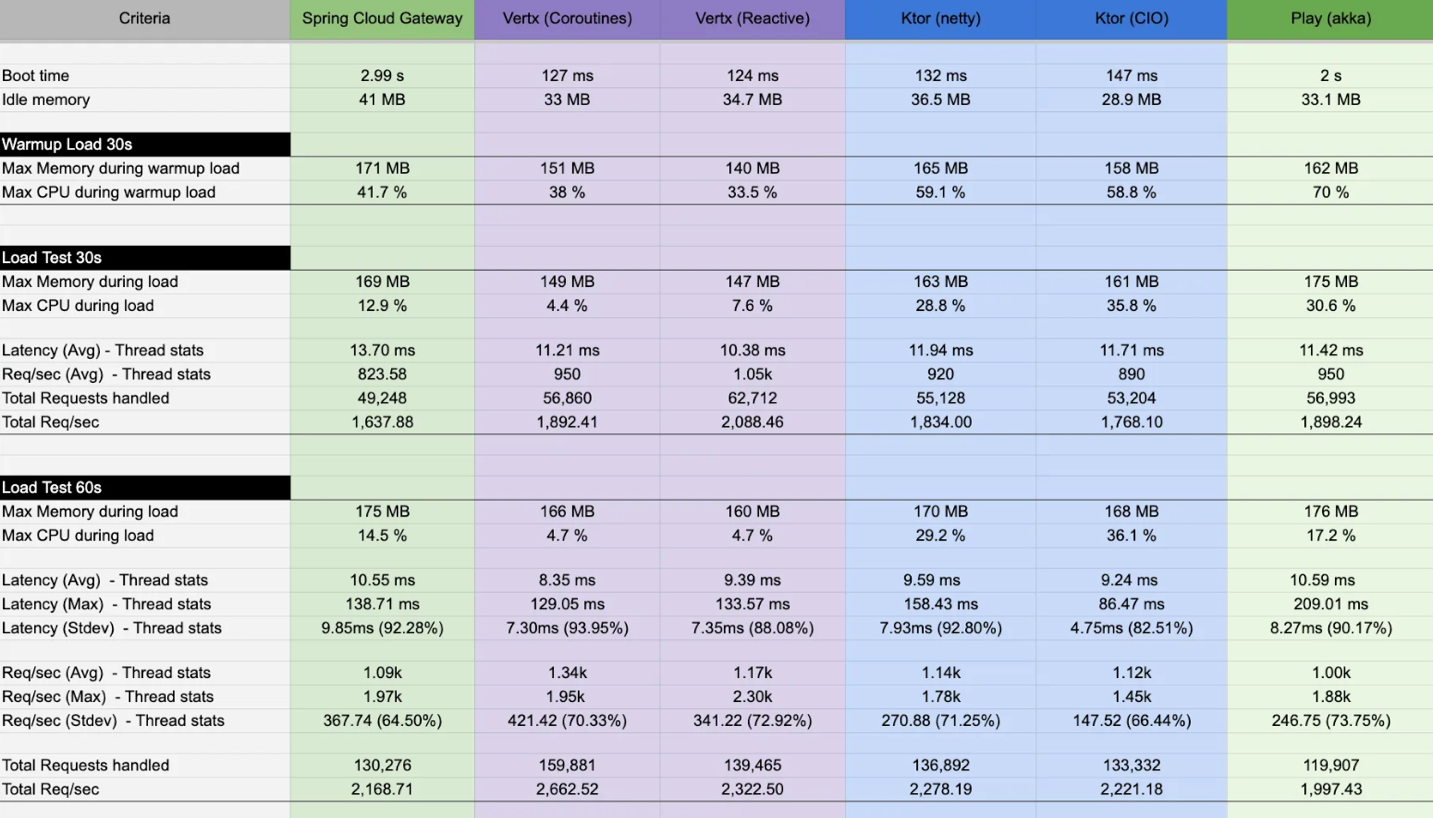
1. **Execution Model**: *Micronaut* adopts a more traditional Java execution model, utilizing compile-time annotation processing to minimize reflection and enhance performance. On the other hand, *Vert.x* is event-driven and non-blocking, making it well-suited for highly concurrent applications.
2. **Language Support**: *Micronaut* primarily focuses on providing support for the Java programming language, although it also offers limited compatibility with Kotlin and Groovy. Alternatively, *Vert.x* is polyglot, meaning it supports multiple languages, including Java, Kotlin, JavaScript, Groovy, Ruby, Python, and more.
3. **Dependency Injection**: *Micronaut* has a built-in dependency injection framework that leverages compile-time DI, leading to faster startup times and decreased memory consumption. Conversely, *Vert.x* does not have its own DI framework and instead encourages the use of external libraries such as Dagger or Spring for dependency injection.
4. **Web Support**: *Micronaut* provides comprehensive support for building RESTful APIs and web applications out-of-the-box, including features like server-side templating and built-in support for HTTP clients. *Vert.x*, on the other hand, offers more low-level control over the web stack and allows for the creation of various types of applications, including not only traditional web apps but also real-time websockets and event-driven web systems.
5. **Concurrency Model**: *Micronaut* leverages thread pools and CompletableFuture for performing tasks concurrently and handling asynchronous operations. *Vert.x*, being an event-driven framework, uses an event loop model and employs a single-threaded model, wherein a single event loop can handle multiple requests concurrently by utilizing non-blocking I/O operations.
6. **Development Approach**: *Micronaut* utilizes a compile-time approach, where it analyzes your project's classpath during the build phase to generate factory classes and metadata. This, in turn, reduces the amount of reflection required at runtime. *Vert.x*, on the other hand, takes a more runtime approach and relies more heavily on dynamic features and runtime reflection.

In Summary, Micronaut Framework focuses on optimizing performance, enables compile-time DI, and provides efficient Java support, while Vert.x emphasizes its polyglot nature, event-driven architecture, and flexible web support.

*[ source : stackshare.io ]*

**Comparing : spring vs vert.x vs ktor vs playframework(akka)**

<https://leondesilva.medium.com/spring-vs-vert-x-vs-ktor-vs-play-framework-the-performance-battle-for-a-jvm-based-api-gateway-6eebbbd4d1b5>



So basically after trying out a number of the technologies listed thereof, I settled on vert.x.



Starting a vertx project is as simple as getting your preferred IDE eg. IntelliJ Community Package version or Eclipse or Visual Studio Code and creating a maven or gradle project and ensuring a dependency to and for io.vertx is set in place.  
  
Or you can start from the vertx starter link to generate a downloadable package or project to start from after successfully importing into the/an IDE as a project.

You can use the Vert.x starter web application at https://start.vertx.io and generate a project skeleton to download



After successfully importing the project,

You can then run the following command :

$ mvn clean install

You can expect some <Build Successful> kind of message at the end to be sure that everything worked out perfectly, as seen below :

[INFO] Copying com.aireceive.firstvertxapp:starter:pom:1.0.0-SNAPSHOT to project local repository

[INFO] Copying com.aireceive.firstvertxapp:starter:jar:1.0.0-SNAPSHOT to project local repository

[INFO] Copying com.aireceive.firstvertxapp:starter:pom:consumer:1.0.0-SNAPSHOT to project local repository

[INFO] --------------------------------------------------------------------------------------------------------------------------

[INFO] **BUILD SUCCESS**

[INFO] --------------------------------------------------------------------------------------------------------------------------

[INFO] Total time: 3.020 s

[INFO] Finished at: 2024-11-04T14:38:33Z

[INFO] --------------------------------------------------------------------------------------------------------------------------

To install the application and the respective dependencies.

**Caution :**

I tried the following commands and it did not work out as expected :

$ vertx run com.arieceive.firstvertxapp.starter.MainMerticle

And the application did not run but rather I had the following error :

Unrecognized VM option 'UseBiasedLocking'

Error: Could not create the Java Virtual Machine.

Error: A fatal exception has occurred. Program will exit.

I also tried the same command and later this error :

Can not find io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider in the classpath, fallback to system defaults. This may result in incorrect DNS resolutions on MacOS. Check whether you have a dependency on 'io.netty:netty-resolver-dns-native-macos'

Failed in deploying verticle

java.lang.ClassNotFoundException: com.arieceive.firstvertxapp.starter.MainMerticle

at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:581)

at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:178)

at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:527)

at io.vertx.core.impl.JavaVerticleFactory.createVerticle(JavaVerticleFactory.java:41)

at io.vertx.core.impl.VerticleManager.doDeployVerticle(VerticleManager.java:217)

at io.vertx.core.impl.VerticleManager.doDeployVerticle(VerticleManager.java:193)

at io.vertx.core.impl.VerticleManager.doDeployVerticle(VerticleManager.java:180)

at io.vertx.core.impl.VerticleManager.deployVerticle(VerticleManager.java:156)

at io.vertx.core.impl.VertxImpl.deployVerticle(VertxImpl.java:794)

at io.vertx.core.impl.VertxImpl.deployVerticle(VertxImpl.java:800)

at io.vertx.core.impl.launcher.commands.VertxIsolatedDeployer.deploy(VertxIsolatedDeployer.java:42)

at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.base/java.lang.reflect.Method.invoke(Method.java:566)

at io.vertx.core.impl.launcher.commands.ClasspathHandler.deploy(ClasspathHandler.java:169)

at io.vertx.core.impl.launcher.commands.RunCommand.deploy(RunCommand.java:398)

at io.vertx.core.impl.launcher.commands.RunCommand.run(RunCommand.java:260)

at io.vertx.core.impl.launcher.VertxCommandLauncher.execute(VertxCommandLauncher.java:248)

at io.vertx.core.impl.launcher.VertxCommandLauncher.dispatch(VertxCommandLauncher.java:383)

at io.vertx.core.impl.launcher.VertxCommandLauncher.dispatch(VertxCommandLauncher.java:346)

at io.vertx.core.Launcher.main(Launcher.java:45)

So the next thing was to try running the generated directly.

I tried : ( from the root folder of the project )

$ java -jar target/starter-1.0.0-SNAPSHOT.jar

And I got this feedback :

no main manifest attribute, in target/starter-1.0.0-SNAPSHOT.jar

What worked out find in getting application running :

But at last, after trying this :

$ java -jar target/starter-1.0.0-SNAPSHOT-fat.jar

I was then able to get the application running with a successful test using a web browser.  
  
Nov 04, 2024 2:34:20 PM io.netty.resolver.dns.DnsServerAddressStreamProviders <clinit>

WARNING: Can not find io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider in the classpath, fallback to system defaults. This may result in incorrect DNS resolutions on MacOS. Check whether you have a dependency on 'io.netty:netty-resolver-dns-native-macos'

HTTP server started on port 8888

Nov 04, 2024 2:34:20 PM io.vertx.core.impl.launcher.commands.VertxIsolatedDeployer

INFO: Succeeded in deploying verticle

My first vertx web app :

[INFO] Copying com.aireceive.firstvertxwebapp:vertx-web-starter:pom:1.0.0-SNAPSHOT to project local repository

[INFO] Copying com.aireceive.firstvertxwebapp:vertx-web-starter:jar:1.0.0-SNAPSHOT to project local repository

[INFO] Copying com.aireceive.firstvertxwebapp:vertx-web-starter:pom:consumer:1.0.0-SNAPSHOT to project local repository

[INFO] --------------------------------------------------------------------------------------------------------------------------

[INFO] **BUILD SUCCESS**

[INFO] --------------------------------------------------------------------------------------------------------------------------

[INFO] Total time: 3.472 s

[INFO] Finished at: 2024-11-04T16:07:02Z

[INFO] --------------------------------------------------------------------------------------------------------------------------

➜ vertx-web-starter git:(main) ✗ java -jar target/vertx-web-starter-1.0.0-SNAPSHOT-fat.jar

Nov 04, 2024 4:07:18 PM io.netty.resolver.dns.DnsServerAddressStreamProviders <clinit>

WARNING: Cannot find io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider in the classpath, fallback to system defaults. This may result in incorrect DNS resolutions on MacOS. Check whether you have a dependency on 'io.netty:netty-resolver-dns-native-macos'

HTTP server started on port 8889

Nov 04, 2024 4:07:18 PM io.vertx.core.impl.launcher.commands.VertxIsolatedDeployer

INFO: Succeeded in deploying verticle



Alternative to Vert.x for asynchronous and reactive programming :

1. NodeJS
2. Akka
3. Spring Framework
4. Quarkus
5. Netty
6. Scripting languages
7. Native languages

Vert.x is the best compared to all the above in so many ways, thanks to the JVM to its advantage. Some of the other options above might have their own benefits in various ways meanwhile for a high performant production system, Vertx would always win in over 82% of the key metrics been checked against.

Summary to this point :

1. Asynchronous programming allows you to handle multiple multiplex networked connections on a single thread.
2. Handling or managing non-blocking I/O is more complex than the equivalent imperative code base on blocking I/O, even for simple protocols.
3. Asynchronous event processing is simplified by the event loop and the reactor pattern.
4. Despite the demanding workloads and failures, a reactive system is both scalable and resilient, producing responses with consistent latencies.
5. Vert.x is an efficient and approachable toolkit for writing asynchronous and reactive applications on the JVM.

**Verticles**

A verticle is the fundamental processing unit in Vert.x, with a life cycle. The role of a verticle is fundamentally to envelope a single technical functional unit for processing events, such as ;

1. Exposing an HTTP API
2. Responding to requests
3. Providing a repository interface on top of a database
4. Issuing requests to a third-party system

Characteristics of Verticles :

1. Verticles exclusively can communicate with other entities by sending and responding to messages.
2. Verticles have private state that may be updated when receiving events, they can deploy other verticles.
3. Verticles can communicate via message-passing
4. Verticles do not necessarily follow the orthodox definition of actors

Now, let’s build a cute little verticle that processes just two types of events for now :

1. Periodic timers
2. Handle HTTP requests

The life cycle of a verticle is basically :

1. Start , and
2. Stop

The start method is typically used in setting things up and fundamentally initializing handlers.

The stop method is used in doing housekeeping tasks, such as closing an opened database connection(s).

To run a verticle, it can basically be run from the mian of the java class that it is in.

Or you can also run it on the command line using Gradle such as :

**$ ./gradlew run -PmainClass=com.example.hello.HelloVerticle**

**Some few experiments :**

So I had to run a few experiments. I decided to check if I could run a single verticle and rather apply two port numbers to see how the behaviour of a dual starting of the same verticle would be and to my surprise, it is only the first verticle start via either the terminal or by directly running it from the “main” method that responds to all the requests to the running verticle.



In the deployment of verticles it is important to pay attention to any action or function or task or tasks that might attempt to block the eventLoop or life cycle of the said verticle. In Vertx, we have a threadchecker that checks to be sure that no process or action or function or task is taking more than a predefined internally set thread checker time limit. The default might be found to be 2000 as of the time I am doing all these tests and experiments. Meanwhile, there are certain environments such as embedded devices, where processing power is a bit slower, and it is normal to increase the thread-checker threshold for such cases.

You can do that by setting the the system properties to change the value as seen below :

-Dvertx.options.blockedThreadCheckInterval=5000

Or if you prefer to disable it :

-Dvertx.threadChecks=false

It must be noted that this configuration is global and cannot be fine-tuned on a per-verticle basis.

It is such a great robust way to or practice to use asynchronous method variants that accept a callback to notify of any errors, example the listen method in the creation of an HTTP server using vertx.

Do check from screenshot below :



**Deploying Verticles:**

We can have a verticle successfully deploy another verticle and repeat same for another verticle. Meanwhile, there is not direct parent or child relationship setup for verticle deployment.

One other thing to note about verticles are that Vert.x creates double number of event-loop threads as the number of CPU Cores present, by default. Basically implies that if you have 4cores, then a Vert.x application has 8 event loops. And the assignment of the verticles to event loops is/are done in a rounded-robin fashion or style.

In the deployment of verticles and by extension the number of event loops, can be managed in a way so as to obtain a certain number of event loops that has to be available. Meanwhile it is not possible to allocate a given verticle to a specific event loop manually. In practice, this should never be a problem whatsoever. It must be noted that we can plan the deployment order of verticles.



**Passing Configuration Data**

**Worker Verticles**

**Threads Contextualisation**

**Threads Mix Monitoring and Management**

**Event Bus : A very important part of Vert.x**

There are three major patterns in event bus communications :

1. Point-to-point messaging
2. Request-reply messaging
3. Publish/subscribe messaging

The event bus is not a message broker but rather carries volatile event that are been processed asynchronously by verticles, more specifically the Vert.x event bus is not able to do the following :

1. Support message acknowledgements
2. Support message priorities
3. Support message durability to recover from crashes
4. Provide routing rules
5. Provide transformation rules (schema adaptation, scatter/gather, etc.)
6. Distribute fewer messages to an overloaded consumer.

If the developer or engineer wants a way to handle communications relating to events without ever loosing them then a middleware would be very necessary.

**Point-to-Point messaging**

Messages from producers are shared proportionally in a round-robin fashion amongst consumers.

**Latter Sudden challenges in getting vert.x running smoothly & the way out :**

As I progressed into the deeper parts of vert.x development, deployment and its general usage. I was trying to run some vert.x application using version 4.5.10 and I had this error below :

Nov 06, 2024 11:58:45 AM io.netty.resolver.dns.DnsServerAddressStreamProviders <clinit>

WARNING: Can not find io.netty.resolver.dns.macos.MacOSDnsServerAddressStreamProvider in the classpath, fallback to system defaults. This may result in incorrect DNS resolutions on MacOS. Check whether you have a dependency on 'io.netty:netty-resolver-dns-native-macos'

I then compared the ‘pom.xml’ (under properties) file from a previous successfully running vert.x application :

<properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <vertx.version>4.0.3</vertx.version>

The new vertx application’s pom.xml file under the properties :

<properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <vertx.version>4.5.10</vertx.version>

The new vertx application’s pom.xml file above was then changed to :

<properties>  
 <maven.compiler.source>1.8</maven.compiler.source>  
 <maven.compiler.target>1.8</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 <vertx.version>4.0.3</vertx.version>

The application was then able to run smoothly without any issues whatsoever.

Access the index.html file :

On a macbook M1 using inteliJ and with the “index.html” at the root of the project, the file could not be rendered.

The file was only able to be rendered after placing the file inside the “resource” folder or package.

**Clustering and distributed event bus**

In attempting to run a cluster that would have brought the application development in vertx to a whole new level of deployment and management, I started getting errors to ensure some infinispan.xml is created before the cluster would run.

I tried running from the main : did not work without the infinispan.xml

I tried running using ./gradlew run : did not work without the infinispan.xml

I tried running using mvn compile exec:java : did not work without the infinispan.xml

**Infinispan Replicated Cache**



**Starting Infinispan Server**

Before you start coding fun stuff, you need to start Infinispan Server. For this tutorial, you need a locally running server instance.

You can do one of the following:

* Pull the container image and run with [Docker](https://www.docker.com/) or [Podman](https://podman.io/).
* Download the server distribution and extract it to your filesystem.

Credentials

By default, Infinispan Server requires user authentication. This tutorial uses admin and secret credentials but you can use any username and password.

**Running the Container Image**

The easiest way to run Infinispan Server locally is to pull the container image.

* Podman

podman run --net=host -p 11222:11222 -e USER="admin" -e PASS="secret" quay.io/infinispan/server:latest

* Docker

docker run -it -p 11222:11222 -e USER="admin" -e PASS="password" infinispan/server:latest

**Running the Server Distribution**

Infinispan Server comes as a bare metal distribution that you can run locally.

1. Download the server distribution from [Infinispan Downloads](https://infinispan.org/download/#stable) and extract it.
2. Open a terminal window in the resulting directory. This is $ISPN\_HOME.
3. Add credentials.

$ ./bin/cli.sh user create admin -p secret

Run Infinispan Server.

$ ./bin/server.sh

**Accessing the Infinispan Console**

Open <http://localhost:11222/> in any browser.

You’ll see the **Welcome to Infinispan Server** page.

[](https://github.com/infinispan/infinispan-server-tutorial/blob/main/images/welcomeConsole.png)

To start using the Infinispan Console, do the following:

1. Select **Go to the console**.
2. Enter your credentials (admin/secret).

Successfully have gotten infinispan running :



I then went on to generate this configuration in XML and JSON :

XML :

<?xml version="1.0"?>

<distributed-cache owners="2" mode="ASYNC" statistics="true">

<encoding media-type="application/x-protostream"/>

<locking concurrency-level="32" acquire-timeout="10"/>

<memory max-size="1024MB" when-full="REMOVE"/>

<persistence passivation="false" availability-interval="1000" connection-attempts="10" connection-interval="50">

<file-store>

<data path="data"/>

<index path="index"/>

</file-store>

</persistence>

</distributed-cache>

Improved to :

<distributed-cache *owners*="2" *mode*="ASYNC" *statistics*="true">  
 <encoding *media-type*="application/x-protostream"/>  
 <locking *concurrency-level*="32" *acquire-timeout*="10"/>  
 <memory *max-size*="1024MB" *when-full*="REMOVE"/>  
 <persistence *passivation*="false" *availability-interval*="1000" *connection-attempts*="10" *connection-interval*="50">  
 <file-store>  
 <data *path*="data"/>  
 <index *path*="index"/>  
 </file-store>  
 </persistence>  
 <infinispan>  
 <cache-container>  
 <transport *initial-cluster-size*="4"  
 *initial-cluster-timeout*="30000" />  
 </cache-container>  
 </infinispan>  
</distributed-cache>

Changed to :

<distributed-cache>

<memory storage="HEAP"/>

</distributed-cache>

Changed to :

<distributed-cache>  
 <memory  
 *storage*="HEAP"  
 *max-count*="500"  
 *when-full*="REMOVE"  
 />  
 <infinispan>  
 <cache-container>  
 <transport *initial-cluster-size*="4"  
 *initial-cluster-timeout*="30000" />  
 </cache-container>  
 </infinispan>  
  
</distributed-cache>

JSON :

{

"distributed-cache": {

"owners": "2",

"mode": "ASYNC",

"statistics": true,

"encoding": {

"media-type": "application/x-protostream"

},

"locking": {

"concurrency-level": "32",

"acquire-timeout": "10"

},

"memory": {

"max-size": "1024MB",

"when-full": "REMOVE"

},

"persistence": {

"passivation": false,

"availability-interval": "1000",

"connection-attempts": "10",

"connection-interval": "50",

"file-store": {

"data": {

"path": "data"

},

"index": {

"path": "index"

}

}

}

}

}

I then copied the XML version into the infinispan.xml file.

**Replicated cache:**

<replicated-cache>

<expiration lifespan="5000" max-idle="1000" />

</replicated-cache>

<replicated-cache>  
 <expiration *lifespan*="5000" *max-idle*="1000" />  
 <memory *storage*="OFF\_HEAP" *max-count*="500"/>  
  
 <infinispan>  
 <cache-container>  
 <transport *initial-cluster-size*="4"  
 *initial-cluster-timeout*="30000" />  
 </cache-container>  
 </infinispan>  
  
</replicated-cache>

<?*xml version*="1.0"?>  
<?*xml version*="1.0" *encoding*="UTF-8"?>  
<infinispan *xmlns:xsi*="http://www.w3.org/2001/XMLSchema-instance"  
 *xsi:schemaLocation*="urn:infinispan:config:7.1 http://www.infinispan.org/schemas/infinispan-config-7.1.xsd"  
 *xmlns*="urn:infinispan:config:7.1">  
  
 <cache-container *default-cache*="default">  
 <local-cache *name*="tableCache">  
 <eviction *max-entries*="200" />  
 <expiration *lifespan*="600000" />  
 </local-cache>  
 <transport  
 *initial-cluster-size*="4"  
 *initial-cluster-timeout*="30000" />  
 </cache-container>  
  
</infinispan>

Other configurations :

<distributed-cache>

<persistence>

<table-jdbc-store xmlns="urn:infinispan:config:store:sql:15.0"

dialect="H2"

shared="true"

table-name="books">

<connection-pool connection-url="jdbc:h2:mem:infinispan"

username="sa"

password="changeme"

driver="org.h2.Driver"/>

<write-behind modification-queue-size="2048"

fail-silently="true"/>

</table-jdbc-store>

</persistence>

</distributed-cache>

<distributed-cache>

<persistence>

<connection-pool connection-url="jdbc:h2:mem:infinispan;DB\_CLOSE\_DELAY=-1"

username="sa"

password="changeme"

driver="org.h2.Driver"/>

</persistence>

</distributed-cache>

##### Managed datasource configuration

XML

JSON

YAML

<server xmlns="urn:infinispan:server:15.0">

<data-sources>

<!-- Defines a unique name for the datasource and JNDI name that you

reference in JDBC cache store configuration.

Enables statistics for the datasource, if required. -->

<data-source name="ds"

jndi-name="jdbc/postgres"

statistics="true">

<!-- Specifies the JDBC driver that creates connections. -->

<connection-factory driver="org.postgresql.Driver"

url="jdbc:postgresql://localhost:5432/postgres"

username="postgres"

password="changeme">

<!-- Sets optional JDBC driver-specific connection properties. -->

<connection-property name="name">value</connection-property>

</connection-factory>

<!-- Defines connection pool tuning properties. -->

<connection-pool initial-size="1"

max-size="10"

min-size="3"

background-validation="1000"

idle-removal="1"

blocking-timeout="1000"

leak-detection="10000"/>

</data-source>

</data-sources>

</server>

##### Connection pool tuning properties

You can tune JDBC connection pools for managed datasources in your Infinispan Server configuration.

| **Property** | **Description** |
| --- | --- |
| initial-size | Initial number of connections the pool should hold. |
| max-size | Maximum number of connections in the pool. |
| min-size | Minimum number of connections the pool should hold. |
| blocking-timeout | Maximum time in milliseconds to block while waiting for a connection before throwing an exception. This will never throw an exception if creating a new connection takes an inordinately long period of time. Default is 0 meaning that a call will wait indefinitely. |
| background-validation | Time in milliseconds between background validation runs. A duration of 0 means that this feature is disabled. |
| validate-on-acquisition | Connections idle for longer than this time, specified in milliseconds, are validated before being acquired (foreground validation). A duration of 0 means that this feature is disabled. |
| idle-removal | Time in minutes a connection has to be idle before it can be removed. |
| leak-detection | Time in milliseconds a connection has to be held before a leak warning. |

After entering the command :

$ ./gradlew run

[ for vertx version 4.0.3 ] : The error message was seen below :

ERROR [vert.x-eventloop-thread-0] VertxImpl - Failed to initialize clustered Vert.x

org.infinispan.commons.CacheConfigurationException: ISPN000327: Cannot find a parser for element 'distributed-cache' in namespace ''. Check that your configuration is up-to date for Infinispan '11.0.5.Final' and if you have the proper dependency in the classpath

[ for vertx version 4.4.4 ] : The error message was seen below :

ERROR [vert.x-eventloop-thread-0] VertxImpl - Failed to initialize clustered Vert.x

org.infinispan.commons.CacheConfigurationException: ISPN000343: Must have a transport set in the global configuration in order to define a clustered cache

##### Distributed caches

XML

JSON

YAML

<distributed-cache owners="2"

segments="256"

capacity-factor="1.0"

l1-lifespan="5000"

mode="SYNC"

statistics="true">

<encoding media-type="application/x-protostream"/>

<locking isolation="REPEATABLE\_READ"/>

<transaction mode="FULL\_XA"

locking="OPTIMISTIC"/>

<expiration lifespan="5000"

max-idle="1000" />

<memory max-count="1000000"

when-full="REMOVE"/>

<indexing enabled="true"

storage="local-heap">

<index-reader refresh-interval="1000"/>

<indexed-entities>

<indexed-entity>org.infinispan.Person</indexed-entity>

</indexed-entities>

</indexing>

<partition-handling when-split="ALLOW\_READ\_WRITES"

merge-policy="PREFERRED\_NON\_NULL"/>

<persistence passivation="false">

<!-- Persistent storage configuration. -->

</persistence>

</distributed-cache>

##### Replicated caches

XML

JSON

YAML

<replicated-cache segments="256"

mode="SYNC"

statistics="true">

<encoding media-type="application/x-protostream"/>

<locking isolation="REPEATABLE\_READ"/>

<transaction mode="FULL\_XA"

locking="OPTIMISTIC"/>

<expiration lifespan="5000"

max-idle="1000" />

<memory max-count="1000000"

when-full="REMOVE"/>

<indexing enabled="true"

storage="local-heap">

<index-reader refresh-interval="1000"/>

<indexed-entities>

<indexed-entity>org.infinispan.Person</indexed-entity>

</indexed-entities>

</indexing>

<partition-handling when-split="ALLOW\_READ\_WRITES"

merge-policy="PREFERRED\_NON\_NULL"/>

<persistence passivation="false">

<!-- Persistent storage configuration. -->

</persistence>

</replicated-cache>

##### Multiple caches

XML

JSON

YAML

<infinispan

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="urn:infinispan:config:15.0 https://infinispan.org/schemas/infinispan-config-15.0.xsd

urn:infinispan:server:15.0 https://infinispan.org/schemas/infinispan-server-15.0.xsd"

xmlns="urn:infinispan:config:15.0"

xmlns:server="urn:infinispan:server:15.0">

<cache-container name="default"

statistics="true">

<distributed-cache name="mycacheone"

mode="ASYNC"

statistics="true">

<encoding media-type="application/x-protostream"/>

<expiration lifespan="300000"/>

<memory max-size="400MB"

when-full="REMOVE"/>

</distributed-cache>

<distributed-cache name="mycachetwo"

mode="SYNC"

statistics="true">

<encoding media-type="application/x-protostream"/>

<expiration lifespan="300000"/>

<memory max-size="400MB"

when-full="REMOVE"/>

</distributed-cache>

</cache-container>

</infinispan>

After trying all possible ways of getting the infinispan right, but with no success, I rather resorted to what I do when I stay with a problem for just so so long and there is still no breakthrough : I ask myself : “Is this going to be needed in production ? and is it the best for the current challenge or problem I ultimately need to solve, globally ?

So I went on to do a quick search :

And in my discovery, I came to realise, I needed to leave the problem behind and rather solve a more relevant one : how to deploy an enterprise application or server solution with all the high standards and not just some fancy localhost or localside ephemeral solutions with infinispan.

Now the twist is that :

Both Hazelcast and Infinispan, by default use multicast communications to discover nodes. This is great for local testing and many bare-metal server deployments, but multicast communications are not possible in a Kubernetes cluster. If you run the containers as is on Kubernetes, the heat sensor services and sensor gateway instances will not be able to communicate over the event bus.

These cluster managers can, of course, be configured to perform service discovery in Kubernetes. We will briefly cover the case of **Hazelcast**, where two discovery modes are possible:

 Hazelcast can connect to the Kubernetes API to listen for and discover pods matching a request, such as a desired label and value.

 Hazelcast can periodically make DNS queries to discover all pods for a given Kubernetes (headless) service.

The DNS approach is more limited.

Instead, let’s use the Kubernetes API and configure Hazelcast to use it. By default, the Hazelcast Vert.x cluster manager reads configuration from a cluster.xml resource. The following listing shows the relevant configuration excerpt of the heat-sensor-service/

src/main/resource/cluster.xml file.

2024/Nov/10th : successful live broadcasting using vertx.

Push and pull/fetch modes.

**Beyond callbacks :**

**Futures & Promises :**

@Override

public void start(Promise<Void> promise) {

vertx.createHttpServer()

.requestHandler(this::handleRequest)

.listen(8080) //returns a future <HttpServer>

.onFailure(promise::fail) //called when the server could not be started

.onSuccess(ok -> {//called on success

System.out.println("http://localhost:8080/");

promise.complete();

});

}

/\*

The listen method(s) or prior httpservers that we saw in earlier examples took a callback, and here it returns a Future<HttpServer>. We then can chain calls to onFailure and onSuccess to define what to do when the server starts, or when an error occurs.

The promise/future interfaces starting from Vert.x 3.8, but the future-based APIs are only available in Vert.x 4.

\*/

**Interoperability with CompletionStage APIs**

CompletionStage<String> cs = promise.future().toCompletionStage();//converts a future to a completionStage

cs

.thenApply(String::toUpperCase) //just like map in Future

.thenApply(str -> "~~~ " + str) //just like map in Future

.whenComplete((str, err) -> {//takes a value or error (ie. str or err)

if (err == null) {

System.out.println(str);

} else {

System.out.println("Oh... " + err.getMessage());

}

});

/\*

we convert the string result to uppercase, prefix it with a string, and eventually call

whenComplete. Note that this is a BiConsumer, and you need to test which of the values or exception parameters is null to know whether the promise completed successfully.

It is also important to note that unless you call an asynchronous CompletionStage

method, the calls are performed on a Vert.x thread.

\*/

**Convert a CompletionStage to a Future**

CompletableFuture<String> cf = CompletableFuture.supplyAsync(() -> {

try {

Thread.sleep(5000);

} catch (InterruptedException e) {

e.printStackTrace();

}

return "5 seconds have elapsed";

});

Future

.fromCompletionStage(cf, vertx.getOrCreateContext())

.onSuccess(System.out::println)

.onFailure(Throwable::printStackTrace);

**Reactive extensions**

Aka : Observable(s) design pattern

Reactive extensions are defined by three things:

 Observing event or data streams (e.g., an incoming HTTP request can be observed)

 Composing operators to transform streams (e.g., merge multiple HTTP request streams as one)

 Subscribing to streams and reacting to events and errors

RxJava v1.x.x does not support back-pressure

RxJava v2.x.x supports back-pressure

**Observable<T>**

* A stream of events of type T. Does not support back-pressure.
* Timer events, observable source where we cannot apply back-pressure like GUI events

**Flowable<T>**

* A stream of events of type T where back-pressure can be applied
* Network data, filesystem inputs

**Single<T>**

* A source that emits exactly one event of type T
* Fetching an entry from a data store by key

**Maybe<T>**

* A source that may emit one event of type T, or none
* Fetching an entry from a data store by key, but the key may not exist

**Completable**

* A source that notifies of some action having completed, but no value is being given
* Deleting files

**Hot Source :**

1. Would still emit event even when there are no subscribers.
2. The subscriber does not get all the events.
3. The subscriber only gets events from the time or period of subscription.
4. Example of hot source(ing) is a periodic timer.

**Cold Source :**

1. Would only emit event if there are subscribers.
2. The subscriber gets all the events.
3. The subscriber only gets events from the time or period of subscription.
4. Example of cold source(ing) is reading file content.

private Single<JsonObject> sendToSnapshot(Single<JsonObject> data) {

return data.flatMap(json -> webClient

.post(4000, "localhost", "")

.expect(ResponsePredicate.SC\_SUCCESS)

.rxSendJsonObject(json)

.flatMap(resp -> Single.just(json))); //and then…

}

**Coroutines**

**Edge Services using coroutines**

**Coroutine trivia :**

“as” is a keyword in Kotlin, so it has to be escaped when used as a method name.

**Choosing asynchronous model appropriate for your project :**

**Some major summary points :**

 Callbacks have expressiveness limitations in relation to composing asynchronous operations, and they can yield that is harder to understand if care is not taken.

 Parallel and sequential asynchronous operations can be composed with other (pure)

asynchronous programming models: futures and promises, reactive extensions,

and coroutines.

 Reactive extensions have an advanced set of composable operators very well suited for event streams.

 Futures and promises are perfect for simple chaining of asynchronous operations.

 Kotlin coroutines affords a language-level support for asynchronous operations.

 There is no universally perfect asynchronous programming model. The choice truly depends on the case of use of such async programming model. The exciting thing about Vert.x is that you can combine these models depending on the challenge or problem domains at hand.

**Beyond the event bus**

**Designing a reactive application**

**Criteria of a reactive application:**

 Back-pressure, as a necessary ingredient in asynchronous stream processing to regulate event throughput.

 Reactive programming as a way to compose asynchronous operations

**One application many services**

**Databases**

PostgresSQL

**Hibernate in Vertx**

<https://how-to.vertx.io/hibernate-reactive-howto/>

MongoDB

**Queue Management Services**

**SMTP**

**Event Stats Services**

**Running the application**

**The Web Stack**

1. The construction of an edge service and a public API
2. The Vert.x web client
3. JSON web tokens (JWT) and cross-origin resource sharing (CORS)
4. Serving and integrating a Vue.js reactive application with Vert.x
5. Testing an HTTP API with REST Assured

**Elements from the vertx web stack :**

Advanced routing

Routing with regex

Authentication

HTTP Client

Cookies

Server side session

Server side template rendering

Cross-site request forgery protection

**Modules required :**

Web (router, request handler, HTTP request dispatcher, handler : {

BodyHandler : {

HTTP authentication,

CORS,

CSRF,

Favicon,

HTTP session,

Serving static files,

Virtual Host,

Template rendering

}

})

Client

Auth-JWT

**Routing HTTP Requests**

BodyHandler bodyHandler = BodyHandler.create();// BodyHandler is a predefined handler that extracts HTTP request body payloads.

router.post().handler(bodyHandler);// Here bodyHandler is called for all HTTP POST and PUT requests.

router.put().handler(bodyHandler);

String prefix = "/api/v1";

router.post(prefix + "/register").handler(this::register);

router.post(prefix + "/token").handler(this::token);

// (...) defines jwtHandler, more later

router.get(prefix + "/:username/:year/:month")//We can extract path parameters by prefixing elements with ":".

.handler(jwtHandler) //Handlers can be chained

.handler(this::checkUser)

.handler(this::monthlySteps);

// (...)

**Making HTTP Requests**

**Creating a JWT Handler**

**JWT Handler in a route**

**Checking that a valid JWT token is present**

**Issuing JWT Tokens in vertx**

Working with JWTs

Common JWT libraries make working with JWTs easy. For example, with JJWT1, creating a

new JWT is straightforward, as shown in Example 6-1.

Example 6-1 Creating and validating a signed JWT using JJWT

import java.time.Instant;

import java.time.temporal.ChronoUnit;

import java.util.Date;

import io.jsonwebtoken.Claims;

import io.jsonwebtoken.Jws;

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

public class JwtExample {

public String createJwt(String secret) throws Exception {

// create and sign the JWT, including a hint

// for the key used to sign the request (kid)

String newJwt = Jwts.builder()

.setHeaderParam("kid", "meaningfulName")

.setSubject("user-12345")

.setAudience("user")

.setIssuedAt(Date.from(Instant.now()))

.setExpiration(Date.from(Instant.now().plus(15, ChronoUnit.MINUTES)))

.signWith(SignatureAlgorithm.HS512, secret)

.compact();

return newJwt;

}

public void validateJwt(String jwtParameter, String secret) throws Exception {

// Validate the Signed JWT!

// Exceptions thrown if not valid

Jws<Claims> jwt = Jwts.parser()

.setSigningKey(secret)

.parseClaimsJws(jwtParameter);

// Inspect the claims, like make a new JWT

// (need a signing key for this)

Claims jwtClaims = jwt.getBody();

System.out.println(jwtClaims.getAudience());

System.out.println(jwtClaims.getIssuer());

System.out.println(jwtClaims.getSubject());

System.out.println(jwtClaims.getExpiration());

System.out.println(jwtClaims.getIssuedAt());

System.out.println(jwtClaims.getNotBefore());

}

}

**Cross-Origin Resource Sharing ( C.O.R.S )**

In vertx Corsehandler we can specify :

1. Allowed Origin path
2. Allowed HTTP Header(s)
3. Allowed HTTP Method(s)

Example :

Set<String> allowedHeaders = new HashSet<>();

allowedHeaders.add("x-requested-with");

allowedHeaders.add("Access-Control-Allow-Origin");

allowedHeaders.add("origin");

allowedHeaders.add("Content-Type");

allowedHeaders.add("accept");

allowedHeaders.add("Authorization");

Set<HttpMethod> allowedMethods = new HashSet<>();

allowedMethods.add(HttpMethod.GET);

allowedMethods.add(HttpMethod.POST);

allowedMethods.add(HttpMethod.OPTIONS);

allowedMethods.add(HttpMethod.PUT);

router.route().handler(

CorsHandler

.create("\*")//A CORS handler for all routes

.allowedHeaders(allowedHeaders)

.allowedMethods(allowedMethods)

);

**Checking C.O.R.S support**

**A modern web frontend with vertx**

**A canvas of Vue.js components**

**VueJS application structure and build integration**

**VueJS & Vertx**

$ npm -version

$ brew upgrade npm

$ npm -version

$ sudo npm install -g @vue/cli

$ vue -version

$ cd <path-to-folder>

$ mkdir <name-of-project>

$ vue create <name-of-vue-project>

$ npm run serve

//sample output :

**➜ basic-vuejs-project** **git:(master) $** npm run serve

> basic-vuejs-project@0.1.0 serve

> vue-cli-service serve

INFO Starting development server...

DONE Compiled successfully in 2848ms 2024-11-25 8:43:06 AM

App running at:

- Local: http://localhost:**8080**/

- Network: http://192.000.111.222:**8080**/

Note that the development build is not optimized.

To create a production build, run

npm run build.

**VueJS + Vite + TailwindCSS**

**CLIENT SIDE LOGIC | JAVASCRIPT | PAGE(S) FLUIDITY**

Then input this into the terminal :

**npm init vite vue**

<press Enter>

? Select a framework: › - Use arrow-keys. Return to submit.

❯   Vanilla

    Vue

    React

    Preact

    Lit

    Svelte

    Solid

    Qwik

    Others

**Choose : Vue**

? Select a variant: › - Use arrow-keys. Return to submit.

❯   TypeScript

    JavaScript

    Customize with create-vue ↗

    Nuxt ↗

**Choose : Javascript**

Done. Now run:

**cd vue**

**npm install**

**npm run dev**

vue % npm install

vue % npm run dev

> vue@0.0.0 dev

> vite

  VITE v5.2.11  ready in 950 ms

  ➜  Local:   http://localhost:5173/

  ➜  Network: use --host to expose

  ➜  press h + enter to show help

h

  Shortcuts

  press r + enter to restart the server

  press u + enter to show server url

  press o + enter to open in browser

  press c + enter to clear console

  press q + enter to quit

u

  ➜  Local:   http://localhost:5173/

  ➜  Network: use --host to expose

o

vue % **npm install -S vuex@next**

or :

**npm install vue-router vuex axios**

Remember to remove the  ‘setup’ tag from the <script setup></script>

Now we can comment out the <pre>{{user}}</pre> in the html of the App.vue

**STYLING :**

Upnext , let’s install Tailwindcss, by following the instructions at :

[tailwindcss.com/docs/guides/vite](http://tailwindcss.com/docs/guides/vite)

Commands :

**npm install -D tailwindcss postcss autoprefixer**

<press Enter>

**npx tailwindcss init -p**

<press Enter>

**npm install @headlessui/vue @heroicons/vue @tailwindcss/forms -S**

<press Enter>

We can get UI components here :

<https://tailwindui.com/components>

LESSON(S) LEARNT :

1. All the npm commands and the installations relating to the tailwindcss should all be made while at the root folder path of “vue” within the project in the terminal for all the designs and CSSes to work perfectly.

All the tailwindcss things with the form design is perfectly working now as of 1119GMT.

1. In creating “vue-routes” do not put a route there when its components have not been created yet. Stick with what exist. The best practice may be to create the component before you ever call it anywhere-else.
2. The  value for component in the set of unique path(s) should match the name given to the import at the top of the line.

At the point of linking the JS and functions such as dispatch, then and so on with promises, it is therefore necessary to install “ axios “ using npm.

Command :

**$ npm i -S axios**

If we ever need icons we can check out heroicons :

<https://www.heroicons.com>

### process 2 :

**npm install**

**CLIENT SIDE LOGIC | JAVASCRIPT | PAGE(S) FLUIDITY**

Then input this into the terminal :

**npm init vite vue**

<press Enter>

? Select a framework: › - Use arrow-keys. Return to submit.

❯   Vanilla

    Vue

    React

    Preact

    Lit

    Svelte

    Solid

    Qwik

    Others

**Choose : Vue**

? Select a variant: › - Use arrow-keys. Return to submit.

❯   TypeScript

    JavaScript

    Customize with create-vue ↗

    Nuxt ↗

**Choose : Javascript**

Scaffolding project in

airsurvey\_app/vue...

Done. Now run:

***cd vue***

***npm install***

***npm run dev***

vue % **npm install**

The next command was to help get vite + vue setup nicely :

**npm install vue-router vuex axios**

After the command is issued, set up the styling, the root folder still set to “ vue “ :

**STYLING :**

Upnext , let’s install Tailwindcss, by following the instructions at :

[tailwindcss.com/docs/guides/vite](http://tailwindcss.com/docs/guides/vite)

Commands :

**npm install -D tailwindcss postcss autoprefixer**

<press Enter>

**npx tailwindcss init -p**

<press Enter>

**npm install @headlessui/vue @heroicons/vue @tailwindcss/forms -S**

<press Enter>

## Had to paste this code into the “ tailwind.config.js”

module.exports = {

content: [

'./index.html',

'./src/\*\*/\*.{vue,js,ts,jsx,tsx}',

'./pages/\*\*/\*.{html,js}',

'./components/\*\*/\*.{html,js}',

],

theme: {

extend: {},

},

plugins: [

require('@tailwindcss/forms')

],

}

Running the app :

**npm run dev**

**npm run watch**

**npm run test**

>>>> 2024/11/26 :

This time it took me far lesser time to setup a vue.js project from scratch … just a few minutes, then testing to make sure it all works.

**VueJS router configuration**

**Backend integration in VueJS**

**HTML Template components**

<https://tailwindtemplates.io/templates>

<https://github.com/PhatStraw/Free-Tailwind-CSS-Templates-and-Components>

<https://github.com/slim-python/tailwind-css-free-components>

<https://github.com/markmead/hyperui>

<https://tw-elements.com/docs/standard/integrations/vue-integration/>

<https://www.tailwind-kit.com/components#forms>

<https://tailwindflex.com/>

<https://flowbite.com/docs/components/alerts/>

<https://flowbite.com/docs/getting-started/vue/>

<https://tailwindui.com/components/preview>

<https://tailwindui.com/components>

**Javascript code components**

**Using JWT Token with axios**

<https://jwt.io>

**Serving static content with Vertx**

**Writing integration tests**

Some fundamental tests set in the IntegrationTest class :

1 Register some users.

2 Get a JWT token for each user.

3 Fetch a user’s data.

4 Try to fetch the data of another user.

5 Update a user’s data.

6 Check some activity stats for a user.

7 Try to check the activity of another user.

Test dependencies to run the integration tests

Preamble of the integration test class

Preparing a REST assured request specification

Utility hash maps for the integration testing

Test for registering users

Test code for retrieving JWT tokens

Extracting JSON with REST assured

\*\*\* Attempt to run the test(s) : 2024 Nov., 25th

Setting up docker for Mac M1 : {

* Docker Desktop is free for small businesses (fewer than 250 employees AND less than $10 million in annual revenue), personal use, education, and non-commercial open source projects.
* Otherwise, it requires a paid subscription for professional use.
* Paid subscriptions are also required for government entities.
* Docker Pro, Team, and Business subscriptions include commercial use of Docker Desktop.

}

Setting up podman for Mac M1

Setting up Colima for Mac M1

install colima, and the docker CLI. We then launch a VM using special configuration flags to use macOS’s virtualisation layer and the translation of x86/amd64 into Apple Silicon.

Colima is only a wrapper to create Lima VMs.

Lima is the virtual machine that will run with the rosetta enhaced compatibility with x86 / amd64 and provide the docker runtime.

brew install colima # we will create the lima vm with this wrapper  
brew install docker # The CLI only

# Create and Configure a super performant vm

Note: adjust your settings (CPU, Memory and Disk according to your needs and hardware)

colima start \  
--profile default \  
--activate \  
--arch aarch64 \  
--cpu 10 \  
--disk 48 \  
--memory 24 \  
--mount ${HOME}:w \  
--mount-inotify \  
--ssh-agent \  
--vm-type vz \  
--vz-rosetta \  
--verbose

# Key Configuration Settings

* **arch**: AARCH64 specifies that we will run an ARM64 machine and not an x86\_64 machine
* **vm-type**: VZ (to use Apple’s Hypervisor.Framework)
* **vz-rosseta**: Enables Rosetta (needs macOS 13.0 or newer)

Reference on how to use Rosetta with Lima and the compatibility modes:

## [Intel-on-ARM and ARM-on-Intel](https://lima-vm.io/docs/config/multi-arch/?source=post_page-----da5100e2557d--------------------------------" \l "slow-mode" \t "_blank)

### [Lima supports two modes for running Intel-on-ARM and ARM-on-Intel: Lima can run a VM with a foreign architecture, just…](https://lima-vm.io/docs/config/multi-arch/?source=post_page-----da5100e2557d--------------------------------" \l "slow-mode" \t "_blank)

[lima-vm.io](https://lima-vm.io/docs/config/multi-arch/?source=post_page-----da5100e2557d--------------------------------" \l "slow-mode" \t "_blank)

# Configure the Shell to replace Docker

Wait for the machine to come up and then let’s setup the docker environment. (Actually, none of this is strictly necessary. You could use the bundled nerdctl tool.)

But follow along if you want a drop-in Docker replacement:

* Place those in your shell’s profile or in the current session at will.

export COLIMA\_VM="default"  
export COLIMA\_VM\_SOCKET="${HOME}/.colima/${COLIMA\_VM}/docker.sock"  
export DOCKER\_HOST="unix://${COLIMA\_VM\_SOCKET}"

# Bonus: Multiple machines simultaneously

At a certain point you might want to run more experiments or even split work / load / whatever.

For example, one limitation of **using Apple’s Hypervisor is that it is not possible to resize the VM’s Disk after being created**. So instead of destroying the VM and recreating it (losing all the containers), you can sidekick another alongside and split the load on that one.

Podman does not officially support this kind of workloads, though it is still possible to achieve the same.

# To create a secondary machine, just do issue another colima command

colima start \  
--profile secondary \  
--activate \  
--arch aarch64 \  
--cpu 1 \  
--disk 20\  
--memory 8 \  
--mount ${HOME}:w \  
--mount-inotify \  
--ssh-agent \  
--vm-type vz \  
--vz-rosetta \  
--verbose

export COLIMA\_VM="secondary"  
export COLIMA\_VM\_SOCKET="${HOME}/.colima/${COLIMA\_VM}/docker.sock"  
export DOCKER\_HOST="unix://${COLIMA\_VM\_SOCKET}"

# Introduction

This post introduces a streamlined method to set up a Podman machine (QEMU) on **Apple Silicon** **for running amd64 (x86\_64) containers**. **We explore two approaches**: **multi-architecture support** and **fully emulated** x86\_64 machines.

**Note**: I will keep every script here and the explanation in this **public repo** too**:** <https://github.com/guillem-riera/podman-machine-x86_64>

# Approach Overview

1. **Mixed Mode, Multi-Architecture Support**: This method enables support for multiple architectures, including x86\_64, on a standard aarch64 machine. It **maintains high performance for the native ARM images and has a performance impact on amd64 images**. It operates on a base aarch64 machine and compatibility with amd64 images is as good as the package qemu-user-static can provide (I haven’t tested for full compatibility).
2. **Full x86\_64 Emulation**: This offers **maximum compatibility at the cost of slower performance**. It’s a fully emulated x86\_64 machine, which means that the containers are also run in fully x86\_64 mode.

**Recommendation**: Always try the **first approach (mixed mode)** before considering the second.

# Requirements

To get started, ensure you have the following installed:

* Homebrew
* Homebrew bundle
* Podman
* QEMU (automatically included as a dependency of Podman)
* jq

The required packages are listed in the Brewfile. Install them using:

brew bundle install

# Setting Up

# 1. Multi-Arch Support on Current Podman Machine

This setup installs the necessary package **qemu-user-static** on your current machine.

This script facilitates this process:

export PODMAN\_MACHINE\_NAME=${PODMAN\_MACHINE\_NAME:-podman-machine-default}  
  
### Stop all podman machine instances  
ALL\_PODMAN\_MACHINES=$(podman machine list | awk '{ print $1 }' | tr -d '\*' | sed 1d | tr '\n' ' ')  
for PODMAN\_MACHINE in ${ALL\_PODMAN\_MACHINES}; do  
 podman machine stop ${PODMAN\_MACHINE}  
done  
  
### Start the target podman machine  
podman machine start ${PODMAN\_MACHINE\_NAME}  
  
### wait for the podman machine to be running  
PODMAN\_MACHINE\_STATUS=$(podman machine inspect ${PODMAN\_MACHINE\_NAME} | jq -r '.[].State')  
while [[ "${PODMAN\_MACHINE\_STATUS}" != "running" ]]; do  
 echo "[Info] Waiting for podman machine '${PODMAN\_MACHINE\_NAME}' to be running, current status: ${PODMAN\_MACHINE\_STATUS}..."  
 sleep 1  
 PODMAN\_MACHINE\_STATUS=$(podman machine inspect ${PODMAN\_MACHINE\_NAME} | jq -r '.[].State')  
done  
  
### Now that the podman machine is running we can install the package  
podman machine ssh "${PODMAN\_MACHINE\_NAME}" 'sudo rpm-ostree install qemu-user-static'  
  
### Stop the podman machine to apply the changes  
podman machine stop ${PODMAN\_MACHINE\_NAME}  
  
### Start the podman machine again  
podman machine start ${PODMAN\_MACHINE\_NAME}  
  
echo "[Info] Done. You can now run multi-architecture images in ${PODMAN\_MACHINE\_NAME}."

Podman can now run multi-architecture images with performance impacts limited to x86\_64 containers.

# How it works?

This bash script automates the setup of multi-architecture support for an existing Podman machine. Here’s a summary of how it works:

1. **Setup**: It sets the PODMAN\_MACHINE\_NAME variable, defaulting to "podman-machine-default" if not already specified.
2. **Stopping All Podman Machine Instances**: The script lists all existing Podman machines, excluding the header line and any active (marked with an asterisk) machines. It then stops each of these machines to ensure a clean setup environment.
3. **Starting the Target Podman Machine**: It starts the target Podman machine specified in PODMAN\_MACHINE\_NAME.
4. **Waiting for the Machine to Run**: The script continuously checks if the target Podman machine has reached the “running” state. It waits in a loop, checking the machine’s status every second.
5. **Installing the Package**: Once the target machine is running, the script remotely connects to it via SSH and installs the qemu-user-static package using sudo rpm-ostree install. This package is crucial for enabling multi-architecture support.
6. **Restarting the Podman Machine**: After the installation, the script stops the Podman machine to apply the changes and then starts it again

# 2. Full x86\_64 Emulation Setup

**Note**: Follow this step only if the first solution doesn’t meet your needs.

## Creating a new emulated Podman Machine (x86\_64)

The following script creates a podman machine and alters it to make it an x86\_64 machine (using QEMU):

# Setup the podman machine for x86\_64 (QEMU), supports only Apple Silicon (Mx) Macs  
  
# Keep all shell arguments in a variable to pass to the podman machine init command:  
EXTRA\_ARGS=${EXTRA\_ARGS:-$@}  
  
## 1. Download Fedora CoreOS image for x86\_64 (QEMU)  
PODMAN\_X86\_64\_MACHINE\_NAME=${PODMAN\_X86\_64\_MACHINE\_NAME:-x86\_64}  
PODMAN\_X86\_64\_MACHINE\_NAME\_EXISTS=$(podman machine list | grep ${PODMAN\_X86\_64\_MACHINE\_NAME} | wc -l | tr -d '[:space:]')  
PODMAN\_QEMU\_IMAGE="fedora-coreos-39.20231101.3.0-qemu.x86\_64.qcow2.xz"  
DOWNLOAD\_DIR=${DOWNLOAD\_DIR:-.}  
  
if [ ${PODMAN\_X86\_64\_MACHINE\_NAME\_EXISTS} -lt 1 ]; then  
 curl -C- -O "https://builds.coreos.fedoraproject.org/prod/streams/stable/builds/39.20231101.3.0/x86\_64/${PODMAN\_QEMU\_IMAGE}"  
 podman machine init --image-path ${DOWNLOAD\_DIR}/${PODMAN\_QEMU\_IMAGE} ${PODMAN\_X86\_64\_MACHINE\_NAME} ${EXTRA\_ARGS}  
else  
 echo "[Info] Machine ${PODMAN\_X86\_64\_MACHINE\_NAME} already exists. If you want to recreate it, run 'podman machine rm ${PODMAN\_X86\_64\_MACHINE\_NAME}'"  
fi  
  
## 2. Change machine settings  
  
### Get the machine config file name  
machineConfigFile="$(podman machine inspect ${PODMAN\_X86\_64\_MACHINE\_NAME} | jq -r '.[].ConfigPath.Path')"  
  
### Change the QEMU binary to x86\_64  
sed -i '' 's/qemu-system-aarch64/qemu-system-x86\_64/g' ${machineConfigFile}  
### Change the firmware to x86\_64  
sed -i '' 's/edk2-aarch64-code/edk2-x86\_64-code/g' ${machineConfigFile}  
### Delete the additional UEFI firmware file (ovmf) and the preceding '-drive' option. The '-drive' option is in a line above the line containing the path to 'x86\_64\_ovmf\_vars.fd'. Both lines must be deleted, but other -drive options must be kept.  
#### using sed to match 2 lines: '-drive' followed by 'x86\_64\_ovmf\_vars.fd'  
sed -i '' '/-drive/{N;/x86\_64\_ovmf\_vars.fd/d;}' ${machineConfigFile}  
### Delete the HVF (Hypervisor Framework) acceleration, which is only available for macOS. This are also 2 lines: '-accel' followed by 'hvf'  
sed -i '' '/-accel/{N;/hvf/d;}' ${machineConfigFile}  
### Delete the TCG acceleration, which seems to work only for Alpha and ARM architectures. This are also 2 lines: '-accel' followed by 'tcg'  
sed -i '' '/-accel/{N;/tcg/d;}' ${machineConfigFile}  
### Change the machine type to q35  
sed -i '' 's/virt,highmem=on/q35/g' ${machineConfigFile}  
### Change the cpu type from 'host' to 'qemu64'  
sed -i '' 's/host/qemu64/g' ${machineConfigFile}

# How it works?

This script is designed to set up a Podman machine specifically for x86\_64 architecture on Apple Silicon (Mx) Macs by modifying the QEMU template that podman generates when it creates a new machine.

Here’s a summary of its functionality and workflow:

**Shell Arguments**: The script stores any arguments passed to it in the EXTRA\_ARGS variable, which will later be used in the podman machine init command.

**Downloading** Fedora CoreOS Image for x86\_64 (QEMU):

* It sets a default name for the Podman x86\_64 machine (PODMAN\_X86\_64\_MACHINE\_NAME) and checks if a machine with this name already exists.
* If the machine does not exist, the script downloads the specified Fedora CoreOS image for x86\_64 using curl.
* After downloading, it initializes a new Podman machine with this image and any extra arguments provided.

**Changing Machine Settings**:

* The script retrieves the configuration file path of the newly created Podman machine.
* Several modifications are made to the machine’s configuration file to adapt it for x86\_64 emulation:
* **Changing QEMU Binary**: Updates the QEMU binary from qemu-system-aarch64 to qemu-system-x86\_64.
* **Changing Firmware**: Adjusts the firmware from edk2-aarch64-code to edk2-x86\_64-code.
* **Removing UEFI Firmware File**: Deletes lines related to the UEFI firmware file (x86\_64\_ovmf\_vars.fd) and its preceding '-drive' option.
* **Removing HVF Acceleration**: Eliminates the Hypervisor Framework (HVF) acceleration settings, as they are only available for macOS.
* **Removing TCG Acceleration**: Removes TCG acceleration settings, which are typically for Alpha and ARM architectures.
* **Changing Machine Type**: Updates the machine type from virt,highmem=on to q35.
* **Changing CPU Type**: Changes the CPU type from host to qemu64.

## Conclusion

The podman offers a convenient way to run x86\_64 containers on Apple Silicon, but you have to do extra steps to enable that.

Whether you require high performance or maximum compatibility, these methods provide a flexible solution to meet your containerization needs.

This is possible because QEMU, the underlaying virtualisation and emulation tool is really awesome!.

# Alternatives

If you are looking for a very high performance and fully open source alternative to Docker Desktop that supports x86\_64 / amd64 architecture with Rosetta, check my newer post on colima:

<https://github.com/guillem-riera/podman-machine-x86_64>

**Messaging and event streaming with Vertx**

Messaging with AMQP

Event streaming with Apache Kafka

Examples of messaging queue brokers :

* 1. AMQP : Advanced Messaging Queuing Protocol
  2. STOMP : Simple Text Oriented Messaging Protocol
  3. RabbitMQ Client
  4. MQTT : Messaging Queuing Telemetric Transport



Sending emails

Integration testing with messaging and event streaming

middleware

Event-Bus TCP Bridge

**Event driven services beyond HTTP with vertx**

**Ingestion from AMQP**

**AMQP client configuration :**

private AmqpClientOptions amqpConfig() {

return new AmqpClientOptions()

.setHost("localhost")//Credentials are the default ones from the docker image

.setPort(5672) //Credentials are the default ones from the docker image

.setUsername("artemis")//Credentials are the default ones from the docker image

.setPassword("simetraehcapa");//Credentials are the default ones from the docker image

}

// (...)

AmqpClientOptions amqpOptions = amqpConfig();

AmqpReceiverOptions receiverOptions = new AmqpReceiverOptions()

.setAutoAcknowledgement(false) //We would manually acknowledge incoming messages

.setDurable(true); //We want durable messages

**AMQP event-processing pipeline :**

AmqpClient.create(vertx, amqpOptions) //Create an AMQP client

.rxConnect()

.flatMap(conn -> conn.rxCreateReceiver("step-events", receiverOptions)) //Create a message receiver from the ‘step-events’ destination

.flatMapPublisher(AmqpReceiver::toFlowable) //Create a flowable AMQP messages

.doOnError(this::logAmqpError) //Error logging

.retryWhen(this::retryLater) //Retry logic

.subscribe(this::handleAmqpMessage); //Subscription that dispatches incoming messages

**Logging AMQP errors**

private void logAmqpError(Throwable err) {

logger.error("Woops AMQP", err);

}

**Recovering from errors with a delayed AMQP resubscription**

private Flowable<Throwable> retryLater(Flowable<Throwable> errs) {

return errs.delay(10, TimeUnit.SECONDS, RxHelper.scheduler(vertx));// It is important to use the scheduler parameter to process events on a Vert.x event loop

}

**Translating AMQP messages to Kafka records**

Handling AMQP messages

private void handleAmqpMessage(AmqpMessage message) {

if (!"application/json".equals(message.contentType()) || invalidIngestedJson(message.bodyAsJsonObject())) {

logger.error("Invalid AMQP message (discarded): {}", message.bodyAsBinary());

message.accepted();

return;

}

JsonObject payload = message.bodyAsJsonObject();

KafkaProducerRecord<String, JsonObject> record = makeKafkaRecord(payload);

updateProducer.rxSend(record).subscribe(

ok -> message.accepted(),

err -> {

logger.error("AMQP ingestion failed", err);

message.rejected();

});

}

Checking for valid JSON data

Preparing a kafka record

Configuring a Kafka Producer

HTTP server for ingestion

Ingesting updates from HTTP

Sending emails using MailHog

Listening for specific update events

Kafka RxJava Pipeline for receiving and processing specific updates

Predicate for certain specific events within the program

Creating an SMTP client

Mail client configuration

Implementation of the sendmail method

Request to retrieve the email address

Preparing an email message / template

Integration Tests

Ingestion testing

Ingest test preparation

AMQP Ingestion test preamble

AMQP Ingestion test: checking for Kafka record

Ingesting a bad JSON document

Sending of email testing

Testing for email NOT sent based on a Boolean check

**PERSISTENT STATE MANAGEMENT WITH DATABASES**

**Data/Object Mapping in Java: Advantages and Disadvantages**

Data/object mapping, often referred to as Object-Relational Mapping (ORM), is a technique that allows developers to interact with databases using object-oriented programming languages like Java. This approach bridges the gap between the relational database schema and the object-oriented model.

**Advantages**

1. **Improved Productivity:**
   * **Reduced boilerplate code:** ORM frameworks automate the repetitive tasks of SQL queries, result set parsing, and object instantiation.
   * **Simplified database interactions:** Developers can interact with the database using object-oriented concepts, making code more readable and maintainable.
2. **Enhanced Data Integrity:**
   * **Data consistency:** ORM frameworks can enforce data integrity constraints, such as unique keys and foreign key relationships, at the object level.
   * **Reduced data inconsistencies:** By abstracting database operations, ORM helps prevent errors that could lead to inconsistent data.
3. **Improved Code Reusability:**
   * **Reusable data access components:** ORM frameworks often provide a layer of abstraction that can be reused across different parts of an application.
   * **Simplified data access logic:** By encapsulating database interactions, ORM promotes code reusability and modularity.
4. **Facilitated Testing:**
   * **Mocking and stubbing:** ORM frameworks can be easily mocked or stubbed for unit testing, making it easier to isolate and test different parts of the application.
   * **Simplified test setup:** By abstracting database interactions, ORM reduces the complexity of setting up test environments.

**Disadvantages**

1. **Increased Complexity:**
   * **Steep learning curve:** ORM frameworks can have a significant learning curve, especially for developers new to the concept.
   * **Configuration overhead:** Configuring ORM frameworks can be complex and time-consuming, especially for large and complex applications.
2. **Performance Overhead:**
   * **Performance impact:** ORM frameworks can introduce performance overhead, particularly for complex queries and large datasets.
   * **Inefficient query generation:** Some ORM frameworks may generate inefficient SQL queries, leading to performance degradation.
3. **Vendor Lock-in:**
   * **Dependency on specific ORM framework:** Once an application is heavily reliant on an ORM framework, it can be difficult to switch to another framework or database system.
   * **Tight coupling:** ORM frameworks can tightly couple the application to a specific database technology, limiting flexibility.
4. **Limited Control Over Database Interactions:**
   * **Reduced flexibility:** ORM frameworks can limit the flexibility to write custom SQL queries or optimize database performance.
   * **Dependency on framework features:** Developers may need to rely on the specific features and limitations of the ORM framework.

**Conclusion**

While data/object mapping offers significant advantages in terms of productivity, data integrity, and code reusability, it's essential to weigh the potential disadvantages, such as increased complexity, performance overhead, and vendor lock-in. The decision to use an ORM framework should be based on the specific needs of the application and the development team's expertise.

<https://github.com/search?q=totp+generator+language%3AJava&type=repositories&l=Java>

**TOTP generation explained**

Time-based verification codes, often referred to as Time-based One-Time Passwords (TOTP), are generated using a cryptographic algorithm that relies on a shared secret key and the current time. Here's a breakdown of how it works:

**1. Shared Secret Key:**

* A unique, randomly generated secret key is created and shared between the user and the authentication server. This key is typically stored securely in the user's authentication app or device.

**2. Time-Based Counter:**

* The current Unix time (seconds since January 1, 1970) is retrieved.
* This time is divided by a predefined time interval (usually 30 or 60 seconds) to obtain a time-based counter value.

**3. HMAC-Based One-Time Password (HOTP) Algorithm:**

* The HOTP algorithm is used to generate the verification code. It takes the following inputs:
  + Shared secret key
  + Time-based counter value

* The algorithm applies a cryptographic hash function (such as HMAC-SHA1 or HMAC-SHA256) to these inputs to produce a hash value.
* The hash value is truncated to a specific number of digits (usually 6 or 8) to form the verification code.

**4. Verification Code Generation:**

* The generated verification code is displayed to the user on their authentication app or device.
* The user enters this code on the authentication website or app to verify their identity.

**5. Server-Side Verification:**

* The server performs the same calculations using the shared secret key and the current time to generate its own verification code.
* The server compares the user-provided code with the generated code.
* If the codes match within a certain time window (to account for clock skew), the authentication is successful.

**Key Points:**

* **Time-Based Validity:** The verification code is valid for a limited time (e.g., 30 or 60 seconds). After this time, a new code is generated.
* **Security:** The security of TOTP relies on the strength of the shared secret key and the cryptographic hash function used.
* **Flexibility:** TOTP can be implemented in various ways, using different time intervals and code lengths.
* **Wide Adoption:** TOTP is widely used in two-factor authentication (2FA) systems due to its simplicity and security.

By following these steps, TOTP provides a secure and convenient way to verify user identity without relying on traditional passwords.

**1. Install the JavaScript Obfuscator Module**

To proceed with the obfuscation of any JS code (for the browser, node.js, etc) with Node.js, you will need to rely on the JavaScript obfuscator module. JavaScript obfuscator is a powerful free obfuscator for JavaScript and Node.js with a wide number of features that provide protection for your source code. This module:

* has no limits or restrictions.
* runs on your local machine - does not send data to a server.
* compatible with es2015, es2016 and partially es2017.

You can install this module using the following command:

npm install javascript-obfuscator

After the installation you will be able to require the module in your scripts using require("javascript-obfuscator"). The obfuscator is free and open source (BSD-2-Clause licensed) and is written in TypeScript, you can see [an online implementation of the module here](https://javascriptobfuscator.herokuapp.com/). For more information about this library, please [visit the official repository at Github here](https://github.com/javascript-obfuscator/javascript-obfuscator).

## 2. Using the Obfuscator

The logic to obfuscate some code with the module is really simple. You create an instance of the Module, from the instance you can use the obfuscate method that expects as first argument the code that you want to obfuscate. This method returns the obfuscated synchronously. Through a series of transformations, such as variable / function / arguments renaming, strings removal, and others, your source code is transformed into something unreadable, while working exactly as before:

// Require the JavaScript obfuscator

var JavaScriptObfuscator = require('javascript-obfuscator');

// Obfuscate the code providen as first argument

var obfuscationResult = JavaScriptObfuscator.obfuscate(`

(function(){

var variable1 = '5' - 3;

var variable2 = '5' + 3;

var variable3 = '5' + - '2';

var variable4 = ['10','10','10','10','10'].map(parseInt);

var variable5 = 'foo ' + 1 + 1;

console.log(variable1);

console.log(variable2);

console.log(variable3);

console.log(variable4);

console.log(variable5);

})();

`);

// Display obfuscated result

console.log(obfuscationResult.getObfuscatedCode());

### Obfuscator Options

The obfuscator can be customized if you provide the configuration object as second argument in the obfuscate method. The following snippet shows all the available properties on the module:

JavaScriptObfuscator.obfuscate(YourCode, {

compact: true,

controlFlowFlattening: false,

controlFlowFlatteningThreshold: 0.75,

deadCodeInjection: false,

deadCodeInjectionThreshold: 0.4,

debugProtection: false,

debugProtectionInterval: false,

disableConsoleOutput: false,

domainLock: [],

log: false,

mangle: false,

renameGlobals: false,

reservedNames: [],

rotateStringArray: true,

seed: 0,

selfDefending: false,

sourceMap: false,

sourceMapBaseUrl: '',

sourceMapFileName: '',

sourceMapMode: 'separate',

stringArray: true,

stringArrayEncoding: false,

stringArrayThreshold: 0.75,

target: 'browser',

unicodeEscapeSequence: false

});

It's worth to read the documentation of the library as in the future new options may appear. The official repository offers already made presets to provide a feeling of "low", "medium" or "high" obfuscation with a special combination of options. Note that the better the obfuscation is, the slower the processing step is:

### A. Low Obfuscation

{

compact: true,

controlFlowFlattening: false,

deadCodeInjection: false,

debugProtection: false,

debugProtectionInterval: false,

disableConsoleOutput: true,

log: false,

mangle: true,

renameGlobals: false,

rotateStringArray: true,

selfDefending: true,

stringArray: true,

stringArrayEncoding: false,

stringArrayThreshold: 0.75,

unicodeEscapeSequence: false

}

### B. Medium Obfuscation

{

compact: true,

controlFlowFlattening: true,

controlFlowFlatteningThreshold: 0.75,

deadCodeInjection: true,

deadCodeInjectionThreshold: 0.4,

debugProtection: false,

debugProtectionInterval: false,

disableConsoleOutput: true,

log: false,

mangle: false,

renameGlobals: false,

rotateStringArray: true,

selfDefending: true,

stringArray: true,

stringArrayEncoding: 'base64',

stringArrayThreshold: 0.75,

unicodeEscapeSequence: false

}

### C. High Obfuscation

{

compact: true,

controlFlowFlattening: true,

controlFlowFlatteningThreshold: 1,

deadCodeInjection: true,

deadCodeInjectionThreshold: 1,

debugProtection: true,

debugProtectionInterval: true,

disableConsoleOutput: true,

log: false,

mangle: false,

renameGlobals: false,

rotateStringArray: true,

selfDefending: true,

stringArray: true,

stringArrayEncoding: 'rc4',

stringArrayThreshold: 1,

unicodeEscapeSequence: false

}

## Example

In the following example we're going to read the content of a JS file and we'll write a new one with the obfuscated version of the code:

// Require Filesystem module

var fs = require("fs");

// Require the Obfuscator Module

var JavaScriptObfuscator = require('javascript-obfuscator');

// Read the file of your original JavaScript Code as text

fs.readFile('./your-original-code.js', "UTF-8", function(err, data) {

if (err) {

throw err;

}

// Obfuscate content of the JS file

var obfuscationResult = JavaScriptObfuscator.obfuscate(data);

// Write the obfuscated code into a new file

fs.writeFile('./your-code-obfuscated.js', obfuscationResult.getObfuscatedCode() , function(err) {

if(err) {

return console.log(err);

}

console.log("The file was saved!");

});

});

Remember that, while it's almost impossible to recover the exact original source code from the obfuscated version, someone with the time, knowledge and patience can reverse-engineer it. The example doesn't use a special obfuscation but the default, so if you want a custom obfuscation, provide a configuration object.

**Critical installations :**

After carefully going through the Terms & Conditions of Docker, I have no desire to use it now nor anytime into the future. It is a very closed system and there are some conditions in the terms such as owner on usage data, certain biased declaration of the who and what a business is, who and what a government institution is, etc.

So I would rather take my chances on how to setup the following very necessary setup directly and configuring them to the level I want without the need for any docker application.

If now or in the very near future, I find a way to use Podman, that is something I could try. Meanwhile, this is also NOT highly desired. The goal is basically to get to the bare bones of the setup for each of the following :

PostgreDB

MongoDB

Kafka

Artemis AMQP

Consul

Infinispan

**Setting up PostgresDB**

**Setting up MongoDB**

**Setting up Kafka**

**Setting up Artemis AMQP**



I am assuming that you have already setup homebrew on your MacOS Apple Silicon system. If not refer to [this](https://stackoverflow.com/questions/66666134/how-to-install-homebrew-on-m1-mac) stackoverflow link to set it up or refer the official documentation.

The first steps are pretty straightforward:

$ brew install java  
$ brew install kafka

Once these commands run successfully your kafka package is installed. Now to run kafka open two seperate terminals to run Zookeeper and Kafka services seperately as:

**Start Zookeeper:**

$ **zookeeper-server-start /opt/homebrew/etc/kafka/zookeeper.properties**

**Start Kafka:**

$ **kafka-server-start /opt/homebrew/etc/kafka/server.properties**

After the services are successfully running, the next step is to create a kafka topic. *A topic is a category or feed name to which records are stored and published.* Before running any producer or consumer API it is mandatory to create a topic. Here is an example to create a sample topic named foobar :

$ **kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic foobar**

**NOTE:** Notice here we aren’t using --zookeeper flag as in standard documentations because Kafka versions for new MacOS do not support/ require that flag anymore.

Once the topic is created we can test out Kafka producer and consumer APIs:

* Open two terminals.
* In the first one initialize a producer console for topic thefoobar . And try to send some test messages as follows:

$ **kafka-console-producer --broker-list localhost:9092 --topic foobar  
>** foo  
**>** bar

* In the second one initalize a consumer console for the topic foobar . This will listen to the bootstrap server at port 9092 at topic foobar:

$ **kafka-console-consumer --bootstrap-server localhost:9092 --topic foobar --from-beginning**foo  
bar

If you can see the outputs kafka is set up and running neatly on your Apple Silicon system.

Cheers !

Terminal output upon running : $ brew install kafka

brew install kafka

==> **Auto-updating Homebrew...**

Adjust how often this is run with HOMEBREW\_AUTO\_UPDATE\_SECS or disable with

HOMEBREW\_NO\_AUTO\_UPDATE. Hide these hints with HOMEBREW\_NO\_ENV\_HINTS (see `man brew`).

==> **Downloading https://ghcr.io/v2/homebrew/portable-ruby/portable-ruby/blobs/sha256:303bed4c7fc431a685db3c3c151d873740114adbdccd23762ea2d1e39ea78f47**

########################################################################################################### 100.0%

==> **Pouring portable-ruby-3.3.6.arm64\_big\_sur.bottle.tar.gz**

==> **Auto-updated Homebrew!**

Updated 3 taps (homebrew/services, homebrew/core and homebrew/cask).

==> **New Formulae**

aliae dum kraftkit pie topiary

cargo-run-bin funzzy lla rshijack

directx-headers harlequin nrm scooter

==> **New Casks**

beaver-notes font-sketchybar-app-font tabtab zen-browser@twilight

font-cica-without-emoji ishare website-audit

You have **56** outdated formulae installed.

Warning: You are using macOS 12.

We (and Apple) do not provide support for this old version.

It is expected behaviour that some formulae will fail to build in this old version.

It is expected behaviour that Homebrew will be buggy and slow.

Do not create any issues about this on Homebrew's GitHub repositories.

Do not create any issues even if you think this message is unrelated.

Any opened issues will be immediately closed without response.

Do not ask for help from Homebrew or its maintainers on social media.

You may ask for help in Homebrew's discussions but are unlikely to receive a response.

Try to figure out the problem yourself and submit a fix as a pull request.

We will review it but may or may not accept it.

==> **Fetching dependencies for kafka: python-packaging, ca-certificates, openssl@3, sqlite, xz, pkgconf, python@3.13, libunistring, bison, meson, python-setuptools, glib, pixman, icu4c@76, gobject-introspection, harfbuzz, openjdk, automake, cppunit, libtool, maven and zookeeper**

==> **Fetching python-packaging**

==> **Downloading https://ghcr.io/v2/homebrew/core/python-packaging/manifests/24.2**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/python-packaging/blobs/sha256:81d0db4704a8a4d53322164f860947baa0b**

########################################################################################################### 100.0%

==> **Fetching ca-certificates**

==> **Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2024-11-26**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/blobs/sha256:7a3b5f75ca44d330e0f37432af09f58e37bf**

########################################################################################################### 100.0%

==> **Fetching openssl@3**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/openssl/openssl/releases/download/openssl-3.4.0/openssl-3.4.0.tar.gz**

==> **Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/7634677/ff52f315**

########################################################################################################### 100.0%

==> **Fetching sqlite**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://www.sqlite.org/2024/sqlite-autoconf-3470100.tar.gz**

########################################################################################################### 100.0%

==> **Fetching xz**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/tukaani-project/xz/releases/download/v5.6.3/xz-5.6.3.tar.gz**

==> **Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/553665726/054f4c**

########################################################################################################### 100.0%

==> **Fetching pkgconf**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://distfiles.ariadne.space/pkgconf/pkgconf-2.3.0.tar.xz**

########################################################################################################### 100.0%

==> **Fetching python@3.13**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://raw.githubusercontent.com/Homebrew/formula-patches/8b5bcbb262d1ea4e572bba55043bf7d2341a682**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/c4/e6/c1ac50fe3eebb38a155155711e6e864e254ce4b6e17fe2429b4c**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/4d/87/fb90046e096a03aeab235e139436b3fe804cdd447ed2093b0d70**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/27/b8/f21073fde99492b33ca357876430822e4800cdf522011f180413**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/b7/a0/95e9e962c5fd9da11c1e28aa4c0d8210ab277b1ada951d2aee33**

########################################################################################################### 100.0%

==> **Downloading https://www.python.org/ftp/python/3.13.0/Python-3.13.0.tgz**

########################################################################################################### 100.0%

==> **Fetching libunistring**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://ftp.gnu.org/gnu/libunistring/libunistring-1.3.tar.gz**

########################################################################################################### 100.0%

==> **Fetching bison**

==> **Downloading https://ghcr.io/v2/homebrew/core/bison/manifests/3.8.2**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/bison/blobs/sha256:78ce4e93936c37005e944b21e4b4d305725bc66f6c675a**

########################################################################################################### 100.0%

==> **Fetching meson**

==> **Downloading https://ghcr.io/v2/homebrew/core/meson/manifests/1.6.0**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/meson/blobs/sha256:77eb91483991e1b615c3fe1dcea843e43706ac3eb011b0**

########################################################################################################### 100.0%

==> **Fetching python-setuptools**

==> **Downloading https://ghcr.io/v2/homebrew/core/python-setuptools/manifests/75.6.0**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/python-setuptools/blobs/sha256:0c0edf5d7de8a5255a90ac7daa168db31d**

########################################################################################################### 100.0%

==> **Fetching glib**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://raw.githubusercontent.com/Homebrew/formula-patches/43467fd8dfc0e8954892ecc08fab131242dca02**

########################################################################################################### 100.0%

==> **Downloading https://download.gnome.org/sources/gobject-introspection/1.82/gobject-introspection-1.82.0.tar.xz**

########################################################################################################### 100.0%

==> **Downloading https://download.gnome.org/sources/glib/2.82/glib-2.82.2.tar.xz**

########################################################################################################### 100.0%

==> **Fetching pixman**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://cairographics.org/releases/pixman-0.44.0.tar.gz**

########################################################################################################### 100.0%

==> **Fetching icu4c@76**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/unicode-org/icu/releases/download/release-76-1/icu4c-76\_1-src.tgz**

==> **Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/49244766/b85410b**

########################################################################################################### 100.0%

==> **Fetching gobject-introspection**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://gitlab.gnome.org/tschoonj/gobject-introspection/-/commit/a7be304478b25271166cd92d110f251a8**

-#O#- # #

==> **Downloading https://files.pythonhosted.org/packages/67/03/fb5ba97ff65ce64f6d35b582aacffc26b693a98053fa831ab43a**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/54/28/3af612670f82f4c056911fbbbb42760255801b3068c48de792d3**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/87/5b/aae44c6655f3801e81aa3eef09dbbf012431987ba564d7231722**

########################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/3e/2c/f0a538a2f91ce633a78daaeb34cbfb93a54bd2132a6de1f6cec0**

########################################################################################################### 100.0%

==> **Downloading https://download.gnome.org/sources/gobject-introspection/1.82/gobject-introspection-1.82.0.tar.xz**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/bec68772f24244dc47ca35bbb5a4e323bc9996c70727b85986d2826626fa4e30--gobject-introspection-1.82.0.tar.xz

==> **Fetching harfbuzz**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/harfbuzz/harfbuzz/raw/fc0daafab0336b847ac14682e581a8838f36a0bf/test/shaping/fon**

==> **Downloading from https://raw.githubusercontent.com/harfbuzz/harfbuzz/fc0daafab0336b847ac14682e581a8838f36a0bf/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/harfbuzz/harfbuzz/archive/refs/tags/10.1.0.tar.gz**

==> **Downloading from https://codeload.github.com/harfbuzz/harfbuzz/tar.gz/refs/tags/10.1.0**

########################################################################################################### 100.0%

==> **Fetching openjdk**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://github.com/openjdk/jdk/commit/ba5a4670b8ad86fefb41a939752754bf36aac9dc.patch?full\_index=1**

########################################################################################################### 100.0%

==> **Downloading https://download.java.net/java/GA/jdk22.0.2/c9ecb94cd31b495da20a27d4581645e8/9/GPL/openjdk-22.0.2\_**

########################################################################################################### 100.0%

==> **Downloading https://github.com/openjdk/jdk23u/archive/refs/tags/jdk-23.0.1-ga.tar.gz**

==> **Downloading from https://codeload.github.com/openjdk/jdk23u/tar.gz/refs/tags/jdk-23.0.1-ga**

-=O=- # # # #

==> **Fetching automake**

==> **Downloading https://ghcr.io/v2/homebrew/core/automake/manifests/1.17**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/automake/blobs/sha256:aaf3cb57d50c48af4886c0cd24340aa6ca5628feac4**

########################################################################################################### 100.0%

==> **Fetching cppunit**

==> **Downloading https://ghcr.io/v2/homebrew/core/cppunit/manifests/1.15.1**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/cppunit/blobs/sha256:7acd81de05bc607287b7133050c269e2ea86ac4e1141**

########################################################################################################### 100.0%

==> **Fetching libtool**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://ftp.gnu.org/gnu/libtool/libtool-2.5.4.tar.xz**

########################################################################################################### 100.0%

==> **Fetching maven**

==> **Downloading https://ghcr.io/v2/homebrew/core/maven/manifests/3.9.9**

########################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/maven/blobs/sha256:106bdaaec0342b1656442dd5d1521b3edf69df22576726**

########################################################################################################### 100.0%

==> **Fetching zookeeper**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://raw.githubusercontent.com/apache/zookeeper/release-3.9.3/conf/logback.xml**

########################################################################################################### 100.0%

==> **Downloading https://www.apache.org/dyn/closer.lua?path=zookeeper/zookeeper-3.9.3/apache-zookeeper-3.9.3.tar.gz**

==> **Downloading from https://dlcdn.apache.org/zookeeper/zookeeper-3.9.3/apache-zookeeper-3.9.3.tar.gz**

########################################################################################################### 100.0%

==> **Fetching kafka**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/a441d45b6da9b1ca05566933c51bf52a99b532a1/**

########################################################################################################### 100.0%

==> **Downloading https://www.apache.org/dyn/closer.lua?path=kafka/3.9.0/kafka\_2.13-3.9.0.tgz**

==> **Downloading from https://dlcdn.apache.org/kafka/3.9.0/kafka\_2.13-3.9.0.tgz**

########################################################################################################### 100.0%

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **Installing dependencies for kafka: python-packaging, ca-certificates, openssl@3, sqlite, xz, pkgconf, python@3.13, libunistring, bison, meson, python-setuptools, glib, pixman, icu4c@76, gobject-introspection, harfbuzz, openjdk, automake, cppunit, libtool, maven and zookeeper**

==> **Installing kafka dependency: python-packaging**

==> **Downloading https://ghcr.io/v2/homebrew/core/python-packaging/manifests/24.2**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/a654ed84d67e434e714ac548959721dfcfc2297e1cb87be9237baee7632a2b76--python-packaging-24.2.bottle\_manifest.json

==> **Pouring python-packaging--24.2.all.bottle.tar.gz**

🍺 /opt/homebrew/Cellar/python-packaging/24.2: 56 files, 491.3KB

==> **Installing kafka dependency: ca-certificates**

==> **Downloading https://ghcr.io/v2/homebrew/core/ca-certificates/manifests/2024-11-26**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/e16b55434e8bc1472ffb41e1a71c1b853417578c631fa1b69e8730f8cafca76c--ca-certificates-2024-11-26.bottle\_manifest.json

==> **Pouring ca-certificates--2024-11-26.all.bottle.tar.gz**

==> **Downloading https://formulae.brew.sh/api/formula.jws.json**

########################################################################################################### 100.0%

==> **Regenerating CA certificate bundle from keychain, this may take a while...**

🍺 /opt/homebrew/Cellar/ca-certificates/2024-11-26: 4 files, 239.4KB

==> **Installing kafka dependency: openssl@3**

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **perl ./Configure --prefix=/opt/homebrew/Cellar/openssl@3/3.4.0 --openssldir=/opt/homebrew/etc/openssl@3 --libd**

==> **make**

==> **make install MANDIR=/opt/homebrew/Cellar/openssl@3/3.4.0/share/man MANSUFFIX=ssl**

==> **make HARNESS\_JOBS=8 test TESTS=-test\_afalg**

🍺 /opt/homebrew/Cellar/openssl@3/3.4.0: 7,227 files, 33.3MB, built in 4 minutes 27 seconds

==> **Installing kafka dependency: sqlite**

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **./configure --enable-dynamic-extensions --enable-readline --disable-editline --enable-session**

**Setting up Kafdrop :**

<https://citizix.com/how-to-install-and-set-up-kafdrop-kafka-web-ui/>   
  
<https://github.com/obsidiandynamics/kafdrop/releases>

**Kafka Cheetsheet**

# Kafka Cheat Sheet

This is a list of commonly used CLI examples, when you work with **Kafka**, **Kafka Connect** and **Schema Registry**. Feel free to use it as well as post extensions to it.

All commands should be executed from **Apache Kafka** or **Confluent Platform** home directory. It is also assumed, that Zookeeper, Brokers, Connect Workers and Schema Registry operate on standard ports. Adjust when necessary.

## Table of contents

* [Kafka Cluster Management](https://github.com/whatsupbros/kafka-cheat-sheet#Kafka-Cluster-Management)
* [Kafka Topics](https://github.com/whatsupbros/kafka-cheat-sheet#Kafka-Topics)
* [Kafka Consumers](https://github.com/whatsupbros/kafka-cheat-sheet#Kafka-Consumers)
* [Kafka Producers](https://github.com/whatsupbros/kafka-cheat-sheet#Kafka-Producers)
* [kafkacat](https://github.com/whatsupbros/kafka-cheat-sheet#kafkacat)
* [Kafka Connect](https://github.com/whatsupbros/kafka-cheat-sheet#Kafka-Connect)
* [Schema Registry](https://github.com/whatsupbros/kafka-cheat-sheet#Schema-Registry)

## Kafka Cluster Management

See also:

* ZooKeeper configuration options: <https://docs.confluent.io/platform/current/zookeeper/deployment.html#configuration-options>
* Brokers configuration options: <https://docs.confluent.io/platform/current/installation/configuration/broker-configs.html>
* REST Proxy v3 for cluster management: <https://docs.confluent.io/platform/current/kafka-rest/api.html#crest-api-v3>

### Environment setup

export KAFKA\_HEAP\_OPTS="-Xmx2G -Xms128M"

export CONFLUENT\_HOME="/path/to/confluent-X.X.X"

export PATH="${CONFLUENT\_HOME}/bin:$PATH"

export LOG\_DIR=/tmp

### Starting Kafka Cluster

Start Zookeeper:

./bin/zookeeper-server-start ./path/to/zookeeper.properties > ./output/zookeeper.out &

Start Kafka Broker:

./bin/kafka-server-start ./path/to/broker0.properties > ./output/broker0.out &

Start Kafka Connect Worker (distributed mode):

./bin/connect-distributed ./path/to/worker0.properties > ./output/worker0.out &

Start Schema Registry:

./bin/schema-registry-start ./path/to/schema-registry.properties > ./output/schema-registry.out &

Start kSQL Server:

./bin/ksql-server-start ./path/to/ksql-server.properties > ./output/ksql-server.out &

Note: Standard output of Kafka Cluster processes will be saved in the specified files in this case. It also can be configured in log4j.properties which logs should be written and where by each process.

### Stopping Kafka Cluster

Note: This command searches for PIDs of all Kafka processes, and stops them gracefully.

#!/bin/bash

jps -m | grep 'QuorumPeerMain\|Kafka\|ConnectDistributed\|SchemaRegistryMain\|KsqlServerMain' | awk '{print $1}' | xargs kill

## Kafka Topics

See also:

* <https://docs.confluent.io/platform/current/installation/configuration/topic-configs.html>
* REST Proxy v3 for topics management: <https://docs.confluent.io/platform/current/kafka-rest/api.html#create-a-topic>

### List topics

./bin/kafka-topics --zookeeper localhost:2181 --list

..or..

./bin/kafka-topics --bootstrap-server localhost:9092 --list

### Describe a topic

./bin/kafka-topics --zookeeper localhost:2181 --describe --topic my-topic

..or..

./bin/kafka-topics --bootstrap-server localhost:9092 --describe --topic my-topic

### Create a topic

Note: To read more about topic configuration options, refer to [official docs](https://docs.confluent.io/platform/current/installation/configuration/topic-configs.html)

./bin/kafka-topics --create --bootstrap-server localhost:9092 --topic my-topic --replication-factor 3 --partitions 3 --config cleanup.policy=compact

### Alter topic config

./bin/kafka-configs --bootstrap-server localhost:9092 --topic my-topic --alter --add-config 'cleanup.policy=compact,retention.ms=86400000,segment.bytes=1073741824'

..or..

# deprecated

./bin/kafka-topics --zookeeper localhost:2181 --topic my-topic --alter --config cleanup.policy=compact --config retention.ms=86400000 --config segment.bytes=1073741824

Note: Usage of kafka-topics command to alter topics configuration is deprecated, usage of kafka-configs command is recommended.

### Delete topic config (reset to default)

./bin/kafka-configs --bootstrap-server localhost:9092 --topic my-topic --alter --delete-config 'cleanup.policy,retention.ms'

### Purge a topic

./bin/kafka-configs --bootstrap-server localhost:9092 --topic my-topic --alter --add-config 'cleanup.policy=delete,retention.ms=100'

...wait a minute...

./bin/kafka-configs --bootstrap-server localhost:9092 --topic my-topic --alter --delete-config 'cleanup.policy,retention.ms'

### Delete a topic

./bin/kafka-topics --bootstrap-server localhost:9092 --delete --topic my-topic

## Kafka Consumers

See also:

* <https://docs.confluent.io/platform/current/tutorials/examples/clients/docs/kafka-commands.html#basic-producer-and-consumer>
* <https://docs.confluent.io/platform/current/installation/configuration/consumer-configs.html>

### Simple consumer console

Plain text:

./bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic my-topic --from-beginning

Avro:

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 --topic my-topic --property schema.registry.url=http://localhost:8081 --from-beginning

### Print key together with value

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property print.key=true \

--property print.value=true \

--property key.separator=":" \

--from-beginning

### Use different deserializers for key and value

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property print.key=true \

--property print.value=true \

--property key.separator=":" \

--key-deserializer "org.apache.kafka.common.serialization.StringDeserializer" \

--value-deserializer "io.confluent.kafka.serializers.KafkaAvroDeserializer" \

--from-beginning

### Use basic auth for Schema Registry

export API\_KEY="USERNAME"

export API\_SECRET="PASSWORD"

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property schema.registry.basic.auth.user.info="$API\_KEY:$API\_SECRET" \

--property basic.auth.credentials.source=USER\_INFO \

--property print.key=true \

--property print.value=true \

--property key.separator=":" \

--key-deserializer "org.apache.kafka.common.serialization.StringDeserializer" \

--value-deserializer "io.confluent.kafka.serializers.KafkaAvroDeserializer" \

--from-beginning

### Use SASL SSL security for Kafka Broker and Schema Registry connection

export API\_KEY="USERNAME"

export API\_SECRET="PASSWORD"

export KAFKA\_TRUSTSTORE\_LOCATION="/path/to/truststore.jks"

export KAFKA\_TRUSTSTORE\_PASSPHRASE="<TRUSTSTORE\_PASSPHRASE>"

export KAFKA\_KEYSTORE\_LOCATION=/path/to/keystore.jks

export KAFKA\_KEYSTORE\_PASSPHRASE="<KEYSTORE\_PASSPHRASE>"

export KAFKA\_KEY\_LOCATION="/path/to/key.pem"

export KAFKA\_KEY\_PASSPHRASE="<KEY\_PASSPHRASE>"

export SCHEMA\_REGISTRY\_OPTS="-Djavax.net.ssl.keyStore=$KAFKA\_KEYSTORE\_LOCATION -Djavax.net.ssl.trustStore=$KAFKA\_TRUSTSTORE\_LOCATION -Djavax.net.ssl.keyStorePassword=$KAFKA\_KEYSTORE\_PASSPHRASE -Djavax.net.ssl.trustStorePassword=$KAFKA\_TRUSTSTORE\_PASSPHRASE"

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 \

--topic my-topic \

--property print.key=true \

--property print.value=true \

--property key.separator=":" \

--key-deserializer "org.apache.kafka.common.serialization.StringDeserializer" \

--value-deserializer "io.confluent.kafka.serializers.KafkaAvroDeserializer" \

--property schema.registry.url=http://localhost:8081 \

--property schema.registry.basic.auth.user.info="$API\_KEY:$API\_SECRET" \

--property basic.auth.credentials.source=USER\_INFO \

--consumer-property security.protocol=SASL\_SSL \

--consumer-property sasl.mechanism=PLAIN \

--consumer-property sasl.jaas.config="org.apache.kafka.common.security.plain.PlainLoginModule required username=\"$API\_KEY\" password=\"$API\_SECRET\";" \

--consumer-property ssl.truststore.location=$KAFKA\_TRUSTSTORE\_LOCATION \

--consumer-property ssl.truststore.password=$KAFKA\_TRUSTSTORE\_PASSPHRASE \

--consumer-property ssl.keystore.location=$KAFKA\_KEYSTORE\_LOCATION \

--consumer-property ssl.keystore.password=$KAFKA\_KEYSTORE\_PASSPHRASE \

--consumer-property ssl.key.password=$KAFKA\_KEY\_PASSPHRASE \

--consumer-property ssl.truststore.type=JKS \

--consumer-property ssl.keystore.type=JKS \

--from-beginning

### Save consumed Kafka messages to a file

./bin/kafka-avro-console-consumer --bootstrap-server localhost:9092 \

--topic my-topic \

--property print.key=true \

--property print.value=true \

--property key.separator=":" \

--property schema.registry.url=http://localhost:8081 \

--from-beginning

--timeout-ms 25000 \

> ./my-topic-data.json

Note: After consuming all topic messages, the console waits for 25 seconds timeout without new messages, and then exits

## Kafka Producers

See also:

* <https://docs.confluent.io/platform/current/tutorials/examples/clients/docs/kafka-commands.html#basic-producer-and-consumer>
* <https://docs.confluent.io/platform/current/installation/configuration/producer-configs.html>

### Simple producer console

Plain text:

./bin/kafka-console-producer --bootstrap-server localhost:9092 --topic my-topic

Avro:

./bin/kafka-avro-console-producer --bootstrap-server localhost:9092 --topic my-topic --property schema.registry.url=http://localhost:8081 --property value.schema='<value-schema-as-json>'

### Parse key together with value

Note: Topic value schema is read from the file /path/to/my-topic-value-schema.json, where schema must be formatted as normal minified JSON (no extra spaces and other whitespace characters).

./bin/kafka-avro-console-producer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property value.schema.file=/path/to/my-topic-value-schema.json \

--property parse.key=true \

--property key.separator=":"

### Use different serializer for key

./bin/kafka-avro-console-producer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property value.schema.file=/path/to/my-topic-value-schema.json \

--property parse.key=true \

--property key.separator=":" \

--property key.serializer="org.apache.kafka.common.serialization.StringSerializer"

### Read Kafka messages to produce from a file

./bin/kafka-avro-console-producer --bootstrap-server localhost:9092 \

--topic my-topic \

--property schema.registry.url=http://localhost:8081 \

--property value.schema.file=/path/to/my-topic-value-schema.json \

--property parse.key=true \

--property key.separator=":" \

--property key.serializer="org.apache.kafka.common.serialization.StringSerializer" \

< ./my-topic-data.json

## [kafkacat](https://github.com/edenhill/kafkacat)

Note: kafkacat is a powerful tool to work with Kafka Cluster written in C. It can operate in both consumer and producer mode, it is fast, it can read topic headers, but it currently does not support all Kafka features (i.e. there is no producer mode for Avro serialized topics).

### kafkacat for Windows

To build kafkacat for Windows from sources, you will need to build it from win32 directory and have these pre-requisites:

1. NuGet package manager (<https://www.nuget.org/downloads>)
2. MS Visual C++ Build Tools 14 for Visual Studio 2015 (<https://visualstudio.microsoft.com/ru/vs/older-downloads/>) - this is not Visual Studio IDE itself, but a subset of stuff to build existing projects. Note this, that exactly this version is currently required to build the project, this can change in future.
3. Add MSBuild location to PATH environment variable (usually it is C:\Program Files (x86)\MSBuild\14.0\Bin) Change version manually to the current one in win32/win32\_config.h (1.6.0 at the moment of writing this)

Now build the project using the official instructions:

cd win32

nuget restore

msbuild # or, with full path: C:\Program Files (x86)\MSBuild\14.0\Bin\msbuild.exe

This should do the trick, you should have the binary then in %kafkacat\_source\_base\_dir%\win32\x64\Debug:

C:\kafkacat\kafkacat-1.6.0\win32\x64\Debug>kafkacat.exe -V

kafkacat - Apache Kafka producer and consumer tool

https://github.com/edenhill/kafkacat

Copyright (c) 2014-2019, Magnus Edenhill

Version 1.6.0 (Transactions, librdkafka 1.5.0 builtin.features=gzip,snappy,ssl,sasl,regex,lz4,sasl\_gssapi,sasl\_plain,sasl\_scram,plugins,zstd,sasl\_oauthbearer)

Note: Currently, the Windows version of kafkacat does not support JSON, Avro and Schema Registry

### kafkacat for Linux

#### Option 1. Install from repository

Ubuntu/Debian example:

sudo apt install kafkacat

Note: kafkacat, installable from Ubuntu (1.5.0-1.1) or Debian (1.6.0-1) repositories does not support working with Avro and Schema Registry, despite the fact that this functionality was added in version 1.5.0.

#### Option 2. Build from sources

To be able to use kafkacat with Avro and Schema registry, download its sources from GitHub and build it yourself from sources with support of libavro-c and libserdes (as it is mentioned here: <https://github.com/edenhill/kafkacat#requirements>)

On Ubuntu 20.04, first, you will need these additional package to build the app:

sudo apt install pkg-config build-essential cmake libtool libssl-dev zlib1g-dev libzstd-dev libsasl2-dev libjansson-dev libcurl4-openssl-dev

Then, you can use bootstrap script to build kafkacat with all dependencies:

./bootstrap.sh

Now you are ready to use it:

$ ./kafkacat -V

kafkacat - Apache Kafka producer and consumer tool

https://github.com/edenhill/kafkacat

Copyright (c) 2014-2019, Magnus Edenhill

Version 1.6.0 (JSON, Avro, Transactions, librdkafka 1.5.0 builtin.features=gzip,snappy,ssl,sasl,regex,lz4,sasl\_gssapi,sasl\_plain,sasl\_scram,plugins,zstd,sasl\_oauthbearer)

### kafkacat examples

See kafkacat usage examples here: <https://github.com/edenhill/kafkacat#examples>

## Kafka Connect

Note: For Kafka Connect and Schema Registry you will need curl and jq utilities to make requests to their APIs.

Note: Alternatively, you can use [this postman workspace](https://www.postman.com/whatsupbros/workspace/03c475b6-174a-40f4-8111-f09d13082d23) to work with Kafka Connect and Schema Registry REST APIs

See also:

* Workers configuration options: <https://docs.confluent.io/platform/current/connect/references/allconfigs.html>
* Connectors configuration options: <https://docs.confluent.io/platform/current/installation/configuration/connect/index.html>
* Kafka Connect REST API Reference: <https://docs.confluent.io/platform/current/connect/references/restapi.html>

### List installed Kafka Connect plugins

curl -Ss -X GET http://localhost:8083/connector-plugins | jq

### List the connectors

curl -Ss -X GET http://localhost:8083/connectors | jq

### Deploy a connector

curl -Ss -X POST -H "Content-Type: application/json" --data @/path/to/my-topic-connector-config.json http://localhost:8083/connectors | jq

### Get connector overview (configuration and tasks overview)

curl -Ss -X GET http://localhost:8083/connectors/<connector-name> | jq

### Get connector config

curl -Ss -X GET http://localhost:8083/connectors/<connector-name>/config | jq

### Get connector status

curl -Ss -X GET http://localhost:8083/connectors/<connector-name>/status | jq

### Restart connector

curl -Ss -X POST http://localhost:8083/connectors/<connector-name>/restart | jq

### Get connector tasks

curl -Ss -X GET http://localhost:8083/connectors/<connector-name>/tasks | jq

### Get connector task status

curl -Ss -X GET http://localhost:8083/connectors/<connector-name>/tasks/0/status | jq

### Restart connector task

curl -Ss -X POST http://localhost:8083/connectors/<connector-name>/tasks/0/restart | jq

### Remove connector

curl -Ss -X DELETE http://localhost:8083/connectors/<connector-name> | jq

### Get current logging levels

curl -Ss http://localhost:8083/admin/loggers | jq

### Set logging level for a particular logger

curl -Ss -X PUT -H "Content-Type:application/json" http://localhost:8083/admin/loggers/<logger-name> -d '{"level": "DEBUG"}' | jq

**Examples:**

# sets debug log level for JDBC connector (source and sink)

curl -Ss -X PUT -H "Content-Type:application/json" http://localhost:8083/admin/loggers/io.confluent.connect.jdbc -d '{"level": "DEBUG"}' | jq

# sets debug log level for Kafka Connect Worker Sink tasks

curl -Ss -X PUT -H "Content-Type:application/json" http://localhost:8083/admin/loggers/org.apache.kafka.connect.runtime.WorkerSinkTask -d '{"level": "DEBUG"}' | jq

# sets debug log level for Kafka Connect in general

curl -Ss -X PUT -H "Content-Type:application/json" http://localhost:8083/admin/loggers/org.apache.kafka.connect -d '{"level": "DEBUG"}' | jq

## Schema Registry

See also:

* Schema Registry REST API Reference: <https://docs.confluent.io/platform/current/schema-registry/develop/api.html>
* REST API Usage Examples: <https://docs.confluent.io/platform/current/schema-registry/develop/using.html#common-sr-api-usage-examples>

### Retrieve currently registered schemas (subjects)

curl -Ss -X GET http://localhost:8081/subjects | jq

### Retrieve schema versions of a subject

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions | jq

### Retrieve schema of a subject of a particular version

With metadata (schema ID, version, schema value as escaped JSON):

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions/1 | jq

Without metadata (schema value as normal JSON):

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions/1/schema | jq

### Retrieve last schema of a subject

With metadata (schema ID, version, schema value as escaped JSON):

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions/latest | jq

Without metadata (schema value as normal JSON):

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions/latest/schema | jq

### Retrieve last schema of a subject, use certificate for the connection to Schema Registry

export API\_KEY="USERNAME"

export API\_SECRET="PASSWORD"

export KEY\_LOCATION="/path/to/key.pem"

export KEY\_PASSPHRASE="<KEY\_PASSPHRASE>"

curl -Ss --cert $KEY\_LOCATION:$KEY\_PASSPHRASE \

-X GET http://$API\_KEY:$API\_SECRET@localhost:8081/subjects/my-topic-value/versions/latest/schema

### Retrieve last schema of a subject and save it into file

Note: Schema saved using this command, is suitable for value.schema.file property of kafka-avro-console-producer utility

curl -Ss -X GET http://localhost:8081/subjects/my-topic-value/versions/latest/schema > ./my-topic-value-schema.json

### Deploy schema for a subject

Note: Schema is read from my-topic-value-schema.json file from the current directory. Mind the fact, that schema value format must be formatted as mimified escaped JSON, i.e.: {"schema":"{\"type\":\"string\"}"}.

curl -Ss -X POST -H "Content-Type: application/json" --data @/path/to/my-topic-value-schema.json http://localhost:8081/subjects/my-topic-value/versions | jq

## About

Kafka, Kafka Connect and Schema Registry commands cheat sheet

## Key Kafka commands

The Kafka CLI is a powerful tool. However, the user experience can be challenging if you don’t already know the exact command needed for your task. The table below shows commonly used CLI commands to interact with Kafka.

|  |  |
| --- | --- |
| Create topics | bin/kafka-topics.sh --bootstrap-server <URL> --create --replication-factor 3 --partitions 4 --topic topic-name |
| List all topics | bin/kafka-topics.sh --bootstrap-server <URL> --list |
| Add a topic partition | bin/kafka-topics.sh --bootstrap-server <URL> --alter --topic <topic-name> --partitions 16 |
| Run a producer | bin/kafka-console-producer.sh --topic <topic-name> --broker-list <URL> |
| Run a consumer | bin/kafka-console-consumer.sh --bootstrap-server <URL> --topic <topic-name> |
| Consume messages from the beginning | bin/kafka-console-consumer.sh --topic <topic-name> --bootstrap-server <URL> --group <group-name>--from-beginning |
| Get information on a specific consumer group | bin/kafka-consumer-groups.sh --bootstrap-server <URL> --describe --group <group-name> |
| Delete a consumer group | /bin/kafka-consumer-groups.sh --bootstrap-server <URL> --delete --group my-group --group <group-name> |
| Reset the offset of a topic | bin/kafka-consumer-groups.sh --bootstrap-server <URL> --reset-offsets --group <group-name> --topic <topic-name> --to-earliest |

## [How to install and set up Kafdrop – Kafka Web UI](https://citizix.com/how-to-install-and-set-up-kafdrop-kafka-web-ui/)

Apr 16, 2022

8 minute read

Kafdrop is a web UI for viewing Kafka topics and browsing consumer groups. The tool displays information such as brokers, topics, partitions, consumers, and lets you view messages.

Apache Kafka is an open-source platform. Kafka was originally developed by Linkedin and was later incubated as the Apache Project. It can process over 1 million messages per second.

Kafka is an amazing platform for processing a huge number of messages very quickly. However, Kafka has one disadvantage that it does not come with an inbuilt User Interface where the users can see the information related to Kafka.

Kafdrop helps us in solving this problem. It gives us a simple, lightweight, and easy-to-use User Interface where one can not only see the required information but can also create and delete Kafka topics.

#### Features

* **View Kafka brokers** - topic and partition assignments, and controller status
* **View topics** - partition count, replication status, and custom configuration
* **Browse messages** - JSON, plain text, Avro and Protobuf encoding
* **View consumer groups** - per-partition parked offsets, combined and per-partition lag
* **Create new topics**
* **View ACLs**
* **Support for Azure Event Hubs**

#### Requirements

* Java 11 or newer
* Kafka (version 0.11.0 or newer) or Azure Event Hubs

Optional, additional integration:

* Schema Registry

Kafdrop can be installed by executing a JAR file or via docker or on Kubernetes. In this guide we will set up Kafdrop JAR directly.

Also checkout:

* [How to install Apache Kafka on Rocky Linux or AlmaLinux8](https://citizix.com/how-to-install-apache-kafka-on-rocky-linux-or-almalinux8/)

## Installing Java

Kafdrop needs Java to run, hence first we need to install that on our local environment and it must be equal or greater than Java 11. Well, we don’t need to add any third repository because the package to get JAVA is already there on the system base repo.

Let us install latest Java on RHEL 8 based distributions with the following command. For other Linux Distributions please consult your package manager manual.

|  |  |
| --- | --- |
|  | sudo dnf install java-11-openjdk |

Type y and press enter when prompted to accept the installation.

## Get the latest Kafdrop

Kafdrop is available as a jar file. Get the latest release from github release page [here](https://github.com/obsidiandynamics/kafdrop/releases).

We are going to download and install kafka in the /opt/kafka directory. First become root and set up the required directory structure

|  |  |
| --- | --- |
| 1  2 | sudo mkdir/opt/kafdrop  cd /opt/kafdrop/ |

Next, download the latest kadrop and rename the file. In this guide we are downloading version 3.30.0.

|  |
| --- |
| curl -LO https://github.com/obsidiandynamics/kafdrop/releases/download/3.30.0/kafdrop-3.30.0.jar  mv kafdrop-3.30.0.jar kafdrop.jar |

## Running the Jar

Once the jar is downloaded, we can run it with the java -jar command. This is the command:

|  |
| --- |
| java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \-jar /opt/kafdrop/kafdrop.jar \--kafka.brokerConnect=localhost:9092 |

If unspecified, kafka.brokerConnect defaults to localhost:9092.

**Note:** As of Kafdrop 3.10.0, a ZooKeeper connection is no longer required. All necessary cluster information is retrieved via the Kafka admin API.

Once it starts, Open a browser and navigate to http://server\_ip:9000/. The port can be overridden by adding the following config:

|  |  |
| --- | --- |
| 1 | --server.port=<port> --management.server.port=<port> |

Optionally, configure a schema registry connection with:

|  |  |
| --- | --- |
| 1 | --schemaregistry.connect=http://localhost:8081 |

and if you also require basic auth for your schema registry connection you should add:

|  |  |
| --- | --- |
| 1 | --schemaregistry.auth=username:password |

Finally, a default message format (e.g. to deserialize Avro messages) can optionally be configured as follows:

|  |  |
| --- | --- |
| 1 | --message.format=AVRO |

Valid format values are DEFAULT, AVRO, PROTOBUF. This can also be configured at the topic level via dropdown when viewing messages.

## Create a Systemd unit for Kafdrop service

When running Kafdrop Service in a production server we have to run it in the background. Hence, create systemd units for both the scripts.

Create a kafdrop systemd service file

|  |  |
| --- | --- |
| 1 | sudo vim /etc/systemd/system/kafdrop.service |

Add this content to the file

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | [Unit]  Description=Kafdrop server  Documentation=https://github.com/obsidiandynamics/kafdrop  Requires=network.target remote-fs.target  After=network.target remote-fs.target  [Service]  Type=simple  ExecStart=/bin/java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \  -jar /opt/kafdrop/kafdrop.jar \  --kafka.brokerConnect=localhost:9092  Restart=on-abnormal  [Install]  WantedBy=multi-user.target |

Finally, save and close the file. To ensure that the service is recognized, reload systemd units:

|  |  |
| --- | --- |
| 1 | sudo systemctl daemon-reload |

## Start and enable Kafdrop systemd service

Now, let’s start and enable the service to make sure they will also get active even after the system reboot.

Start kafdrop

|  |  |
| --- | --- |
| 1 | sudo systemctl start kafdrop |

Confirm the services status to ensure that they are both running as expected:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | $ sudo systemctl status kafdrop  kafdrop.service - Kafdrop server  Loaded: loaded (/etc/systemd/system/kafdrop.service; disabled; vendor preset: disabled)  Active: active (running) since Sat 2022-04-16 10:49:28 UTC; 21s ago  Docs: https://github.com/obsidiandynamics/kafdrop  Main PID: 78642 (java)  Tasks: 22 (limit: 23167)  Memory: 334.7M  CGroup: /system.slice/kafdrop.service  &#x2514;&#x2500;78642 /bin/java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED -jar /opt/kafdrop/kafdrop.jar --kafka.brokerConnect=localhost:9092  Apr 16 10:49:35 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:35.046 INFO 78642 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks  Apr 16 10:49:35 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:35.046 INFO 78642 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties  Apr 16 10:49:35 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:35.128 INFO 78642 [ main] k.s.BuildInfo : Kafdrop version: 3.30.0, build time: 2022-04->  Apr 16 10:49:36 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:36.039 INFO 78642 [ main] o.s.b.a.e.w.EndpointLinksResolver : Exposing 13 endpoint(s) beneath base path '/a>  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:36.996 INFO 78642 [ main] i.u.Undertow : starting server: Undertow - 2.2.16.Final  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:37.011 INFO 78642 [ main] o.x.Xnio : XNIO version 3.8.6.Final  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:37.029 INFO 78642 [ main] o.x.n.NioXnio : XNIO NIO Implementation Version 3.8.6.Final  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:37.075 INFO 78642 [ main] o.j.t.Version : JBoss Threads version 3.1.0.Final  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:37.139 INFO 78642 [ main] o.s.b.w.e.u.UndertowWebServer : Undertow started on port(s) 9000 (http)  Apr 16 10:49:37 rockysrv.citizix.com java[78642]: 2022-04-16 10:49:37.727 INFO 78642 [ main] o.s.b.StartupInfoLogger : Started Kafdrop in 7.411 seconds (JVM runnin |

Finally enable the service on boot:

|  |  |
| --- | --- |
| 1 | sudo systemctl enable kafdrop |

Once the service is successfully started, you can access the UI.

## Navigating the UI

Open a browser and navigate to http://server\_ip:9000/.

The **Cluster Overview** screen is the landing page of the web UI.

You get to see the overall layout of the cluster - the individual brokers that make it up, their addresses and some key broker stats - whether they are a controller and the number of partitions each broker owns. The latter is quite important - as your cluster size and the number of topics (and therefore partitions) grows, you generally want to see an approximately level distribution of partitions across the cluster.

Next is the **Topics List**, which in most cases is what you’re really here for. Any reasonably-sized microservices-based ecosystem might have hundreds, if not thousands of topics. As you’d expect, the list is searchable. The stats displayed alongside each topic are fairly ho-hum. The one worth noting is the under-replicated column. Essentially, it’s telling us the number of partition replicas that have fallen behind the primary. Zero is a good figure. Anything else is indicative of either a broker or a network issue that requires immediate attention.

Click on a topic in the list to get to the **Topic Overview** screen.

The screen is subdivided into four sections.

On the top-left, there is a summary of the topic stats - a handy view, not dissimilar to what you would have seen in the cluster overview.

On the top-right, you can view the custom configuration. In the example above, the topic runs a stock-standard config, so there’s nothing to see. Had the configuration been overridden, you’d see a set of custom values like in the example below.

The bottom-left section enumerates over the partitions. The partition indexes are links - clicking through will reveal the first 100 messages in the topic.

The **consumers** section on the bottom-right lists the consumer group names as well as their aggregate lag (the sum of all individual partition lags).

Clicking on the consumer group on the Topic Overview gets you into the **Consumer View**. This screen provides a comprehensive breakdown of a single consumer group.

The view is sectioned by topic. For each topic, a separate table lists the underlying partitions. Against each partition, we see the committed offset, which we can compare against the first and last offsets to see how our consumer is tracking. Conveniently, Kafdrop displays the computed lag for each partition, which is aggregated at the footer of each topic table.

The **Message View** screen is the coveted topic viewer that has in all likelihood brought you here. You can get to the message view in one of two ways:

1. Click the **View Messages** button in the Topic Overview screen.
2. Click the individual partition link in the Topic Overview.

It’s exactly what you’d expect - a chronologically-ordered list of messages (or records, in Kafka parlance) for a chosen partition.

Each entry conveniently displays the offset, the record key (if one is set), the timestamp of publication, and any headers that may have been appended by the producer.

There’s another little trick up Kafdrop’s sleeve. If the message happens to be a valid JSON document, the topic viewer can nicely format it. Click on the green arrow on the left of the message to expand it.

## Conclusion

In this guide we learnt how to install and use the Kafdrop Kafka UI.



·

Sep 29, 2019

Do you make changes to your Kafka cluster using the CLI?  
Do you always have to look up the CLI commands and options?  
My Apache Kafka, CLI cheat sheet might be helpful for you!

In this short blog post, you find my Kafka CLI cheatsheet (I’ll add more commands to the cheat sheet regularly).

# Overview

[**Docker**](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#471a)

* [Kafka & Zookeeper Docker Compose file](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#e260)
* [Start Kafka & Zookeeper using Docker Compose](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#e260)
* [Attach to the Kafka Broker running in Docker](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#5745)
* [Unset the JMX port in the Kafka Docker container](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#5745)
* [Stop Kafka & Zookeeper using Docker Compose](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#84e3)

[**Cluster and Broker(s)**](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#a689)

* [Show the version of the Kafka broker](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#8198)
* [Connect to Zookeeper](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#8198)
* [Show the Kafka cluster id](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#6260)
* [List brokers in the Kafka cluster](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#497b)
* [Show details of a Kafka broker](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#81cb)
* [Show all the topics that exist in the cluster](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#93c2)
* [Show details of a specific topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#cd67)
* [Exit the zookeeper shell](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#c481)

[**Topics**](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#6fd2)

* [Create a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#8c2f)
* [Create a Kafka topic in case the topic doesn’t exist](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#09e8)
* [Create a Kafka topic with a short retention period](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#47c2)
* [List the Kafka topics](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#a1a2)
* [List Kafka topics and exclude internal topics](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#c9bd)
* [Show the Kafka topic details](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#477b)
* [Increase the number of partitions of a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#1bbb)
* [Change the retention time of a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#2020)
* [Purge a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#b94e)
* [List Kafka topics (with configuration values) that have specific configuration overrides](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#38bc)
* [Show specific Kafka topic configuration overrides](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#f61a)
* [Show partition offsets of a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#0bf4)
* [Show offset for specific partition(s) of a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#f2c5)
* [Delete a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#0d0c)

[**Producers**](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#0bd9)

* [Produce a message on a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#f574)
* [Produce messages on a Kafka topic from a file](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#abe2)
* [Produce messages to Kafka with both key and value](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#8719)

[**Consumers**](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#5c14)

* [Consume from a Kafka topic](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#fe4f)
* [Consume a Kafka topic and show both key, value and timestamp](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#8719)
* [Consume a topic from a Kafka topic from the beginning](https://medium.com/@TimvanBaarsen/apache-kafka-cli-commands-cheat-sheet-a6f06eac01b#995a)

## Kafka CLI

All the commands used in this blogpost are included in the Apache Kafka distribution. The file extension of the scripts in the Apache Kafka distribution is .sh

Be aware that the Confluent Kafka distribution dropped the .sh file extension!

# Docker

For every command, I also provide a practical working example you can execute in a running Kafka Docker container. I’m using the Confluent Kafka distribution in this cheat sheet.

## Kafka & Zookeeper Docker Compose file

You start a Kafka broker (including Zookeeper) using this docker-compose example file:

Copy the content in a file with the name: docker-compose.yml

docker-compose.yml file to run Kafka and Zookeeper. <https://gist.github.com/j-tim/cb630c3646b5987e45e2b8cf847f4149>

## Start Kafka & Zookeeper using Docker Compose

Start the official [Confluent Kafka](https://hub.docker.com/r/confluentinc/cp-kafka/tags) and [Zookeeper Docker](https://hub.docker.com/r/confluentinc/cp-zookeeper/tags) containers using [Docker Compose](https://docs.docker.com/compose/):

docker-compose up -d

In case you don’t have the docker images on your system, docker will pull the images from Dockerhub.

## Attach to the Kafka Broker running in Docker

Attach to the Kafka Docker container to execute operations on your Apache Kafka cluster.

docker exec -it kafka bash

## Unset the JMX port in the Kafka Docker container

In case you configured a JMX\_PORT in your docker-compose file for the Kafka Docker container you have to unset the JMX\_PORT environment variable

unset JXM\_PORT

otherwise, you will run into issues when executing the CLI like:

JMX port already in use

## Stop Kafka & Zookeeper using Docker Compose

docker-compose down -v

# Cluster and broker(s)

## Show the version of the Kafka broker

$KAFKA\_HOME/bin/kafka-broker-api-versions.sh --bootstrap-server localhost:9092 --version

🐳 Example to execute in Docker:

./usr/bin/kafka-broker-api-versions --bootstrap-server localhost:9092 --version

## Connect to Zookeeper

Apache Kafka is depending\* on Apache Zookeeper for:

* cluster membership
* controller election
* topic configuration
* access control lists
* quotas.

The meta-data for those operations are stored outside of Kafka in a separate Zookeeper cluster.

\* from version 2.8 onwards Apache Kafka is not depending on Zookeeper anymore. For more details see: [Apache Kafka Needs No Keeper: Removing the Apache ZooKeeper Dependency](https://www.confluent.io/blog/removing-zookeeper-dependency-in-kafka/) on the Confluent blog.

To be able to see metadata of the Kafka cluster from Zookeeper first connect to Zookeeper using the zookeeper-shell command that ships with the Kafka distribution.

$KAFKA\_HOME/bin/zookeeper-shell.sh $ZK\_HOSTS:2181

🐳 Example to execute in Docker:

./usr/bin/zookeeper-shell zookeeper:2181

Output:

Connecting to zookeeper:2181  
Welcome to ZooKeeper!  
JLine support is disabledWATCHER::WatchedEvent state:SyncConnected type:None path:null

You are now connected to the ZooKeeper cluster.

## Show the Kafka cluster id

$KAFKA\_HOME/bin/zookeeper-shell $ZK\_HOSTS:2181 get /cluster/id

🐳 Example to execute in Docker:

./usr/bin/zookeeper-shell zookeeper:2181 get /cluster/id

Output:

Connecting to zookeeper:2181WATCHER::WatchedEvent state:SyncConnected type:None path:null  
{"version":"1","id":"cDpWBoJpQraTnNqSwB\_4Tg"}

In this example, the id of the Kafka cluster is: cpWBoJpQraTnNqSwB\_4Tg

## List brokers in the Kafka cluster

Let’s list the broker in the Kafka cluster

ls /brokers/ids

Output:

[1]

In my docker setup, there is only one Kafka broker in the cluster.  
The id of the Kafka broker is: 1

## Show details of a Kafka broker

get /brokers/ids/1

## Show all the topics that exist in the cluster

ls /brokers/topics

Output:

[my-first-topic]

## Show details of a specific topic

get /brokers/topics/my-first-topic

## **Exit the zookeeper shell**

exit

From Kafka version 3.0.0. onwards ‘exit’ doesn’t work ‘quit’ should be used instead.

# Topics

In this chapter, you find the CLI command and options that are related to Kafka topics.

## Create a Kafka topic

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS **--create** --topic $TOPIC\_NAME --partitions 3 --replication-factor 1

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 **--create** --topic my-first-topic --partitions 3 --replication-factor 1

## Create a Kafka topic in case the topic doesn’t exist

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS --create --topic $TOPIC\_NAME --partitions 3 --replication-factor 1 **--if-not-exists**

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 --create --topic my-first-topic --partitions 3 --replication-factor 1 **--if-not-exists**

Note the CLI will not give an error in case the topic already exists.

## Create a Kafka topic with a short retention

By default, a Kafka topic has a retention period of [7 days](https://kafka.apache.org/documentation/#topicconfigs_retention.ms).  
This example shows the command to create a topic with a retention period of 10 seconds.

./usr/bin/kafka-topics --bootstrap-server localhost:9092 --create --topic my-topic-with-short-retention-period --partitions 3 --replication-factor 1 --config retention.ms=10000 --config segment.ms=10000

## List Kafka topics

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS **--list**

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 **--list**

## List Kafka topics and exclude internal topics

Exclude listing of external topics like “\_\_consumer\_offsets”

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS **--list** --exclude-internal

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 **--list** --exclude-internal

## Show the Kafka topic details

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS --topic $TOPIC\_NAME **--describe**

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 --topic my-first-topic **--describe**

## Increase the number of partitions of a Kafka topic

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS   
**--alter** --topic my-first-topic **--partitions** 5

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 **--alter** --topic my-first-topic **--partitions** 5

Note:

* The number of partitions for a topic can only be increased.
* If partitions are increased for a topic that has a key, the partition logic or ordering of the messages will be affected.

## Change the retention time of a Kafka topic

259200000 ms = 3 days

$KAFKA\_HOME/bin/kafka-configs.sh --zookeeper $ZK\_HOSTS **--alter** --entity-type topics --entity-name my-first-topic **--add-config retention.ms=259200000**

🐳 Example to execute in Docker:

./usr/bin/kafka-configs --zookeeper zookeeper:2181 **--alter** --entity-type topics --entity-name my-first-topic **--add-config retention.ms=259200000**

## Purge a Kafka topic

At the time of writing, there is no single command to purge a topic.  
As a workaround, you can purge a topic by changing the retention time of a topic to one minute.

$KAFKA\_HOME/bin/kafka-configs.sh --zookeeper $ZK\_HOSTS **--alter** --entity-type topics --entity-name my-first-topic **--add-config retention.ms=1000**

Wait one minute. Kafka will now remove records older than one minute from the topic. Now delete the retention.ms configuration from the topic so it will default back to the retention configuration of the Kafka cluster. The default retention period of a Kafka topic is seven days (unless you configured it differently).

$KAFKA\_HOME/bin/kafka-configs.sh --zookeeper $ZK\_HOSTS **--alter** --entity-type topics --entity-name my-first-topic **--delete-config retention.ms**

In case you had a specific retention period specified on the topic. You need to apply that retention period again (in this example 3 days retention period)

$KAFKA\_HOME/bin/kafka-configs.sh --zookeeper $ZK\_HOSTS --alter --entity-type topics --entity-name my-first-topic **--add-config retention.ms=259200000**

🐳 Example to execute in Docker:

./usr/bin/kafka-configs --zookeeper zookeeper:2181 **--alter** --entity-type topics --entity-name my-first-topic **--add-config retention.ms=1000**

Wait one minute:

./usr/bin/kafka-configs --zookeeper zookeeper:2181 **--alter** --entity-type topics --entity-name my-first-topic **--delete-config retention.ms**

## List Kafka topics (with configuration values) that have specific configuration overrides

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS --describe --topics-with-overrides

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 --describe --topics-with-overrides

## Show specific Kafka topic configuration overrides

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper $ZK\_HOSTS --describe --entity-type topics --entity-name my-first-topic

🐳 Example to execute in Docker:

./usr/bin/kafka-configs --zookeeper zookeeper:2181 --describe --entity-type topics --entity-name my-first-topic

Note in case there a no particular property overrides for the topic this command will not show default cluster properties applied to the topic.

## Show partition offsets of a Kafka topic

$KAFKA\_HOME/bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic $TOPIC\_NAME

🐳 Example to execute in Docker:

./usr/bin/kafka-run-class kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic my-first-topic

Example output:

my-first-topic:0:3  
my-first-topic:1:3  
my-first-topic:2:3

Format is: topicname:partition-id:offset  
In this example, the output means: on all 3 partitions of topic with the name my-first-topicthere are 3 records.

By default, the latest offset for all partitions is shown.   
Configuration options:

* latest offset: --time -1
* earliest offset: --time -2

to show the earliest offset:

./usr/bin/kafka-run-class kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic my-first-topic --time -2

Example output:

my-first-topic:0:0  
my-first-topic:1:0  
my-first-topic:2:0

## Show offset for specific partition(s) of a Kafka topic

$KAFKA\_HOME/bin/kafka-run-class.sh kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic $TOPIC\_NAME --partitions partition-id, another-partition-id

🐳 Example to execute in Docker:

./usr/bin/kafka-run-class kafka.tools.GetOffsetShell --broker-list localhost:9092 --topic my-first-topic --partitions 0,2

Example output:

my-first-topic:0:3  
my-first-topic:2:3

## Delete a Kafka topic

$KAFKA\_HOME/bin/kafka-topics.sh — zookeeper $ZK\_HOSTS **--delete**   
--topic $TOPIC\_NAME

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 **--delete** --topic my-first-topic

# Producers

## Produce a message to a Kafka topic

$KAFKA\_HOME/usr/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic $TOPIC\_NAME

A prompt will open:

>

Type in a message and press enter to publish it to the topic:

>Hello World

Note:

* Repeat the same step to publish more messages
* The message published to Kafka has a null key
* Press Ctrl+C to exit

🐳 Example to execute in Docker:

./usr/bin/kafka-console-producer --broker-list localhost:9092 --topic my-first-topic

## Produce messages to a Kafka topic from a file

Example file topic-input.txt (make sure each message is on a new line):

Hello World  
Kafka Rocks!  
Happy Streaming

Produce messages to the topic from the file:

$KAFKA\_HOME/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic my-first-topic < topic-input.txt

🐳 Example to execute in Docker:

./usr/bin/kafka-console-producer --broker-list localhost:9092 --topic my-first-topic < topic-input.txt

## Produce messages to Kafka with both key and value

By default messages sent to a Kafka topic will result in messages with nullkeys. In this example, the separator between the key and the value is: :

$KAFKA\_HOME/usr/bin/kafka-console-producer.sh --broker-list localhost:9092 --topic $TOPIC\_NAME **--property parse.key=true --property key.separator=:**

🐳 Example to execute in Docker:

./usr/bin/kafka-console-producer --broker-list localhost:9092 --topic some-topic **--property parse.key=true** **--property key.separator=\***

Example input:

>key:value>foo:bar>anotherKey:another value

# Consumers

## Consume a Kafka topic

$KAFKA\_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic $TOPIC\_NAME

🐳 Example to execute in Docker:

./usr/bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic my-first-topic

If a consumer group id is not specified, the kafka-console-consumer generates a random consumer group.

## Consume a Kafka topic and show both key, value and timestamp

By default, the console consumer will only the value of the Kafka record.  
Using this command you can show both the key and value.

$KAFKA\_HOME/bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic some-topic **--formatter kafka.tools.DefaultMessageFormatter --property print.timestamp=true --property print.key=true --property print.value=true**

🐳 Example to execute in Docker:

./usr/bin/kafka-console-consumer --bootstrap-server localhost:9092 \ --topic some-topic \  
**--formatter kafka.tools.DefaultMessageFormatter \  
--property print.timestamp=true** **\  
--property print.key=true \   
--property print.value=true**

## Consume a Kafka topic from the beginning

$KAFKA\_HOME/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic my-first-topic **--from-beginning**

🐳 Example to execute in Docker:

./usr/bin/kafka-console-consumer --bootstrap-server localhost:9092 --topic my-first-topic **--from-beginning**

# Miscellaneous

Find all the partitions where one or more of the replicas for the partition are not in-sync with the leader.

$KAFKA\_HOME/bin/kafka-topics.sh --zookeeper localhost:2181 --describe --under-replicated-partitions

🐳 Example to execute in Docker:

./usr/bin/kafka-topics --zookeeper zookeeper:2181 --describe --under-replicated-partitions

**Setting up Artemis AMQP**

# High availability and failover configurations are set up on two VMs of ActiveMQ Artemis.

Aug 17, 2023

--

High availability and Failover is when the system can keep working even if one or more servers stop working.

Failover is a part of high availability and it means that if one server (live server) fails, the client connections can move to another server (backup server) so that the client applications can keep working without any interruption.

Apache ActiveMQ Artemis supports two different strategies for backing up a server shared store and replication. Which is configured via the ha-policy configuration element.

**Replication**

Replicated clusters in ActiveMQ Artemis are setups where the data is synchronized between the master and slave brokers. When messages are sent to the master broker, the data is replicated to all the configured slave brokers. This ensures that even if the master broker goes down, the data is available on the slave brokers, allowing them to take over the role of the master and continue serving clients.

**Shared-store**

When using a shared store, both live and backup servers share the same entire data directory using a shared file system. When failover occurs and a backup server takes over, it will load the persistent storage from the shared file system and clients can connect to it.

Both servers we use replicated clustered configurations.

# Pre-Installations

Before install Artemis make sure you have Java 11 or later version installed on your machine. If not you can install JDK from the following link. <https://www.oracle.com/java/technologies/downloads/>

In this project, I used two different operating systems. It is not a necessity for them to be different, but I personally use MacOS and set up a Linux virtual machine through Parallels. For this reason, I will provide installation details for both operating systems. If you also want to install Parallels for Mac (M1), you can do so through this website: <https://www.parallels.com/eu/>

# Install and Run ActiveMQ on MacOS (M1)

• **Download Apache ActiveMQ Artemis**: Visit the official Apache ActiveMQ Artemis website (<https://activemq.apache.org/components/artemis/download/>) and download the latest version of the software.

• **Extract the Archive**: Once the download is complete, extract the downloaded archive to a location of your choice on your Mac.

* Set Environment Variables: Open the Terminal and set the ARTEMIS\_HOME environment variable to point to the extracted folder’s location. For example:

$ export ARTEMIS\_HOME=/path/to/extracted/folder

* **Create a Broker Instance**: Run the following command to create a new broker instance:

$ cd $ARTEMIS\_HOME/bin  
$ ./artemis create <instance-name>

**Error! Filename not specified.**

Creating a broker instance

* **Start the Server:** After creating the broker instance, navigate to its bin directory and run the following command to start broker:

$ ./artemis run

* Now you can open the login page <http://localhost:8161/console> in your browser.

**Error! Filename not specified.**

Login page

# Install and Run ActiveMQ on Ubuntu 22.04

* We should not run Artemis under the root user for security reasons. So, we need to create a group artemis and add a user artemis and we are going to install Artemis under /opt/artemis directory:

$ sudo groupadd artemis  
$ sudo useradd -s /bin/false -g artemis -d /opt/artemis artemis

* Now we can download Artemis version 2.29.0 on the /opt directory

$ cd /opt  
$ sudo wget https://archive.apache.org/dist/activemq/activemq-artemis/2.29.0/apache-artemis-2.29.0-bin.tar.gz

Version of slave device should be same version as master. My master’s version is 2.29.0.

* Ater finish the download extract the tar package and rename the extracted directory to **artemis**

$ sudo tar -xvzf apache-artemis-2.8.1-bin.tar.gz  
$ sudo mv apache-artemis-2.8.1 artemis

* We need to change the permission and the ownership of the Artemis Home directory. We will also give executed permission to opt/artemis/bin/ directory.

$ sudo chown -R artemis: artemis  
$ sudo chmod o+x /opt/artemis/bin/

* And finally we can create an instance like we do in the Mac. I will create into /var/lib directory.

$ cd /var/lib  
$ sudo /opt/artemis/bin/artemis create test-broker

**Error! Filename not specified.**

Create a broker instance

* We can run using the below command

$ sudo /var/lib/test-broker/bin/artemis run

* Once the broker instance starts we can access the admin console at <http://localhost:8161/console> and the login page will appear.

**Error! Filename not specified.**

Login page

# Create Replicated Brokers

After setting up the installations and test brokers for the two different operating systems, we can now proceed to configure the replica cluster configurations. To make servers replicated we can create as replicated clustered. On master device we need to create master instance like the below command:

$ ./artemis create –-replicated –-clustered master

For slave broker we need to run slave command and create the backup instance on Ubuntu.

$ ./artemis create –-replicated –-clustered slave

After we run these commands both of the systems ask you for additional inputs:

* **— user :** it is the default username to login Artemis.
* **— password :** it is the login password.
* **— host :** your device IP (master’s or slave’s).
* **— cluster-user :** a username for cluster connection.
* **— cluster-password :** password for cluster connection.
* **— allow-anonymous | — require-login** **:** allow anonymous access or not.

**Error! Filename not specified.**

Replicated broker master example

However, some manual changes may still be required to set up the replication. We need to edit broker.xml files to each instances as I will explained below.

# Cluster Configurations Of Master And Slave Brokers

Artemis provides cluster configuration to create a group of brokers that work together. This cluster setup allows multiple brokers to share the workload and handle failover seamlessly. In case one broker fails, another can take over to ensure uninterrupted message processing.

**Both brokers include these three main components:**

## <broadcast-groups>

* Helps brokers discover each other even if they are on different network segments

<broadcast-groups>  
 <broadcast-group name="gss-broadcast">  
 <group-address>${udp-address:231.7.7.7}</group-address>  
 <group-port>9876</group-port>  
 <broadcast-period>100</broadcast-period>  
 <connector-ref>artemis</connector-ref>  
 </broadcast-group>  
 </broadcast-groups>

## <discovery-groups>

* Allows brokers to learn about each other dynamically and join the cluster.

<discovery-groups>  
 <discovery-group name="gss-discovery">   
 <group-address>${udp-address:231.7.7.7}</group-address>  
 <group-port>9876</group-port>  
 <refresh-timeout>10000</refresh-timeout>  
 </discovery-group>  
 </discovery-groups>

## <cluster-connections>

* Enables brokers to share information, such as message state and data replication, to achieve high availability and failover. Slave cluster can discover if a live server is already running, see [check-for-live-server](https://activemq.apache.org/components/artemis/documentation/1.1.0/ha.html).
* Master broker: include [<use-duplicate-detection>](https://activemq.apache.org/components/artemis/documentation/1.1.0/ha.html) for filtering out duplicate messages without you having to code your own fiddly duplicate detection logic at the application level.

<cluster-connections>  
 <cluster-connection name="gss-cluster">  
 <address>jms</address>  
 <connector-ref>artemis</connector-ref>  
 <retry-interval>500</retry-interval>  
 <use-duplicate-detection>true</use-duplicate-detection>  
 <message-load-balancing>ON\_DEMAND</message-load-balancing>  
 <max-hops>1</max-hops>  
 <static-connectors>  
 <connector-ref>artemis</connector-ref>  
 </static-connectors>  
 </cluster-connection>  
 </cluster-connections>

* Slave broker : same configuration but without duplicate detection.
* To connect clusters we need to add user and password information

<cluster-user>admin</cluster-user>  
 <cluster-password>admin</cluster-password>

We should make the master and slave configurations to allows to be linked together as live — backup groups.

# Configuration Of Master Broker

Master cluster is the live server until it is down… After we done with cluster and groups we need to add HA Policies.

This is the HA policies of master:

<ha-policy>  
 <replication>  
 <master>  
 <check-for-live-server>true</check-for-live-server>  
 </master>  
 </replication>  
 </ha-policy>

**<check-for-live-server> :** This option is only necessary for performing ‘fail-back’ on replicating servers. If set, backup servers will only pair with live servers with matching group-name.

## **NOTE:**

We need to start with <connectors> and <acceptors>. For master cluster we need to change the connection and also acceptors IP’s with master device IP.

<connectors>  
 <connector name="artemis">tcp://Master\_device\_IP:61616</connector>   
</connectors>

<acceptor name= "artemis">tcp://Master\_device\_IP:61616?tcpSendBufferSize= 1048576;tcpReceiveBufferSize=1048576;amqpMinLargeMessageSize=102400;protocols=CORE,AMQP,STOMP,HORNETQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpLrnalManagementObjects=false</acceptor>

# Configuration Of Slave Broker

Slave broker is the backup server that when live server is down the backup server will replace and become the current live server. Here are some HA policies for the slave broker that we need to make:

* **< allow-failback> :** Whether a server will automatically stop when a another places a request to take over its place. The use case is when the backup has failed over

<ha-policy>  
 <replication>  
 <slave>  
 <allow-failback>true</allow-failback>  
 </slave>  
 </replication>  
 </ha-policy>

* **<connectors> :** These connectors allow clients to connect to both the master and slave brokers in our messaging system.

<connectors>  
 <connector name="master-broker">tcp://Master\_device\_IP:61616</connector>  
 <connector name="slave-broker">tcp://Slave\_device\_IP:61616</connector>  
 </connectors>

* **<acceptors>** : Allows TCP settings and protocol options for Artemis messaging

<acceptor name= "artemis">tcp://Slave\_device\_IP:61616?tcpSendBufferSize= 1048576;tcpReceiveBufferSize=1048576;amqpMinLargeMessageSize=102400;protocols=CORE,AMQP,STOMP,HORNETQ,MQTT,OPENWIRE;useEpoll=true;amqpCredits=1000;amqpLrnalManagementObjects=false</acceptor>

# Run Brokers

After we done with configurations we can test by running each broker.

* First run the master broker on Mac and open the login page <http://localhost:8161/console> from your browser. Enter the user and password as you configured.
* Now we can run the slave broker on Ubuntu and open the login page.
* We can see from Cluster Info section that backup server is connected to the master server.
* If we kill the master the slave automatically will be the current live server (but like copy of master).

## **Master broker before kill**

It shows is live and backup true.

* Lives : 1
* Backups : 1
* HA Policy : Replicated
* replicating : true

**Error! Filename not specified.**

Master broker

## **Slave broker after kill the Master broker**

The master broker is no longer live, and the slave broker has taken its place. All addresses and clients that were previously connected are now linked to the slave broker (the new live broker).

* Lives : 1
* Backups : 0
* HA Policy : Replica

**Error! Filename not specified.**

Slave broker

**In conclusion, it’s not relevant where your master and slave brokers are located; these configurations are equally effective for both scenarios.**

# My own ActiveMQ Artemis Cheat Sheet

[ActiveMQ Artemis](https://activemq.apache.org/components/artemis/) is a high-performance messaging system for highly scalable microservices or asynchronous messaging between different systems.

This cheat sheet includes the most common commands to install, deploy, administrate or operate a messaging system based in ActiveMQ Artemis.

**NOTE**: ActiveMQ Artemis is the upstream project of [Red Hat AMQ 7 Broker](https://access.redhat.com/products/red-hat-amq#broker-gs), so these commands are also valid for this product.

## Cluster Topologies

ActiveMQ Artemis allows defining different of topologies to build simple or complex messaging solutions. The most standard topologies to cover High Availability, scalable and failover scenarios are:

* Symmetric cluster with Replicated journal
* Symmetric cluster with Shared journal

For other cluster topologies, review [Cluster Topologies](https://activemq.apache.org/components/artemis/documentation/latest/clusters.html#cluster-topologies).

### Symmetric Cluster with Replicated Journal

Topology based in:

* Set of live brokers with network journal replication (Symmetric Cluster and High Availability)
* Set of backup brokers for the live brokers (Failover)

This diagram shows us this topology:

### Symmetric Cluster with Shared Journal

Topology based in:

* Set of live brokers with a shared journal (Symmetric Cluster and High Availability)
* Set of backup brokers for the live brokers (Failover)

This diagram shows us this topology:

## Deploying Cluster Topologies

### Deploying a Symmetric Cluster with Replicated Journal

This command will create a live broker instance in a replicated journal topology:

$ARTEMIS\_HOME/bin/artemis create /opt/brokers/live-replicated-broker-XX \

--http-host $HOSTNAME \

--host $HOSTNAME \

--aio \

--clustered \

--cluster-user $ARTEMIS\_CLUSTER\_USER \

--cluster-password $ARTEMIS\_CLUSTER\_PASSWORD \

--name live-replicated-broker-XX \

--max-hops 1 \

--user $ARTEMIS\_ADMIN\_USER \

--password $ARTEMIS\_ADMIN\_PASSWORD \

--require-login \

--port-offset 0 \

--no-autocreate \

--replicated \

--failover-on-shutdown

This other command will create a backup broker instance for a replicated journal topology:

$ARTEMIS\_HOME/bin/artemis create /opt/brokers/backup-replicated-broker-XX \

--http-host $HOSTNAME \

--host $HOSTNAME \

--aio \

--clustered \

--cluster-user $ARTEMIS\_CLUSTER\_USER \

--cluster-password $ARTEMIS\_CLUSTER\_PASSWORD \

--name live-replicated-broker-XX \

--max-hops 1 \

--user $ARTEMIS\_ADMIN\_USER \

--password $ARTEMIS\_ADMIN\_PASSWORD \

--require-login \

--port-offset 0 \

--no-autocreate \

--replicated \

--failover-on-shutdown \

--slave

By default, ActiveMQ Artemis will use a broadcast network to discover the instances of the topology. The values are identified at runtime of the command and store them in the configuration files.

In case you need to identify the different instances of the topology, the staticCluster property is needed. This property identifies the static list of other brokers of the topology.

--staticCluster $ARTEMIS\_STATIC\_CLUSTER\_LIST

Samples:

* **ARTEMIS\_STATIC\_CLUSTER\_LIST**: tcp://live-broker-01:61616,...,tcp://live-broker-XX:61616

### Deploying a Symmetric Cluster with Shared Journal

This command will create a live broker of a live/backup pair using a shared journal:

$ARTEMIS\_HOME/bin/artemis create /opt/brokers/live-broker \

--http-host $HOSTNAME \

--host $HOSTNAME \

--aio \

--clustered \

--cluster-user $ARTEMIS\_CLUSTER\_USER \

--cluster-password $ARTEMIS\_CLUSTER\_PASSWORD \

--name live-broker \

--max-hops 1 \

--user $ARTEMIS\_ADMIN\_USER \

--password $ARTEMIS\_ADMIN\_PASSWORD \

--require-login \

--port-offset 0 \

--no-autocreate \

--data $ARTEMIS\_SHARED\_STORAGE\_PATH \

--shared-store \

--failover-on-shutdown

For the backup broker instance:

$ARTEMIS\_HOME/bin/artemis create /opt/brokers/backup-broker \

--http-host $HOSTNAME \

--host $HOSTNAME \

--aio \

--clustered \

--cluster-user $ARTEMIS\_CLUSTER\_USER \

--cluster-password $ARTEMIS\_CLUSTER\_PASSWORD \

--name backup-broker \

--max-hops 1 \

--user $ARTEMIS\_ADMIN\_USER \

--password $ARTEMIS\_ADMIN\_PASSWORD \

--require-login \

--port-offset 100 \

--no-autocreate \

--data $ARTEMIS\_SHARED\_STORAGE\_PATH \

--shared-store \

--slave \

--failover-on-shutdown \

Here, the key point is that $ARTEMIS\_SHARED\_STORAGE\_PATH mount a shared storage between both brokers. It is very common to use a NFSv4 mounted folder.

More details in [Persistence](https://activemq.apache.org/components/artemis/documentation/latest/persistence.html).

## HA and Failover life cycle

ActiveMQ Artemis includes a failover feature based in the combination of Live and Backup brokers. This feature allows defining an automatic client failover connection.

A client can receive information about all live and backup brokers, so that in the event of a connection failure, it can reconnect to the backup broker. The backup broker then automatically re-creates any sessions and consumers that existed on each connection before failover. This feature saves you from having to hand-code manual reconnection logic in your applications.

When a session is re-created on the backup, it does not have any knowledge of messages already sent or acknowledged. Any in-flight sends or acknowledgements at the time of failover might also be lost. However, even without 100% transparent failover, it is simple to guarantee once and only once delivery, even in the case of failure, by using a combination of duplicate detection and retrying of transactions.

Client connection strings identify a connection string using the Core protocol with the right values to enable the high availability and failover features from the client side. For example:

* Single broker: tcp://HOSTNAME1:61616?ha=true&reconnectAttempts=10
* Failover connection: (tcp://HOSTNAME1:616161,tcp://HOSTNAME2:61616)?ha=true&reconnectAttempts=10

Every time a live broker shutdown, the backup instance will be promoted as lived instance to continue the service to producers and consumers, and also to distribute the messages to other members of the topology. The backup instance could delegate again the role to the live instance as soon is is ready. The following definition in the backup instance automate that life cycle:

<ha-policy>

<shared-store>

<slave>

<allow-failback>true</allow-failback>

</slave>

</shared-store>

</ha-policy>

More details in [High Availability and Failover](https://activemq.apache.org/components/artemis/documentation/latest/ha.html).

## Message Redistribution

To enable [messaging redistribution](https://activemq.apache.org/components/artemis/documentation/latest/clusters.html#message-redistribution) between brokers in a cluster, the **redistribution-delay** property must be enabled to zero in the <address-setting> in broker.xml file:

<!--default for catch all-->

<address-setting match="#">

<redistribution-delay>0</redistribution-delay>

</address-setting>

## Protocols

ActiveMQ Artemis has a pluggable protocol architecture, so that it can easily enable one or more protocols for a network connection. The broker supports the following protocols:

* Core
* AMQP
* MQTT
* OpenWire
* STOMP
* HornetQ

Each protocol could be defined as secure channel (SSL). These channels could be used to define one-way or two-ways to validate the communications from the clients.

More details in [Protocols and Interoperability](https://activemq.apache.org/components/artemis/documentation/latest/protocols-interoperability.html).

## Starting a broker

Start a broker instance is very simple:

$ARTEMIS\_BROKER\_INSTANCE\_HOME/bin/artemis run

Or you can run the broker in the background using:

$ARTEMIS\_BROKER\_INSTANCE\_HOME/bin/artemis-service start

## Installing as OS service (systemd)

Create a file called artemis.service file in the /etc/systemd/system folder. This file will have the following content:

[Unit]

Description = ActiveMQ Artemis - Broker

After = syslog.target network.target

[Service]

ExecStart = /opt/brokers/live/bin/artemis run

ExecStop = /opt/brokers/live/bin/artemis stop

User = amq-broker

Group = amq-broker

SuccessExitStatus = 0 143

RestartSec = 60

Restart = on-failure

LimitNOFILE = 102642

[Install]

WantedBy = multi-user.target

To Enable the OS services:

systemctl enable artemis

To start, stop, check the status or restart the OS service:

systemctl (start|stop|status|restart) artemis

## Modularizing Broker Configuration

ActiveMQ Artemis supports XML inclusions so the configuration can be broken out into separate files. It is an interesting feature to prevent errors and to ease populating common configuration between clustered brokers.

By default, the etc/broker.xml file declares XInclude XML namespace.

<configuration xmlns="urn:activemq"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xi="http://www.w3.org/2001/XInclude"

...

For example, we could move the addresses tag to an external directory in each host. The live and backup brokers of the same host will share an addresses.xml configuration file.

❯ cat /opt/brokers/config/addresses.xml

<addresses xmlns="urn:activemq:core">

<address name="DLQ">

<anycast>

<queue name="DLQ" />

</anycast>

</address>

<address name="ExpiryQueue">

<anycast>

<queue name="ExpiryQueue" />

</anycast>

</address>

<!-- Queues and Topics -->

<address name="SampleQueue">

<anycast>

<queue name="SampleQueue" />

</anycast>

</address>

<address name="SampleTopic">

<multicast/>

</address>

</addresses>

This file must be copied in any ActiveMQ Artemis cluster hosts. Finally each broker.xml file will include it using the xi:include tag:

<xi:include href="/opt/brokers/config/addresses.xml"/>

**WARNING**: External files are not monitored by the broker so it is needed to refresh or restart it to take the latest status. touch command could be useful to modify the timestamp of the broker.xml configuration file:

touch /opt/brokers/live/etc/broker.xml

## Automatic Configuration

ActiveMQ Artemis is defined to automatically create an address/queue when a new sender/receiver is connected. It is a great feature because it allows us to avoid having to manage the address in the broker.xml file. Also ActiveMQ Artemis also deletes an address/queue when there is not a sender/receiver connected and there are no messages persisted. This feature is also great however it includes some extra staff to manage this process.

The following properties in the address-setting section of broker.xml file manages the automatic configuration:

<address-setting match="#">

<auto-create-addresses>false</auto-create-addresses>

<auto-create-queues>false</auto-create-queues>

<auto-create-jms-queues>false</auto-create-jms-queues>

<auto-create-jms-topics>false</auto-create-jms-topics>

<auto-delete-addresses>false</auto-delete-addresses>

<auto-delete-queues>false</auto-delete-queues>

</address-setting>

More details in [Automatic Configuration](https://activemq.apache.org/components/artemis/documentation/latest/address-model.html#automatic-configuration)

## Producer Command

To send messages to a resource (and easy way to test the broker):

./bin/artemis producer --url '$ARTEMIS\_CORE\_PROTOCOL\_URL' \

--destination $ARTEMIS\_RESOURCE \

--user $ARTEMIS\_ADMIN\_USER --password $ARTEMIS\_ADMIN\_PASSWORD \

--message-size 1024 --message-count 10 --verbose

Where:

* **ARTEMIS\_CORE\_PROTOCOL\_URL**: Identifies a connection string using the Core protocol. Valid values:
  + Single broker: tcp://HOSTNAME1:61616?ha=true&reconnectAttempts=10
  + Failover connection: (tcp://HOSTNAME1:616161,tcp://HOSTNAME2:61616)?ha=true&reconnectAttempts=10
* **ARTEMIS\_RESOURCE**: Identifies the resource. Sample values for:
  + A Queue: queue://SampleQueue
  + A Topic: topic://SampleTopic

## Consumer Command

To consume messages from a resource (and easy way to test the broker):

./bin/artemis consumer --url '$ARTEMIS\_CORE\_PROTOCOL\_URL' \

--destination $ARTEMIS\_RESOURCE \

--user $ARTEMIS\_ADMIN\_USER --password $ARTEMIS\_ADMIN\_PASSWORD \

--message-count 10 --verbose

Where:

* **ARTEMIS\_CORE\_PROTOCOL\_URL**: Identifies a connection string using the Core protocol. Valid values:
  + Single broker: tcp://HOSTNAME1:61616?ha=true&reconnectAttempts=10
  + Failover connection: (tcp://HOSTNAME1:616161,tcp://HOSTNAME2:61616)?ha=true&reconnectAttempts=10
* **ARTEMIS\_RESOURCE**: Identifies the resource. Valid values:
  + Queue: queue://SampleQueue
  + Topic: topic://SampleTopic

## Durable Subscription Queue

The broker saves messages for any inactive subscribers when a queue is configured as a Durable Subscription. The broker delivers them to the subscribers when they reconnect. Clients are therefore guaranteed to receive each message delivered to the queue after subscribing to it.

A sample definition will be similar to:

<address name="topic">

<multicast>

<queue name="subscription1">

<durable>true</durable>

</queue>

<queue name="subscription2">

<durable>true</durable>

</queue>

</multicast>

</address>

This is a sample definition of a Durable Subscription Queue to be used with the Artemis producer and consumer commands:

<address name="topic.events">

<multicast>

<queue name="c1.Consumer ActiveMQTopic[topic.foo], thread=0">

<durable>true</durable>

</queue>

<queue name="c2.Consumer ActiveMQTopic[topic.foo], thread=0">

<durable>true</durable>

</queue>

</multicast>

</address>

This command produces a number of messages to the topic:

./bin/artemis producer --url tcp://$HOSTNAME:5672 \

--user admin --password admin \

--protocol amqp \

--destination topic://topic.events \

--threads 1 \

--message-count 100 \

--text-size 1024

This command consumes the messages from the c1 subscription:

./bin/artemis consumer --url tcp://$HOSTNAME:5672 \

--user admin --password admin \

--protocol amqp \

--destination topic://topic.events \

--threads 1 \

--message-count 100 \

--durable \

--clientID c1

This other command consumes the messages from the c2 subscription:

./bin/artemis consumer --url tcp://$HOSTNAME:5672 \

--user admin --password admin \

--protocol amqp \

--destination topic://topic.events \

--threads 1 \

--message-count 100 \

--durable \

--clientID c2

**NOTE**: These queues could be consumed using JMS Message Consumers using [Fully Qualified Queue Names (FQQN)](https://activemq.apache.org/components/artemis/documentation/latest/address-model.html) as:

String FQQN = "topic::subscription1";

Queue queueDestination session.createQueue(FQQN);

MessageConsumer consumer = session.createConsumer(queueDestination);

More details in [Addressing Model](https://activemq.apache.org/components/artemis/documentation/latest/address-model.html).

## Clustered Message Grouping

This feature allows to process messages with a particular group ID in the same order by the consumers. Each clustered broker therefore uses a grouping handler to manage the complexity of routing of grouped messages. Each clustered broker should choose should choose a grouping handler type: Local or Remote.

Local grouping handler broker (only one per cluster topology):

<grouping-handler name="my-grouping-handler">

<type>LOCAL</type>

<address>SampleQueue</address>

<timeout>5000</timeout>

</grouping-handler>

Remote grouping handler broker (the rest of instances of the cluster topology):

<grouping-handler name="my-grouping-handler">

<type>REMOTE</type>

<address>SampleQueue</address>

<timeout>5000</timeout>

</grouping-handler>

To produce grouped messages:

./bin/artemis producer --url $ARTEMIS\_CORE\_PROTOCOL\_URL \

--destination $ARTEMIS\_RESOURCE \

--user $ARTEMIS\_ADMIN\_USER --password $ARTEMIS\_ADMIN\_PASSWORD \

--message "Sample Grouped Message" \

--message-count 10 \

--group mygroup

To consume grouped messages:

./bin/artemis consumer --url $ARTEMIS\_CORE\_PROTOCOL\_URL \

--destination $ARTEMIS\_RESOURCE \

--user $ARTEMIS\_ADMIN\_USER --password $ARTEMIS\_ADMIN\_PASSWORD \

--message-count 10 \

--verbose

More details in [Clustered Message Grouping](https://activemq.apache.org/components/artemis/documentation/latest/message-grouping.html)

## AMQP Secured Connection

Client connection string for AMQP secure protocol:

amqps://HOSTNAME:5671?sslEnabled=true&transport.trustAll=true&transport.verifyHost=false

More details in [Configuring Transports](https://activemq.apache.org/components/artemis/documentation/latest/configuring-transports.html)

## Monitoring

ActiveMQ Artemis includes [Jolokia](https://jolokia.org/) endpoints to execute administrative tasks or query administrative information.

Samples:

* Query the up time of the broker:

❯ curl -u admin:admin 'http://localhost:8161/console/jolokia/read/org.apache.activemq.artemis:broker=!%22live!%22/Uptime' | jq

{

"request": {

"mbean": "org.apache.activemq.artemis:broker=\"live\"",

"attribute": "Uptime",

"type": "read"

},

"value": "1 minute",

"timestamp": 1653917729,

"status": 200

}

* Query to get the total number of messages added:

❯ curl -u admin:admin 'http://localhost:8161/console/jolokia/read/org.apache.activemq.artemis:broker=!%22live!%22/TotalMessagesAdded' | jq

{

"request": {

"mbean": "org.apache.activemq.artemis:broker=\"live\"",

"attribute": "TotalMessagesAdded",

"type": "read"

},

"value": 700,

"timestamp": 1653917823,

"status": 200

}

## Performance and Limits

ActiveMQ Artemis manages several resources (descriptors, connections, …) and it is needed to define the OS limits to allow it for the user that runs it.

Review the /etc/security/limits.conf file to add the following definition for this user:

amq-broker soft nofile 65001

amq-broker hard nofile 65001

**WARNING**: In RHEL 8 this step is no longer needed since nofile defaults have been increased to 1048576 max open files.

ActiveMQ Artemis includes a general server thread pool used for most asynchronous actions on the server side. This pool is defined by default to use only 30 threads and it is very useful to improve the performance.

Definition at broker.xml file:

<thread-pool-max-size>120</thread-pool-max-size>

There are a few things that can go wrong in a production environment (bugs, IO errors, memory issues, …), so ActiveMQ Artemis includes a protection to shut itself down when bad things happen (as a safeguard). This method includes different policies:

* **LOG** (default): Log messages into artemis.log to inform that something is wrong.
* **HALT** (default at broker creation): Stop the messaging process but not the VM.
* **SHUTDOWN**: Shutdown the VM process.

To check easily if a broker suffered an issue the best practice is to use SHUTDOWN policy. It is very easy to check if the broker is running or not checking the service or the java process in OS.

Definition at broker.xml file:

<critical-analyzer-policy>SHUTDOWN</critical-analyzer-policy>

More details about performance, tunning and extra features, please review the following references:

* [Performance Tuning](https://activemq.apache.org/components/artemis/documentation/latest/perf-tuning.html)
* [Thread management](https://activemq.apache.org/components/artemis/documentation/latest/thread-pooling.html)
* [Critical Analysis of the broker](https://activemq.apache.org/components/artemis/documentation/latest/critical-analysis.html)

## Managing users

Add a user:

./bin/artemis user add --url $ARTEMIS\_CORE\_PROTOCOL\_URL \

--user admin --password admin \

--user-command-user user1 \

--user-command-password user1-password1 \

--role role1

Reset a user (change user password and/or role/s):

./bin/artemis user reset --url $ARTEMIS\_CORE\_PROTOCOL\_URL \

--user admin --password admin \

--user-command-user user1 \

--user-command-password user1-password2 \

--role role2,role3

Remove a user:

./bin/artemis user rm --url $ARTEMIS\_CORE\_PROTOCOL\_URL \

--user admin --password admin \

--user-command-user user1

## Mask passwords

To mask password to be added in broker.xml file:

$ARTEMIS\_HOME/bin/artemis mask $PASSWORD

More details in [Generating a masked password](https://activemq.apache.org/components/artemis/documentation/latest/masking-passwords.html#generating-a-masked-password).

## Getting more Help

ActiveMQ Artemis CLI tool includes an option to get in line help and documentation for each command implemented.

To get detailed information of any kind of command:

./bin/artemis help COMMAND

## ActiveMQ Artemis version

This cheat-sheet was tested and verified with the ActiveMQ Artemis 2.18 version.

**Writing test cases for DB : { mongoDB & PostgresDB }**

**End to End realtime reactive event processing**

1.Combining RxJava operators and Vert.x clients to support advanced processing

2.Using RxJava operators to perform content enrichment and aggregate data processing on top of event streams

3.Extending the Vert.x event bus to web applications to unify backend and frontend communication models

4.Managing state in a stream-processing setting

**Load Balancing**

Setting up HAProxy for the whole set(s) of modules and server as a whole.

HAProxy Architecture

Frontend – anything in front of haproxy

* 1. Timeout client – if the client does not do any activity for a certain period, disconnect them.
  2. Bind – port binding
  3. Access Control list ( ACL ) – which controller path or paths are allowed? Which IP addresses and from which countries are allowed ? which other sub paths and parameters are allowed ?

Backend – anything behind proxy

1. Timeout Connect – how long do you want the haproxy to keep waiting for a successful server connection ? the timeout value should not be too long. You could use eg. 300ms. And then HAproxy should switch to another server.
2. Timeout Server – how long should HAProxy wait for a server process ? if the server is taking too long than expected for an output, then kill it.
3. Balance ( roundrobin, leastconn, source ) – RoundRobin : 1,2,3, 1,2,3, 1,2,3 ,…. We can also decide to forward requests to the least connections. There is a way that you can decide to make some client to server relationship sticky, such that anytime there is a request from a particular client or IP address or from a particular country do connect to a particular server allocated in the HAProxy for such purpose(s).

**HAProxy modes ( TCP and HTTP)**

HAProxy supports layer 4 and layer 7

Layer :

* + 1. - Physical network
    2. - Data layer
    3. Network layer
    4. Transport layer
    5. Session layer
    6. Presentation layer
    7. Application layer

TCP -> layer 4 proxy

HTTP -> layer 7 proxy : has to look at the data, and that requires terminating the TLS and the proxy is looking at your data.

If you do not trust your proxy but you have well certified backend server, then you should only use layer 4, where you can to TLS pass through.



**ACL**

ACL can be applied to layer 4 and I can see the IP address and route traffic based on countries and client location (sources).

**TLS Termination vs TLS passthrough**

**TLS Termination**

1.Frontend can be TLS and the backend can rather be HTTP

2. HAProxy can terminate the TLS and decrypt and send encrypted data or response.

3. Can look at the data, Layer 7 ACL, re-write headers ( which could be an expensive approach), cache but require certificate(s).

**TLS Passthrough**

Backend is TLS :: TCP mode

HAProxy frontend proxy the packets directly to the backend.

No caching, Layer 4 ACL only, but more secure, HAProxy doesn’t need cert.

**How to install HAProxy**

We would be using brew :

$ brew install haproxy

Sample output(s) on terminal after hitting enter to install HAProxy using Brew. :

**➜ ~** brew install haproxy

==> **Auto-updating Homebrew...**

Adjust how often this is run with HOMEBREW\_AUTO\_UPDATE\_SECS or disable with

HOMEBREW\_NO\_AUTO\_UPDATE. Hide these hints with HOMEBREW\_NO\_ENV\_HINTS (see `man brew`).

==> **Auto-updated Homebrew!**

Updated 2 taps (homebrew/core and homebrew/cask).

==> **New Formulae**

aws-c-auth aws-c-io cargo-chef imgp television

aws-c-cal aws-c-mqtt cloudquery meli victorialogs

aws-c-common aws-c-s3 cobo-cli minio-warp

aws-c-compression aws-c-sdkutils cpp-peglib rogcat

aws-c-event-stream aws-checksums ducker sqlitecpp

aws-c-http aws-crt-cpp eventpp tabulate

==> **New Casks**

bananas font-playwrite-de-la-guides font-playwrite-nl-guides

font-agu-display font-playwrite-de-sas-guides font-playwrite-no-guides

font-badeen-display font-playwrite-de-va-guides font-playwrite-nz-guides

font-playwrite-ar-guides font-playwrite-dk-loopet-guides font-playwrite-pe-guides

font-playwrite-at-guides font-playwrite-dk-uloopet-guides font-playwrite-pl-guides

font-playwrite-au-nsw-guides font-playwrite-es-deco-guides font-playwrite-pt-guides

font-playwrite-au-qld-guides font-playwrite-es-guides font-playwrite-ro-guides

font-playwrite-au-sa-guides font-playwrite-fr-moderne-guides font-playwrite-sk-guides

font-playwrite-au-tas-guides font-playwrite-fr-trad-guides font-playwrite-tz-guides

font-playwrite-au-vic-guides font-playwrite-gb-j-guides font-playwrite-us-modern-guides

font-playwrite-be-vlg-guides font-playwrite-hr-guides font-playwrite-us-trad-guides

font-playwrite-be-wal-guides font-playwrite-hr-lijeva-guides font-playwrite-vn-guides

font-playwrite-br-guides font-playwrite-hu-guides font-playwrite-za-guides

font-playwrite-ca-guides font-playwrite-in-guides meridiem

font-playwrite-cl-guides font-playwrite-is-guides satyrn

font-playwrite-co-guides font-playwrite-it-moderna-guides sys-pc-tool

font-playwrite-cu-guides font-playwrite-it-trad-guides windsurf

font-playwrite-cz-guides font-playwrite-mx-guides

font-playwrite-de-grund-guides font-playwrite-ng-modern-guides

You have **44** outdated formulae installed.

Warning: You are using macOS 12.

We (and Apple) do not provide support for this old version.

It is expected behaviour that some formulae will fail to build in this old version.

It is expected behaviour that Homebrew will be buggy and slow.

Do not create any issues about this on Homebrew's GitHub repositories.

Do not create any issues even if you think this message is unrelated.

Any opened issues will be immediately closed without response.

Do not ask for help from Homebrew or its maintainers on social media.

You may ask for help in Homebrew's discussions but are unlikely to receive a response.

Try to figure out the problem yourself and submit a fix as a pull request.

We will review it but may or may not accept it.

==> **Fetching haproxy**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/d774bea8470ad0ab8885ba25fbeda5df2b44ca5**

######################################################################################################### 100.0%

==> **Downloading https://www.haproxy.org/download/3.1/src/haproxy-3.1.0.tar.gz**

######################################################################################################### 100.0%

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **make USE\_PCRE2=1 USE\_PCRE2\_JIT=1 USE\_OPENSSL=1 USE\_ZLIB=1 TARGET=osx**

==> **Caveats**

To start haproxy now and restart at login:

brew services start haproxy

Or, if you don't want/need a background service you can just run:

/opt/homebrew/opt/haproxy/bin/haproxy -f /opt/homebrew/etc/haproxy.cfg

==> **Summary**

🍺 /opt/homebrew/Cellar/haproxy/3.1.0: 10 files, 4.4MB, built in 13 seconds

==> **Running `brew cleanup haproxy`...**

Disable this behaviour by setting HOMEBREW\_NO\_INSTALL\_CLEANUP.

Hide these hints with HOMEBREW\_NO\_ENV\_HINTS (see `man brew`).

**➜ ~**

**HAProxy configuration**

Frontend -> to -> Backend with many servers load balance

Create a simple folder on your system.

Frontend & Backend configuration

$ vim test\_haproxy.cfg

frontend http80

bind 0.0.0.0:80 or bind \*:80

timeout client 10s

default\_backend allservers

backend allservers

timeout connect 5s

timeout server 100s

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

### and then save the file appropriately

wq

How to start the HAProxy

$ haproxy -f test\_haproxy.cfg

With the above configuration, because we are using TCP at layer 4, the moment the browser connects and communicates with the HAProxy, it would forward the client to the one of the server connections. The browser wants to keep the connection as alive as possible. So basically the browser would want to use and keep connected to the already existing connection established.

But by using Telnet in the teminal :

$ telnet 127.0.0.1 80

GET /

Using HTTP mode :

We did not specify the mode on the side of the frontend configuration, therefore the default in used is the TCP.

Meanwhile right under the timeout client under frontend, we can add :

mode http

And it all has to match, so we also have to set the mode for the backend, right under the timeout server, we then add :

mode http

Do well not to mix mode(s) for the frontend and backend. The mode has to be the same for seamless and easy configuration.

The final configuration would now look like :

frontend http80

bind 0.0.0.0:80 or bind \*:80

timeout client 10s

mode http

default\_backend allservers

backend allservers

timeout connect 5s

timeout server 100s

mode http

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

**How setup multiple backend for a particular frontend**

frontend http80

bind 0.0.0.0:80 or bind \*:80

timeout client 10s

mode http

acl payments path\_end -i /mobilepymnt

acl paymenthistory path\_end -i /trnxhistory

use\_backend paymentsServers if payments

use\_backend trnxHistoryServers if paymenthistory

default\_backend allservers

backend paymentsServers

timeout connect 5s

timeout server 4s

mode http

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

backend trnxHistoryServers

timeout connect 5s

timeout server 10s

mode http

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

backend allservers

timeout connect 5s

timeout server 100s

mode http

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

We can make a particular client to connect to a particular server by using the code :

balance source

By adding this to the any of the backend server, we make the client sticky with/to that particular server.

**Implementing ACL**

ACL based on conditional service/app0 and service/app1

ACL based admin access prevention

**How to block a path in haproxy**

http-request deny if { path -i -m beg /admin }

Do note the above line of code or config for haproxy would not necessarily block :

localhost:12435/admin

Meanwhile,

localhost/admin

would definitely be blocked.

The final haproxy would then look like :

frontend http80

bind 0.0.0.0:80 or bind \*:80

timeout client 10s

mode http

acl payments path\_end -i /mobilepymnt

acl paymenthistory path\_end -i /trnxhistory

http-request deny if { path -i -m beg /admin }

use\_backend paymentsServers if payments

use\_backend trnxHistoryServers if paymenthistory

default\_backend allservers

backend paymentsServers

timeout connect 5s

timeout server 4s

mode http

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

backend trnxHistoryServers

timeout connect 5s

timeout server 10s

mode http

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

backend allservers

timeout connect 5s

timeout server 100s

mode http

server server5872 127.0.0.1:5872

server server5873 127.0.0.1:5873

server server5874 127.0.0.1:5874

server server5875 127.0.0.1:5875

server server5876 127.0.0.1:5876

We can use :

<https://noip.com>

to connect to a free domain name over an unsecured domain name in the browser to set things up and test.

**Enabling HTTPS on HAProxy**

How to setup Letsencrypt

$ brew install letsencrypt

Up next :

We need to generate public key & private key

$ sudo certbot certonly –standalone

Then enter password of your system.

Then confirm : the email address

Then confirm : the T&C , and access to use your email

Then enter in your domain name(s) ; comma separated

Setting letsencrypt communicate using port 8080 , so do make sure that port 8080 is not in use anywhere else on the system.

HAProxy expects the two keys to be merged as one. Ie. the public key and the private key generated by letsencrypt.

We would use the command below :

$ sudo cat path\_to\_public\_key\_with\_the\_file\_extension path\_to\_private\_key\_with\_the\_file\_extension | sudo tee /Users/projectname/proxy/haproxy\_tls\_keys.pem

Now lets go the configuration file for haproxy

$ vim test.cfg

And then bind the port 443 to be fully secured , under the frontend configuration area in the ‘ test.cfg ‘.

bind \*:443 ssl crt /Users/projectname/proxy/haproxy\_tls\_keys.pem

The secured layer is using the Diffie Helman algorithm, hence the ‘ -dh- ‘ in the tune.ssl.default-dh-param.

**Enabling HTTP/2 on HAProxy**

Defining Application Layer Protocol Negotiation ( alpn ) in haproxy configuration.

bind \*:443 ssl crt /Users/projectname/proxy/haproxy\_tls\_keys.pem alpn h2,http/1.1

We want to make sure we do not get a downgrade attack or http downgrade attack.

alpn is an extension that can exchange protocols during handshakes not after.

Sample output for installing letsencrypt :

**~** brew install letsencrypt

==> **Downloading https://formulae.brew.sh/api/formula.jws.json**

######################################################################################################### 100.0%

Warning: Formula letsencrypt was renamed to certbot.

==> **Downloading https://formulae.brew.sh/api/cask.jws.json**

######################################################################################################### 100.0%

Warning: You are using macOS 12.

We (and Apple) do not provide support for this old version.

It is expected behaviour that some formulae will fail to build in this old version.

It is expected behaviour that Homebrew will be buggy and slow.

Do not create any issues about this on Homebrew's GitHub repositories.

Do not create any issues even if you think this message is unrelated.

Any opened issues will be immediately closed without response.

Do not ask for help from Homebrew or its maintainers on social media.

You may ask for help in Homebrew's discussions but are unlikely to receive a response.

Try to figure out the problem yourself and submit a fix as a pull request.

We will review it but may or may not accept it.

==> **Fetching dependencies for certbot: augeas, python@3.11, python@3.12, certifi, pycparser, cffi, libssh2, cmake, libgit2, z3, swig, llvm, rust, maturin, cryptography and dialog**

==> **Fetching augeas**

==> **Downloading https://ghcr.io/v2/homebrew/core/augeas/manifests/1.14.1-1**

######################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/augeas/blobs/sha256:6ce2ccf218f4ac51eae364b50a74eae014820ddb0e2**

######################################################################################################### 100.0%

==> **Fetching python@3.11**

==> **Downloading https://ghcr.io/v2/homebrew/core/python/3.11/manifests/3.11.10**

######################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/python/3.11/blobs/sha256:83ff46a5ab6f3008bef20946081b47b83b4c94**

######################################################################################################### 100.0%

==> **Fetching python@3.12**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://raw.githubusercontent.com/Homebrew/formula-patches/6d2fba8de3159182025237d373a6f4f78b8bd**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/c4/e6/c1ac50fe3eebb38a155155711e6e864e254ce4b6e17fe2429b**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/240a7924cd4b28a95eb7967148f1a55a9fa38b65f3722a6e5e16193319616b0f--flit\_core-3.9.0.tar.gz

==> **Downloading https://files.pythonhosted.org/packages/4d/87/fb90046e096a03aeab235e139436b3fe804cdd447ed2093b0d**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/1d254ab6f3faf1adbba5e7a1c80fdc683cc524ca2953fe7dbb04e7bbbe5ebc22--pip-24.2.tar.gz

==> **Downloading https://files.pythonhosted.org/packages/27/b8/f21073fde99492b33ca357876430822e4800cdf522011f1804**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/5c0af5acc79f514d9e861f222b6ce549dc1b3f5ca243f04d0ad9e0a66a4288fa--setuptools-75.1.0.tar.gz

==> **Downloading https://files.pythonhosted.org/packages/b7/a0/95e9e962c5fd9da11c1e28aa4c0d8210ab277b1ada951d2aee**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/ce6706d401907ae2b7fffebfd68689fabab5db9f2c4c68c163fa97804627e61d--wheel-0.44.0.tar.gz

==> **Downloading https://www.python.org/ftp/python/3.12.7/Python-3.12.7.tgz**

######################################################################################################### 100.0%

==> **Fetching certifi**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/b0/ee/9b19140fe824b367c04c5e1b369942dd754c4c5462d5674002**

######################################################################################################### 100.0%

==> **Fetching pycparser**

==> **Downloading https://ghcr.io/v2/homebrew/core/pycparser/manifests/2.22\_1**

######################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/pycparser/blobs/sha256:96eddd22a812be4f919562d6525ab8fa19b03b70**

######################################################################################################### 100.0%

==> **Fetching cffi**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/fc/97/c783634659c2920c3fc70419e3af40972dbaf758daa229a7d6**

######################################################################################################### 100.0%

==> **Fetching libssh2**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://libssh2.org/download/libssh2-1.11.1.tar.gz**

######################################################################################################### 100.0%

==> **Fetching cmake**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/Kitware/CMake/releases/download/v3.31.1/cmake-3.31.1.tar.gz**

==> **Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/537699/ed388d6**

######################################################################################################### 100.0%

==> **Fetching libgit2**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/libgit2/libgit2/archive/refs/tags/v1.8.4.tar.gz**

==> **Downloading from https://codeload.github.com/libgit2/libgit2/tar.gz/refs/tags/v1.8.4**

######################################################################################################### 100.0%

==> **Fetching z3**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/Z3Prover/z3/archive/refs/tags/z3-4.13.3.tar.gz**

==> **Downloading from https://codeload.github.com/Z3Prover/z3/tar.gz/refs/tags/z3-4.13.3**

######################################################################################################### 100.0%

==> **Fetching swig**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://downloads.sourceforge.net/project/swig/swig/swig-4.3.0/swig-4.3.0.tar.gz**

==> **Downloading from https://liquidtelecom.dl.sourceforge.net/project/swig/swig/swig-4.3.0/swig-4.3.0.tar.gz?via**

######################################################################################################### 100.0%

==> **Fetching llvm**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/llvm/llvm-project/commit/1682c99a8877364f1d847395cef501e813804caa.patch?full\_**

######################################################################################################### 100.0%

==> **Downloading https://github.com/llvm/llvm-project/commit/88dd0d33147a7f46a3c9df4aed28ad4e47ef597c.patch?full\_**

######################################################################################################### 100.0%

==> **Downloading https://github.com/llvm/llvm-project/commit/a3e8b860788934d7cc1489f850f00dcfd9d8b595.patch?full\_**

######################################################################################################### 100.0%

==> **Downloading https://github.com/llvm/llvm-project/releases/download/llvmorg-19.1.4/llvm-project-19.1.4.src.ta**

==> **Downloading from https://objects.githubusercontent.com/github-production-release-asset-2e65be/75821432/721d4**

######################################################################################################### 100.0%

==> **Fetching rust**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/rust-lang/cargo/archive/refs/tags/0.84.0.tar.gz**

==> **Downloading from https://codeload.github.com/rust-lang/cargo/tar.gz/refs/tags/0.84.0**

######################################################################################################### 100.0%

==> **Downloading https://static.rust-lang.org/dist/2024-10-17/rustc-1.82.0-aarch64-apple-darwin.tar.xz**

######################################################################################################### 100.0%

==> **Downloading https://static.rust-lang.org/dist/2024-10-17/cargo-1.82.0-aarch64-apple-darwin.tar.xz**

######################################################################################################### 100.0%

==> **Downloading https://static.rust-lang.org/dist/2024-10-17/rust-std-1.82.0-aarch64-apple-darwin.tar.xz**

######################################################################################################### 100.0%

==> **Downloading https://static.rust-lang.org/dist/rustc-1.83.0-src.tar.gz**

######################################################################################################### 100.0%

==> **Fetching maturin**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://github.com/PyO3/maturin/archive/refs/tags/v1.7.6.tar.gz**

==> **Downloading from https://codeload.github.com/PyO3/maturin/tar.gz/refs/tags/v1.7.6**

-=O=- # # # #

==> **Fetching cryptography**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/91/4c/45dfa6829acffa344e3967d6006ee4ae8be57af746ae2eba1c**

######################################################################################################### 100.0%

==> **Fetching dialog**

==> **Downloading https://ghcr.io/v2/homebrew/core/dialog/manifests/1.3-20240619**

######################################################################################################### 100.0%

==> **Downloading https://ghcr.io/v2/homebrew/core/dialog/blobs/sha256:0a639f7053e72045fd49d4ea0d48d89d56d82261c88**

######################################################################################################### 100.0%

==> **Fetching certbot**

==> **Downloading https://raw.githubusercontent.com/Homebrew/homebrew-core/37b08a4fbfa09c1ac8bef7ba2b704662bd9f6ed**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/2a/34/d6d7064afabe02f0043e5015aab664b67fc8bf049e81ac9220**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/bc/a0/e156803d229c63c317442d1a23f6218201971fa2f41261e5fa**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/2f/fe/dd474bb102881362703de7cb3b00c5fb147f8de1eb72a52542**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/f2/4f/e1808dc01273379acc506d18f1504eb2d299bd4131743b9fc5**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/70/8a/73f1008adfad01cb923255b924b1528727b8270e67cb4ef41e**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/f5/c4/c7f9e41bc2e5f8eeae4a08a01c91b2aea3dfab40a3e14b25e8**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/fc/f8/98eea607f65de6527f8a2e8885fc8015d3e6f5775df186e443**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/f1/70/7703c29685631f5a7590aa73f1f1d3fa9a380e654b86af429e**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/2c/cd/684c45107851da4507854ef4b16fcdce448e02668f0e7c359d**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/a8/20/cb587f6672dbe585d101f590c3871d16e7aec5a576a1694997**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/5d/70/ff56a63248562e77c0c8ee4aefc3224258f1856977e0c14726**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/8c/d5/e5aeee5387091148a19e1145f63606619cb5f20b83fccb63ef**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/f0/d2/6587e8ec3951cbd97c56333d11e0f8a3a4cb64c0d6ed101882**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/af/cc/5064a3c25721cd863e6982b87f10fdd91d8bcc62b6f7f36f52**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/3a/31/3c70bf7603cc2dca0f19bdc53b4537a797747a58875b552c8c**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/63/70/2bf7780ad2d390a8d301ad0b550f1581eadbd9a20f896afe06**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/c8/db/722a42ffdc226e950c4757b3da7b56ff5c090bb265dccd707f**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/ed/63/22ba4ebfe7430b76388e7cd448d5478814d3032121827c12a2**

######################################################################################################### 100.0%

==> **Downloading https://files.pythonhosted.org/packages/a7/09/cb6243a1bf761916a24d0cf58b7719f68d8975ba4491916d5a**

######################################################################################################### 100.0%

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **Installing dependencies for certbot: augeas, python@3.11, python@3.12, certifi, pycparser, cffi, libssh2, cmake, libgit2, z3, swig, llvm, rust, maturin, cryptography and dialog**

==> **Installing certbot dependency: augeas**

==> **Downloading https://ghcr.io/v2/homebrew/core/augeas/manifests/1.14.1-1**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/338b8e0bd10e0c97d2f41e197eb4f04ceb79e8449df0331ebe7e8a4925ce4111--augeas-1.14.1-1.bottle\_manifest.json

==> **Pouring augeas--1.14.1.arm64\_monterey.bottle.1.tar.gz**

🍺 /opt/homebrew/Cellar/augeas/1.14.1: 493 files, 3.9MB

==> **Installing certbot dependency: python@3.11**

==> **Downloading https://ghcr.io/v2/homebrew/core/python/3.11/manifests/3.11.10**

Already downloaded: /Users/<user>/Library/Caches/Homebrew/downloads/dbceba8ebdfd7d47453a640759660b014f161373922ee800cf0591569c2ef58c--python@3.11-3.11.10.bottle\_manifest.json

==> **Pouring python@3.11--3.11.10.arm64\_monterey.bottle.tar.gz**

==> **Downloading https://formulae.brew.sh/api/formula.jws.json**

######################################################################################################### 100.0%

==> **/opt/homebrew/Cellar/python@3.11/3.11.10/bin/python3.11 -Im ensurepip**

==> **/opt/homebrew/Cellar/python@3.11/3.11.10/bin/python3.11 -Im pip install -v --no-index --upgrade --isolated -**

🍺 /opt/homebrew/Cellar/python@3.11/3.11.10: 3,306 files, 62.8MB

==> **Installing certbot dependency: python@3.12**

Warning: Your Xcode (13.1) is outdated.

Please update to Xcode 14.2 (or delete it).

Xcode can be updated from the App Store.

Warning: A newer Command Line Tools release is available.

Update them from Software Update in System Preferences.

If that doesn't show you any updates, run:

sudo rm -rf /Library/Developer/CommandLineTools

sudo xcode-select --install

Alternatively, manually download them from:

https://developer.apple.com/download/all/.

You should download the Command Line Tools for Xcode 14.2.

==> **Patching**

==> **Applying 3.11-sysconfig.diff**

==> **./configure --enable-ipv6 --datarootdir=/opt/homebrew/Cellar/python@3.12/3.12.7\_1/share --datadir=/opt/homeb**

==> **make**

**Entities using HAProxy**

Many large enterprises and high-profile websites, such as Airbnb, Reddit, and Stack Overflow, rely on HAProxy for their load balancing needs.

## Performance Benchmarks

One of the most important factors in choosing a load balancer is performance. Both HAProxy and Nginx are known for their high performance and ability to handle a large number of concurrent connections, but how do they compare head-to-head?

In a 2022 benchmark study conducted by the Cloud Native Computing Foundation (CNCF), HAProxy and Nginx were tested under a variety of workloads and configurations. The results showed that HAProxy consistently outperformed Nginx in terms of throughput and connection handling.

| **Proxy** | **Requests/sec** | **Connections/sec** |
| --- | --- | --- |
| HAProxy | 1,200,000 | 100,000 |
| Nginx | 1,000,000 | 80,000 |

As shown in [this test run on AWS ARM-based Graviton2](https://www.haproxy.com/blog/haproxy-forwards-over-2-million-http-requests-per-second-on-a-single-aws-arm-instance/), HAProxy scales very well with threads and was shown to be able to reach 2 million requests/s over SSL and 100 Gbps for forwarded traffic.

### Load Balancing Algorithms

HAProxy offers a wide range of load balancing algorithms, making it well-suited for a variety of workloads and requirements. Some of the notable algorithms include:

* Round Robin: Distributes requests sequentially across a set of backend servers.
* Least Connections: Sends requests to the server with the fewest active connections.
* Source Hash: Pins each client (based on IP address) to a specific backend server.
* URL Hash: Pins each URL to a specific backend server for caching purposes.
* Consistent Hash: Maps requests to servers based on a hash of the request, ensuring requests for the same resource are sent to the same server.

**A lot more readings on HAProxy**

<https://www.haproxy.org/>

<https://docs.haproxy.org/3.0/configuration.html>

<https://docs.haproxy.org/3.0/intro.html>

<https://github.com/haproxy/haproxy>

**Other Examples of haproxy cfg scripts**

global

log 127.0.0.1 local0

log 127.0.0.1 local1 notice

defaults

log global

mode http

option httplog

option dontlognull

timeout connect 5000ms

timeout client 50000ms

timeout server 50000ms

listen stats

bind 0.0.0.0:70

stats enable

stats uri /

frontend balancer

bind 0.0.0.0:80

mode http

default\_backend web\_backends

backend web\_backends

mode http

option forwardfor

balance roundrobin

server weba weba:80 check

server webb webb:80 check

server webc webc:80 check

option httpchk GET /

http-check expect status 200

Varnish git repos : v6.0 :

<https://github.com/varnishcache/varnish-cache/tree/6.0>

**Realtime Reactive Web Applications**

**Vertx weds VueJS**

– Serving vue.js resources

– Connecting to Kafka and forwarding updates to the Vert.x event bus

– Bridging the connected web browsers and the Vert.x event bus

**Load testing, Chaos testing, Testing, Testing, Testing, Benchmarking**

**Running the whole server setup without any dependency on any container**

The sequence has always been to do the following :

1 – Source for environmental variables

2 - Start zookeeper

3 - Start kafka

4 - Start kafdrop

**Sample output , If zookeeper is started appropriately :**

t/homebrew/etc/kafka/zookeeper.properties

[2024-12-06 09:07:30,067] INFO Reading configuration from: /opt/homebrew/etc/kafka/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,076] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,076] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,076] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,076] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,077] INFO autopurge.snapRetainCount set to 3 (org.apache.zookeeper.server.DatadirCleanupManager)

[2024-12-06 09:07:30,077] INFO autopurge.purgeInterval set to 0 (org.apache.zookeeper.server.DatadirCleanupManager)

[2024-12-06 09:07:30,077] INFO Purge task is not scheduled. (org.apache.zookeeper.server.DatadirCleanupManager)

[2024-12-06 09:07:30,077] WARN Either no config or no quorum defined in config, running in standalone mode (org.apache.zookeeper.server.quorum.QuorumPeerMain)

[2024-12-06 09:07:30,078] INFO Log4j 1.2 jmx support not found; jmx disabled. (org.apache.zookeeper.jmx.ManagedUtil)

[2024-12-06 09:07:30,078] INFO Reading configuration from: /opt/homebrew/etc/kafka/zookeeper.properties (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,078] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,078] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,078] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,078] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.apache.zookeeper.server.quorum.QuorumPeerConfig)

[2024-12-06 09:07:30,078] INFO Starting server (org.apache.zookeeper.server.ZooKeeperServerMain)

[2024-12-06 09:07:30,085] INFO ServerMetrics initialized with provider org.apache.zookeeper.metrics.impl.DefaultMetricsProvider@41fecb8b (org.apache.zookeeper.server.ServerMetrics)

[2024-12-06 09:07:30,087] INFO ACL digest algorithm is: SHA1 (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)

[2024-12-06 09:07:30,087] INFO zookeeper.DigestAuthenticationProvider.enabled = true (org.apache.zookeeper.server.auth.DigestAuthenticationProvider)

[2024-12-06 09:07:30,088] INFO zookeeper.snapshot.trust.empty : false (org.apache.zookeeper.server.persistence.FileTxnSnapLog)

[2024-12-06 09:07:30,099] INFO (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO \_\_\_\_\_\_ \_ (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO |\_\_\_ / | | (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO / / \_\_\_ \_\_\_ | | \_\_ \_\_\_ \_\_\_ \_ \_\_ \_\_\_ \_ \_\_ (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO / / / \_ \ / \_ \ | |/ / / \_ \ / \_ \ | '\_ \ / \_ \ | '\_\_| (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO / /\_\_ | (\_) | | (\_) | | < | \_\_/ | \_\_/ | |\_) | | \_\_/ | | (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO /\_\_\_\_\_| \\_\_\_/ \\_\_\_/ |\_|\\_\ \\_\_\_| \\_\_\_| | .\_\_/ \\_\_\_| |\_| (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO | | (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO |\_| (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,099] INFO (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:host.name=192.168.244.228 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:java.version=11.0.24 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:java.vendor=Amazon.com Inc. (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:java.home=/Users/<user>/.sdkman/candidates/java/11.0.24-amzn (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,101] INFO Server environment:java.class.path=/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/activation-1.1.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/aopalliance-repackaged-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/argparse4j-0.7.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/audience-annotations-0.12.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/caffeine-2.9.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-beanutils-1.9.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-cli-1.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-collections-3.2.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-digester-2.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-io-2.14.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-lang3-3.12.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-logging-1.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-validator-1.7.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-basic-auth-extension-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-json-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-mirror-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-mirror-client-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-runtime-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-transforms-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/error\_prone\_annotations-2.10.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-api-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-locator-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-utils-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-annotations-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-core-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-databind-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-dataformat-csv-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-datatype-jdk8-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-jaxrs-base-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-jaxrs-json-provider-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-afterburner-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-jaxb-annotations-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-scala\_2.13-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.activation-api-1.2.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.annotation-api-1.3.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.inject-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.validation-api-2.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.ws.rs-api-2.1.6.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.xml.bind-api-2.3.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javassist-3.29.2-GA.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.activation-api-1.2.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.annotation-api-1.3.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.servlet-api-3.1.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.ws.rs-api-2.1.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jaxb-api-2.3.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-client-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-common-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-container-servlet-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-container-servlet-core-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-hk2-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-server-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-client-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-continuation-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-http-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-io-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-security-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-server-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-servlet-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-servlets-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-util-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-util-ajax-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jline-3.25.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jopt-simple-5.0.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jose4j-0.9.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jsr305-3.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-clients-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-group-coordinator-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-group-coordinator-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-metadata-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-raft-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-server-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-server-common-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-shell-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-storage-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-storage-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-examples-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-scala\_2.13-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-test-utils-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-tools-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-tools-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-transaction-coordinator-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka\_2.13-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/lz4-java-1.8.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/maven-artifact-3.9.6.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/metrics-core-2.2.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/metrics-core-4.1.12.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-buffer-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-codec-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-common-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-handler-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-resolver-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-classes-epoll-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-native-epoll-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-native-unix-common-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/opentelemetry-proto-1.0.0-alpha.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/osgi-resource-locator-1.0.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/paranamer-2.8.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/pcollections-4.0.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/plexus-utils-3.5.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/protobuf-java-3.25.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/reflections-0.10.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/reload4j-1.2.25.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/rocksdbjni-7.9.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-collection-compat\_2.13-2.10.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-java8-compat\_2.13-1.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-library-2.13.14.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-logging\_2.13-3.9.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-reflect-2.13.14.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/slf4j-api-1.7.36.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/slf4j-reload4j-1.7.36.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/snappy-java-1.1.10.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/swagger-annotations-2.2.8.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/trogdor-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zookeeper-3.8.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zookeeper-jute-3.8.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zstd-jni-1.5.6-4.jar (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,126] INFO Server environment:java.library.path=/Users/<user>/Library/Java/Extensions:/Library/Java/Extensions:/Network/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/java:. (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:java.io.tmpdir=/var/folders/dn/zhprrwy90m7dn0kk6gqm82mr0000gp/T/ (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:os.name=Mac OS X (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:os.arch=aarch64 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:os.version=12.5.1 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:user.name=<user> (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,128] INFO Server environment:user.home=/Users/<user> (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO Server environment:user.dir=/Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafka/kafka-3.0-li/bin (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO Server environment:os.memory.free=503MB (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO Server environment:os.memory.max=512MB (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO Server environment:os.memory.total=512MB (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO zookeeper.enableEagerACLCheck = false (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO zookeeper.digest.enabled = true (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO zookeeper.closeSessionTxn.enabled = true (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,145] INFO zookeeper.flushDelay = 0 ms (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,146] INFO zookeeper.maxWriteQueuePollTime = 0 ms (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,146] INFO zookeeper.maxBatchSize=1000 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,146] INFO zookeeper.intBufferStartingSizeBytes = 1024 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,147] INFO Weighed connection throttling is disabled (org.apache.zookeeper.server.BlueThrottle)

[2024-12-06 09:07:30,147] INFO minSessionTimeout set to 6000 ms (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,147] INFO maxSessionTimeout set to 60000 ms (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,148] INFO getData response cache size is initialized with value 400. (org.apache.zookeeper.server.ResponseCache)

[2024-12-06 09:07:30,148] INFO getChildren response cache size is initialized with value 400. (org.apache.zookeeper.server.ResponseCache)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.slotCapacity = 60 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.slotDuration = 15 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.maxDepth = 6 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.initialDelay = 5 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.delay = 5 (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,149] INFO zookeeper.pathStats.enabled = false (org.apache.zookeeper.server.util.RequestPathMetricsCollector)

[2024-12-06 09:07:30,150] INFO The max bytes for all large requests are set to 104857600 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,151] INFO The large request threshold is set to -1 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,151] INFO zookeeper.enforce.auth.enabled = false (org.apache.zookeeper.server.AuthenticationHelper)

[2024-12-06 09:07:30,151] INFO zookeeper.enforce.auth.schemes = [] (org.apache.zookeeper.server.AuthenticationHelper)

[2024-12-06 09:07:30,151] INFO Created server with tickTime 3000 ms minSessionTimeout 6000 ms maxSessionTimeout 60000 ms clientPortListenBacklog -1 datadir /opt/homebrew/var/lib/zookeeper/version-2 snapdir /opt/homebrew/var/lib/zookeeper/version-2 (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,156] INFO Using org.apache.zookeeper.server.NIOServerCnxnFactory as server connection factory (org.apache.zookeeper.server.ServerCnxnFactory)

[2024-12-06 09:07:30,157] WARN maxCnxns is not configured, using default value 0. (org.apache.zookeeper.server.ServerCnxnFactory)

[2024-12-06 09:07:30,157] INFO Configuring NIO connection handler with 10s sessionless connection timeout, 2 selector thread(s), 16 worker threads, and 64 kB direct buffers. (org.apache.zookeeper.server.NIOServerCnxnFactory)

[2024-12-06 09:07:30,164] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)

[2024-12-06 09:07:30,173] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)

[2024-12-06 09:07:30,173] INFO Using org.apache.zookeeper.server.watch.WatchManager as watch manager (org.apache.zookeeper.server.watch.WatchManagerFactory)

[2024-12-06 09:07:30,174] INFO zookeeper.snapshotSizeFactor = 0.33 (org.apache.zookeeper.server.ZKDatabase)

[2024-12-06 09:07:30,174] INFO zookeeper.commitLogCount=500 (org.apache.zookeeper.server.ZKDatabase)

[2024-12-06 09:07:30,176] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)

[2024-12-06 09:07:30,177] INFO Reading snapshot /opt/homebrew/var/lib/zookeeper/version-2/snapshot.4d (org.apache.zookeeper.server.persistence.FileSnap)

[2024-12-06 09:07:30,179] INFO The digest value is empty in snapshot (org.apache.zookeeper.server.DataTree)

[2024-12-06 09:07:30,197] INFO 138 txns loaded in 13 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)

[2024-12-06 09:07:30,197] INFO Snapshot loaded in 23 ms, highest zxid is 0xd7, digest is 308996950402 (org.apache.zookeeper.server.ZKDatabase)

[2024-12-06 09:07:30,198] INFO Snapshotting: 0xd7 to /opt/homebrew/var/lib/zookeeper/version-2/snapshot.d7 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)

[2024-12-06 09:07:30,199] INFO Snapshot taken in 1 ms (org.apache.zookeeper.server.ZooKeeperServer)

[2024-12-06 09:07:30,205] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)

[2024-12-06 09:07:30,206] INFO zookeeper.request\_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.server.RequestThrottler)

[2024-12-06 09:07:30,214] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)

[2024-12-06 09:07:30,214] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)

[2024-12-06 09:07:31,141] INFO Creating new log file: log.d8 (org.apache.zookeeper.server.persistence.FileTxnLog)

[2024-12-06 09:07:50,119] INFO Expiring session 0x1000097de9b0000, timeout of 18000ms exceeded (org.apache.zookeeper.server.ZooKeeperServer)

**If kafka is started properly, then kafdrop also started, the sample output looks like :**

bin kafka-server-start /opt/homebrew/etc/kafka/server.properties

[2024-12-06 09:08:15,440] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$)

[2024-12-06 09:08:15,584] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS renegotiation (org.apache.zookeeper.common.X509Util)

[2024-12-06 09:08:15,643] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalHandler)

[2024-12-06 09:08:15,644] INFO starting (kafka.server.KafkaServer)

[2024-12-06 09:08:15,644] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)

[2024-12-06 09:08:15,665] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeeper.ZooKeeperClient)

[2024-12-06 09:08:15,668] INFO Client environment:zookeeper.version=3.8.4-9316c2a7a97e1666d8f4593f34dd6fc36ecc436c, built on 2024-02-12 22:16 UTC (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,668] INFO Client environment:host.name=192.168.244.228 (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,668] INFO Client environment:java.version=11.0.24 (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,668] INFO Client environment:java.vendor=Amazon.com Inc. (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,668] INFO Client environment:java.home=/Users/<user>/.sdkman/candidates/java/11.0.24-amzn (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,668] INFO Client environment:java.class.path=/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/activation-1.1.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/aopalliance-repackaged-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/argparse4j-0.7.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/audience-annotations-0.12.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/caffeine-2.9.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-beanutils-1.9.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-cli-1.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-collections-3.2.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-digester-2.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-io-2.14.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-lang3-3.12.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-logging-1.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/commons-validator-1.7.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-basic-auth-extension-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-json-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-mirror-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-mirror-client-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-runtime-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/connect-transforms-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/error\_prone\_annotations-2.10.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-api-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-locator-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/hk2-utils-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-annotations-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-core-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-databind-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-dataformat-csv-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-datatype-jdk8-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-jaxrs-base-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-jaxrs-json-provider-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-afterburner-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-jaxb-annotations-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jackson-module-scala\_2.13-2.16.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.activation-api-1.2.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.annotation-api-1.3.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.inject-2.6.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.validation-api-2.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.ws.rs-api-2.1.6.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jakarta.xml.bind-api-2.3.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javassist-3.29.2-GA.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.activation-api-1.2.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.annotation-api-1.3.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.servlet-api-3.1.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/javax.ws.rs-api-2.1.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jaxb-api-2.3.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-client-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-common-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-container-servlet-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-container-servlet-core-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-hk2-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jersey-server-2.39.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-client-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-continuation-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-http-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-io-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-security-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-server-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-servlet-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-servlets-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-util-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jetty-util-ajax-9.4.56.v20240826.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jline-3.25.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jopt-simple-5.0.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jose4j-0.9.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/jsr305-3.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-clients-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-group-coordinator-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-group-coordinator-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-metadata-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-raft-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-server-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-server-common-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-shell-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-storage-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-storage-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-examples-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-scala\_2.13-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-streams-test-utils-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-tools-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-tools-api-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka-transaction-coordinator-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/kafka\_2.13-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/lz4-java-1.8.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/maven-artifact-3.9.6.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/metrics-core-2.2.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/metrics-core-4.1.12.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-buffer-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-codec-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-common-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-handler-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-resolver-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-classes-epoll-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-native-epoll-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/netty-transport-native-unix-common-4.1.111.Final.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/opentelemetry-proto-1.0.0-alpha.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/osgi-resource-locator-1.0.3.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/paranamer-2.8.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/pcollections-4.0.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/plexus-utils-3.5.1.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/protobuf-java-3.25.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/reflections-0.10.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/reload4j-1.2.25.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/rocksdbjni-7.9.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-collection-compat\_2.13-2.10.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-java8-compat\_2.13-1.0.2.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-library-2.13.14.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-logging\_2.13-3.9.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/scala-reflect-2.13.14.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/slf4j-api-1.7.36.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/slf4j-reload4j-1.7.36.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/snappy-java-1.1.10.5.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/swagger-annotations-2.2.8.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/trogdor-3.9.0.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zookeeper-3.8.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zookeeper-jute-3.8.4.jar:/opt/homebrew/Cellar/kafka/3.9.0/libexec/bin/../libs/zstd-jni-1.5.6-4.jar (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,692] INFO Client environment:java.library.path=/Users/<user>/Library/Java/Extensions:/Library/Java/Extensions:/Network/Library/Java/Extensions:/System/Library/Java/Extensions:/usr/lib/java:. (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:java.io.tmpdir=/var/folders/dn/zhprrwy90m7dn0kk6gqm82mr0000gp/T/ (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:java.compiler=<NA> (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:os.name=Mac OS X (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:os.arch=aarch64 (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:os.version=12.5.1 (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:user.name=<user> (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:user.home=/Users/<user> (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,694] INFO Client environment:user.dir=/Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafka/kafka-3.0-li/bin (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,695] INFO Client environment:os.memory.free=1010MB (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,695] INFO Client environment:os.memory.max=1024MB (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,695] INFO Client environment:os.memory.total=1024MB (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,697] INFO Initiating client connection, connectString=localhost:2181 sessionTimeout=18000 watcher=kafka.zookeeper.ZooKeeperClient$ZooKeeperClientWatcher$@5ef6ae06 (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:15,701] INFO jute.maxbuffer value is 4194304 Bytes (org.apache.zookeeper.ClientCnxnSocket)

[2024-12-06 09:08:15,705] INFO zookeeper.request.timeout value is 0. feature enabled=false (org.apache.zookeeper.ClientCnxn)

[2024-12-06 09:08:15,707] INFO [ZooKeeperClient Kafka server] Waiting until connected. (kafka.zookeeper.ZooKeeperClient)

[2024-12-06 09:08:15,708] INFO Opening socket connection to server localhost/0:0:0:0:0:0:0:1:2181. (org.apache.zookeeper.ClientCnxn)

[2024-12-06 09:08:15,713] INFO Socket connection established, initiating session, client: /0:0:0:0:0:0:0:1:54374, server: localhost/0:0:0:0:0:0:0:1:2181 (org.apache.zookeeper.ClientCnxn)

[2024-12-06 09:08:15,736] INFO Session establishment complete on server localhost/0:0:0:0:0:0:0:1:2181, session id = 0x1000067d65d0003, negotiated timeout = 18000 (org.apache.zookeeper.ClientCnxn)

[2024-12-06 09:08:15,738] INFO [ZooKeeperClient Kafka server] Connected. (kafka.zookeeper.ZooKeeperClient)

[2024-12-06 09:08:16,125] INFO Cluster ID = YjR8\_Uo8SNmQ8jLNRlPCrQ (kafka.server.KafkaServer)

[2024-12-06 09:08:16,151] INFO KafkaConfig values:

advertised.listeners = null

alter.config.policy.class.name = null

alter.log.dirs.replication.quota.window.num = 11

alter.log.dirs.replication.quota.window.size.seconds = 1

authorizer.class.name =

auto.create.topics.enable = true

auto.include.jmx.reporter = true

auto.leader.rebalance.enable = true

background.threads = 10

broker.heartbeat.interval.ms = 2000

broker.id = 0

broker.id.generation.enable = true

broker.rack = null

broker.session.timeout.ms = 9000

client.quota.callback.class = null

compression.gzip.level = -1

compression.lz4.level = 9

compression.type = producer

compression.zstd.level = 3

connection.failed.authentication.delay.ms = 100

connections.max.idle.ms = 600000

connections.max.reauth.ms = 0

control.plane.listener.name = null

controlled.shutdown.enable = true

controlled.shutdown.max.retries = 3

controlled.shutdown.retry.backoff.ms = 5000

controller.listener.names = null

controller.quorum.append.linger.ms = 25

controller.quorum.bootstrap.servers = []

controller.quorum.election.backoff.max.ms = 1000

controller.quorum.election.timeout.ms = 1000

controller.quorum.fetch.timeout.ms = 2000

controller.quorum.request.timeout.ms = 2000

controller.quorum.retry.backoff.ms = 20

controller.quorum.voters = []

controller.quota.window.num = 11

controller.quota.window.size.seconds = 1

controller.socket.timeout.ms = 30000

create.topic.policy.class.name = null

default.replication.factor = 1

delegation.token.expiry.check.interval.ms = 3600000

delegation.token.expiry.time.ms = 86400000

delegation.token.master.key = null

delegation.token.max.lifetime.ms = 604800000

delegation.token.secret.key = null

delete.records.purgatory.purge.interval.requests = 1

delete.topic.enable = true

early.start.listeners = null

eligible.leader.replicas.enable = false

fetch.max.bytes = 57671680

fetch.purgatory.purge.interval.requests = 1000

group.consumer.assignors = [org.apache.kafka.coordinator.group.assignor.UniformAssignor, org.apache.kafka.coordinator.group.assignor.RangeAssignor]

group.consumer.heartbeat.interval.ms = 5000

group.consumer.max.heartbeat.interval.ms = 15000

group.consumer.max.session.timeout.ms = 60000

group.consumer.max.size = 2147483647

group.consumer.migration.policy = disabled

group.consumer.min.heartbeat.interval.ms = 5000

group.consumer.min.session.timeout.ms = 45000

group.consumer.session.timeout.ms = 45000

group.coordinator.append.linger.ms = 10

group.coordinator.new.enable = false

group.coordinator.rebalance.protocols = [classic]

group.coordinator.threads = 1

group.initial.rebalance.delay.ms = 0

group.max.session.timeout.ms = 1800000

group.max.size = 2147483647

group.min.session.timeout.ms = 6000

group.share.delivery.count.limit = 5

group.share.enable = false

group.share.heartbeat.interval.ms = 5000

group.share.max.groups = 10

group.share.max.heartbeat.interval.ms = 15000

group.share.max.record.lock.duration.ms = 60000

group.share.max.session.timeout.ms = 60000

group.share.max.size = 200

group.share.min.heartbeat.interval.ms = 5000

group.share.min.record.lock.duration.ms = 15000

group.share.min.session.timeout.ms = 45000

group.share.partition.max.record.locks = 200

group.share.record.lock.duration.ms = 30000

group.share.session.timeout.ms = 45000

initial.broker.registration.timeout.ms = 60000

inter.broker.listener.name = null

inter.broker.protocol.version = 3.9-IV0

kafka.metrics.polling.interval.secs = 10

kafka.metrics.reporters = []

leader.imbalance.check.interval.seconds = 300

leader.imbalance.per.broker.percentage = 10

listener.security.protocol.map = SASL\_SSL:SASL\_SSL,PLAINTEXT:PLAINTEXT,SSL:SSL,SASL\_PLAINTEXT:SASL\_PLAINTEXT

listeners = PLAINTEXT://:9092

log.cleaner.backoff.ms = 15000

log.cleaner.dedupe.buffer.size = 134217728

log.cleaner.delete.retention.ms = 86400000

log.cleaner.enable = true

log.cleaner.io.buffer.load.factor = 0.9

log.cleaner.io.buffer.size = 524288

log.cleaner.io.max.bytes.per.second = 1.7976931348623157E308

log.cleaner.max.compaction.lag.ms = 9223372036854775807

log.cleaner.min.cleanable.ratio = 0.5

log.cleaner.min.compaction.lag.ms = 0

log.cleaner.threads = 1

log.cleanup.policy = [delete]

log.dir = /tmp/kafka-logs

log.dir.failure.timeout.ms = 30000

log.dirs = /opt/homebrew/var/lib/kafka-logs

log.flush.interval.messages = 9223372036854775807

log.flush.interval.ms = null

log.flush.offset.checkpoint.interval.ms = 60000

log.flush.scheduler.interval.ms = 9223372036854775807

log.flush.start.offset.checkpoint.interval.ms = 60000

log.index.interval.bytes = 4096

log.index.size.max.bytes = 10485760

log.initial.task.delay.ms = 30000

log.local.retention.bytes = -2

log.local.retention.ms = -2

log.message.downconversion.enable = true

log.message.format.version = 3.0-IV1

log.message.timestamp.after.max.ms = 9223372036854775807

log.message.timestamp.before.max.ms = 9223372036854775807

log.message.timestamp.difference.max.ms = 9223372036854775807

log.message.timestamp.type = CreateTime

log.preallocate = false

log.retention.bytes = -1

log.retention.check.interval.ms = 300000

log.retention.hours = 168

log.retention.minutes = null

log.retention.ms = null

log.roll.hours = 168

log.roll.jitter.hours = 0

log.roll.jitter.ms = null

log.roll.ms = null

log.segment.bytes = 1073741824

log.segment.delete.delay.ms = 60000

max.connection.creation.rate = 2147483647

max.connections = 2147483647

max.connections.per.ip = 2147483647

max.connections.per.ip.overrides =

max.incremental.fetch.session.cache.slots = 1000

max.request.partition.size.limit = 2000

message.max.bytes = 1048588

metadata.log.dir = null

metadata.log.max.record.bytes.between.snapshots = 20971520

metadata.log.max.snapshot.interval.ms = 3600000

metadata.log.segment.bytes = 1073741824

metadata.log.segment.min.bytes = 8388608

metadata.log.segment.ms = 604800000

metadata.max.idle.interval.ms = 500

metadata.max.retention.bytes = 104857600

metadata.max.retention.ms = 604800000

metric.reporters = []

metrics.num.samples = 2

metrics.recording.level = INFO

metrics.sample.window.ms = 30000

min.insync.replicas = 1

node.id = 0

num.io.threads = 8

num.network.threads = 3

num.partitions = 1

num.recovery.threads.per.data.dir = 1

num.replica.alter.log.dirs.threads = null

num.replica.fetchers = 1

offset.metadata.max.bytes = 4096

offsets.commit.required.acks = -1

offsets.commit.timeout.ms = 5000

offsets.load.buffer.size = 5242880

offsets.retention.check.interval.ms = 600000

offsets.retention.minutes = 10080

offsets.topic.compression.codec = 0

offsets.topic.num.partitions = 50

offsets.topic.replication.factor = 1

offsets.topic.segment.bytes = 104857600

password.encoder.cipher.algorithm = AES/CBC/PKCS5Padding

password.encoder.iterations = 4096

password.encoder.key.length = 128

password.encoder.keyfactory.algorithm = null

password.encoder.old.secret = null

password.encoder.secret = null

principal.builder.class = class org.apache.kafka.common.security.authenticator.DefaultKafkaPrincipalBuilder

process.roles = []

producer.id.expiration.check.interval.ms = 600000

producer.id.expiration.ms = 86400000

producer.purgatory.purge.interval.requests = 1000

queued.max.request.bytes = -1

queued.max.requests = 500

quota.window.num = 11

quota.window.size.seconds = 1

remote.fetch.max.wait.ms = 500

remote.log.index.file.cache.total.size.bytes = 1073741824

remote.log.manager.copier.thread.pool.size = -1

remote.log.manager.copy.max.bytes.per.second = 9223372036854775807

remote.log.manager.copy.quota.window.num = 11

remote.log.manager.copy.quota.window.size.seconds = 1

remote.log.manager.expiration.thread.pool.size = -1

remote.log.manager.fetch.max.bytes.per.second = 9223372036854775807

remote.log.manager.fetch.quota.window.num = 11

remote.log.manager.fetch.quota.window.size.seconds = 1

remote.log.manager.task.interval.ms = 30000

remote.log.manager.task.retry.backoff.max.ms = 30000

remote.log.manager.task.retry.backoff.ms = 500

remote.log.manager.task.retry.jitter = 0.2

remote.log.manager.thread.pool.size = 10

remote.log.metadata.custom.metadata.max.bytes = 128

remote.log.metadata.manager.class.name = org.apache.kafka.server.log.remote.metadata.storage.TopicBasedRemoteLogMetadataManager

remote.log.metadata.manager.class.path = null

remote.log.metadata.manager.impl.prefix = rlmm.config.

remote.log.metadata.manager.listener.name = null

remote.log.reader.max.pending.tasks = 100

remote.log.reader.threads = 10

remote.log.storage.manager.class.name = null

remote.log.storage.manager.class.path = null

remote.log.storage.manager.impl.prefix = rsm.config.

remote.log.storage.system.enable = false

replica.fetch.backoff.ms = 1000

replica.fetch.max.bytes = 1048576

replica.fetch.min.bytes = 1

replica.fetch.response.max.bytes = 10485760

replica.fetch.wait.max.ms = 500

replica.high.watermark.checkpoint.interval.ms = 5000

replica.lag.time.max.ms = 30000

replica.selector.class = null

replica.socket.receive.buffer.bytes = 65536

replica.socket.timeout.ms = 30000

replication.quota.window.num = 11

replication.quota.window.size.seconds = 1

request.timeout.ms = 30000

reserved.broker.max.id = 1000

sasl.client.callback.handler.class = null

sasl.enabled.mechanisms = [GSSAPI]

sasl.jaas.config = null

sasl.kerberos.kinit.cmd = /usr/bin/kinit

sasl.kerberos.min.time.before.relogin = 60000

sasl.kerberos.principal.to.local.rules = [DEFAULT]

sasl.kerberos.service.name = null

sasl.kerberos.ticket.renew.jitter = 0.05

sasl.kerberos.ticket.renew.window.factor = 0.8

sasl.login.callback.handler.class = null

sasl.login.class = null

sasl.login.connect.timeout.ms = null

sasl.login.read.timeout.ms = null

sasl.login.refresh.buffer.seconds = 300

sasl.login.refresh.min.period.seconds = 60

sasl.login.refresh.window.factor = 0.8

sasl.login.refresh.window.jitter = 0.05

sasl.login.retry.backoff.max.ms = 10000

sasl.login.retry.backoff.ms = 100

sasl.mechanism.controller.protocol = GSSAPI

sasl.mechanism.inter.broker.protocol = GSSAPI

sasl.oauthbearer.clock.skew.seconds = 30

sasl.oauthbearer.expected.audience = null

sasl.oauthbearer.expected.issuer = null

sasl.oauthbearer.jwks.endpoint.refresh.ms = 3600000

sasl.oauthbearer.jwks.endpoint.retry.backoff.max.ms = 10000

sasl.oauthbearer.jwks.endpoint.retry.backoff.ms = 100

sasl.oauthbearer.jwks.endpoint.url = null

sasl.oauthbearer.scope.claim.name = scope

sasl.oauthbearer.sub.claim.name = sub

sasl.oauthbearer.token.endpoint.url = null

sasl.server.callback.handler.class = null

sasl.server.max.receive.size = 524288

security.inter.broker.protocol = PLAINTEXT

security.providers = null

server.max.startup.time.ms = 9223372036854775807

socket.connection.setup.timeout.max.ms = 30000

socket.connection.setup.timeout.ms = 10000

socket.listen.backlog.size = 50

socket.receive.buffer.bytes = 102400

socket.request.max.bytes = 104857600

socket.send.buffer.bytes = 102400

ssl.allow.dn.changes = false

ssl.allow.san.changes = false

ssl.cipher.suites = []

ssl.client.auth = none

ssl.enabled.protocols = [TLSv1.2, TLSv1.3]

ssl.endpoint.identification.algorithm = https

ssl.engine.factory.class = null

ssl.key.password = null

ssl.keymanager.algorithm = SunX509

ssl.keystore.certificate.chain = null

ssl.keystore.key = null

ssl.keystore.location = null

ssl.keystore.password = null

ssl.keystore.type = JKS

ssl.principal.mapping.rules = DEFAULT

ssl.protocol = TLSv1.3

ssl.provider = null

ssl.secure.random.implementation = null

ssl.trustmanager.algorithm = PKIX

ssl.truststore.certificates = null

ssl.truststore.location = null

ssl.truststore.password = null

ssl.truststore.type = JKS

telemetry.max.bytes = 1048576

transaction.abort.timed.out.transaction.cleanup.interval.ms = 10000

transaction.max.timeout.ms = 900000

transaction.partition.verification.enable = true

transaction.remove.expired.transaction.cleanup.interval.ms = 3600000

transaction.state.log.load.buffer.size = 5242880

transaction.state.log.min.isr = 1

transaction.state.log.num.partitions = 50

transaction.state.log.replication.factor = 1

transaction.state.log.segment.bytes = 104857600

transactional.id.expiration.ms = 604800000

unclean.leader.election.enable = false

unclean.leader.election.interval.ms = 300000

unstable.api.versions.enable = false

unstable.feature.versions.enable = false

zookeeper.clientCnxnSocket = null

zookeeper.connect = localhost:2181

zookeeper.connection.timeout.ms = 18000

zookeeper.max.in.flight.requests = 10

zookeeper.metadata.migration.enable = false

zookeeper.metadata.migration.min.batch.size = 200

zookeeper.session.timeout.ms = 18000

zookeeper.set.acl = false

zookeeper.ssl.cipher.suites = null

zookeeper.ssl.client.enable = false

zookeeper.ssl.crl.enable = false

zookeeper.ssl.enabled.protocols = null

zookeeper.ssl.endpoint.identification.algorithm = HTTPS

zookeeper.ssl.keystore.location = null

zookeeper.ssl.keystore.password = null

zookeeper.ssl.keystore.type = null

zookeeper.ssl.ocsp.enable = false

zookeeper.ssl.protocol = TLSv1.2

zookeeper.ssl.truststore.location = null

zookeeper.ssl.truststore.password = null

zookeeper.ssl.truststore.type = null

(kafka.server.KafkaConfig)

[2024-12-06 09:08:16,181] INFO [ThrottledChannelReaper-Fetch]: Starting (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,181] INFO [ThrottledChannelReaper-Produce]: Starting (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,182] INFO [ThrottledChannelReaper-Request]: Starting (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,183] INFO [ThrottledChannelReaper-ControllerMutation]: Starting (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,194] ERROR [KafkaServer id=0] Fatal error during KafkaServer startup. Prepare to shutdown (kafka.server.KafkaServer)

org.apache.kafka.common.KafkaException: Failed to acquire lock on file .lock in /opt/homebrew/var/lib/kafka-logs. A Kafka instance in another process or thread is using this directory.

at kafka.log.LogManager.$anonfun$lockLogDirs$1(LogManager.scala:270)

at scala.collection.StrictOptimizedIterableOps.flatMap(StrictOptimizedIterableOps.scala:118)

at scala.collection.StrictOptimizedIterableOps.flatMap$(StrictOptimizedIterableOps.scala:105)

at scala.collection.mutable.ArrayBuffer.flatMap(ArrayBuffer.scala:42)

at kafka.log.LogManager.lockLogDirs(LogManager.scala:265)

at kafka.log.LogManager.<init>(LogManager.scala:128)

at kafka.log.LogManager$.apply(LogManager.scala:1599)

at kafka.server.KafkaServer.startup(KafkaServer.scala:325)

at kafka.Kafka$.main(Kafka.scala:112)

at kafka.Kafka.main(Kafka.scala)

[2024-12-06 09:08:16,195] INFO [KafkaServer id=0] shutting down (kafka.server.KafkaServer)

[2024-12-06 09:08:16,197] INFO [ZooKeeperClient Kafka server] Closing. (kafka.zookeeper.ZooKeeperClient)

[2024-12-06 09:08:16,324] INFO Session: 0x1000067d65d0003 closed (org.apache.zookeeper.ZooKeeper)

[2024-12-06 09:08:16,324] INFO EventThread shut down for session: 0x1000067d65d0003 (org.apache.zookeeper.ClientCnxn)

[2024-12-06 09:08:16,325] INFO [ZooKeeperClient Kafka server] Closed. (kafka.zookeeper.ZooKeeperClient)

[2024-12-06 09:08:16,325] INFO [ThrottledChannelReaper-Fetch]: Shutting down (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Fetch]: Stopped (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Fetch]: Shutdown completed (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Produce]: Shutting down (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Produce]: Stopped (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Produce]: Shutdown completed (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Request]: Shutting down (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Request]: Stopped (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-Request]: Shutdown completed (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-ControllerMutation]: Shutting down (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,328] INFO [ThrottledChannelReaper-ControllerMutation]: Stopped (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,329] INFO [ThrottledChannelReaper-ControllerMutation]: Shutdown completed (kafka.server.ClientQuotaManager$ThrottledChannelReaper)

[2024-12-06 09:08:16,338] INFO Metrics scheduler closed (org.apache.kafka.common.metrics.Metrics)

[2024-12-06 09:08:16,338] INFO Metrics reporters closed (org.apache.kafka.common.metrics.Metrics)

[2024-12-06 09:08:16,340] INFO Broker and topic stats closed (kafka.server.BrokerTopicStats)

[2024-12-06 09:08:16,344] INFO App info kafka.server for 0 unregistered (org.apache.kafka.common.utils.AppInfoParser)

[2024-12-06 09:08:16,344] INFO [KafkaServer id=0] shut down completed (kafka.server.KafkaServer)

[2024-12-06 09:08:16,344] ERROR Exiting Kafka due to fatal exception during startup. (kafka.Kafka$)

org.apache.kafka.common.KafkaException: Failed to acquire lock on file .lock in /opt/homebrew/var/lib/kafka-logs. A Kafka instance in another process or thread is using this directory.

at kafka.log.LogManager.$anonfun$lockLogDirs$1(LogManager.scala:270)

at scala.collection.StrictOptimizedIterableOps.flatMap(StrictOptimizedIterableOps.scala:118)

at scala.collection.StrictOptimizedIterableOps.flatMap$(StrictOptimizedIterableOps.scala:105)

at scala.collection.mutable.ArrayBuffer.flatMap(ArrayBuffer.scala:42)

at kafka.log.LogManager.lockLogDirs(LogManager.scala:265)

at kafka.log.LogManager.<init>(LogManager.scala:128)

at kafka.log.LogManager$.apply(LogManager.scala:1599)

at kafka.server.KafkaServer.startup(KafkaServer.scala:325)

at kafka.Kafka$.main(Kafka.scala:112)

at kafka.Kafka.main(Kafka.scala)

[2024-12-06 09:08:16,344] INFO [KafkaServer id=0] shutting down (kafka.server.KafkaServer)

**Sample output, if kafdrop is started appropriately :**

Using java version 24.ea.15-open in this shell.

➜ kafdrop java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \-jar kafdrop-4.0.2.jar \--kafka.brokerConnect=localhost:9092

2024-12-06 07:44:54.353 INFO 30542 [kground-preinit] o.h.v.i.u.Version : HV000001: Hibernate Validator 8.0.1.Final

2024-12-06 07:44:54.390 INFO 30542 [ main] o.s.b.StartupInfoLogger : Starting Kafdrop v4.0.2 using Java 24-ea with PID 30542 (/Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop/kafdrop-4.0.2.jar started by <user> in /Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop)

2024-12-06 07:44:54.392 INFO 30542 [ main] o.s.b.SpringApplication : No active profile set, falling back to 1 default profile: "default"

2024-12-06 07:44:55.599 INFO 30542 [ main] i.u.s.s.ServletContextImpl : Initializing Spring embedded WebApplicationContext

2024-12-06 07:44:55.599 INFO 30542 [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1160 ms

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 07:44:55.831 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:55.831 INFO 30542 [ main] k.s.BuildInfo : Kafdrop version: 4.0.2, build time: 2024-07-09T13:11:53.509Z

2024-12-06 07:44:55.934 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.137 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.339 INFO 30542 [ main] o.s.b.a.e.w.EndpointLinksResolver : Exposing 14 endpoints beneath base path '/actuator'

2024-12-06 07:44:56.390 INFO 30542 [ main] i.u.Undertow : starting server: Undertow - 2.3.13.Final

2024-12-06 07:44:56.397 INFO 30542 [ main] o.x.Xnio : XNIO version 3.8.8.Final

2024-12-06 07:44:56.403 INFO 30542 [ main] o.x.n.NioXnio : XNIO NIO Implementation Version 3.8.8.Final

2024-12-06 07:44:56.421 INFO 30542 [ main] o.j.t.Version : JBoss Threads version 3.5.0.Final

2024-12-06 07:44:56.441 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.463 INFO 30542 [ main] o.s.b.w.e.u.UndertowWebServer : Undertow started on port 9000 (http) with context path '/'

2024-12-06 07:44:56.474 INFO 30542 [ main] o.s.b.StartupInfoLogger : Started Kafdrop in 2.329 seconds (process running for 3.063)

2024-12-06 07:44:56.848 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:57.661 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:58.675 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:59.590 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:00.504 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:01.519 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:02.433 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:03.348 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

^C2024-12-06 07:45:03.561 INFO 30542 [ionShutdownHook] i.u.Undertow : stopping server: Undertow - 2.3.13.Final

➜ kafdrop java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \-jar kafdrop-4.0.2.jar \--kafka.brokerConnect=localhost:9092

2024-12-06 09:08:34.835 INFO 13410 [kground-preinit] o.h.v.i.u.Version : HV000001: Hibernate Validator 8.0.1.Final

2024-12-06 09:08:34.875 INFO 13410 [ main] o.s.b.StartupInfoLogger : Starting Kafdrop v4.0.2 using Java 24-ea with PID 13410 (/Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop/kafdrop-4.0.2.jar started by <user> in /Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop)

2024-12-06 09:08:34.877 INFO 13410 [ main] o.s.b.SpringApplication : No active profile set, falling back to 1 default profile: "default"

2024-12-06 09:08:36.127 INFO 13410 [ main] i.u.s.s.ServletContextImpl : Initializing Spring embedded WebApplicationContext

2024-12-06 09:08:36.127 INFO 13410 [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1199 ms

2024-12-06 09:08:36.249 INFO 13410 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 09:08:36.249 INFO 13410 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 09:08:36.249 INFO 13410 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 09:08:36.334 INFO 13410 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 09:08:36.334 INFO 13410 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 09:08:36.334 INFO 13410 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 09:08:36.368 INFO 13410 [ main] k.s.BuildInfo : Kafdrop version: 4.0.2, build time: 2024-07-09T13:11:53.509Z

2024-12-06 09:08:36.935 INFO 13410 [ main] o.s.b.a.e.w.EndpointLinksResolver : Exposing 14 endpoints beneath base path '/actuator'

2024-12-06 09:08:36.993 INFO 13410 [ main] i.u.Undertow : starting server: Undertow - 2.3.13.Final

2024-12-06 09:08:36.999 INFO 13410 [ main] o.x.Xnio : XNIO version 3.8.8.Final

2024-12-06 09:08:37.006 INFO 13410 [ main] o.x.n.NioXnio : XNIO NIO Implementation Version 3.8.8.Final

2024-12-06 09:08:37.025 INFO 13410 [ main] o.j.t.Version : JBoss Threads version 3.5.0.Final

2024-12-06 09:08:37.068 INFO 13410 [ main] o.s.b.w.e.u.UndertowWebServer : Undertow started on port 9000 (http) with context path '/'

2024-12-06 09:08:37.079 INFO 13410 [ main] o.s.b.StartupInfoLogger : Started Kafdrop in 2.454 seconds (process running for 3.159)

**Screenshot of kafdrop if started appropriately :**

Accessible via : http://127.0.0.1:9000



**If you try to start kafdrop before kafka is started properly, you would potentially see the sample output below :**

➜ kafdrop java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \-jar kafdrop-4.0.2.jar \--kafka.brokerConnect=localhost:9092

Error: LinkageError occurred while loading main class org.springframework.boot.loader.launch.PropertiesLauncher

java.lang.UnsupportedClassVersionError: org/springframework/boot/loader/launch/PropertiesLauncher has been compiled by a more recent version of the Java Runtime (class file version 61.0), this version of the Java Runtime only recognizes class file versions up to 55.0

➜ kafdrop java -version

openjdk version "11.0.24" 2024-07-16 LTS

OpenJDK Runtime Environment Corretto-11.0.24.8.1 (build 11.0.24+8-LTS)

OpenJDK 64-Bit Server VM Corretto-11.0.24.8.1 (build 11.0.24+8-LTS, mixed mode)

➜ kafdrop sdk use java 24.ea.15-open

Using java version 24.ea.15-open in this shell.

➜ kafdrop java --add-opens=java.base/sun.nio.ch=ALL-UNNAMED \-jar kafdrop-4.0.2.jar \--kafka.brokerConnect=localhost:9092

2024-12-06 07:44:54.353 INFO 30542 [kground-preinit] o.h.v.i.u.Version : HV000001: Hibernate Validator 8.0.1.Final

2024-12-06 07:44:54.390 INFO 30542 [ main] o.s.b.StartupInfoLogger : Starting Kafdrop v4.0.2 using Java 24-ea with PID 30542 (/Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop/kafdrop-4.0.2.jar started by <user> in /Users/<user>/\_\_work/\_source\_code\_goldmine/\_\_microservices\_tools/kafdrop)

2024-12-06 07:44:54.392 INFO 30542 [ main] o.s.b.SpringApplication : No active profile set, falling back to 1 default profile: "default"

2024-12-06 07:44:55.599 INFO 30542 [ main] i.u.s.s.ServletContextImpl : Initializing Spring embedded WebApplicationContext

2024-12-06 07:44:55.599 INFO 30542 [ main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1160 ms

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 07:44:55.718 INFO 30542 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking truststore file kafka.truststore.jks

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking keystore file kafka.keystore.jks

2024-12-06 07:44:55.799 INFO 30542 [ main] k.c.KafkaConfiguration : Checking properties file kafka.properties

2024-12-06 07:44:55.831 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:55.831 INFO 30542 [ main] k.s.BuildInfo : Kafdrop version: 4.0.2, build time: 2024-07-09T13:11:53.509Z

2024-12-06 07:44:55.934 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.137 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.339 INFO 30542 [ main] o.s.b.a.e.w.EndpointLinksResolver : Exposing 14 endpoints beneath base path '/actuator'

2024-12-06 07:44:56.390 INFO 30542 [ main] i.u.Undertow : starting server: Undertow - 2.3.13.Final

2024-12-06 07:44:56.397 INFO 30542 [ main] o.x.Xnio : XNIO version 3.8.8.Final

2024-12-06 07:44:56.403 INFO 30542 [ main] o.x.n.NioXnio : XNIO NIO Implementation Version 3.8.8.Final

2024-12-06 07:44:56.421 INFO 30542 [ main] o.j.t.Version : JBoss Threads version 3.5.0.Final

2024-12-06 07:44:56.441 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:56.463 INFO 30542 [ main] o.s.b.w.e.u.UndertowWebServer : Undertow started on port 9000 (http) with context path '/'

2024-12-06 07:44:56.474 INFO 30542 [ main] o.s.b.StartupInfoLogger : Started Kafdrop in 2.329 seconds (process running for 3.063)

2024-12-06 07:44:56.848 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:57.661 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:58.675 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:44:59.590 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:00.504 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:01.519 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:02.433 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

2024-12-06 07:45:03.348 WARN 30542 [| kafdrop-admin] o.a.k.c.NetworkClient : [AdminClient clientId=kafdrop-admin] Connection to node -1 (localhost/127.0.0.1:9092) could not be established. Node may not be available.

^C2024-12-06 07:45:03.561 INFO 30542 [ionShutdownHook] i.u.Undertow : stopping server: Undertow - 2.3.13.Final

**Setting up PostgreSQL on pgAdmin**

CREATE ROLE user\_00 WITH

LOGIN

SUPERUSER

CREATEDB

CREATEROLE

INHERIT

NOREPLICATION

CONNECTION LIMIT -1

PASSWORD 'xxxxxx';

CREATE DATABASE fs\_stepevent\_db

WITH

OWNER = user\_00

ENCODING = 'UTF8'

CONNECTION LIMIT = -1;

CREATE TABLE IF NOT EXISTS stepevent  
(  
 device\_id VARCHAR,  
 device\_sync BIGINT,  
 sync\_timestamp timestamptz,  
 steps\_count INTEGER,  
 PRIMARY KEY (device\_id, device\_sync)  
);

**Setting up MongoDB on Mongo Compass**

After successfully downloading from :

<https://www.mongodb.com/try/download/community>

Before starting the process, remember to add the full path to the ‘ mongod ‘ to your environmental variables :

/Users/<user>/mongodb/bin

And also export the variable “ MONGODB “ ;

Pointing to the path : /Users/<user>/mongodb/bin

As seen below :

export MONGODB=/Users/<user>/mongodb/bin

Now open a terminal session and test that the path addition and variable setting are all working fine by entering the command :

$ mongod --version

And you should be getting an output such as :

db version v6.0.6

Build Info: {

"version": "6.0.6",

"gitVersion": "26b4851a412cc8b9b4a18cdb6cd0f9f642e06aa7",

"modules": [],

"allocator": "system",

"environment": {

"distarch": "aarch64",

"target\_arch": "aarch64"

}

}

Follow these steps to run MongoDB Community Edition. These instructions assume that you are using the default settings.

1 - **Create the data directory.**

Before you start MongoDB for the first time, you must create the directory to which the [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) process will write data.

For example, to create the ~/data/db directory:

|  |
| --- |
| sudo mkdir -p ~/data/db |

2 - **Create the log directory.**

You must also create the directory in which the mongod process will write its log file:

For example, to create the ~/data/log/mongodb directory:

|  |
| --- |
| sudo mkdir -p ~/data/log/mongodb |

3 - **Set permissions for the data and log directories.**

Ensure that the user account running [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) has read and write permissions for these two directories. If you are running [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) as your own user account, and you just created the two directories above, they should already accessible to your user. Otherwise, you can use chown to set ownership, substituting the appropriate *user*:

|  |
| --- |
| sudo chown <user> ~/data/db |
| sudo chown <user> ~/data/log/mongodb |

4 - **Run MongoDB.**

To run MongoDB, run the [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) process at the system prompt, providing the two parameters dbpath and logpath from above, and the fork parameter to run [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) in the background. Alternatively, you may choose to store the values for dbpath, logpath, fork, and many other parameters in a [configuration file.](https://www.mongodb.com/docs/manual/reference/configuration-options/#std-label-configuration-options)

**Run mongod with command-line parameters**

Run the [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) process at the system prompt, providing the three necessary parameters directly on the command-line:

|  |
| --- |
| $ mongod --port 27017 --dbpath ~/data/db --logpath ~/data/log/mongodb/mongo.log --fork |

**Run mongod with a configuration file**

Run the [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) process at the system prompt, providing the path to a [configuration file](https://www.mongodb.com/docs/manual/reference/configuration-options/#std-label-configuration-options) with the config parameter:

|  |
| --- |
| $ mongod --config /usr/local/etc/mongod.conf |

macOS may prevent [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) from running after installation. If you receive a security error when starting [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) indicating that the developer could not be identified or verified, do the following to grant [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) access to run:

* Open *System Preferences*
* Select the *Security and Privacy* pane.
* Under the *General* tab, click the button to the right of the message about [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod), labelled either Open Anyway or Allow Anyway depending on your version of macOS.

5 - **Verify that MongoDB has started successfully.**

1. Verify that MongoDB has started successfully:

|  |
| --- |
| $ ps aux | grep -v grep | grep mongod |

If you do not see a mongod process running, check the logfile for any error messages.

1. You can also verify that mongodb is running by entering the command :

$ lsof -i

And you should be able to see this in the list somewhere/somehow :

mongod 27959 speedo 10u IPv4 0x52dea907a69ff1c9 0t0 TCP localhost:27017 (LISTEN)

6 - **Begin using MongoDB.**

Start a [mongosh](https://www.mongodb.com/docs/mongodb-shell/#mongodb-binary-bin.mongosh) session on the same host machine as the [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod). You can run [mongosh](https://www.mongodb.com/docs/mongodb-shell/#mongodb-binary-bin.mongosh) without any command-line options to connect to a [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) that is running on your *localhost* with the default port of *27017*:

|  |
| --- |
| $ mongosh |

For more information on connecting using [mongosh](https://www.mongodb.com/docs/mongodb-shell/#mongodb-binary-bin.mongosh), such as to connect to a [mongod](https://www.mongodb.com/docs/manual/reference/program/mongod/#mongodb-binary-bin.mongod) instance running on a different host and/or port, see the [mongosh documentation.](https://www.mongodb.com/docs/mongodb-shell/)

To help you start using MongoDB, MongoDB provides [Getting Started Guides](https://www.mongodb.com/docs/manual/tutorial/getting-started/#std-label-getting-started) in various driver editions. See [Getting Started with MongoDB](https://www.mongodb.com/docs/manual/tutorial/getting-started/#std-label-getting-started) for the available editions.

https://www.slingacademy.com/article/mongodb-create-admin-user-enable-authentication/

## Introduction

Securing your MongoDB database is an essential step to protect your data from unauthorized access. One of the first steps in securing your MongoDB instance is creating an admin user and enabling authentication. This tutorial will guide you through the process of setting up an administrative user for your MongoDB database and turning on authentication to ensure that only authorized users have access to your database’s functionalities.

## Prerequisites

Before you begin, ensure you have the following:

* MongoDB installed on your system
* Access to the MongoDB command line interface (CLI)
* Basic understanding of command-line operations and MongoDB concepts

## Step 1: Starting MongoDB Without Access Control

Access your MongoDB instance without access control to create the initial admin user. Typically, after installation, MongoDB starts without access control enabled. You can start a MongoDB instance without access control using the following command:

mongod --port 27017 --dbpath /data/db1

Make sure to replace /data/db1 with the actual path to your MongoDB data directory.

## Step 2: Connecting to MongoDB Instance

Once your MongoDB instance is running, connect to it using the mongo shell with the following command:

mongo

## Step 3: Creating the Admin User

Upon connecting to the MongoDB instance, create the administrative user by switching to the admin database and running the createUser command: ( or within MongoDB Compass : you can switch to “ >\_MONGOSH “ and use the terminal-like parts to excute this.

use admin

db.createUser({

user: "UserName",

pwd: "UserPassword",

roles: [{ role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase"]

});

Replace " UserName" and "UserPassword" with your desired admin username and secure password respectively.

## Step 4: Enabling Access Control and Testing Authentication

With the admin user created, access control needs to be enabled. You do this by restarting the MongoDB instance with the --auth flag:

mongod --auth --port 27017 --dbpath /data/db1

Once restarted, attempt to connect to the database using the admin credentials to verify that authentication is working:

mongo -u UserName -p UserPassword --authenticationDatabase admin

OR :

mongo –-username UserName -–password UserPassword -–authenticationDatabase admin

You should be connected to the database if the credentials are correct. If you encounter errors, ensure that the username and password are as you set in the createUser command.

## Step 5: Managing User Roles

After the administrative user has been set up, you can manage various user roles. MongoDB provides a flexible role-based access control system. Below is an example of how to provide a user with read-only access to a particular database:

use someDatabase

db.createUser({

user: "readOnlyUser",

pwd: "readOnlyPassword",

roles: [{ role: "read", db: "someDatabase" }]

});

Again, ensure that you are using secure passwords and replace "readOnlyUser" and "readOnlyPassword" with your chosen credentials.

## Step 6: Advanced User Management

For more advanced user management, you can create roles with customized permissions. Below is an example of creating a role:

use admin

db.createRole({

role: "myCustomRole",

privileges: [

{ resource: { db: "myDatabase", collection: "" }, actions: [ "find", "update", "insert", "remove" ] }

],

roles: []

});

Once the role is created, you can assign it to a user with the following command:

db.grantRolesToUser("existingUser", ["myCustomRole"]);

Ensure that you replace "existingUser" with the actual username of the user to whom you wish to grant the newly created role.

**Running Mongo Compass**

After ensuring that mongod is running very well.

You can then click and run the Mongo Compass app on your mac

And by using the mongosh area :

You can issue commands to create the ;

Database,

Then the

Collection(s).

Sample outputs from Mongo Compass’ mongosh :

db.createCollection("user");

|  |
| --- |
| { ok: 1 } |

db.user.createIndex({username: 1}, {unique: true});

|  |
| --- |
| **'username\_1'** |

db.user.createIndex({deviceId: 1}, {unique: true});

|  |
| --- |
| **'deviceId\_1'** |

user\_profiles\_db

**ReferenceError:** user\_profiles\_db is not defined

help

Shell Help

|  |  |
| --- | --- |
| **use** | Set current database |
| **show** | 'show databases'/'show dbs': Print a list of all available databases. 'show collections'/'show tables': Print a list of all collections for current database. 'show profile': Prints system.profile information. 'show users': Print a list of all users for current database. 'show roles': Print a list of all roles for current database. 'show log <type>': log for current connection, if type is not set uses 'global' 'show logs': Print all logs. |
| **exit** | Quit the MongoDB shell with exit/exit()/.exit |
| **quit** | Quit the MongoDB shell with quit/quit() |
| **Mongo** | Create a new connection and return the Mongo object. Usage: new Mongo(URI, options [optional]) |
| **connect** | Create a new connection and return the Database object. Usage: connect(URI, username [optional], password [optional]) |
| **it** | result of the last line evaluated; use to further iterate |
| **version** | Shell version |
| **load** | Loads and runs a JavaScript file into the current shell environment |
| **enableTelemetry** | Enables collection of anonymous usage data to improve the mongosh CLI |
| **disableTelemetry** | Disables collection of anonymous usage data to improve the mongosh CLI |
| **passwordPrompt** | Prompts the user for a password |
| **sleep** | Sleep for the specified number of milliseconds |
| **print** | Prints the contents of an object to the output |
| **printjson** | Alias for print() |
| **cls** | Clears the screen like console.clear() |
| **isInteractive** | Returns whether the shell will enter or has entered interactive mode |

For more information on usage: [**https://docs.mongodb.com/manual/reference/method**](https://docs.mongodb.com/manual/reference/method)

use user\_profiles\_db

|  |
| --- |
| **'switched to db user\_profiles\_db'** |

db.createCollection("user");

**MongoServerError:** Collection user\_profiles\_db.user already exists.

db.user.createIndex({username: 1}, {unique: true});

db.user.createIndex({deviceId: 1}, {unique: true});

|  |
| --- |
| **'deviceId\_1'** |

**How to upgrade/update mongod or mongodb**

**Lessons from attempting to connect to a mongoDB**

Some error messages that might be encountered :

//I was getting this error msg :

//( MongoParseError: options usecreateindex, usefindandmodify are not supported

/\*

After a successful run of the application, I got this further msg :

(node:85040) [MONGODB DRIVER] Warning: useNewUrlParser is a deprecated option: useNewUrlParser has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

(Use `node --trace-warnings ...` to show where the warning was created)

(node:85040) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopology has no effect since Node.js Driver version 4.0.0 and will be removed in the next major version

\*/

**How to connect to mongoDB (ref: )**

<https://mongoosejs.com/docs/connections.html>

<https://www.geeksforgeeks.org/how-to-use-mongodb-and-mongoose-with-node-js/>

**Switching from InteliJ to VSCode**

**Setting up Mailhog**

To install mailhog :

$ brew update && brew install mailhog

To run mailhog on a mac via terminal :

$ mailhog

Sample output via mac terminal :

2024/12/08 07:03:09 Using in-memory storage

[HTTP] Binding to address: 0.0.0.0:8025

2024/12/08 07:03:09 Serving under http://0.0.0.0:8025/

2024/12/08 07:03:09 [SMTP] Binding to address: 0.0.0.0:1025

Creating API v1 with WebPath:

Creating API v2 with WebPath:

How to access mailhog in the browser :

http://127.0.0.1:8025





**Troubleshooting kafka : Groups, Topics & Partitions for issue of duplicate emails or repetitive actions upon receiving an event trigger.**

**Alternatives to Mailhog**

1 - Mailcatcher,

2- FakeSMTP,

3 - LunaticSMTP,

4 - SMTPBucket,

5 - Mailtrap{

Free: [$0, 1000emails/mo, 500contacts],

Basic10K: [$15, 10000emails/mo, 2500contacts],

Business100K: [$85, 100000emails/mo, 25000contacts],

Enterprise: [$750, 1.5Memails/mo, 375000contacts]

}

**Other functionalities with SMTP setup and testing**

* 1. - Email templating
  2. – Open rate tracking
  3. – Click rate tracking
  4. – HTTPS link branding
  5. – Full Deliverability Analytics
  6. – Statistics for each mailbox provider
  7. – Logging
  8. – GDPR compliance
  9. – ISO 27001

**Interesting discoveries with Kafka : event streaming and processing**

After a very thorough testing and checking the application developed, and monitoring how the kafka producers and consumers are communicating with each other, the concerns of an override based on “key” similarities within the Kafka records for the specific Kafka topic.

After adding “Order(x)” in the test class, it becomes very clear that the vertx code receives the event trigger twice for single kafka event trigger for two methods with one method making a call to the kafka ‘engine’ twice.

The screenshot below shows what is been described :



**The Data Structure behaviour of Kafka**

**Kafka topics are not queues** in the computer science manner.

Queue means First in First out (FIFO) - this is not what you get in Kafka in the topic level.

Order is guaranteed for each partition in topic based on the partition key. So if order matters, you need to partition by the value on which order matters. This is actually stronger ordering guarantees than rabbitmq, which may only have one consumer to guarantee ordering.

One consumer per partition, not per topic. The issue is in rabbitmq as well. If you want messages to be processed in guaranteed order, then you can only have one consumer for that queue. You cannot process work in order with parallel consumers.

Kafka main advantage is in streaming of huge amount of data. If u r not streaming huge amount of data - Kafka is probably a bad choice

Order is not guaranteed when you have multiple consumers in any meaningful way. What if one consumer fails and the task gets requeued? What if a consumer A finishes a task before consumer B, even though they received them in the opposite order? Kafka has iron clad ordering guarantees. The vast majority of message queues do not, including rabbit mq, unless you have a single producer and a single consumer.

The experience of using kafka has led me to rally checking all sorts of other event streaming and queue-ing systems all over the internet, such as RabbitMQ, ActiveMQ.

# Kafka:

* Real-time Analytics: Companies like LinkedIn use Kafka for real-time monitoring and analytics.
* Event Sourcing: Kafka can be used to capture and store every change, allowing systems to be rebuilt from scratch if needed.

# RabbitMQ:

* Decoupling Applications: Used by companies to decouple their microservices, ensuring one service’s failure doesn’t affect others.
* Task Queues: Offloading tasks to be processed in the background.

# ActiveMQ:

* Financial Transactions: Ensuring reliable message delivery for critical financial transactions.
* Order Processing Systems: Ensuring orders are processed reliably and in sequence.

And the following links were very helpful :

* [Introduction to Apache Kafka](https://kafka.apache.org/intro)
* [RabbitMQ Official Documentation](https://www.rabbitmq.com/documentation.html)
* [ActiveMQ Overview](https://activemq.apache.org/)

**Reviewing and Considerations for Samza and or Storm**

<https://samza.apache.org/startup/hello-samza/latest/>

<https://samza.apache.org/learn/documentation/0.7.0/comparisons/storm.html>

<https://samza.apache.org/startup/download/>

<https://samza.apache.org/startup/hello-samza/latest/>

<https://samza.apache.org/startup/quick-start/latest/samza.html>

### Checking out and Building

If you’re interested in working on Samza, or building the JARs from scratch, then you’ll need to checkout and build the code. Samza does not have a binary release at this time. To check out and build Samza, run these commands.

git clone http://git-wip-us.apache.org/repos/asf/samza.git

cd samza

./gradlew clean build

### Maven

All Samza JARs are published through [Apache’s Maven repository](https://repository.apache.org/content/groups/public/org/apache/samza/).

#### Artifacts

A Maven-based Samza project can pull in all required dependencies Samza dependencies this XML block:

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-api</artifactId>

<version>1.6.0</version>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-core\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-shell</artifactId>

<classifier>dist</classifier>

<type>tgz</type>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-yarn\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-kv\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-kv-rocksdb\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-kv-inmemory\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-kafka\_2.11</artifactId>

<version>1.6.0</version>

<scope>runtime</scope>

</dependency>

Samza versions less than 0.12 should use artifacts with scala version 2.10 as suffix. For example,

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-yarn\_2.10</artifactId>

<version>0.11.0</version>

</dependency>

Samza versions less than 0.9 should include this additional dependency.

<dependency>

<setId>org.apache.samza</setId>

<artifactId>samza-serializers\_2.10</artifactId>

<version>0.8.1</version>

</dependency>

[Hello Samza](https://samza.apache.org/startup/hello-samza/latest) is a working Maven project that illustrates how to build projects that have Samza jobs in them.

#### Repositories

Samza is available in the Apache Maven repository.

<repository>

<id>apache-releases</id>

<url>https://repository.apache.org/content/groups/public</url>

</repository>

Snapshot builds are available in the Apache Maven snapshot repository.

<repository>

<id>apache-snapshots</id>

<url>https://repository.apache.org/content/groups/snapshots</url>

</repository>

## Hello Samza

The [hello-samza](https://github.com/apache/samza-hello-samza) project is an example project designed to help you run your first Samza job.

### Get the Code

Check out the hello-samza project:

git clone https://gitbox.apache.org/repos/asf/samza-hello-samza.git hello-samza

cd hello-samza

This project contains everything you’ll need to run your first Samza jobs.

### Start a Grid

A Samza grid usually comprises three different systems: [YARN](http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html), [Kafka](http://kafka.apache.org/), and [ZooKeeper](http://zookeeper.apache.org/). The hello-samza project comes with a script called “grid” to help you setup these systems. Start by running:

bin/grid bootstrap

This command will download, install, and start ZooKeeper, Kafka, and YARN. It will also check out the latest version of Samza and build it. All package files will be put in a sub-directory called “deploy” inside hello-samza’s root folder.

If you get a complaint that JAVA\_HOME is not set, then you’ll need to set it to the path where Java is installed on your system.

Once the grid command completes, you can verify that YARN is up and running by going to <http://localhost:8088>. This is the YARN UI.

### Build a Samza Job Package

Before you can run a Samza job, you need to build a package for it. This package is what YARN uses to deploy your jobs on the grid.

**(Optional)** NOTE: if you want the hello-samza jobs to run with a local Samza build (e.g., if you are a Samza developer), make sure that you run the following steps, otherwise skip them.

In your hello-samza project,

git checkout latest

In your local Samza project,

./gradlew publishToMavenLocal

Then, you can continue w/ the following command in hello-samza project:

mvn clean package

mkdir -p deploy/samza

tar -xvf ./target/hello-samza-1.9.0-SNAPSHOT-dist.tar.gz -C deploy/samza

### Run a Samza Job

After you’ve built your Samza package, you can start a job on the grid using the run-app.sh script.

deploy/samza/bin/run-app.sh --config-path=$PWD/deploy/samza/config/wikipedia-feed.properties

The job will consume a feed of real-time edits from Wikipedia, and produce them to a Kafka topic called “wikipedia-raw”. Give the job a minute to startup, and then tail the Kafka topic:

deploy/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic wikipedia-raw

Pretty neat, right? Now, check out the YARN UI again (<http://localhost:8088>). This time around, you’ll see your Samza job is running!

If you can not see any output from Kafka consumer, you may have connection problem. Check [here](https://samza.apache.org/learn/tutorials/latest/run-hello-samza-without-internet.html).

### Generate Wikipedia Statistics

Let’s calculate some statistics based on the messages in the wikipedia-raw topic. Start two more jobs:

deploy/samza/bin/run-app.sh --config-path=$PWD/deploy/samza/config/wikipedia-parser.properties

deploy/samza/bin/run-app.sh --config-path=$PWD/deploy/samza/config/wikipedia-stats.properties

The first job (wikipedia-parser) parses the messages in wikipedia-raw, and extracts information about the size of the edit, who made the change, etc. You can take a look at its output with:

deploy/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic wikipedia-edits

The last job (wikipedia-stats) reads messages from the wikipedia-edits topic, and calculates counts, every ten seconds, for all edits that were made during that window. It outputs these counts to the wikipedia-stats topic.

deploy/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic wikipedia-stats

The messages in the stats topic look like this:

{"is-talk":2,"bytes-added":5276,"edits":13,"unique-titles":13}

{"is-bot-edit":1,"is-talk":3,"bytes-added":4211,"edits":30,"unique-titles":30,"is-unpatrolled":1,"is-new":2,"is-minor":7}

{"bytes-added":3180,"edits":19,"unique-titles":19,"is-unpatrolled":1,"is-new":1,"is-minor":3}

{"bytes-added":2218,"edits":18,"unique-titles":18,"is-unpatrolled":2,"is-new":2,"is-minor":3}

If you check the YARN UI, again, you’ll see that all three jobs are now listed.

### Shutdown

To shutdown one of the jobs, use the same script with an extra ‘–operation=kill’ argument

deploy/samza/bin/run-app.sh --config-path=$PWD/deploy/samza/config/wikipedia-feed.properties --operation=kill

After you’re done, you can clean everything up using the same grid script.

bin/grid stop all

Congratulations! You’ve now setup a local grid that includes YARN, Kafka, and ZooKeeper, and run a Samza job on it. Next up, check out the [Background](https://samza.apache.org/learn/documentation/latest/introduction/background.html) and [API Overview](https://samza.apache.org/learn/documentation/latest/api/overview.html) pages.

**Comparing .Net and Vertx**

<https://www.stackshare.io/stackups/dot-net-vs-vert-x>

# .NET vs Vert.x: What are the differences?

### Introduction

In the world of web development, .NET and Vert.x are both popular frameworks that offer different features and capabilities. Understanding the key differences between these two can help developers make informed decisions when choosing the right technology for their projects.

1. **\*\*Language Support\*\***: .NET primarily supports C# and F# while Vert.x supports Java, JavaScript, Groovy, Ruby, Scala, and Kotlin. This difference in language support can influence developers' choices based on their expertise and project requirements.

2. **\*\*Concurrency Model\*\***: .NET relies on a multi-threaded model for handling concurrency, whereas Vert.x uses an event loop-based model. The event loop model in Vert.x allows for high concurrency and non-blocking operations, making it more suitable for highly scalable applications.

3. **\*\*Performance and Scalability\*\***: Vert.x is known for its lightweight and high-performance nature, making it a preferred choice for building reactive and highly scalable applications. On the other hand, .NET, while powerful, may require more resources for similar scalability and performance levels.

4. **\*\*Community and Ecosystem\*\***: .NET has a strong community and a vast ecosystem of tools, libraries, and support due to its long-standing presence in the industry. Vert.x, although gaining popularity, may have a smaller community and ecosystem in comparison, impacting the availability of resources and community-driven support.

5. **\*\*Deployment Flexibility\*\***: .NET applications are typically deployed on Windows servers, while Vert.x allows for more flexibility in deployment options, including containers, cloud platforms, and different operating systems. This gives Vert.x an edge in terms of deployment flexibility and compatibility with diverse environments.

### Summary

In summary, .NET and Vert.x differ in language support, concurrency models, performance, community size, deployment flexibility, and ecosystem, offering developers a range of options based on their specific project requirements.

**Decisions about .NET and Vert.x**

Ing. Alvaro Rodríguez Scelza

**Ing. Alvaro Rodríguez Scelza**

Software Systems Engineer at Ripio · [May 21, 2020 | 9 upvotes · 504.4K views](https://www.stackshare.io/loaderon/decisions/104204597162612663)

**Chose**

[.NET](https://www.stackshare.io/dot-net)

over

[Django](https://www.stackshare.io/django)[Python](https://www.stackshare.io/python)

in

**[complete-knowledge-stack](https://www.stackshare.io/loaderon/complete-knowledge-stack" \t "_blank)**

Decided to change all my stack to microsoft technologies for they behave just great together. It is very easy to set up and deploy projects using visual studio and azure. Visual studio is also an amazing IDE, if not the best, when used for C#, it allows you to work in every aspect of your software.

Visual studio templates for ASP.NET MVC are the best I've found compared to django, rails, laravel, and others.

See more

**Manage your open source components, licenses, and vulnerabilities**

[Learn More](https://fossa.com/fossa-developers)

**Pros of .NET**

**Pros of Vert.x**

* **272**

Tight integration with visual studio

* **261**

Stable code

* **190**

Great community

* **182**

Reliable and strongly typed server side language.

* **140**

Microsoft

* **119**

Fantastic documentation

* **89**

Great 3rd party libraries

* **80**

Speedy

* **71**

Great azure integration

* **63**

Great support

* **34**

Highly productive

* **34**

Linq

* **34**

C#

* **31**

High Performance

* **28**

Great programming languages (C#, VB)

* **26**

Open source

* **19**

Powerful Web application framework (ASP.NET MVC)

* **16**

Fast

* **16**

Clean markup with razor

* **15**

Powerful ORM (EntityFramework)

* **14**

Dependency injection

* **10**

Visual studio + Resharper = <3

* **10**

Constantly improving to keep up with new trends

* **9**

High-Performance

* **8**

TFS

* **8**

Security

* **7**

Job opportunities

* **7**

Integrated and Reliable

* **7**

Huge ecosystem and communities

* **6**

Light-weight

* **6**

Lovely

* **5**

{get; set;}

* **5**

Variations

* **5**

Asynchrony

* **4**

Scaffolding

* **4**

Support and SImplicity

* **4**

Default Debuging tools

* **4**

Concurrent

* **4**

Useful IoC

* **4**

Entity framework

* **3**

Nuget package manager

* **3**

Blazor

* **2**

F♯

* **13**

Light weight

* **12**

Fast

* **8**

Java

* **6**

Developers Are Super

* **5**

Extensible

* **2**

Easy Socks.js integration

* **2**

Asynchronous

* **1**

Strong concurrency model

* **1**

Great tooling

* **1**

Easy integration

* **1**

Central Config (Redis)

* **1**

Good documentation

* **1**

Abstract data grid API

* **1**

Unopinionated

* **1**

Clustering Infrastructure

* **1**

Scalable

* **1**

Parallelism

* **1**

Actor-like model

# Sign up to add or upvote prosMake informed product decisions

Sign up now

**Cons of .NET**

**Cons of Vert.x**

* **13**

C#

* **12**

Too expensive to deploy and maintain

* **8**

Microsoft dependable systems

* **8**

Microsoft itself

* **5**

Hard learning curve

* **3**

Tight integration with visual studio

* **3**

Not have a full fledged visual studio for linux

* **1**

Microsoft itself 🤡🥲

* **2**

Steep Learning Curve

* **2**

Too Many Conflicting Versions And Suggestions

# Sign up to add or upvote consMake informed product decisions

Sign up now

14.5K

537.5K

338K

493

6

[21K](https://github.com/dotnet/core" \t "_blank)

[4.9K](https://github.com/dotnet/core" \t "_blank)

[6y](https://github.com/dotnet/core" \t "_blank)

*- No public GitHub repository available -*

## What is .NET?

.NET is a general purpose development platform. With .NET, you can use multiple languages, editors, and libraries to build native applications for web, mobile, desktop, gaming, and IoT for Windows, macOS, Linux, Android, and more.

## What is Vert.x?

It is event driven and non blocking application framework. This means your app can handle a lot of concurrency using a small number of kernel threads. It lets your app scale with minimal hardware.

<https://www.codereliant.io/battle-of-the-frameworks-benchmarking-high-performance-http-libraries/>

Oct 23, 2023 7 min read [Performance](https://www.codereliant.io/tag/performance/)

# Battle of the Frameworks: Benchmarking High-Performance HTTP Libraries



Photo by [Riccardo Pierri](https://unsplash.com/@rieppi?utm_source=ghost&utm_medium=referral&utm_campaign=api-credit) / [Unsplash](https://unsplash.com/?utm_source=ghost&utm_medium=referral&utm_campaign=api-credit)

Have you ever wondered about how high performance HTTP servers will do across different languages? In this post we will try to compare many http frameworks from different languages: Nodejs, Java, C#, and Go.

The world of web development is filled with endless options for HTTP frameworks across programming languages. But with so many choices, how do developers know which ones deliver truly high-performance results? In this blog post, we'll cut through the noise and directly compare some of the top contenders for speed. Looking at popular options in Javascript/Bun, Java, C#, Go, and rust, we will benchmark and evaluate their throughput and response times when put to the test. The frameworks we've selected have a reputation for performance, but we'll see how they stack up across languages.

* **Java** 21 + [**vertex**](https://vertx.io/?ref=codereliant.io) 4.4.6
* **JS/Bun** 1.0.6 + [**elysiajs**](https://elysiajs.com/?ref=codereliant.io) 0.7
* **C#** 12 + [**dotnet**/**ASP.NET**](https://dotnet.microsoft.com/en-us/?ref=codereliant.io) 8.0 RC2
* **Go** 1.21.3 + [**fiber**](https://gofiber.io/?ref=codereliant.io) 2.49.2
* **Rust** 1.73.0 + [**actix-web**](https://actix.rs/?ref=codereliant.io) 4

By stress testing them, we'll get hard data on their capabilities. This head-to-head comparison focuses on raw speed and scalability with **zero tuning**, so you can pick the right framework for your next high-traffic web project. Whether you're looking to turbo-charge an API, build lower-latency systems, or squeeze the most out of your servers, this evaluation aims to help you choose a high-performant HTTP framework tailored to your tech stack.

Join our weekly Reliability Newsletter!

Top of Form

Bottom of Form

Top of Form

Bottom of Form

## Environment & Test Set up:

The test will be done by spinning the minimal version of the HTTP server that return a hello world response when pining / .

We will run server on a [Hetzner](https://hetzner.com/?ref=codereliant.io) Machine:

* **OS**: Ubuntu 22.04.3 LTS
* **Kernel**: 5.15.0-86-generic
* **Arch**: ARM aarch64
* **Resources**: 4 vcpu & 8 GB Ram

The **client** load generator will be on a separate machine that has similar properties except resources will be higher 8 vcpu & 16 GB Ram.

### Java & Vertex:

We will generate the vertex starter project using the [starter site](https://start.vertx.io/?ref=codereliant.io) from vertex. Also, we will use java 21-oracle that we just installed using [sdkman](https://sdkman.io/?ref=codereliant.io).

package io.codereliant.performance;

import io.vertx.core.AbstractVerticle;

import io.vertx.core.Promise;

public class MainVerticle extends AbstractVerticle {

@Override

public void start(Promise<Void> startPromise) throws Exception {

vertx.createHttpServer().requestHandler(req -> {

req.response()

.putHeader("content-type", "text/plain")

.end("Hello World!");

}).listen(80, http -> {

if (http.succeeded()) {

startPromise.complete();

System.out.println("HTTP server started on port 80");

} else {

startPromise.fail(http.cause());

}

});

}

}

After just updating the port to 80 instead of 8888 and modify the string return to be Hello World instead of the default text; we build our server using mvn package and run it using java -jar target/performance-1.0.0-SNAPSHOT-fat.jar.

java -jar target/performance-1.0.0-SNAPSHOT-fat.jar

HTTP server started on port 80

Oct 15, 2023 10:56:15 PM io.vertx.core.impl.launcher.commands.VertxIsolatedDeployer

INFO: Succeeded in deploying verticle

### Bun & Elysia:

Elysia makes it easier to create a project by just invoking bun create elysia perf-app, then invoke bun run index.ts.

For reference we are using Bun 1.0.6 and Elysia 0.7.0.

import { Elysia } from "elysia";

const app = new Elysia().get("/", () => "Hello World").listen(80);

console.log(

`🦊 Elysia is running at ${app.server?.hostname}:${app.server?.port}`

);

Create Elysia Perf app using Bun

### C# & Dotnet:

For C# 12 and dotnet 8.0 RC, we will use ASP.NET Core [Minimal APIs](https://learn.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis?view=aspnetcore-8.0&ref=codereliant.io).

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.MapGet("/", () => "Hello World!");

app.Run();

dotnet new web -o perf-app

cd perf-app

dotnet build -c Release

ASPNETCORE\_URLS="http://\*:80/" ./bin/Release/net8.0/dotnet-app

### Golang & Fiber:

[Fiber](https://gofiber.io/?ref=codereliant.io) is a Go web framework built on top of Fasthttp, which was designed for high performance.

mkdir go-app

cd go-app

go mod init github.com/codereliant/go-app

go get -u github.com/gofiber/fiber/v2

touch main.go

Then in main.go, we will use the hello world example from teh landing page of [https://gofiber.io/](https://gofiber.io/?ref=codereliant.io) .

package main

import (

"log"

"github.com/gofiber/fiber/v2"

)

func main() {

app := fiber.New()

app.Get("/", func (c \*fiber.Ctx) error {

return c.SendString("Hello, World!")

})

log.Fatal(app.Listen(":80"))

}

main.go

for building and running the example, you can use these 2 lines:

go build main.go

./main

### Rust & Actix-web:

From the main site of actix-web:

Actix Web is a powerful, pragmatic, and extremely fast web framework for Rust

We can create a project using:

cargo new actix-hello

cd actix-hello

Then replacing `src/main.rs` content with the content below, which is taking from actix-web [getting started](https://actix.rs/docs/getting-started?ref=codereliant.io) page:

use actix\_web::{web, App, HttpResponse, HttpServer, Responder};

async fn manual\_hello() -> impl Responder {

HttpResponse::Ok().body("Hey there!")

}

#[actix\_web::main]

async fn main() -> std::io::Result<()> {

HttpServer::new(|| {

App::new()

.route("/", web::get().to(manual\_hello))

})

.bind(("0.0.0.0", 80))?

.run()

.await

}

Then we just build the binary and run it:

cargo build --release

# .......

# Finished release [optimized] target(s) in 1m 58s

./target/release/actix-hello

🔥

Elevate your Linux performance expertise! Our [cheat sheet](https://www.codereliant.io/linux-performance-cli-tools/) is a must-have for every SRE and Software Engineer. Grab yours for [free](https://www.codereliant.io/linux-performance-cli-tools/)!

### Client:

For the client we will use [oha](https://github.com/hatoo/oha?ref=codereliant.io) an HTTP benchmarking tool written in rust and inspired by [Hey](https://github.com/rakyll/hey?ref=codereliant.io).

For how to install [oha](https://github.com/hatoo/oha?ref=codereliant.io) you can check its installation steps on the [github page](https://github.com/hatoo/oha?ref=codereliant.io#installation).

We will run the bombardier with 500 connections for a total of 3 millions requests, we will repeat this experiment 3 times.

oha -c 500 -n 3000000 http://perf-experiment-host/

The response should look something like this:



## Results:

For each of the language + framework option above we benchmarked the application 3 times, then we grabbed the best data to compare.

### Throughput:

Req/s by language/Framework

1. **Rust/Actix-web** leads the pack with the highest throughput, closely followed by **Go/fiber**.
2. **C#/ASP.NET**, despite its popularity, lags behind the top performers in this benchmark.
3. **Java/Vertex** and **Bun/Elysia** demonstrate comparable and mid-range throughput values.

### Latency:

Latencies comparison by framework

The graph showcases the latencies of various HTTP frameworks across three different metrics:

1. **99.9% Latency**: Represents the latency at which 99.9% of the requests are processed.
2. **99% Latency**: Represents the latency at which 99% of the requests are processed.
3. **Mean Latency**: Represents the average latency for all requests.

From the graph, a few observations can be made:

* **Rust/Actix-web** and **Go/fiber** not only display impressive throughput but also exhibit lower latencies across all three metrics.
* **C#/ASP.NET** demonstrates relatively lower mean latency compared to some other frameworks, despite its lower throughput.
* There's a noticeable difference between the 99% and 99.9% latencies, highlighting the variability of request processing times within the frameworks.

## Conclusion:

In our exploration of the performance of various HTTP frameworks, **Rust/Actix-web** is obviously a clear winner, getting not only the highest throughput but also super low latencies across all metrics. Following closely was **Go/fiber**, which impressed with its blend of high request processing speeds and efficient response times. While **C#/ASP.NET** might not have matched the top 2 in throughput, its mean latency was notably competitive, suggesting that it remains a viable choice for many applications. On the other hand, frameworks like **Java/Vertex** and **Bun/Elysia** presented okay performance and they might probably need more tuning to compete with other frameworks.

We are sure that these numbers and comparison might have looked different if we spent some time tuning the language/frameworks with all of their bells and whistles; however, we wanted this comparison to be done with **zero tuning**.

<https://www.techempower.com/benchmarks/#hw=ph&test=plaintext&section=data-r22&l=yykha5-cn3&p=zik0zi-zdjvnj-zijocf-zik0vz-zih7un-sf>









**A quick discussion on choosing between vert.x and spring boot**

<https://www.reddit.com/r/java/comments/jik6io/should_i_have_any_reason_to_prefer_vertx_over/>

**In love with vert.x or what ?**

<https://kislayverma.com/programming/why-i-like-the-vert-x-framework/>

**Going Deep(er) into VueJS + Vertx**

**Responsiveness with load and chaos testing**

1 – Simulating users with locust

2 – Load testing endpoints with Hey or Wrk

3 – Chaos testing with Pumba

4 – Mitigating failures with explicit timeouts, circuit breakers and caches

**Pre-Locust Test warmup**

The Java Virtual Machine (JVM) running the various services would need to have some workload before it can start to run code efficiently. After that, you can run a bigger workload to get a first estimation of how your services are faring.

**Cons of Locust**

1- Locust’s network stack is not very efficient, so we quickly reach limits in the number of concurrent users.

2 - Like many load testing tools, Locust suffers from coordinated omission, a problem where time measures are incorrect due to ignoring the wait time before the

requests are actually made

**How NOT to measure latency**

www.youtube.com/watch?v=lJ8ydIuPFeU

**The difference btwn Service time and Response time**

**Sustainable throughput**

**Latency at saturation**

**Comparing behavior under different throughputs and/or configurations**

**Problem decomposition**

**Load Testing Visualisation**

**High Level Chaos Testing**

**Using a circuit breaker**

A circuit breaker is a very useful tool for avoiding cascading failures, but you don’t have to wrap every operation over the network in a circuit breaker. Every abstraction has a cost, and circuit breakers do add a level of indirection. Instead, it is best to use chaos testing and identify where they are most likely to have a positive effect on the overall system behavior.

We now have a reactive service: it is not just resource-efficient and scalable, but is also

resilient to failures. The service keeps responding in all situations, and the latency is

kept under control.

Fundamentally :

A reactive service must not just be scalable, but also resilient and responsive.

Load testing and chaos testing tools are super critical ad key to analyzing the service behaviour during nominal operation conditions and when inevitably stymied by failures as a result of network and services it relies on/upon.

Circuit breakers are the most efficient tools for shielding a service from unresponsive services and network failures.

A resilient service is not just responsive when it can quickly notify of an error; it may still be able to respond successfully, such as using cached data if the application domain allows it.

**Final Notes / Conclusion**

Efficiently building container images with jib

Configuring vert.x clustering to work in a Kubernetes cluster

Deploying vert.x services to a Kubernetes cluster

Using Skaffold and Minikube for local development

Exposing health checks and metrics

**Deploying to a local cluster**

**Minikubes**

https://minikube.sigs.k8s.io/docs/

**Creating a minikube cluster**

$ minikube start --cpus=4 --memory=8G --addons ingress

minikube v1.9.2 on Darwin 10.15.4

MINIKUBE\_ACTIVE\_DOCKERD=minikube

Automatically selected the hyperkit driver. Other choices:

➥ docker, virtualbox

Starting control plane node m01 in cluster minikube

Creating hyperkit VM (CPUs=4, Memory=8192MB, Disk=20000MB) ...

Preparing Kubernetes v1.14.0 on Docker 19.03.8 ...

Enabling addons: default-storageclass, ingress, storage-provisioner

Done! kubectl is now configured to use "minikube"

**Skaffold-ing**

apiVersion: skaffold/v1

kind: Config

metadata:

name: chapter13

build:

artifacts:

- image: vertx-in-action/heat-sensor-service

jib:

type: gradle

project: heat-sensor-service

context: .

- image: vertx-in-action/sensor-gateway

jib:

type: gradle

project: sensor-gateway

context: .

- image: vertx-in-action/heat-api

jib:

type: gradle

project: heat-api

context: .

deploy:

kubectl:

manifests:

- "\*\*/k8s/\*.yaml"

**Checking the exposed services**

$ minikube tunnel

$ kubectl get services

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| NAME | TYPE | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE |
| heat-api | LoadBalancer | 10.103.127.60 | 10.103.127.60 | 8080:31673/TCP | 102s |
| Heat-sensor-service | ClusterIP | None | <none> | 8080/TCP,5701/TCP | 102s |
| Kubernetes | ClusterIP | 10.96.0.1 | <none> | 443/TCP | 42m |
| Sensor-gateway | ClusterIP | 10.108.31.235 | <none> | 8080/TCP,5701/TCP | 102s |

$ minikube service heat-api --url

http://192.168.64.12:31673

**Interacting with exposed services**

$ http 10.103.127.60:8080/all

HTTP/1.1 200 OK

Content-Type: application/json

content-length: 402

<JSON DATA>

$ http 10.103.127.60:8080/warnings

HTTP/1.1 200 OK

Content-Type: application/json

content-length: 11

<JSON DATA>

**Interacting with service gateways**

$ kubectl port-forward services/sensor-gateway 8080

$ http :8080/data

HTTP/1.1 200 OK

Content-Type: application/json

content-length: 400

<JSON data>

The kubectl port-forward command must be run in another terminal, and as long

as it is running, the local port 8080 forwards to the sensor gateway service inside the

cluster. This is very convenient for accessing anything that is running in the cluster

without being exposed as a LoadBalancer service.

**DNS query to discover headless application services**

$ kubectl run --image tutum/dnsutils dns -it --rm -- bash

root@dns:/# dig +short heat-sensor-service.default.svc.cluster.local

172.17.0.8

172.17.0.12

172.17.0.11

172.17.0.9

root@dns:/#

**Replication in Kubernetes k8s :**

**Increasing the number of application service replicas**

$ kubectl scale deployment/heat-sensor-service --replicas 5

$ kubectl run --image tutum/dnsutils dns -it --rm -- bash

root@dns:/# dig +short heat-sensor-service.default.svc.cluster.local

172.17.0.11

172.17.0.12

172.17.0.8

172.17.0.13

172.17.0.9

root@dns:/#

Also, if we make HTTP requests like in listing 13.14, we can see that we have data from

five sensors.

Now that we have deployed the services and interacted with them, let’s look at how

deployment in Kubernetes works for Vert.x services.

**Making the services work in Kubernetes**

**Building Container Images**

**Container image layers with jib**

Jib offers Maven and Gradle plugins, and it builds container images by deriving information from a project. Jib is also great because it is purely written in Java and it does not need Docker to build images, so you can produce container images without any third-party tools. It can also publish container images to registries and Docker daemons, which is useful in development.

**Configuring the jib gradle plugin**

jib {

from {

image = "adoptopenjdk/openjdk11:ubi-minimal-jre"

}

to {

image = "vertx-in-action/heat-sensor"

tags = setOf("v1", "latest")

}

container {

mainClass = "chapter13.sensor.HeatSensor"

jvmFlags = listOf("-noverify",

➥ "-Djava.security.egd=file:/dev/./urandom")

ports = listOf("8080", "5701")

user = "nobody:nobody"

}

}

**Building a service container image to a Docker daemon**

**Clustering and Kubernetes**

**Kubernetes configuration for Hazelcast discovery**

The DNS approach is more limited.

Instead, let’s use the Kubernetes API and configure Hazelcast to use it. By default, the

Hazelcast Vert.x cluster manager reads configuration from a cluster.xml resource. The

following listing shows the relevant configuration excerpt of the

heat-sensor-service/src/main/resource/cluster.xml file.

(...)

<join>

<multicast enabled="false"/>

<tcp-ip enabled="false" />

<discovery-strategies>

<discovery-strategy enabled="true"

class="com.hazelcast.kubernetes.HazelcastKubernetesDiscoveryStrategy">

<properties>

<property name="service-label-name">vertx-in-action</property>

<property name="service-label-value">chapter13</property>

</properties>

</discovery-strategy>

</discovery-strategies>

</join>

(...)

**Role-based access control ( RBAC ) to grant view access to the Kubernetes API**

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: default-cluster

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: view

subjects:

- kind: ServiceAccount

name: default

namespace: default

**Service/Application deployment descriptor**

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: heat-sensor-service

name: heat-sensor-service

spec:

selector:

matchLabels:

app: heat-sensor-service

replicas: 4

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

template:

metadata:

labels:

app: heat-sensor-service

spec:

containers:

- image: vertx-in-action/heat-sensor-service:latest

name: heat-sensor-service

**Application or Service definition**

apiVersion: v1

kind: Service

metadata:

labels:

app: heat-sensor-service

vertx-in-action: chapter13

name: heat-sensor-service

spec:

clusterIP: None

selector:

app: heat-sensor-service

ports:

- name: http

port: 8080

- name: hazelcast

port: 5701

**Application or Service gateway deployment descriptor**

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: sensor-gateway

name: sensor-gateway

spec:

selector:

matchLabels:

app: sensor-gateway

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

template:

metadata:

labels:

app: sensor-gateway

spec:

containers:

- image: vertx-in-action/sensor-gateway:latest

name: sensor-gateway

**Application or Service gateway definition**

apiVersion: v1

kind: Service

metadata:

labels:

app: sensor-gateway

vertx-in-action: chapter13

name: sensor-gateway

spec:

type: ClusterIP

selector:

app: sensor-gateway

ports:

- name: http

port: 8080

- name: hazelcast

port: 5701

**Service API deployment excerpt**

spec:

containers:

- image: vertx-in-action/heat-api:latest

name: heat-api

env:

- name: LOW\_TEMP

value: "12.0"

- name: HIGH\_TEMP

value: "32.0"

- name: GATEWAY\_HOST

valueFrom:

configMapKeyRef:

name: sensor-gateway-config

key: gateway\_hostname

- name: GATEWAY\_PORT

valueFrom:

configMapKeyRef:

name: sensor-gateway-config

key: gateway\_port

**Configuration map example**

apiVersion: v1

kind: ConfigMap

metadata:

name: sensor-gateway-config

data:

gateway\_hostname: sensor-gateway.default.svc.cluster.local

gateway\_port: "8080"

**API Service definition**

apiVersion: v1

kind: Service

metadata:

labels:

app: heat-api

name: heat-api

spec:

type: LoadBalancer

selector:

app: heat-api

ports:

- name: http

port: 8080

**Exposing health, readiness checks and metrics**

**Liveness checks** : service reports if it is working correctly, or if it is failing and needs to be restarted.

**Readiness checks** : service reports that it is ready to accept traffic.

**Simple HTTP health-check probe**

// In the verticle start method:

router.get("/health").handler(this::healthCheck);

// (...)

private final JsonObject okStatus = new JsonObject().put("status", "UP");

private void healthCheck(RoutingContext ctx) {

logger.info("Health check");

ctx.response()

.putHeader("Content-Type", "application/json")

.end(okStatus.encode());

}

**Application or Service liveness probe**

# (...)

spec:

containers:

- image: vertx-in-action/heat-sensor-service:latest

name: heat-sensor-service

livenessProbe:

httpGet:

path: /health

port: 8080

initialDelaySeconds: 15

periodSeconds: 15

timeoutSeconds: 5

**Health checks in logs**

$ kubectl logs -f heat-sensor-service-6944f78b84-2tpnx | grep

➥ 'Health check'

2024-05-02 17:27:54,218 INFO [vert.x-eventloop-thread-1]

➥ chapter13.sensor.HeatSensor - Health check

2024-05-02 17:28:09,182 INFO [vert.x-eventloop-thread-1]

➥ chapter13.sensor.HeatSensor - Health check

2024-05-02 17:28:24,181 INFO [vert.x-eventloop-thread-1]

➥ chapter13.sensor.HeatSensor - Health check

2024-05-02 17:28:39,182 INFO [vert.x-eventloop-thread-1]

➥ chapter13.sensor.HeatSensor - Health check

**Health-check routes of the application or service API service**

router.get("/health/ready").handler(this::readinessCheck);

router.get("/health/live").handler(this::livenessCheck);

**Readiness check of the application or service API service**

private void readinessCheck(RoutingContext ctx) {

webClient.get("/health")

.expect(ResponsePredicate.SC\_OK)

.timeout(5000)

.send(ar -> {

if (ar.succeeded()) {

logger.info("Readiness check complete");

ctx.response().setStatusCode(200)

.putHeader("Content-Type","application/json")

.end(okStatus.encode());

} else {

logger.error("Readiness check failed", ar.cause());

ctx.response().setStatusCode(503)

.putHeader("Content-Type","application/json")

.end(new JsonObject()

.put("status", "DOWN")

.put("reason", ar.cause().getMessage()).encode());

}

});

}

**Configuring health checks for the application API service**

# (...)

spec:

containers:

- image: vertx-in-action/heat-api:latest

name: heat-api

# (...)

livenessProbe:

httpGet:

path: /health/live

port: 8080

initialDelaySeconds: 1

periodSeconds: 15

timeoutSeconds: 5

readinessProbe:

httpGet:

path: /health/ready

port: 8080

initialDelaySeconds: 5

periodSeconds: 10

timeoutSeconds: 5

**Scaling down the sensor gateway to 0 replicas**

$ kubectl scale deployment/sensor-gateway --replicas 0

deployment.extensions/sensor-gateway scaled

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

heat-api-5dbcc84795-ccb8d 0/1 Running 0 55m

heat-sensor-service-6946bc8f6f-2k7lv 1/1 Running 0 55m

heat-sensor-service-6946bc8f6f-d9hd8 1/1 Running 0 55m

heat-sensor-service-6946bc8f6f-rhdbg 1/1 Running 0 55m

heat-sensor-service-6946bc8f6f-xd28p 1/1 Running 0 55m

**Scaling up the sensor gateway to one replica**

$ kubectl scale deployment/sensor-gateway --replicas 1

deployment.extensions/sensor-gateway scaled

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

heat-api-5dbcc84795-ccb8d 1/1 Running 0 63m

heat-sensor-service-6946bc8f6f-2k7lv 1/1 Running 0 63m

heat-sensor-service-6946bc8f6f-d9hd8 1/1 Running 0 63m

heat-sensor-service-6946bc8f6f-rhdbg 1/1 Running 0 63m

heat-sensor-service-6946bc8f6f-xd28p 1/1 Running 0 63m

sensor-gateway-6b7cd8bbcb-btl4k 1/1 Running 0 2m18s

**Metrics**

Vert.x can be configured to report metrics on various items such as :

Event-bus communications,

Network communications, et. al.

We can decide to trigger alerts if some metrics is not met nor received nor measured as expected.

We could also have an alert that causes Kubernetes to sacles up a service when the throughput or latency of a given URL endpoint is above a certain threshold.

Other topics to be considered in metrics :

* + - 1. – visualization
      2. – alerting
      3. – auto-scaling

Vert.x exposes metrics over popular technologies such as :

JMX

Dropwizard

Jolokia

[Micrometer](https://micrometer.io/)

[Prometheus](https://prometheus.io/)

**Adding metrics support**

Vert.x micrometer support

implementation("io.vertx:vertx-micrometer-metrics:$vertxVersion")

implementation("io.micrometer:micrometer-registry-prometheus:$mpromVersion")

The sensor gateway needs clustering and metrics when starting Vert.x from the main

method. We need to enable metrics as follows.

**Enabling Micrometer/Prometheus metrics**

**Exposing a metrics endpoint over HTTP**

**May be just IDE trivial :**

So in the way I like to use my IDE (eg. InteliJ) is fundamentally that I have taken time to define different types of elements in the said programming language by certain specific colours. Now with all of the above I shared so far on my progress with Vert.x, the colours were not coming up . . . everything was just plain white.



In order to get the colours to show, after trying to set the project properties, or try to set the compiler levels and or types, none of those helped until , I added a pom to the root of my project and defined the respective modules by name. After adding the pom file to identifying each of the elements as expected. the root, just like magic, the IDE suddenly seems to have recovered and then started



**Site Reliability Engineering ( S.R.E ) with java microservices**

**Intrusion Detection System ( IDS )**

IDS Resources:

snort.org

tcpdump.org

[www.sans.org/dosstep/cidr.php](http://www.sans.org/dosstep/cidr.php).

**References**

HAProxy Crash Course : https://www.youtube.com/@hnasr

Quick Links :

<https://github.com/containers/buildah>

<https://github.com/GoogleContainerTools/jib>

<https://github.com/wagoodman/dive>

<https://github.com/derailed/k9s>

### **The Ultimate List of Types of Testing:**

Let’s see different types of Testing one by one.

**1. Functional testing**: In simple words, what the system actually does is functional testing. To verify that each function of the software application behaves as specified in the requirement document. Testing all the functionalities by providing appropriate input to verify whether the actual output is matching the expected output or not. It falls within the scope of black-box testing and the testers need not concern about the source code of the application.

**2. Non-functional testing**: In simple words, how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility, etc., Main focus is to improve the user experience on how fast the system responds to a request.

**3. Manual testing**: Manual testing is the process of testing the software manually to find the defects. A tester should have the perspective of an end-user and to ensure all the features are working as mentioned in the requirement document. In this process, testers execute the [test cases](https://www.softwaretestingmaterial.com/test-case-template-with-explanation/)and generate the reports manually without using any automation tools.

**4. Automated testing**: Automation testing is the process of testing the software using an automation tool to find the defects. In this process, executing the test scripts and generating the results are performed automatically by automation tools. Some most popular tools to do automation testing are HP QTP/UFT, [Selenium WebDriver](https://www.softwaretestingmaterial.com/install-selenium-webdriver/), etc.,

Learn the [Difference between Manual & Automated Testing](https://www.softwaretestingmaterial.com/manual-testing-vs-automation-testing/) here…

**5. Black box testing**: Black Box Testing is a [software testing](https://www.softwaretestingmaterial.com/software-testing/) method in which testers evaluate the functionality of the software under test without looking at the internal code structure. This can be applied to every level of software testing such as Unit, Integration, System and Acceptance Testing.

[Read more on black box testing here](https://www.softwaretestingmaterial.com/black-box-and-white-box-testing/)…

**6. Glass box testing:** The glass box testing is a methodology of testing that scans the program structure and then creates the test data based on the flow and logic of the program. This type of testing is mainly done by utilizing the internal programming logic to select the proper test data and to remove design errors.

The glass box testing is called open box testing, logic-driven testing, path driven testing, or clear box testing. The techniques of glass box testing are Path Coverage, Branch Coverage, and Statement Coverage.

**7. White box testing**: White Box Testing is also called as Glass Box, Clear Box, and Structural Testing. It is based on applications internal code structure. In white-box testing, an internal perspective of the system, as well as programming skills, are used to design test cases. This testing usually was done at the unit level.

[Click here for more details.](https://www.softwaretestingmaterial.com/black-box-and-white-box-testing/)

**8. Specification-based testing**: Specification-based testing is similar to behavior-driven testing and black-box testing. The testers perform this testing by viewing the application as a black box and they do not have any understanding of the internal logic or the flow of the program. Thus in short the testers are more concerned with the behavior of the application.

Specification-based testing includes both nonfunctional and functional testing. A specification may be in the form of a prototype, a written document, a group of use cases. It is considered a reference for building test data. The techniques for specification-based testing are Decision Table, Equivalence Partitioning, Boundary Value Analysis, and State Transitioning.

**9. Structure-based testing**:

The structure-based testing requires the technical know-how of the program logic and flow. This type of testing ensures that there is maximum test coverage and test design is proper. The structure-based testing is adopted in the initial phase of the project development to determine the amount of testing that is needed.

The structure-based testing also assists in creating some additional test cases to increase the test coverage, different from preexisting test cases. So it achieves more depth in testing. The techniques for structure-based testing are Path testing, Condition Testing, Multiple Condition Testing, Decision Testing, and Statement Testing.

**10. Gray box testing**: Grey box is the combination of both White Box and Black Box Testing. The tester who works on this type of testing needs to have access to design documents. This helps to create better test cases in this process.

**11. Unit testing**: Unit Testing is also called Module Testing or Component Testing. It is done to check whether the individual unit or module of the source code is working properly. It is done by the developers in the developer’s environment.

**12. Component testing:**Refer Unit Testing

**13. Module testing**: Refer Unit Testing

**14. Integration testing**: Integration Testing is the process of testing the interface between the two software units. Integration testing is done by multiple approaches such as Big Bang Approach, Top-Down Approach, Bottom-Up Approach, and Hybrid Integration approach.

[Integration Testing Complete Guide](https://www.softwaretestingmaterial.com/integration-testing/)

**15. System testing**: Testing the fully integrated application to evaluate the system’s compliance with its specified requirements is called System Testing AKA End to End testing. Verifying the completed system to ensure that the application works as intended or not.

**16. Acceptance testing**: It is also known as pre-production testing.  This is done by the end-users along with the testers to validate the functionality of the application. After successful acceptance testing. Formal testing conducted to determine whether an application is developed as per the requirement. It allows the customer to accept or reject the application. Types of acceptance testing are Alpha, Beta & Gamma.

**17. Big bang Integration Testing**: Combining all the modules once and verifying the functionality after completion of individual module testing.

Top-down and bottom-up are carried out by using dummy modules known as Stubs and Drivers. These Stubs and Drivers are used to stand-in for missing components to simulate data communication between modules.

**18. Top-down Integration Testing**: Testing takes place from top to bottom. High-level modules are tested first and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended. Stubs are used as a temporary module if a module is not ready for integration testing.

**19. Bottom-up Integration Testing**: It is a reciprocate of the Top-Down Approach. Testing takes place from bottom to up. Lowest level modules are tested first and then high-level modules and finally integrating the high-level modules to a low level to ensure the system is working as intended. Drivers are used as a temporary module for integration testing.

**20. Hybrid Integration Testing**: Hybrid integration testing is the combination of both Top-down and bottom-up integration testing.

**21. Alpha testing**: Alpha testing is done by the in-house developers (who developed the software) and testers. Sometimes alpha testing is done by the client or outsourcing team with the presence of developers or testers.

**22. Beta testing**: Beta testing is done by a limited number of end-users before delivery. Usually, it is done in the client’s place.

**23. Gamma Testing:** Gamma testing is done when the software is ready for release with specified requirements. It is done at the client’s place. It is done directly by skipping all the in-house testing activities.

**24. Equivalence partitioning testing**: Equivalence Partitioning is also known as Equivalence Class Partitioning. In equivalence partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior, so they are likely to be proposed in the same way. Hence selecting one input from each group to design the test cases.

[Read more on Equivalence Partitioning Testing Technique](https://www.softwaretestingmaterial.com/equivalence-partitioning-testing-technique/)…

**25. Boundary value analysis testing**: Boundary value analysis (BVA) is based on testing the boundary values of valid and invalid partitions. The Behavior at the edge of each equivalence partition is more likely to be incorrect than the behavior within the partition, so boundaries are an area where testing is likely to yield defects. Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition. A boundary value for a valid partition is a valid boundary value. Similarly, a boundary value for an invalid partition is an invalid boundary value.

[Read more on Boundary Value Analysis Testing Technique](https://www.softwaretestingmaterial.com/boundary-value-analysis-testing-technique/)…

**26. Decision tables testing**: Decision Table is aka Cause-Effect Table. This test technique is appropriate for functionalities which has logical relationships between inputs (if-else logic). In the Decision table technique, we deal with combinations of inputs. To identify the test cases with a decision table, we consider conditions and actions. We take conditions as inputs and actions as outputs.

[Read more on the Decision Table Testing Technique](https://www.softwaretestingmaterial.com/decision-table-test-design-technique/)…

**27. Cause-effect graph testing:** The cause-effect graph testing is a test case development methodology that begins with a collection of requirements and then identifies the optimal number of test cases needed to achieve maximum coverage with minimum time and cost.

**28. State transition testing**: Using state transition testing, we pick test cases from an application where we need to test different system transitions. We can apply this when an application gives a different output for the same input, depending on what has happened in the earlier state.

[Read more on State Transition Test Design Technique](https://www.softwaretestingmaterial.com/state-transition-test-design-technique/)…

**29.** **Exhaustive Testing:** Testing all the functionalities using all valid and invalid inputs and preconditions is known as Exhaustive testing.

**30. Early Testing:** Defects detected in the early phases of SDLC are less expensive to fix. So conducting early testing reduces the cost of fixing defects.

**31. Use case testing**: Use case testing is carried out with the help of a use case document. It is done to identify test scenarios to test end to end testing

**32. Scenario testing**: Scenario testing is a software testing technique that is based on a scenario. It involves converting business requirements to test scenarios for better understanding and achieve an end to end testing. A well-designed scenario should be motivating, credible, complex, and the outcome of which is easy to evaluate.

**33. Documentation testing**: Documentation testing is done to validate the documented artifacts such as requirements, test plan, traceability matrix, test cases.

**34. Statement coverage testing**: Statement coverage testing is a white box testing technique which is to validate whether each and every statement in the code is executed at least once.

**35. Decision coverage testing/branch coverage testing**: Decision coverage or branch coverage testing is a white box testing technique which is to validate every possible branch is executed at least once.

**36. Path testing**: Path coverage testing is a white box testing technique which is to validate that all the paths of the program are executed at least once.

**37. Mutation testing**: Mutation testing is a type of white box testing which is to change (mutate) certain statements in the source code and verify if the tests are able to find the errors.

**38. Loop testing**: Loop testing is a white box testing technique which is to validate a different kind of loops such as simple loops, nested loops, concatenated loops, and unstructured loops.

**39. Performance testing**: This type of testing determines or validates the speed, scalability, and/or stability characteristics of the system or application under test. Performance is concerned with achieving response times, throughput, and resource-utilization levels that meet the performance objectives for the project or product.

**40. Load testing**: It is to verify that the system/application can handle the expected number of transactions and to verify the system/application behavior under both normal and peak load conditions.

**41. Stress testing**: It is to verify the behavior of the system once the load increases more than its design expectations.

**42. Soak testing**: Running a system at high load for a prolonged period of time to identify the performance problems is called Soak Testing.

**43. Endurance testing**: Endurance testing is a kind of nonfunctional testing. It is also called soak testing. An application is given a considerable load for a considerable duration to check its behavior and performance under such a condition.

**44. Stability testing**: Stability testing is a testing methodology used to check the capacity of the application to do the required actions under a specific state or stress. It is a type of non-functional testing and is used to detect performance bugs.

Stability testing is sometimes called endurance testing. It is an optional testing technique used to verify if the application can perform uninterruptedly for a specific duration of time with a significant number of users and stress. It also checks memory leaks or other issues that degrade the stability of the application.

**45. Scalability Testing:** Scalability testing is a type of non-functional testing. It is to determine how the application under test scales with the increasing workload.

**46. Volume testing**: It is to verify that the system/application can handle a large amount of data

**47. Robustness testing**: Robustness testing is a type of testing that is performed to validate the robustness of the application.

**48. Vulnerability testing**: Vulnerability testing is the process of identifying the vulnerabilities or weaknesses in the application.

**49. Adhoc testing**: Ad-hoc testing is quite opposite to the formal testing. It is an informal testing type. In Adhoc testing, testers randomly test the application without following any documents and test design techniques. This testing is primarily performed if the knowledge of testers in the application under test is very high. Testers randomly test the application without any test cases or any business requirement document.

**50. Exploratory testing**: Usually, this process will be carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements.

**51. Retesting**: To ensure that the defects which were found and posted in the earlier build were fixed or not in the current build. Say, Build 1.0 was released. The test team found some defects (Defect Id 1.0.1, 1.0.2) and posted. Build 1.1 was released, now testing the defects 1.0.1 and 1.0.2 in this build is retesting.

**52. Regression testing**: Repeated testing of an already tested program, after modification, to discover any defects introduced or uncovered as a result of the changes in the software being tested or in another related or unrelated software components.

**53. Smoke testing**: Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as “Day 0” check. It is done at the “build level”. It helps not to waste the testing time to simply testing the whole application when the key features don’t work or the key bugs have not been fixed yet.

**54. Sanity testing**: Sanity Testing is done during the release phase to check for the main functionalities of the application without going deeper. It is also called as a subset of Regression testing. It is done at the “release level”. At times due to release time constraints rigorous regression testing can’t be done to the build, sanity testing does that part by checking main functionalities.

**55. Dynamic testing**: Dynamic testing involves the execution of code. It validates the output with the expected outcome

**56. Static testing**: Static Testing involves in reviewing the documents to identify the defects in the early stages of SDLC.

**57. Monkey testing**: Perform abnormal action on the application deliberately in order to verify the stability of the application.

**58. Gorilla testing**: Gorilla testing is done by testers, sometimes developers also join hands with testers. It involves testing a system repeatedly to test the robustness of the system.

**59. Usability testing**: To verify whether the application is user-friendly or not and was comfortably used by an end-user or not. The main focus of this testing is to check whether the end-user can understand and operate the application easily or not. An application should be self-exploratory and must not require training to operate it.

**60. Accessibility testing**: Accessibility testing is a subset of usability testing. It aims to discover how easily people with disabilities (such as visual Impairments, Physical Impairment, Hearing Impairment, Cognitive Impairment, Learning Impairment) can use a system.

**61. Compatibility testing**: It is to deploy and check whether the application is working as expected in a different combination of environmental components.

**62. Configuration testing**: Configuration testing is the process of testing an application with each one of the supported hardware and software configurations to find out whether the application can work without any issues.

**63. Localization testing**: Localization is a process of adapting globalization software for a specific region or language by adding local specific components.

**64. Globalization testing**: Globalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes.

**65. Internationalization testing**– Refer Globalization testing

**66. Positive Testing:** It is to determine what system supposed to do. It helps to check whether the application is justifying the requirements or not.

**67. Negative testing**: It is to determine what system not supposed to do. It helps to find the defects from the software.

**68. Security testing**: Security testing is a process to determine whether the system protects data and maintains functionality as intended.

[Security Testing Complete Guide](https://www.softwaretestingmaterial.com/security-testing-tutorial/)

**69. Penetration testing**: Penetration testing is also known as pen testing. It is a type of security testing. It is performed to evaluate the security of the system.

[Penetration Testing Complete Guide](https://www.softwaretestingmaterial.com/penetration-testing-tutorial/)

**70. Database testing**: Database testing is done to validate the data in the UI is matched with the data stored in the database. It involves in checking the schema, tables, triggers etc., of the database.

**71. Bucket Testing:** Bucket testing is a method to compare two versions of an application against each other to determine which one performs better.

**72. A/B testing**: Refer Bucket Testing…

**73. Split testing**– Refer bucket testing…

**74. Reliability Testing:** Perform testing on the application continuously for a long period of time in order to verify the stability of the application

**75. Interface Testing:** Interface testing is performed to evaluate whether two intended modules pass data and communicate correctly to one another.

**76. Concurrency testing**: Concurrency testing means accessing the application at the same time by multiple users to ensure the stability of the system. This is mainly used to identify deadlock issues.

**77. Fuzz testing**: Fuzz testing is used to identify coding errors and security loopholes in an application. By inputting a massive amount of random data to the system in an attempt to make it crash to identify if anything breaks in the application.

**78. GUI Testing:** Graphical User Interface Testing is to test the interface between the application and the end user. Mainly testers concern about the appearance of the elements such as fonts and colors conforms to design specifications.

**79. API testing**: API stands for Application Programming Interface. API testing is a type of software testing that involves testing APIs using some tools like SOAPUI, PostMan.

**80. Agile testing**: Agile testing is a type of testing that involves following principles of agile software development methodology. In this agile testing, testing is conducted throughout the lifecycle of the continuously evolving project instead of being confined to a particular phase.

**81. End to end testing:**The end to end testing is a testing methodology to check if the flow of the software from beginning till the end is as per the expected result. It helps to determine the dependencies in the system and ensures there is no data loss or corruption while the interaction between multiple components.

While doing end to end testing, the key features like interaction among the database, other systems, network, and so on are tested and verified if they are happening as per expected results.

**82. Recovery testing**: Recovery testing is performed in order to determine how quickly the system can recover after the system crash or hardware failure. It comes under the type of non-functional testing.

**83. Risk-based testing**: Identify the modules or functionalities which are most likely cause failures and then testing those functionalities.

**84. Installation testing**: It is to check whether the application is successfully installed and it is working as expected after installation.

**85. Formal Testing:** It is a process where the testers test the application by having pre-planned procedures and proper documentation.

**86. Pilot testing**: Pilot testing is testing carried out under a real-time operating condition by the company in order to gain the confidence of the client

**87. Backend testing**: Backend testing is a testing technique for the database and server-side validation. It is often known as database testing. It is done to test if the entered data in the front end is stored and reflected in the database. The backend testing is used to prevent data truncation and loss.

The database testing can be nonfunctional [which deals with the performance of the database], functional [which deals with application characteristics from the backend] and structural [which deals with the testing of the database structure like tables, views, and so on]. Thus the backend testing deals with the back end items which are not viewable from the front end.

**88. Cross-browser testing**: Cross Browser Testing is a type of non-functional test which helps us to ensure that our website or web application works as expected in various web browsers.

[Read more on Cross Browser Testing](https://www.softwaretestingmaterial.com/what-is-cross-browser-testing/)…

**89. Browser compatibility testing**: The browser compatibility testing is an important part of the testing phase. It is done to verify the application in multiple web browsers. There must be sufficient resources allocated to carry out this testing.

The most essential points to check in a browser compatibility testing are the font look and feel in browsers, header and footer, styles of the page, formats in date, image positioning, HTML and CSS validation, zoom in and zoom out and alignment of the elements on the page and so on.

**90. Forward compatibility testing**: Forward compatibility testing is to validate the application under test is working as intended in the later versions of the software’s current version.

**91. Backward compatibility testing**: Backward compatibility testing is to validate the application under test is working as intended in the earlier versions of the software’s current version.

**92. Downward compatibility testing**: Refer to Backward compatibility testing…

**93. Compliance testing**: Compliance testing is non-functional testing which is done to validate whether the software meets a defined set of standards.

**94. Conformance testing**: Conformance testing is a testing technique to check that a product meets certain standards before its release. These standards are defined by organizations like IEEE to ensure that the software is compliant.

The conformance testing features include the below points:

* It is carried out by external organizations that are certified in their domain.
* It checks the robust testing process.
* It carries out testing with approved testing processes.

**95. UI testing**: In UI testing, testers aim to test both GUI and Command Line Interfaces (CLIs)

Also, refer to GUI Testing…

**96. Destructive testing**: Destructive testing is a testing technique that aims to validate the robustness of the application by testing continues until the application breaks.

**97. Dependency testing**: Dependency testing is a testing technique that examines the requirements of an application for pre-conditions, initial states, and configuration for the proper functioning of the application.

**98. Crowdsourced testing**: Crowdsourced testing is carried out by a community of expert quality assurance testers through an online platform.

**99. ETL testing**: ETL (Extract, Transform, and Load) testing involves invalidating the data movement from source to destination and verifying the data count in both source and destination and verifying data extraction, transformation, and also verifying the table relations.

**100. Data warehouse testing**: Refer to ETL testing…

**101. Fault injection testing**: Fault injection testing is a testing technique in which fault is intentionally introduced in the code in order to improve the test coverage.

**102. Failover testing**: Failover testing is a testing technique that validates a system’s ability to be able to allocate extra resource during the server failure and transferring of the processing part to back-up systems

**103. All pair testing**: All pair testing approach is to test the application with all possible combination of the values of input parameters.

**104. Pairwise Testing:**The pairwise testing is a testing method to test an application with permutation and combination of parameters. It is done to test all the feasible discrete combinations of specifications. By using the normal exhaustive testing approach, it may become impossible to test the complete product. But by following the permutation and combination of inputs, the testing of the product is achievable.

For example, let us consider a product to be tested with 15 inputs and there are 15 possible configurations for each input. So there are a total 15^15 inputs to be tested. In such a scenario, complete testing is not possible and we have to choose combinations of inputs.

Here I am going to conclude different types of software testing types. If you like this post, please share it with your friends.

Here I have hand-picked a few posts which will help you to learn more interview related stuff:

* [Manual Testing Tutorial](https://www.softwaretestingmaterial.com/manual-testing-tutorial/)
* [Agile Tutorial](https://www.softwaretestingmaterial.com/category/agile/)
* [Manual Testing Interview Questions](https://www.softwaretestingmaterial.com/100-software-testing-interview-questions/)
* [Agile Interview Questions](https://www.softwaretestingmaterial.com/agile-testing-interview-questions/)
* [Why You Choose Software Testing As A Career](https://www.softwaretestingmaterial.com/choose-software-testing-as-a-career/)
* [General Interview Questions](https://www.softwaretestingmaterial.com/6-important-interview-questions/)

If you have any more questions, feel free to ask via comments. If you find this post useful, do share it with your friends on Social Networking.

**A QUICK ASSERTION : NODEJS ADOPTION FOR MIDDLEWARE DEVELOPMENT**

After a very thorough experience since 2012 in software development, leading a team of multi-talented software developers, software engineers, devops guys, design gurus, etc., and on top of all that the amount of load nodeJS can handle, it is NOT and would not be an option I would consider for development of any middleware especially not for a financial sector type of middleware.

Some results from $ npm audit

# npm audit report

axios <=1.8.1

Severity: high

Axios vulnerable to Server-Side Request Forgery - https://github.com/advisories/GHSA-4w2v-q235-vp99

Axios Cross-Site Request Forgery Vulnerability - https://github.com/advisories/GHSA-wf5p-g6vw-rhxx

axios Inefficient Regular Expression Complexity vulnerability - https://github.com/advisories/GHSA-cph5-m8f7-6c5x

axios Requests Vulnerable To Possible SSRF and Credential Leakage via Absolute URL - https://github.com/advisories/GHSA-jr5f-v2jv-69x6

Depends on vulnerable versions of follow-redirects

No fix available

node\_modules/axios

node\_modules/flutterwave-node-v3/node\_modules/axios

node\_modules/kuda-node/node\_modules/axios

flutterwave-node-v3 \*

Depends on vulnerable versions of axios

Depends on vulnerable versions of request

node\_modules/flutterwave-node-v3

kuda-node \*

Depends on vulnerable versions of axios

node\_modules/kuda-node

cookie <0.7.0

cookie accepts cookie name, path, and domain with out of bounds characters - https://github.com/advisories/GHSA-pxg6-pf52-xh8x

fix available via `npm audit fix`

node\_modules/engine.io/node\_modules/cookie

engine.io 1.8.0 - 6.6.1

Depends on vulnerable versions of cookie

Depends on vulnerable versions of debug

node\_modules/engine.io

socket.io 1.0.0-pre - 3.0.4

Depends on vulnerable versions of debug

Depends on vulnerable versions of engine.io

Depends on vulnerable versions of socket.io-client

Depends on vulnerable versions of socket.io-parser

node\_modules/socket.io

cross-spawn <6.0.6

Severity: high

Regular Expression Denial of Service (ReDoS) in cross-spawn - https://github.com/advisories/GHSA-3xgq-45jj-v275

No fix available

node\_modules/precise-commits/node\_modules/cross-spawn

execa 0.5.0 - 0.9.0

Depends on vulnerable versions of cross-spawn

node\_modules/precise-commits/node\_modules/execa

precise-commits \*

Depends on vulnerable versions of execa

node\_modules/precise-commits

debug 4.0.0 - 4.3.0

Regular Expression Denial of Service in debug - https://github.com/advisories/GHSA-gxpj-cx7g-858c

fix available via `npm audit fix`

node\_modules/engine.io/node\_modules/debug

node\_modules/socket.io-parser/node\_modules/debug

node\_modules/socket.io/node\_modules/debug

socket.io-parser 3.4.0 - 4.0.2

Depends on vulnerable versions of debug

node\_modules/socket.io-parser

dicer \*

Severity: high

Crash in HeaderParser in dicer - https://github.com/advisories/GHSA-wm7h-9275-46v2

No fix available

node\_modules/dicer

busboy <=0.3.1

Depends on vulnerable versions of dicer

node\_modules/busboy

multer <=2.0.0-rc.3

Depends on vulnerable versions of busboy

node\_modules/multer

follow-redirects <=1.15.5

Severity: high

Exposure of Sensitive Information to an Unauthorized Actor in follow-redirects - https://github.com/advisories/GHSA-pw2r-vq6v-hr8c

Follow Redirects improperly handles URLs in the url.parse() function - https://github.com/advisories/GHSA-jchw-25xp-jwwc

follow-redirects' Proxy-Authorization header kept across hosts - https://github.com/advisories/GHSA-cxjh-pqwp-8mfp

Exposure of sensitive information in follow-redirects - https://github.com/advisories/GHSA-74fj-2j2h-c42q

No fix available

node\_modules/kuda-node/node\_modules/follow-redirects

parseuri <2.0.0

Severity: moderate

parse-uri Regular expression Denial of Service (ReDoS) - https://github.com/advisories/GHSA-6fx8-h7jm-663j

fix available via `npm audit fix`

node\_modules/parseuri

engine.io-client 1.0.2 - 6.1.1

Depends on vulnerable versions of parseuri

node\_modules/engine.io-client

socket.io-client 1.0.0-pre - 4.4.1

Depends on vulnerable versions of engine.io-client

Depends on vulnerable versions of parseuri

node\_modules/socket.io-client

request \*

Severity: moderate

Server-Side Request Forgery in Request - https://github.com/advisories/GHSA-p8p7-x288-28g6

Depends on vulnerable versions of tough-cookie

No fix available

node\_modules/request

requestretry \*

Depends on vulnerable versions of request

node\_modules/requestretry

slack-node >=0.1.2

Depends on vulnerable versions of requestretry

node\_modules/slack-node

tough-cookie <4.1.3

Severity: moderate

tough-cookie Prototype Pollution vulnerability - https://github.com/advisories/GHSA-72xf-g2v4-qvf3

No fix available

node\_modules/request/node\_modules/tough-cookie

useragent \*

Severity: moderate

useragent Regular Expression Denial of Service vulnerability - https://github.com/advisories/GHSA-mgfv-m47x-4wqp

No fix available

node\_modules/useragent

23 vulnerabilities (4 low, 7 moderate, 12 high)

To address issues that do not require attention, run:

npm audit fix

Some issues need review, and may require choosing

a different dependency.

<https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=node-fetch>

## Search Results

There are **3** CVE Records that match your search.

| **Name** | **Description** |
| --- | --- |
| [CVE-2022-2596](https://www.cve.org/CVERecord?id=CVE-2022-2596) | Inefficient Regular Expression Complexity in GitHub repository node-fetch/node-fetch prior to 3.2.10. |
| [CVE-2022-0235](https://www.cve.org/CVERecord?id=CVE-2022-0235) | node-fetch is vulnerable to Exposure of Sensitive Information to an Unauthorized Actor |
| [CVE-2020-15168](https://www.cve.org/CVERecord?id=CVE-2020-15168) | node-fetch before versions 2.6.1 and 3.0.0-beta.9 did not honor the size option after following a redirect, which means that when a content size was over the limit, a FetchError would never get thrown and the process would end without failure. For most people, this fix will have a little or no impact. However, if you are relying on node-fetch to gate files above a size, the impact could be significant, for example: If you don't double-check the size of the data after fetch() has completed, your JS thread could get tied up doing work on a large file (DoS) and/or cost you money in computing. |

<https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=axios>

## Search Results

There are **11** CVE Records that match your search.

| **Name** | **Description** |
| --- | --- |
| [CVE-2025-27152](https://www.cve.org/CVERecord?id=CVE-2025-27152) | axios is a promise based HTTP client for the browser and node.js. The issue occurs when passing absolute URLs rather than protocol-relative URLs to axios. Even if &#8288;baseURL is set, axios sends the request to the specified absolute URL, potentially causing SSRF and credential leakage. This issue impacts both server-side and client-side usage of axios. This issue is fixed in 1.8.2. |
| [CVE-2024-57965](https://www.cve.org/CVERecord?id=CVE-2024-57965) | \*\* DISPUTED \*\* In axios before 1.7.8, lib/helpers/isURLSameOrigin.js does not use a URL object when determining an origin, and has a potentially unwanted setAttribute('href',href) call. NOTE: some parties feel that the code change only addresses a warning message from a SAST tool and does not fix a vulnerability. |
| [CVE-2024-39338](https://www.cve.org/CVERecord?id=CVE-2024-39338) | axios 1.7.2 allows SSRF via unexpected behavior where requests for path relative URLs get processed as protocol relative URLs. |
| [CVE-2023-45857](https://www.cve.org/CVERecord?id=CVE-2023-45857) | An issue discovered in Axios 1.5.1 inadvertently reveals the confidential XSRF-TOKEN stored in cookies by including it in the HTTP header X-XSRF-TOKEN for every request made to any host allowing attackers to view sensitive information. |
| [CVE-2021-3749](https://www.cve.org/CVERecord?id=CVE-2021-3749) | axios is vulnerable to Inefficient Regular Expression Complexity |
| [CVE-2020-28168](https://www.cve.org/CVERecord?id=CVE-2020-28168) | Axios NPM package 0.21.0 contains a Server-Side Request Forgery (SSRF) vulnerability where an attacker is able to bypass a proxy by providing a URL that responds with a redirect to a restricted host or IP address. |
| [CVE-2019-7693](https://www.cve.org/CVERecord?id=CVE-2019-7693) | Axios Italia Axios RE 1.7.0/7.0.0 devices have XSS via the RELogOff.aspx Error\_Parameters parameter. In some situations, the XSS would be on the family.axioscloud.it cloud service; however, the vendor also supports "Sissi in Rete (con server)" for offline operation. |
| [CVE-2019-25069](https://www.cve.org/CVERecord?id=CVE-2019-25069) | A vulnerability, which was classified as problematic, has been found in Axios Italia Axios RE 1.7.0/7.0.0. This issue affects some unknown processing of the component Error Message Handler. The manipulation leads to information disclosure (ASP.NET). The attack may be initiated remotely. |
| [CVE-2019-25068](https://www.cve.org/CVERecord?id=CVE-2019-25068) | A vulnerability classified as critical was found in Axios Italia Axios RE 1.7.0/7.0.0. This vulnerability affects unknown code of the file REDefault.aspx of the component Connection Handler. The manipulation of the argument DBIDX leads to privilege escalation. The attack can be initiated remotely. |
| [CVE-2019-10742](https://www.cve.org/CVERecord?id=CVE-2019-10742) | Axios up to and including 0.18.0 allows attackers to cause a denial of service (application crash) by continuing to accepting content after maxContentLength is exceeded. |
| [CVE-2018-18437](https://www.cve.org/CVERecord?id=CVE-2018-18437) | In AXIOS ITALIA Axioscloud Sissiweb Registro Elettronico 1.7.0, secret/relogoff.aspx has XSS via the Error\_Desc parameter. |

**Axios vs Fetch**

In the article “[How to make HTTP requests like a pro with Axios](https://blog.logrocket.com/how-to-make-http-requests-like-a-pro-with-axios/#new_tab),” I discussed the benefits of using the Axios library.

However, Axios isn’t always the ideal solution for making HTTP requests. A popular alternative to Axios for handling HTTP/GET/POST/etc. requests in JavaScript is the native fetch() API. Understanding these technologies’ strengths, differences, and use cases is crucial for modern web development.

Here are some differences worth noting between the two solutions:

| **Characteristic** | **fetch() API** | **Axios library** |
| --- | --- | --- |
| Origin | Native JavaScript API | Third-party library |
| Installation | Natively available to browsers and Node.js v18+ | Requires npm install |
| JSON parsing (see code below) | Manual (need to use .json()) | Automatic |
| Error handling | Minimal (only network errors) | Comprehensive |
| Request interceptors | Not available (see [this article to implement them in fetch()](https://blog.logrocket.com/intercepting-javascript-fetch-api-requests-responses/)) | Available |
| Request cancellation (see code below) | Requires AbortController | Built-in method |
| Response transformation | Manual | Automatic |
| Platform support | Once browser-only but now available in Node.js v18+ | Browser and Node.js |

Some developers prefer Axios over built-in APIs for their ease of use. But many overestimate the need for such a library. The fetch() API is perfectly capable of reproducing the key features of Axios, and it has the added advantage of being readily available in all modern browsers.

In this article, we’ll compare fetch() and Axios to see how they can be used to perform different tasks. At the end of the article, you should have a better understanding of both APIs.

## Understanding the basic syntax of Axios and fetch()

Before we delve into more advanced features of Axios, let’s compare its basic syntax to fetch().  
Here’s how you can use [Axios to send a POST request](https://blog.logrocket.com/understanding-axios-post-requests/) with custom headers to a URL. Axios automatically converts the data to JSON, so you don’t have to:

// axios

const url = 'https://jsonplaceholder.typicode.com/posts'

const data = {

a: 10,

b: 20,

};

axios

.post(url, data, {

headers: {

Accept: "application/json",

"Content-Type": "application/json;charset=UTF-8",

},

})

.then(({data}) => {

console.log(data);

});

Now compare this code to the fetch() version, which produces the same result:

// fetch()

const url = "https://jsonplaceholder.typicode.com/todos";

const options = {

method: "POST",

headers: {

Accept: "application/json",

"Content-Type": "application/json;charset=UTF-8",

},

body: JSON.stringify({

a: 10,

b: 20,

}),

};

fetch(url, options)

.then((response) => response.json())

.then((data) => {

console.log(data);

});

Notice that:

* To send data, fetch() uses the body property for a post request to send data to the endpoint, while Axios uses the data property
* The data in fetch() is transformed into a string using the JSON.stringify method
* Axios automatically transforms the data returned from the server, but with fetch() you have to call the [response.json method to parse the data to a JavaScript object](https://developer.mozilla.org/en-US/docs/Web/API/Response/json)
* With Axios, the data response provided by the server can be accessed within the [data object](https://github.com/axios/axios#response-schema), while for the fetch() method, the final data can be named any variable
* Axios and fetch() handle headers in the same way

## Backward compatibility

One of the main selling points of Axios is its wide browser support. Even old browsers like IE11 can run Axios without any issues. This is because it uses [XMLHttpRequest](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) under the hood. fetch(), on the other hand, only supports Chrome 42+, Firefox 39+, Edge 14+, and Safari 10.1+ (you can see the full compatibility table on [CanIUse.com](https://caniuse.com/fetch)).

If your only reason for using Axios is backward compatibility, you don’t need an HTTP library. Instead, you can [use fetch() with a polyfill](https://github.com/JakeChampion/fetch) to implement similar functionality on web browsers that don’t support fetch().

To use the fetch() polyfill, install it via the npm command like so:

npm install whatwg-fetch --save

Then, you can make requests like this:

import 'whatwg-fetch'

window.fetch(...)

Keep in mind that you might also need a promise polyfill in some old browsers.

## Response timeouts in Axios vs. fetch()

The simplicity of setting a timeout in Axios is one of the reasons some developers prefer it to fetch(). In Axios, you can use the optional timeout property in the config object to set the number of milliseconds before the request is aborted.

Here’s an example:

axios({

method: 'post',

url: '/login',

timeout: 4000, // 4 seconds timeout

data: {

firstName: 'David',

lastName: 'Pollock'

}

})

.then(response => {/\* handle the response \*/})

.catch(error => console.error('timeout exceeded'))

fetch() provides similar functionality through the AbortController interface. However, it’s not as simple as the Axios version:

const controller = new AbortController();

const options = {

method: 'POST',

signal: controller.signal,

body: JSON.stringify({

firstName: 'David',

lastName: 'Pollock'

})

};

const promise = fetch('/login', options);

const timeoutId = setTimeout(() => controller.abort(), 4000);

promise

.then(response => {/\* handle the response \*/})

.catch(error => console.error('timeout exceeded'));

Here, we created an AbortController object using the AbortController.abort() constructor, which allows us to abort the request later. Signal is a read-only property of AbortController, providing a means to communicate with a request or abort it. If the server doesn’t respond in less than four seconds, controller.abort() is called, and the operation is terminated.

## Automatic JSON data transformation

As we saw earlier, Axios automatically stringifies the data when sending requests (though you can override the default behavior and define a different transformation mechanism). When using fetch(), however, you’d have to do it manually.

Compare the two below:

// axios

axios.get('https://api.github.com/orgs/axios')

.then(response => {

console.log(response.data);

}, error => {

console.log(error);

});

// fetch()

fetch('https://api.github.com/orgs/axios')

.then(response => response.json()) // one extra step

.then(data => {

console.log(data)

})

.catch(error => console.error(error));

Automatic data transformation is a nice feature, but again, it’s not something you can’t do with fetch().

## HTTP interceptors

One of Axios’s key features is its ability to intercept HTTP requests. HTTP interceptors come in handy when you need to examine or change HTTP requests from your application to the server or vice versa (e.g., logging, authentication, or retrying a failed HTTP request).

interceptors are helpful when you want to set a global strategy for how you handle requests and responses.

Here’s how you can declare a request interceptor in Axios:

axios.interceptors.request.use(config => {

// log a message before any HTTP request is sent

console.log('Request was sent');

return config;

});

// sent a GET request

axios.get('https://api.github.com/users/sideshowbarker')

.then(response => {

console.log(response.data);

});

In this code, the axios.interceptors.request.use() method is used to define code to be run before an HTTP request is sent. Also, axios.interceptors.response.use() can be used to intercept the response from the server. Let’s say there is a network error; using the response interceptors, you can retry that same request using interceptors.

By default, fetch() doesn’t provide a way to intercept requests, but it’s not hard to come up with a workaround. You can overwrite the global fetch() method and define your interceptor, like this:

fetch = (originalFetch => {

return (...arguments) => {

const result = originalFetch.apply(this, arguments);

return result.then(console.log('Request was sent'));

};

})(fetch);

fetch('https://api.github.com/orgs/axios')

.then(response => response.json())

.then(data => {

console.log(data)

});

## Download progress with Axios vs. fetch()

Progress indicators are very useful when loading large assets, especially for users with slow internet. Previously, JavaScript programmers used the XMLHttpRequest.onprogress callback handler to implement progress indicators.

The [Fetch API](https://blog.logrocket.com/patterns-for-data-fetching-in-react-981ced7e5c56/#:~:text=these%20alternative%20implementations.-,Using%20the%20Fetch%20API,-I%E2%80%99ve%20used%20Fetch) doesn’t have an onprogress handler. Instead, it provides an instance of ReadableStream via the body property of the response object.

The following example illustrates the use of ReadableStream to provide users with immediate feedback during image download:

index.html

<!-- Wherever you html is -->

<div id="progress" src="">progress</div>

<img id="img">

script.js

'use strict'

const element = document.getElementById('progress');

fetch('https://fetch-progress.anthum.com/30kbps/images/sunrise-baseline.jpg')

.then(response => {

if (!response.ok) {

throw Error(response.status+' '+response.statusText)

}

// ensure ReadableStream is supported

if (!response.body) {

throw Error('ReadableStream not yet supported in this browser.')

}

// store the size of the entity-body, in bytes

const contentLength = response.headers.get('content-length');

// ensure contentLength is available

if (!contentLength) {

throw Error('Content-Length response header unavailable');

}

// parse the integer into a base-10 number

const total = parseInt(contentLength, 10);

let loaded = 0;

return new Response(

// create and return a readable stream

new ReadableStream({

start(controller) {

const reader = response.body.getReader();

read();

function read() {

reader.read().then(({done, value}) => {

if (done) {

controller.close();

return;

}

loaded += value.byteLength;

progress({loaded, total})

controller.enqueue(value);

read();

}).catch(error => {

console.error(error);

controller.error(error)

})

}

}

})

);

})

.then(response =>

// construct a blob from the data

response.blob()

)

.then(data => {

// insert the downloaded image into the page

document.getElementById('img').src = URL.createObjectURL(data);

})

.catch(error => {

console.error(error);

})

function progress({loaded, total}) {

element.innerHTML = Math.round(loaded/total\*100)+'%';

}

Implementing a progress indicator in Axios is simpler, especially if you use the [Axios Progress Bar](https://github.com/rikmms/progress-bar-4-axios/) module. First, you need to include the following style and scripts:

// the head of your HTML

<link rel="stylesheet" type="text/css"

href="https://cdn.rawgit.com/rikmms/progress-bar-4-axios/0a3acf92/dist/nprogress.css" />

// the body of your HTML

<img id="img" />

<button onclick="downloadFile()">Get Resource</button>

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

<script src="https://cdn.rawgit.com/rikmms/progress-bar-4-axios/0a3acf92/dist/index.js"></script>

// add the following to customize the style

<style>

#nprogress .bar {

background: red !important;

}

#nprogress .peg {

box-shadow: 0 0 10px red, 0 0 5px red !important;

}

#nprogress .spinner-icon {

border-top-color: red !important;

border-left-color: red !important;

}

</style>

Then you can implement the progress bar like this:

<script type="text/javascript">

loadProgressBar();

function downloadFile() {

getRequest(

"https://fetch-progress.anthum.com/30kbps/images/sunrise-baseline.jpg"

);

}

function getRequest(url) {

axios

.get(url, { responseType: "blob" })

.then(function (response) {

const reader = new window.FileReader();

reader.readAsDataURL(response.data);

reader.onload = () => {

document.getElementById("img").setAttribute("src", reader.result);

};

})

.catch(function (error) {

console.log(error);

});

}

</script>

This code uses the FileReader API to asynchronously read the downloaded image. The readAsDataURL method returns the image’s data as a Base64-encoded string, which is then inserted into the src attribute of the img tag to display the image.

## Making simultaneous requests

To make multiple, simultaneous requests, Axios provides the axios.all() method. Simply pass an array of requests to this method, then use axios.spread() to assign the properties of the response array to separate variables:

axios.all([

axios.get('https://api.github.com/users/iliakan'),

axios.get('https://api.github.com/users/taylorotwell')

])

.then(axios.spread((obj1, obj2) => {

// Both requests are now complete

console.log(obj1.data.login + ' has ' + obj1.data.public\_repos + ' public repos on GitHub');

console.log(obj2.data.login + ' has ' + obj2.data.public\_repos + ' public repos on GitHub');

}));

You can achieve the same result by using the built-in Promise.all() method. Pass all fetch requests as an array to Promise.all(). Next, handle the response by using an async function, like this:

Promise.all([

fetch('https://api.github.com/users/iliakan'),

fetch('https://api.github.com/users/taylorotwell')

])

.then(async([res1, res2]) => {

const a = await res1.json();

const b = await res2.json();

console.log(a.login + ' has ' + a.public\_repos + ' public repos on GitHub');

console.log(b.login + ' has ' + b.public\_repos + ' public repos on GitHub');

})

.catch(error => {

console.log(error);

});

## How to configure CORS

[Cross-Origin Resource Sharing (CORS)](https://blog.logrocket.com/the-ultimate-guide-to-enabling-cross-origin-resource-sharing-cors/) is a mechanism available in HTTP to enable a server to permit the loading of its resources from any origins other than itself. For example, you need CORS when you want to pull data from external APIs that are public or authorized.

If the CORS mechanism is not properly enabled on the server, any request from a different server — regardless of whether or not it is made with Axios or fetch() — will receive the No Access-Control-Header-Present error.

To properly handle CORS, the first step is to configure the server, which depends on your environment/server. Once the server has been properly configured, it will automatically include the Access-Control-Allow-Origin header in response to all requests ([see the documentation for more information](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Access-Control-Allow-Origin)).

A common error, in both Axios and fetch(), is to add the Access-Control-Allow-Origin to the request — this is a response parameter and is used by the server to specify the permitted access control for the origin.

Another aspect to be aware of, when you add the headers to your Axios request, is that the request is handled differently: the browser performs a preflight request before the actual request and this preflight request is an [OPTIONS](https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/OPTIONS) request that verifies if CORS is honored and if the actual request is safe to send the real request.

## Effectively handling responses

Response management is a critical part of every application invoking an API. In this section, we will briefly look at the two aspects of it: getting the error code and manipulating response data.

Error management is different in Axios and fetch(). Specifically, fetch() doesn’t automatically reject the promise in the event of server-side errors, such as HTTP 404 or 500 status codes. This means that these errors don’t trigger the .catch() block, unlike in Axios where such responses would typically be considered exceptions.

### More great articles from LogRocket:

* Don't miss a moment with [The Replay](https://lp.logrocket.com/subscribe-thereplay), a curated newsletter from LogRocket
* [Learn](https://blog.logrocket.com/rethinking-error-tracking-product-analytics/) how LogRocket's Galileo cuts through the noise to proactively resolve issues in your app
* Use React's useEffect [to optimize your application's performance](https://blog.logrocket.com/understanding-react-useeffect-cleanup-function/)
* Switch between [multiple versions of Node](https://blog.logrocket.com/switching-between-node-versions-during-development/)
* [Discover](https://blog.logrocket.com/using-react-children-prop-with-typescript/) how to use the React children prop with TypeScript
* [Explore](https://blog.logrocket.com/creating-custom-mouse-cursor-css/) creating a custom mouse cursor with CSS
* Advisory boards aren’t just for executives. [Join LogRocket’s Content Advisory Board.](https://lp.logrocket.com/blg/content-advisory-board-signup) You’ll help inform the type of content we create and get access to exclusive meetups, social accreditation, and swag.

Instead, fetch() will resolve the promise normally with the ok status in the response set to false. The call to fetch() will only fail on network failures or if anything has prevented the request from completing.

In the following code, you can see how to handle errors in fetch():

try {

const res = await fetch('...');

if (!res.ok) {

// Error on the response (5xx, 4xx)

switch (res.status) {

case 400: /\* Handle \*/ break;

case 401: /\* Handle \*/ break;

case 404: /\* Handle \*/ break;

case 500: /\* Handle \*/ break;

}

}

// Here the response can be properly handled

} catch (err) {

// Error on the request (Network error)

}

Meanwhile, in Axios, you can discriminate all errors in a proper catch block as shown in the following example:

try {

let res = await axios.get('...');

// Here the response can be properly handled

} catch (err) {

if (err.response) {

// Error on the response (5xx, 4xx)

} else if (err.request) {

// Error on the request (Network error)

}

}

Once the request has been served with a proper response without any errors, you can handle the response payload that will be accessible by using two different mechanisms.

In fetch(), the request/response payload is accessible in the body field and must be stringified, while in Axios it is in the data field as a proper JavaScript object. This difference is captured in the following, stripped-down examples:

// Using Fetch API

fetch('...')

.then(response => response.json())

.then(data => console.log(data))

.catch(error => console.error('Error:', error));

// Using Axios

axios.get('...')

.then(response => console.log(response.data))

.catch(error => console.error('Error:', error));

The key difference in fetch() lies in the use of the .json() method. Despite the name, this method does not produce JSON but instead, it will take JSON as an input and parse it to produce a JavaScript object.

## Conclusion

Axios provides an easy-to-use API in a compact package for most HTTP communication needs. However, if you prefer to stick with native APIs, nothing is stopping you from implementing Axios features.

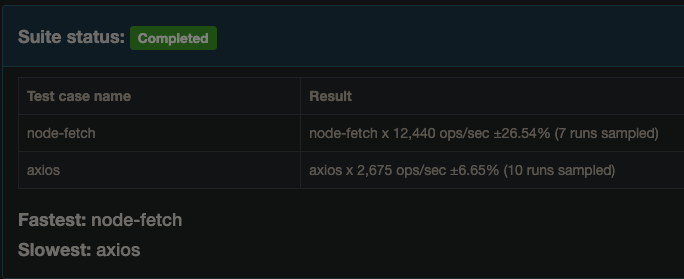
As discussed in this article, it’s possible to reproduce the key features of the Axios library using the fetch() method provided by web browsers. Whether it’s worth loading a client HTTP API depends on whether you’re comfortable working with built-in APIs.

https://www.measurethat.net/Benchmarks/Show/16084/0/axios-vs-node-fetch

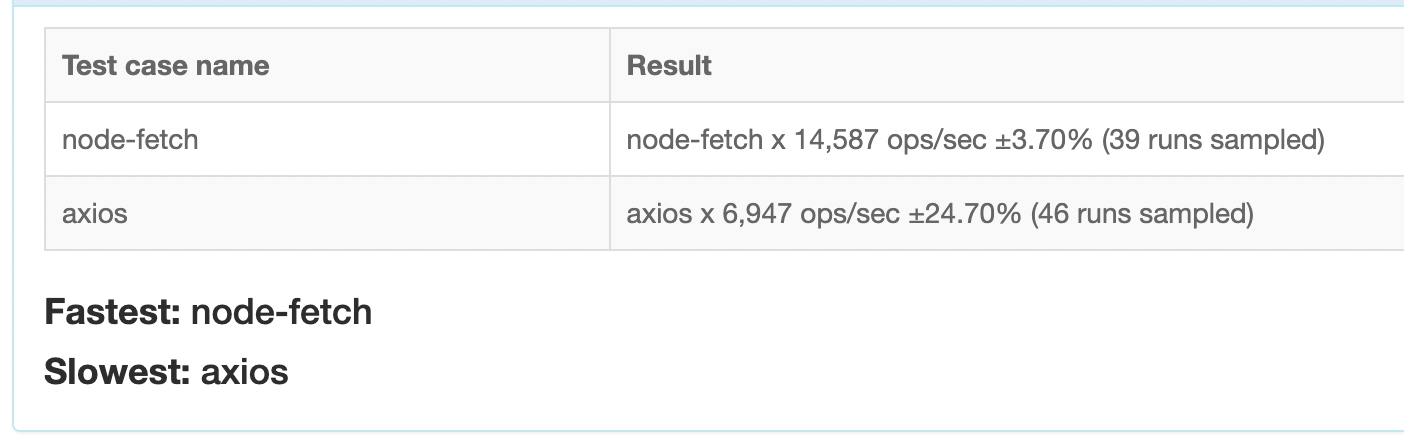
| **Test name** | **Executions per second** |
| --- | --- |
| **node-fetch** | 4472.3 Ops/sec |
| **axios** | 1488.7 Ops/sec |



Tor Browser Results :



Google chrome (incognito) results :



**FRONT END ENGINEERING**

So far so good, great and super awesome backend development. A big thanks to techempower for helping with the decision points in commencing this awesome journey of developing a solution that scales on a single instance at double, triple and quadripple millions of requests per second type of load handling.

At this point of my journey, I have tried a lot of frontend technologies including good old naturally occurring HTML, CSS, JS, VueJS, JSP, htmx, and so much more, but I needed something ( a frontend technology ), that is very fast, very scalable, very easy to develop, very cool to work with as a java software engineer, and the search is pointing to something pretty awesome.

JTE : Java Template Engine

So I came across across JTE or jte after working pain-stakingly with JSP with some very old project.

So far I like what I have seen from the likes of danvega, casid, and all the few people I saw as at April 2025 7th. Github only returned just about 8 repositories globally of those that are dabbling in/with jte. For me that was ok, it meant , jte is or was new and at its nascent stages.

I have already done my git clone command a few choices and quick ZIP downloads on those I just want to try out.

And now I am about to fully roll up my sleeves and get my hands quite muddy not just dirty. And let’s see what I build in just under 72 hours. And yeah, 72 hours because I have a lot of other responsibilities : my two kids, home chores, etc.

**Benchmarkings in relating to JTE**

With the sorts of technologies I am looking forward to rendering to the world via the companies and brands I do my ‘wizardry’ for, I always interested in know how they compare with or to other technologies or wizardries out there.

So I also took some time to checkout what the bench-markers say about JTE. See below :

## Why Choose JTE over Thymeleaf?

While Thymeleaf is a popular choice for Spring Boot applications, JTE offers several advantages:

1. **Performance**: JTE compiles templates to Java bytecode, resulting in very efficient runtime execution. This can lead to better performance, especially for high-traffic applications.
2. **Compile-time checking**: JTE performs extensive compile-time checks, catching many errors (like missing parameters or incorrect types) during compilation rather than at runtime. This can significantly improve development productivity and reduce runtime errors.
3. **Simpler syntax**: JTE uses a syntax that's very close to Java, making it easier for Java developers to pick up. As seen in the index.jte file, the syntax is intuitive and closely resembles Java code.
4. **Lightweight**: JTE is a more lightweight library compared to Thymeleaf, with fewer dependencies and a smaller footprint.
5. **Better IDE support**: Because JTE compiles to Java, it can leverage existing Java tooling, often resulting in better IDE support, including features like code completion and refactoring.
6. **Hot reloading**: JTE supports hot reloading of templates in development mode without requiring a server restart, which can speed up the development process.
7. **Precompilation**: JTE allows you to precompile templates, which can be useful for detecting errors early and improving startup times in production.
8. **Explicit over implicit**: JTE favors explicit declarations (like @param for parameters) over implicit conventions, making templates more self-documenting and easier to understand.

Source : <https://github.com/bruce-mig/java-template-engine>

**Some of the very few reasons for considering leaving JSP**<https://www.cve.org/CVERecord/SearchResults?query=jsp>

**My JTE journey**

After the git cloning and git zipping,

I quickly headed to cve.mitre.org to find out if there are any vulnerabilities in relation to jte or java template engine, so I can ensure my wire-fence is up and all guards up heavily before I get deep into ‘this’ jte.

CVE finding on jte :

<https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=jte>

|  |  |
| --- | --- |
| [CVE-2025-23026](https://www.cve.org/CVERecord?id=CVE-2025-23026) | jte (Java Template Engine) is a secure and lightweight template engine for Java and Kotlin. In affected versions Jte HTML templates with `script` tags or script attributes that include a Javascript template string (backticks) are subject to XSS. The `javaScriptBlock` and `javaScriptAttribute` methods in the `Escape` class do not escape backticks, which are used for Javascript template strings. Dollar signs in template strings should also be escaped as well to prevent undesired interpolation. HTML templates rendered by Jte's `OwaspHtmlTemplateOutput` in versions less than or equal to `3.1.15` with `script` tags or script attributes that contain Javascript template strings (backticks) are vulnerable. Users are advised to upgrade to version 3.1.16 or later to resolve this issue. There are no known workarounds for this vulnerability. References 3 Total  * [github.com: https://github.com/casid/jte/security/advisories/GHSA-vh22-6c6h-rm8q](https://github.com/casid/jte/security/advisories/GHSA-vh22-6c6h-rm8q)     [developer.mozilla.org: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template\_literals#description](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Template_literals#description)    [github.com: https://github.com/casid/jte/blob/main/jte-runtime/src/main/java/gg/jte/html/escape/Escape.java#L43-L83](https://github.com/casid/jte/blob/main/jte-runtime/src/main/java/gg/jte/html/escape/Escape.java#L43-L83) |

|  |  |
| --- | --- |
| [CVE-2023-33962](https://www.cve.org/CVERecord?id=CVE-2023-33962) | JStachio is a type-safe Java Mustache templating engine. Prior to version 1.0.1, JStachio fails to escape single quotes `'` in HTML, allowing an attacker to inject malicious code. This vulnerability can be exploited by an attacker to execute arbitrary JavaScript code in the context of other users visiting pages that use this template engine. This can lead to various consequences, including session hijacking, defacement of web pages, theft of sensitive information, or even the propagation of malware. Version 1.0.1 contains a patch for this issue. To mitigate this vulnerability, the template engine should properly escape special characters, including single quotes. Common practice is to escape `'` as `&#39`. As a workaround, users can avoid this issue by using only double quotes `"` for HTML attributes. |

[github.com: https://github.com/jstachio/jstachio/security/advisories/GHSA-gwxv-jv83-6qjr](https://github.com/jstachio/jstachio/security/advisories/GHSA-gwxv-jv83-6qjr)

  [github.com: https://github.com/jstachio/jstachio/issues/157](https://github.com/jstachio/jstachio/issues/157)

  [github.com: https://github.com/jstachio/jstachio/pull/158](https://github.com/jstachio/jstachio/pull/158)

  [github.com: https://github.com/jstachio/jstachio/commit/7b2f78377d1284df14c580be762a25af5f8dcd66](https://github.com/jstachio/jstachio/commit/7b2f78377d1284df14c580be762a25af5f8dcd66)

  [github.com: https://github.com/jstachio/jstachio/releases/tag/v1.0.1](https://github.com/jstachio/jstachio/releases/tag/v1.0.1)

Some channels worth checking out :

<https://www.youtube.com/@DanVega>

2025/04/23

Would you recommend using an internally database managed http sessions and uniquely generated code that has to be submitted with a form for the form to be valid ?

That's an interesting approach to form validation! While it offers some security benefits, it also comes with potential drawbacks compared to standard HTTP session management and CSRF tokens. Let's break down the pros and cons:

**Internally Managed HTTP Sessions and Unique Codes for Form Validation:**

**Potential Advantages:**

* **Enhanced State Control:** Managing sessions internally in your database gives you very fine-grained control over session data, lifecycle, and persistence. You're not reliant on the web server's session management.
* **Custom Security Logic:** You can implement custom security checks and logic directly within your session management and code generation process.
* **Potentially More Resilient to Stateless Environments:** In highly distributed or stateless backend architectures, managing session state in a central database can be more robust than relying on server-local session storage.
* **Unique Code as a Strong Anti-CSRF Measure:** The uniquely generated code, tied to the session and required for form submission, acts as a strong defense against Cross-Site Request Forgery (CSRF) attacks. Since the attacker wouldn't have access to the server-generated code for a legitimate session, they couldn't forge a valid request.
* **Simplified Statelessness for Forms:** This method can make form handling more stateless from the application server's perspective, as the validation logic relies on the database rather than in-memory session data.

**Potential Disadvantages:**

* **Increased Complexity:** Implementing and maintaining a custom session management system and unique code generation process adds significant complexity to your application. You'll need to handle session creation, storage, retrieval, expiration, and secure code generation.
* **Performance Overhead:** Every form submission will require a database query to validate the submitted code against the active session. For high-traffic applications, this could introduce significant database load compared to simpler session-based CSRF protection.
* **Scalability Concerns:** While a database can be scaled, poorly optimized session management and code validation queries could become a bottleneck under heavy load. Efficient indexing and database design are crucial.
* **Security Risks of Custom Implementation:** Rolling your own security mechanisms is generally discouraged unless you have deep security expertise. There's a higher risk of introducing vulnerabilities compared to using well-established and tested libraries for session management and CSRF protection.
* **Maintenance Burden:** Custom systems require ongoing maintenance, updates, and security patching. You'll be responsible for addressing any vulnerabilities discovered in your implementation.
* **Loss of Standard HTTP Session Features:** You might lose some built-in features and conveniences provided by standard HTTP session management frameworks (e.g., session cookies, automatic expiration, integration with web server features).
* **Potential for Code Exhaustion or Collision (though unlikely with proper generation):** While statistically improbable with strong random code generation, you'd need to ensure your code generation process avoids collisions and handles potential exhaustion if you have extremely high volumes of form submissions.

**Recommendation:**

Generally, **I would not recommend this approach as the primary method for managing HTTP sessions and form validation in most standard web applications.**

Here's why:

* **Standard HTTP session management with well-vetted CSRF protection (like synchronizer tokens) is a well-established, secure, and often more performant solution.** Web frameworks typically provide built-in support for these mechanisms, reducing development complexity and the risk of security vulnerabilities. 1

[1. Secure Coding: Top Risks, Best Practices, and Techniques - Jit.io](https://www.jit.io/resources/app-security/secure-coding-top-risks-best-practices-and-techniques" \t "_blank)

[[Source icon](https://www.jit.io/resources/app-security/secure-coding-top-risks-best-practices-and-techniques)](https://www.jit.io/resources/app-security/secure-coding-top-risks-best-practices-and-techniques" \t "_blank)

[www.jit.io](https://www.jit.io/resources/app-security/secure-coding-top-risks-best-practices-and-techniques" \t "_blank)

* **The added complexity of a custom session management and code validation system often outweighs the potential benefits, especially for common use cases.**
* **The potential performance overhead of database lookups for every form submission can be a significant concern for scalability.**

**When might this approach be considered?**

There might be specific niche scenarios where this level of control and custom logic could be beneficial:

* **Highly Specialized Security Requirements:** If your application has extremely stringent and unusual security requirements that standard session management and CSRF protection don't fully address.
* **Stateless Backend Architectures with Specific Needs:** In certain complex, stateless backend setups where traditional session cookies are difficult to manage, and a database-centric approach offers a more consistent way to handle state and prevent forgery.
* **Specific Audit or Compliance Requirements:** If your application has very specific audit trails or compliance needs related to session activity and form submissions.

**In conclusion, for most web development scenarios, leveraging the built-in session management and CSRF protection mechanisms provided by your web framework is the more secure, efficient, and maintainable approach. Building a custom system introduces unnecessary complexity and potential security risks unless you have very specific and compelling reasons to do so and possess the necessary security expertise.**

**Perfect Forward Secrecy is a critical security property that ensures that past encrypted communications cannot be decrypted even if the long-term private key of the server is compromised in the future. It achieves this by using ephemeral session keys that are uniquely generated for each session and are not derived in a way that depends on the long-term private key for retroactive decryption.** When you see that a connection uses a cipher suite with DHE or ECDHE, you know that Perfect Forward Secrecy is in play, providing a significant boost to the security and privacy of your communication.

### Session Hijacking Prevention

Session hijacking occurs when an attacker gains unauthorized access to a user’s session by stealing their session identifier, typically stored in cookies. This attack can compromise sensitive user data or account control.

**Techniques to prevent session hijacking:**

* **Secure flag**: Set the 'Secure' flag on cookies to ensure they are only transmitted over HTTPS, protecting them from interception.
* **HTTP-only flag**: Use the 'HttpOnly' flag to prevent client-side scripts, such as those from XSS attacks, from accessing session cookies.
* **SameSite policies**: Apply the 'SameSite' attribute to cookies, restricting their use to requests originating from the same domain.
* **Session Security**: Implement absolute and sliding session timeouts; Use secure session ID generation with sufficient entropy (at least 128 bits); Regenerate session IDs after privilege changes; Validate session data integrity with signed tokens

## The Shift-Left Approach to Secure Coding: A New Standard for Development

The [shift-left security approach](https://www.jit.io/resources/devsecops/what-is-shift-left-security) integrates security practices throughout the entire development lifecycle, starting from the initial planning and design phases. This methodology:

* Embeds security considerations into requirements gathering
* Implements security controls during architecture design
* Automates security testing in development workflows
* Validates security measures during code review
* Monitors security metrics throughout deployment
* **Error rates:** If you see a spike in failed requests (e.g., 401 Unauthorized, 403 Forbidden), attackers may be probing your API for vulnerabilities.

You can also **set up automated rules to shut down specific attack patterns before they escalate.** For instance, consider blocking IPs that hit /admin without an admin token. Consider implementing risk-based traffic rules:

* Monitor and analyze traffic patterns from cloud providers and Tor exit nodes
* Use adaptive rate limiting based on source IP reputation
* Implement CAPTCHA or additional verification for high-risk sources
* Allow legitimate API gateway and cloud service traffic based on your architecture

## Strengthen Your Defenses Against Standard API Attack Methods

With incidents like T-Mobile’s breach, which exposed 37 million records, failing to secure your API is a risk you can’t afford. A comprehensive checklist will help you take action now to protect your API security before you become the next major headline.

Additionally, product security platforms like Jit help you leverage tools like OWASP ZAP to scan your API vulnerabilities after every pull request automatically. If the platform finds issues, you get code-specific remediation steps directly within GitHub.

Download the API Security Checklist now and [visit Jit's homepage](https://www.jit.io/) to discover more about how developer-centric tools can transform your API security strategy.

2025/07/4

Comparing C/C++, Rust and Java :

When comparing C/C++, Rust, and Java for high-performance computing and general programming, each language offers distinct advantages and trade-offs:

### 1. High Performance Computing (HPC)

* **C/C++:**
  + **Performance:** Generally considered the gold standard for raw performance due to direct memory access, manual memory management, and compilation to highly optimized machine code. Offers fine-grained control over hardware resources.
  + **HPC Use Cases:** Widely used in scientific simulations, financial modeling, operating systems, game engines, and embedded systems where absolute maximum speed and minimal overhead are critical.
* **Rust:**
  + **Performance:** Aims to match C/C++'s performance. It compiles to native code, has zero-cost abstractions, and avoids runtime overhead like garbage collection. Its focus on memory safety at compile time doesn't typically incur a runtime performance penalty.
  + **HPC Use Cases:** Increasingly adopted for systems programming, web assembly, and areas where C/C++ traditionally dominates, especially when memory safety and concurrency are paramount.
* **Java:**
  + **Performance:** Runs on the Java Virtual Machine (JVM), which uses a Just-In-Time (JIT) compiler to optimize bytecode during runtime. While not as consistently fast as C/C++ or Rust for raw CPU-bound tasks due to JVM overhead and garbage collection, it can achieve very high performance for many enterprise and server-side applications, especially after "warm-up" (JIT optimization).
  + **HPC Use Cases:** Less common for extreme HPC (e.g., supercomputing) but used in large-scale data processing (e.g., Apache Spark, Hadoop), financial trading systems, and other high-throughput, low-latency enterprise applications.

### 2. Memory Safety

* **C/C++:**
  + **Safety:** **Manual memory management.** This gives immense control but is the primary source of common bugs like buffer overflows, use-after-free errors, null pointer dereferences, and memory leaks. These are runtime errors that can lead to crashes or security vulnerabilities. Requires diligent programming and extensive testing.
* **Rust:**
  + **Safety:** **Guaranteed memory safety at compile time** through its "borrow checker." This unique system enforces strict rules about memory ownership, borrowing, and lifetimes, preventing common memory errors (like data races in concurrent code) without a garbage collector. If the code compiles, it's memory-safe.
* **Java:**
  + **Safety:** **Automatic memory management (Garbage Collection).** The JVM handles memory allocation and deallocation, significantly reducing the risk of memory leaks and common memory-related errors found in C/C++. This makes it much safer in terms of memory access compared to C/C++. However, it doesn't prevent all logical memory issues (e.g., holding onto references to objects longer than needed, leading to higher memory usage).

### 3. Concurrency and Threading

* **C/C++:**
  + **Concurrency:** Offers low-level threading primitives (e.g., std::thread, mutexes, condition variables). Requires careful manual synchronization to prevent data races and deadlocks. Debugging concurrent issues can be extremely challenging.
* **Rust:**
  + **Concurrency:** Provides high-level concurrency primitives (e.g., std::thread, Arc, Mutex). Crucially, its borrow checker **guarantees data-race freedom at compile time**. This means that if your concurrent code compiles, it's impossible to have a data race, making concurrent programming much safer and less error-prone than in C/C++.
* **Java:**
  + **Concurrency:** Has robust built-in support for concurrency with rich libraries (e.g., java.util.concurrent). Uses threads, locks, and higher-level constructs like executors, concurrent collections, and atomic variables. While it provides tools for safe concurrency, developers must still manually manage locks and synchronization to prevent deadlocks and other issues. The JVM's memory model helps, but data races are still possible if not handled correctly.

### 4. Database CRUDing (Create, Read, Update, Delete)

* **C/C++:**
  + **Database Access:** Typically uses C-style APIs or C++ wrappers for database drivers (e.g., ODBC, libpq for PostgreSQL, MySQL C API). Requires more boilerplate code and manual handling of connections, queries, and result sets. ORM (Object-Relational Mapping) libraries are less mature or common compared to higher-level languages.
  + **Suitability:** Good for high-throughput, low-latency database interactions where every millisecond counts, but development can be slower.
* **Rust:**
  + **Database Access:** Has growing support for database drivers (e.g., diesel ORM, tokio-postgres for async PostgreSQL). The ecosystem is maturing rapidly, offering modern, safe, and performant asynchronous database access.
  + **Suitability:** Excellent for building high-performance, reliable backend services that interact with databases, leveraging its concurrency safety.
* **Java:**
  + **Database Access:** Extremely mature and comprehensive ecosystem for database interaction. JDBC (Java Database Connectivity) is the standard API, with numerous robust drivers for virtually every database. Extensive ORM frameworks like Hibernate, JPA, and Spring Data make CRUD operations highly productive and abstract away much of the boilerplate.
  + **Suitability:** Dominant in enterprise applications requiring complex database interactions, transaction management, and high developer productivity.

### 5. Maintainability

* **C/C++:**
  + **Maintainability:** Can be challenging. Manual memory management, complex header files, and the need for careful resource handling can lead to intricate and error-prone code. Debugging memory-related issues is notoriously difficult. Large C++ codebases often have a steep learning curve.
* **Rust:**
  + **Maintainability:** Generally high. The strict compile-time checks (borrow checker) enforce good design patterns and catch many errors early, leading to more reliable code that is easier to refactor. Its strong type system and clear error messages aid development. The learning curve can be steep initially due to the borrow checker, but it pays off in long-term stability.
* **Java:**
  + **Maintainability:** Generally high. Automatic memory management, strong type safety, clear object-oriented paradigms, and a vast ecosystem of well-documented libraries and frameworks contribute to highly maintainable code. IDE support is excellent, and its verbose nature can sometimes aid readability.

### 6. Community Support

* **C/C++:**
  + **Community:** Massive and mature. Decades of development mean a vast amount of existing code, libraries, forums, and experienced developers. However, the community is fragmented across various standards versions (C++98, C++11, C++17, C++20, etc.) and build systems.
* **Rust:**
  + **Community:** Rapidly growing, enthusiastic, and very supportive. Excellent official documentation, active forums, and a strong focus on developer experience. The ecosystem of libraries (crates.io) is expanding quickly.
* **Java:**
  + **Community:** One of the largest and most established programming communities globally. Extensive documentation, countless tutorials, active forums, and a huge number of open-source projects and enterprise solutions. Finding resources and developers is generally easy.

### 7. General Programming Operations

* **C/C++:**
  + **Garbage Collection:** None (manual memory management).
  + **Ecosystem:** Rich in low-level libraries, system utilities, and domain-specific libraries (e.g., scientific computing, graphics). Build systems (CMake, Make) can be complex.
  + **Typical Domains:** Operating systems, embedded systems, game development, high-frequency trading, scientific computing, compilers, databases.
* **Rust:**
  + **Garbage Collection:** None (ownership and borrowing system).
  + **Ecosystem:** Growing rapidly with a focus on safety, performance, and concurrency. Excellent tooling (Cargo package manager, rustfmt, clippy). Strong for web assembly, command-line tools, and backend services.
  + **Typical Domains:** Systems programming, web services (backend), command-line tools, WebAssembly, game development (increasingly), embedded systems.
* **Java:**
  + **Garbage Collection:** Automatic (JVM handles it). This simplifies development but can introduce occasional pauses (though modern GCs are highly optimized).
  + **Ecosystem:** Enormous and mature for enterprise applications. Rich standard library, vast array of third-party libraries and frameworks (Spring, Apache projects, etc.). Excellent cross-platform compatibility.
  + **Typical Domains:** Enterprise applications, Android mobile development, big data processing, web services (backend), desktop applications, cloud-native applications.

In summary:

* For **absolute raw performance and ultimate hardware control**, **C/C++** remains a top choice, but at the cost of manual memory management and higher complexity.
* **Rust** offers comparable performance to C/C++ with **unparalleled memory safety and data-race freedom at compile time**, making it an excellent choice for highly reliable and performant systems. It has a steeper initial learning curve.
* **Java** provides a highly productive development environment with **automatic memory management, robust concurrency features, and a vast ecosystem**, making it ideal for large-scale enterprise applications and distributed systems where slightly higher runtime overhead is acceptable for increased safety and developer velocity.