Google Summer of Code 2023 Ideas

JBoss is not accepted to GSoC for the 2023 program. Thanks everyone for your interest.

The JBoss Community is planning to participate in Google Summer of Code in 2023.

All contributors & developers are welcome to participate in the https://summerofcode.withgoogle.com/ program with the JBoss Community. See https://opensource.googleblog.com/2022/11/get-ready-for-google-summer-of-code-2023.html for more information.

If you are a contributor looking forward to participating in the GSoC 2023 with the JBoss Community:

- Feel free to browse the growing idea list below.
- Please don't hesitate to contact the mentor(s) indicated in the proposal for any related clarification and to discuss proposals.
- You can have a look at ideas list of previous years for inspiration.
- Please see our contributor guide.
- You may find a sample GSoC proposal document here which was for this idea.

Contributors: Please read the list above and also read our contributor guide.

A note to mentors

MENTORS: Red Hat employees can change this page directly to add ideas. Please be extra careful to not get other mentor's edits discarded.
Red Hatters should have linked their jboss.org account with Red Hat and can be checked on https://sso.jboss.org/login

Non-Red Hatters can add a comment to the page and admins will make sure the idea is added to the page.

Table of Contents

Administrators and Mentors

We will list the potential mentors in this place. For now, if you have any questions, please contact the GSoC administrators:

George Zaronikas (gzaronikas) and Ali Ok (@aliok_tr) .

Communication channels

Gitter : **JBossOutreach/GSoC - Gitter**

Please take note - These channels are about generic doubts. For project-specific doubts you will need to contact project mentors and channels specified in the project description.

Idea template (for mentors)

Project title

Summary of idea:

-Idea

-Feature A

-Feature B

Knowledge prerequisite: Languages/Technologies goes here

Github repo:

Project size: medium (~175 hours) or large (~350 hours)

Skill level: Beginner/Intermediate/Advanced

Contact(s) / potential mentors(s): Mentor(s) name and contact details

Associated JBoss community project(s):

Idea Proposals

Proof of Concept of Kroxylicious implementing Kubernetes Ingress or Gateway API specifications

Summary of idea:

- Understand the differences between Kubernetes ingress and Gateway API
- Proof of concept ingress API implementation
- Proof of concept Gateway API implementation
- Server Name Indicator (SNI) aware Null proxy implementation

Knowledge prerequisite:
Java or Go experience.
Kubernetes basic experience.

Github repo: https://github.com/kroxylicious/

Project size: medium (~175 hours) or large (~350 hours)

Skill level: Intermediate/Advanced

Contact(s) / potential mentors(s): Mentor(s) name and contact details
 - Sam Barker: sbarker@redhat.com

Associated JBoss community project(s):
kroxylicious.io

EAT - Testing Infinite Software Project Versions

Summary of idea:

The innovative part of EAT is creating the test once and testing with any version of the tested software. It may be firstly applied for the JBoss Servers, but, in general, a similar structure, can be used for creating tests about any software with multiple versions or for multiple software programs that have a part of the testsuite in common. EAT is a project under the statement.

Possible tasks for this project :

- extend the existing AT testsuites with latest JBOSS Community server snapshot
- extend the android related functionality
- update the workshop
- the contributors are also welcome to make their proposals
- proposals should be also added at the EAT-PROPOSALS mobile app (https://play.google.com/store/apps/details?id=edu.eatproposals.eatapp&hl=en&gl=US)

Project size: large (~350 hours)

Github repo: https://github.com/EAT-JBCOMMUNITY/EAT

Contact / potential mentors: Panagiotis Sotiropoulos (psotirop@redhat.com)

Associated JBoss community project: EAT

Horreum - Migrate project to use the latest web stack

Summary of idea:

The task of this project is to migrate Horreum from the current web stack to the latest version, f.ex. React 17 to React 18.

Tasks for this project :

- Update the platform to use the latest web stack
- Improve existing API usage to use the new libraries
- Refactor UI with new features

Difficulty level: Medium

Project size: Long (~350h)

Github repo: https://github.com/Hyperfoil/Horreum

Contact / potential mentors: Jesper Pedersen - jesper (dot) pedersen (at) redhat (dot) com

Horreum - Security model

Summary of idea:

The task of this project is to create a security model for Horreum that will allow for an improved multi-tenant platform. The initial task is described in https://github.com/Hyperfoil/Horreum/issues/78

Tasks for this project :

- Create a new security model

Difficulty level: Hard

Project size: Long (~350 hours)

Github repo: https://github.com/Hyperfoil/Horreum

Contact / potential mentors: Jesper Pedersen - jesper (dot) pedersen (at) redhat (dot) com

Knative - Multiple Knative Eventing control planes

Summary of idea:

We see more users asking for complete isolation in Knative Eventing deployments. While there are Knative Eventing components that support isolated data planes, we are interested in to see isolated control planes as well. There were discussions about this in the community before and many of the asks were left unadressed. However, we have tools that support multitenancy today, such as kcp. We see this project as an experiment. Expected outcome is finding issues blocking multiple control planes, and possibly fixing them. More info available at https://github.com/knative/eventing/issues/6601

Knowledge prerequisite: Golang, Kubernetes, Knative, Kubernetes Controllers

Github repo: https://github.com/knative/eventing/

Project size: 350 hours

Skill level: Advanced

Contact(s) / potential mentors(s): Ali Ok - aliok AT redhat DOT com, Christoph Stäbler- cstabler AT redhat DOT com

Associated JBoss community project(s): Knative / OpenShift Serverless

NOTE: This idea is also listed on CNCF GSoC ideas document for more visibility.

Knative - Dataplane migration for Apache Kafka communications: From Vert.x to Project Loom

Summary of idea:

The Knative Broker's data-plane communication with Apache Kafka for consuming and producing records is currently done via the Vertx Kafka client library. The library is basically a wrapper for communications with Apache Kafka inside of the Vertx threading model.The project requires an evaluation the OpenJDK 19 "Project Loom" itself and how to leverage it for efficient communications with an Apache Kafka cluster. The goal of the project is to migrate the usage of vertx for any communications with Apache Kafka to pure OpenJDK 19's Loom, and reduce the number of 3rd party frameworks, such as vertx for Apache Kafka communication.

Expected outcome is having a Knative Kafka Broker data plane implementation working on Java 19, leveraging the Project Loom APIs while communicating with Apache Kafka. No usage of the vertx-kafka-client, since all Kafka comms are done via Loom.

See https://github.com/knative-sandbox/eventing-kafka-broker/issues/2928 for the upstream ticket.

Knowledge prerequisite:

Github repo: https://github.com/knative-sandbox/eventing-kafka-broker/

Project size: 350 hours

Skill level: Advanced

Contact(s) / potential mentors(s): Matthias Wessendorf - mwessend AT redhat DOT com, Pierangelo Di Pilato - pierdipi AT redhat DOT com

Associated JBoss community project(s): Knative / OpenShift Serverless

NOTE: This idea is also listed on CNCF GSoC ideas document for more visibility.

Knative - Porting Knative Serving to Microshift

Summary of idea:

More and more workload is moving towards running on the edge. We saw experiments running Kubernetes on vehicles, fighter jets, 5G antenna and various far edge, near edge and fat edge environments. We would like to see what the challenges are when Knative is run in a resource limited environment. While there are multiple edge-friendly Kubernetes distributions, we would like to see Microshift used as the base platform. Knative consists of Serving and Eventing modules but focusing on Knative Serving as a first step should be more approachable. Stages:

- Run Knative on Microshift with minimal resources: Find out problems here, solve them.
- Stretch goal: Find out what happens with architectures other than x86_64.

Expected outcome for this project is having Knative Serving with an ingress layer running on top of Microshift. Having a Hello-World Knative Service running on top. Finding issues blocking the edge setup, and possibly fixing them.

See upstream ticket for more information: https://github.com/knative/serving/issues/12718

Knowledge prerequisite: Golang, Kubernetes, Knative, good understanding of networking, good understanding of CI/CD

Github repo: https://github.com/knative/serving/

Project size: 350 hours

Skill level: Advanced

Contact(s) / potential mentors(s): Stavros Kontopoulos - skontopo AT redhat DOT com, Reto Lehmann - rlehmann AT redhat DOT com

Associated JBoss community project(s): Knative / OpenShift Serverless

NOTE: This idea is also listed on CNCF GSoC ideas document for more visibility.

Knative - Self-Balancing Knative Kafka Broker partitions

Summary of idea:

Creating a Knative Kafka Broker requires developers to specify the number of partitions the backing Kafka topic should have, however, this is not an easy decision to make and it requires planning based on the expected load the Knative Broker could receive.
This project aims to remove this configuration setting by having an autoscaler that is responsible to add or remove partitions based on the effective load the Knative Kafka Broker receives while preserving ordered and unordered delivery features for Triggers.

Expected outcome for this project is having a Knative Kafka Broker can be created without setting the number of partitions and the number of partitions for a topic backing the Knative Kafka Broker increases or decreases based on the observed load received.

See https://github.com/knative-sandbox/eventing-kafka-broker/issues/2917 for more information.

Knowledge prerequisite: Kubernetes, Apache Kafka, Go, Java

Github repo: https://github.com/knative-sandbox/eventing-kafka-broker/

Project size: 350 hours

Skill level: Advanced

Contact(s) / potential mentors(s): Pierangelo Di Pilato - pierdipi AT redhat DOT com, Ali Ok - aliok AT redhat DOT com

Associated JBoss community project(s): Knative / OpenShift Serverless

NOTE: This idea is also listed on CNCF GSoC ideas document for more visibility.

Project Starfix - Open anything anywhere in any IDE/editor

Summary of idea:

StarFix is a cross-platform client-side application that lets you open a file in the Editor of your choice (vscode, eclipse, intellij, emacs, vi, etc.) and other commands locally directly from the browser or file system. This will enable an option to “Open in IDE” through browser extension on various websites like Github, Gitlab, etc. Moving forward we intend to add more features and capabilities. You can learn more about Starfix at : https://github.com/starfixdev/starfix

Possible Tasks for this project:

- **macOS installer:** Currently the end user isn't able to install the app using the generated installer due to signing issue.
- **Setup Release Mechanism for Browser Extension via Github Actions:** Currently Browser Extension release works well for Firefox but is not in place for Chrome.
- **Make Starfix more useful:** Starfix can offer more features making it a more powerful cli tool. Look at : [https://github.com/starfixdev/starfix/issues/7] and [https://github.com/starfixdev/starfix/issues/9] for example.
- **Webpage in case of Issues:** If starfix isn't configured or encounters an issue, we open a generated web page with instructions. Those instructions could even have ide:// based links to configure starfix. i.e. [ide://config?editor=emacs&clone=~/code] which when clicked launches startfix and set that config. Also we can have a help page for the user.
- **Versioning and Changelogs:** Setup mechanism for tracking the changelogs(feature updates, bug fixes) for each release and maintaining it with version history. Would be good if the latest changelog is included in the readme file under a heading like What's New @v2.1.1
- **Stable Release:** After putting the missing pieces in place and testing everything make a stable release.
- **Tests:** Add more tests considering the added features and big fixes to ensure future developments don't break the existing ones.
- **Documentation:** Update documentation elaborating about existing/added features with examples. Also fix issues like logo and existing demos not rendering in the readme on github.

You can look at more tasks/issues to work here : https://github.com/starfixdev/starfix/issues

Knowledge prerequisite:

- Java
- Basic Idea of CLI Tools
- Basic understanding of IDE and Browser extensions.
- Access to more than one of the Operating Systems will be helpful
- Other tools/technologies used (knowledge would be a plus but not a prerequisite) : Quarkus, JBang, Picocli, Github Actions, Maven Assembly, JUnit5 .

Github repo: https://github.com/starfixdev/starfix

If you want to explore/work in this area and show your experience/interest you can look into contributing to projects like JBang, Quarkus or to Project Starfix directly. Basically any project that relates to developer tools, desktop tools, scripts or browser extensions will be useful experience.

Skill level: Beginner/Intermediate

Project size: Short (~175 hours)

Contact(s) / potential mentors(s): Fahad Israr (fahad00cms@gmail.com)

(Can also reach out in the "gsoc" stream at Zulip-Quarkus: https://quarkusio.zulipchat.com/)

Associated JBoss community project(s): Quarkus, vscode extensions, JBoss Tools

3scale - Proof of Concept: authorino for role-based access

Summary of idea:

Authorisation of 3scale using authorino

- Understand the current RBAC implementation (Ruby on Rails)
- Proof of concept using authorino for RBAC instead of current 3scale auth implementation (Go)

Knowledge prerequisite: Go or Ruby experience.

Github repos: https://github.com/3scale/porta, https://github.com/Kuadrant/authorino

Project size: large (~350 hours)

Skill level: Intermediate/Advanced

Contact(s) / potential mentors(s): Thales Miguel thmartin@redhat.com

Associated JBoss community project(s): 3scale

3scale - Auto-generate OpenAPI v3 API Docs

Summary of idea:

We are migrating our API Docs from OpenAPI v1 to v3. As part of this, we need to have a way to automatically update OpenAPI json files every time a change is made in any endpoint. We are currently using the source2swagger gem for that, but it's abandoned and doesn't support OAS3. A possible way to implement auto-generation for OAS3 is by using the rswag gem, which creates the docs based on the test suite.

Knowledge prerequisite: Ruby experience.

Github repos: https://github.com/3scale/porta, https://github.com/rswag/rswag

Project size: medium (~150 hours)

Skill level: Intermediate

Further info: https://issues.redhat.com/browse/THREESCALE-8904

Contact(s) / potential mentors(s): Joan Lledo: jlledo@redhat.com

Associated JBoss community project(s): 3Scale

Proof of Concept of an MQTT to Apache Kafka bridge for producing messages

Summary of idea:

- Enabling an IoT scenario with MQTT based devices and using an Apache Kafka cluster as the events and storage platform.
- Providing an idea of mapping between MQTT to Apache Kafka protocols.
- Developing a pure Netty based MQTT server component (not a full MQTT broker) able to accept MQTT client connections and handling the corresponding communication based on the MQTT 3.1.1 specification.
- Developing the producer part able to get messages from MQTT clients and sending them to an Apache Kafka cluster.

Knowledge prerequisite: Java, MQTT protocol (not mandatory but to be learned)

Github repo: It would be part of the CNCF Strimzi project https://github.com/strimzi

Project size: medium (~175 hours) or large (~350 hours)

Skill level: Intermediate/Advanced

Contact(s) / potential mentors(s): Paolo Patierno (ppatiern@redhat.com)

Associated JBoss community project(s): Strimzi https://strimzi.io/

WildFly Elytron - A GUI to simplify client side security configuration

Summary of idea:

The WildFly Elytron project is a security framework for Java clients and application servers. Within the WildFly application server, Elytron is used to secure applications that are deployed to the server and to secure management access to the server. Banks, retail stores, and governments are just some examples of end-users of the enterprise version of the WildFly application server.

The WildFly Elytron Client allows remote clients to authenticate using Elytron. When a connection is being established, the WildFly Elytron Client makes use of rules to determine the information that should be used when attempting to authenticate. This includes things like the username, credentials, and authentication mechanisms that should be used. The WildFly Elytron Client also makes it possible to specify TLS configuration. For example, it's possible to specify the client certificate to present to the server.

The configuration used by the WildFly Elytron Client can be specified using an XML file, as described in detail here. The purpose of this project is to create a GUI (graphical user interface) that can be used to generate this XML file automatically rather than requiring a user to write this XML file on their own. The idea is to walk the user through providing the various types of information that can be specified and to use that information to write the XML file.

Possible tasks for this project include:

- Create a plan for what the GUI will look like and the information it will request from the user.
- Implement the GUI (we recommend using JavaFX to implement the GUI).
- Determine how to test the GUI (i.e., verifying that the GUI creates a valid XML file for the WildFly Elytron Client from the inputs provided).
- Consider what other things could be added to the GUI. As an example, the WildFly Elytron Client allows a user to reference a credential from a credential store. However, this relies on the credential store already existing. Perhaps the GUI can also guide the user through providing the information needed to automatically create the credential store.
- Create documentation that describes how to use the GUI.
- Write a blog post and/or create a YouTube video for the WildFlyAS YouTube Channel that shows how to use your GUI to create the XML file for WildFly Elytron Client and make use of the resulting XML file to establish a connection to a server.

The WildFly Elytron team is a diverse, distributed team that has a lot of experience working with interns and junior engineers.

Knowledge prerequisites:

- Experience with Java
- Git
- Maven
- Prior experience with JavaFX or other GUI frameworks would be useful but not required

Github repo: https://github.com/wildfly-security/wildfly-elytron

Other useful links:

- Elytron Contribution Guide
- Elytron website

Project size: Large (~350 hours)

Skill level: Intermediate

Project chat: https://wildfly.zulipchat.com/#narrow/stream/173102-wildfly-elytron

Contact(s) / potential mentors(s): Farah Juma (fjuma@redhat.com), and Diana Krepinska (dvilkola@redhat.com)

Associated JBoss community project(s): WildFly Elytron

General purpose programming language that transpiles to C++

Summary of idea:

Liam is a strongly typed compiled language intended for high performance. It is intended to give the control and speed of languages such as C or C++ but removing the headaches, over complexity and difficult to use build systems. Liam gives modern features without the legacy weighing it down like generic types, function expressions and union types.

Possible tasks for this project :

- Extend the existing vim code highlighting package
- Create a VsCode extension and Liam language server
- Improve UX with better error messages and type inference
- Extend on the standard library which ships with the compiler to increase the feature set of the language
- Increase the size of the automated test suite to improve the robustness of the compiler
- Work on the internals of the compier starting from the lexer and parser to the more complex type checker and C++ backend.

Project size: Medium (~200 hours)

Github repo: https://github.com/jackdelahunt/Liam

Contact / potential mentors: Jack Delahunt (jdelahun@redhat.com)