



GOTC 2023

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

OPEN SOURCE, INTO THE FUTURE

Cloud Native Summit 专场

Container tooling for WebAssembly

Michael Yuan 2023年05月28日



Develop faster. Run anywhere.

The most-loved Tool in Stack Overflow's 2022 Developer Survey.

Download Docker Desktop

Intel Chip



Apple Chip



Windows



Linux

WASM IS AWESOME

WHAT'S NEW

Docker + Wasm = Awesome!

Wasm is a new, fast, and light alternative to the Linux/Windows containers you're using in Docker today – give it a try with the Docker+Wasm Beta.

→ Try it



Solomon Hykes / @shykes@hachyderm...
@solomonstre

The Docker+wasm announcement makes perfect sense. We no longer live in a single-runtime world: there are linux containers, windows containers and wasm containers. OCI can package them all, I should be able to build and run them all with @docker.

3:02 AM · 10/25/22

298 Likes 70 Retweets 7 Quotes

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

<https://docs.docker.com/desktop/wasm/>

KEY FINDINGS 3/3

CONTAINERS ARE THE NEW NORMAL, AND WEBASSEMBLY IS THE FUTURE

END USER ORGANIZATIONS THAT HAVE USED WEBASSEMBLY, WITH WASMEDGE AND WAMR BEING THE TOP RUNTIMES

37%

Have you used or are aware of the following WebAssembly runtimes?

WASMEDGE



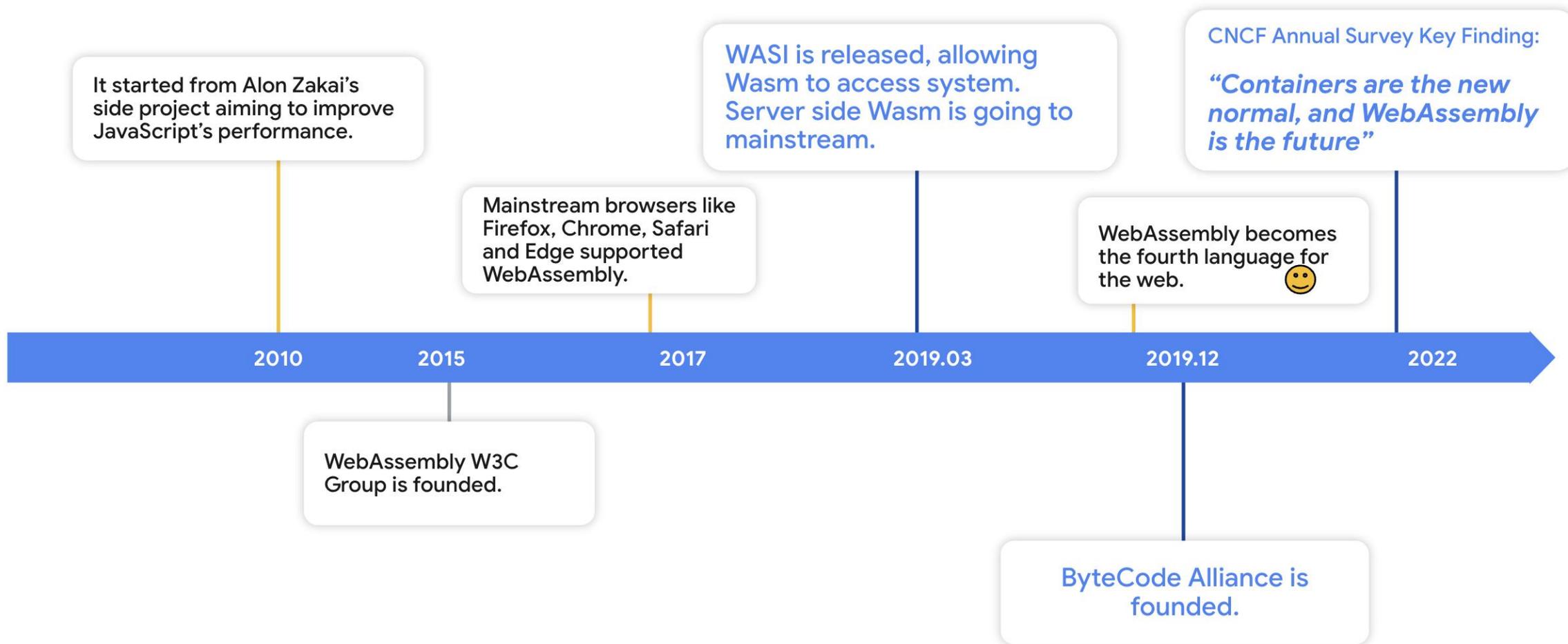
WEBASSEMBLY MICRO RUNTIME (WAMR)



WASMER



● Yes, using or have used
 ● Yes, aware of the runtime
 ● No, not aware



▶ Wasm as an alternative to Linux containers

- 1/100 the size of typical LXC images
- 1000x faster startup time
- Near native runtime performance
- Secure by default and very small attack surface
- Completely portable across platforms
- Programming language agnostic
- Plays well with k8s, service mesh, distributed runtimes etc.

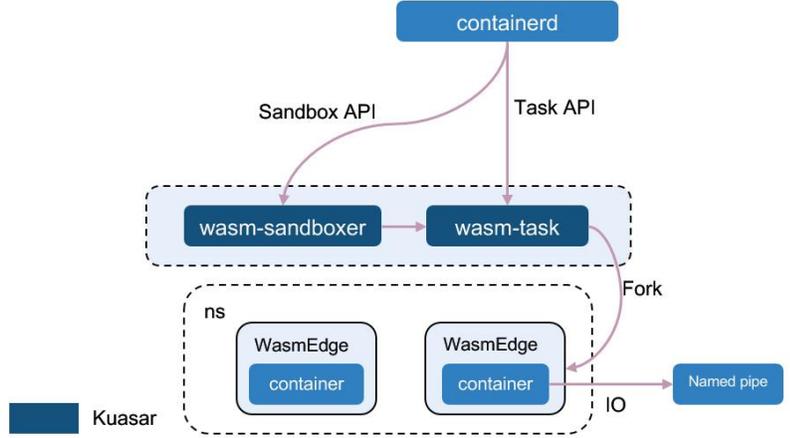
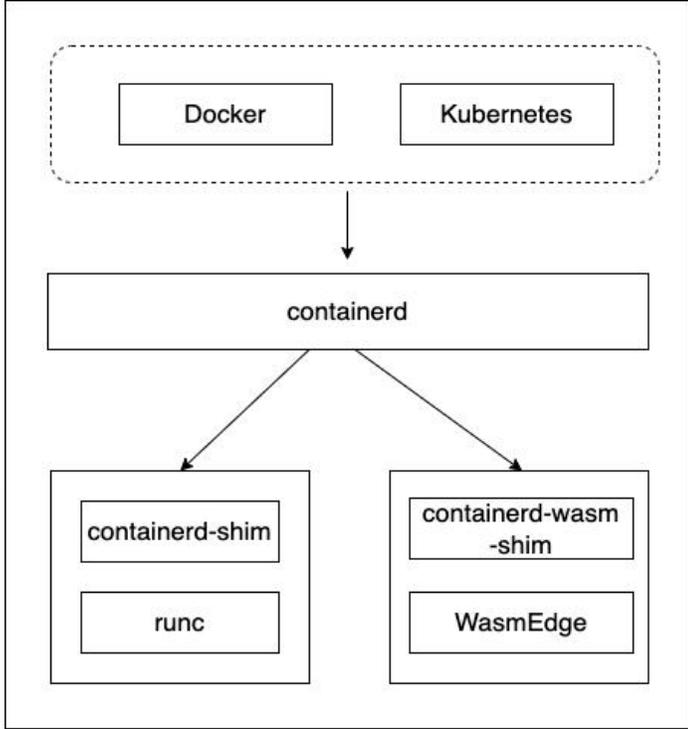
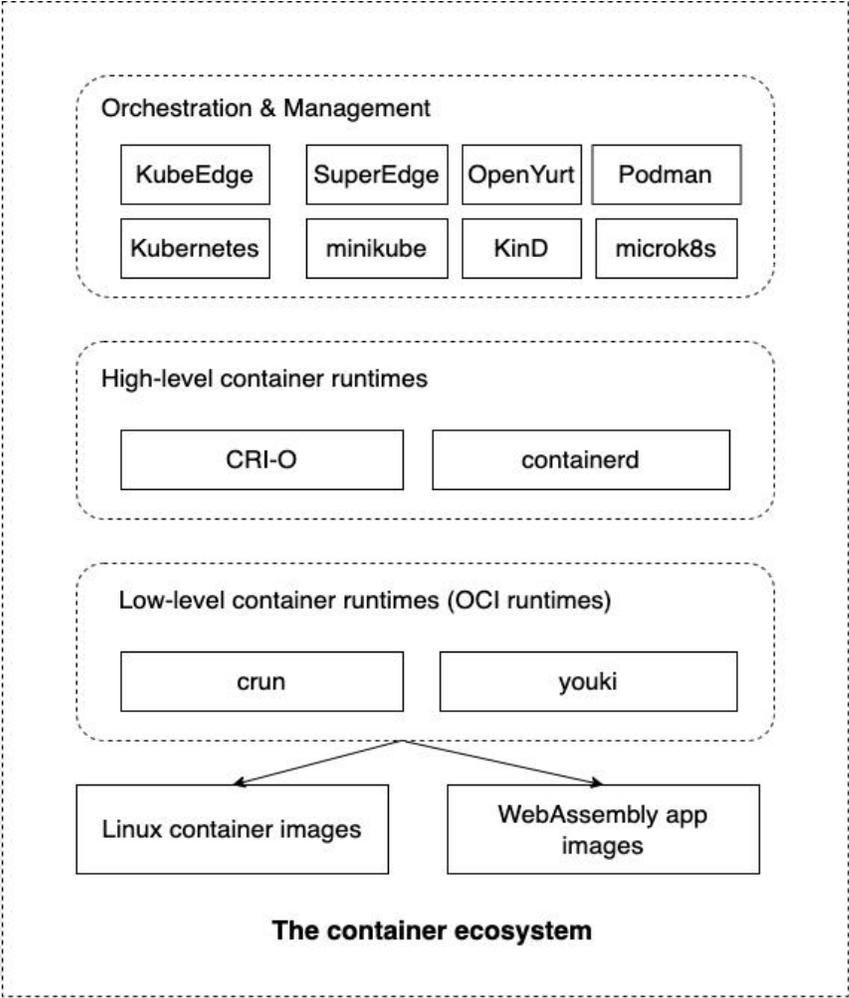


A lightweight, secure, high-performance and extensible WebAssembly Runtime

- Support networking socket and web services
- Support databases, caches, and DOs
- Support AI inference in Tensorflow, OpenVino, PyTorch etc.
- Seamlessly integrates into the existing cloud-native infra
- Support writing wasm programs using JS

<https://github.com/WasmEdge/WasmEdge>



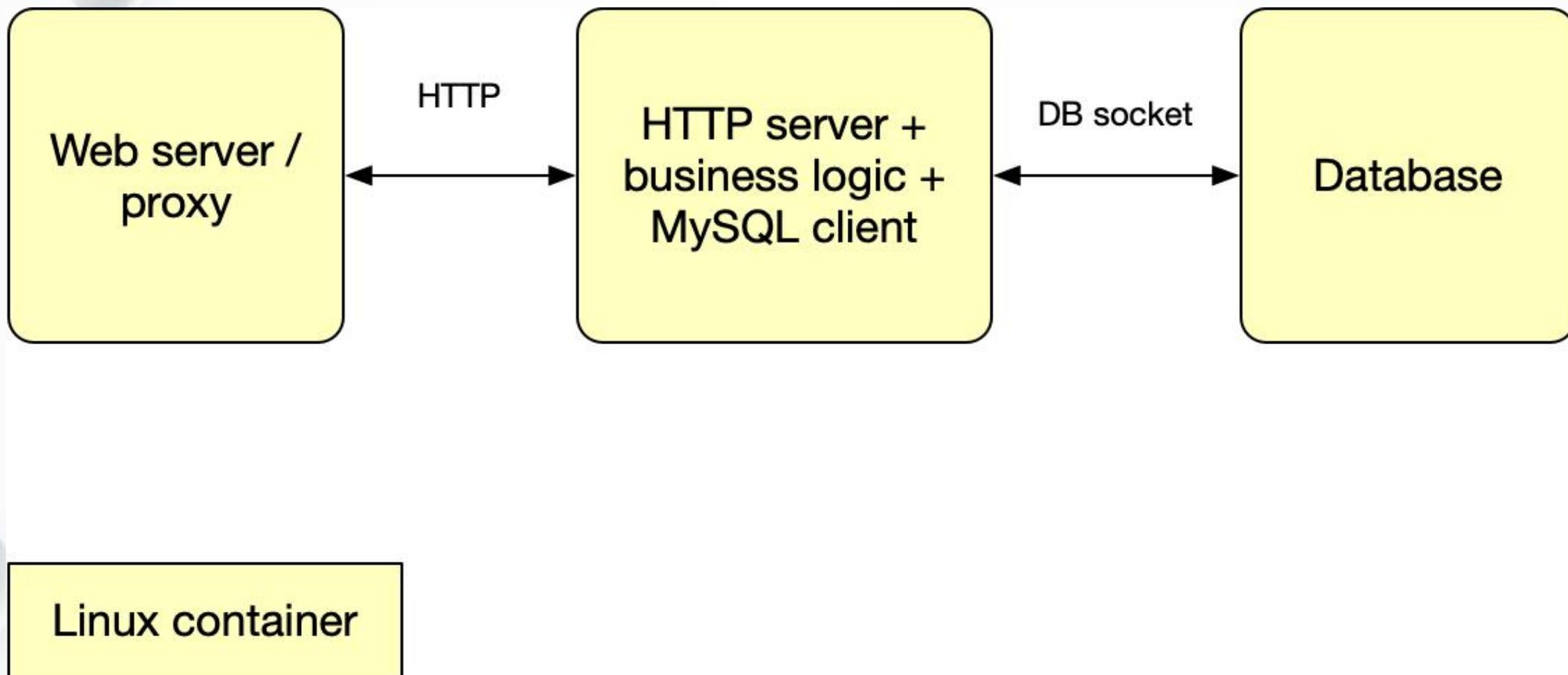




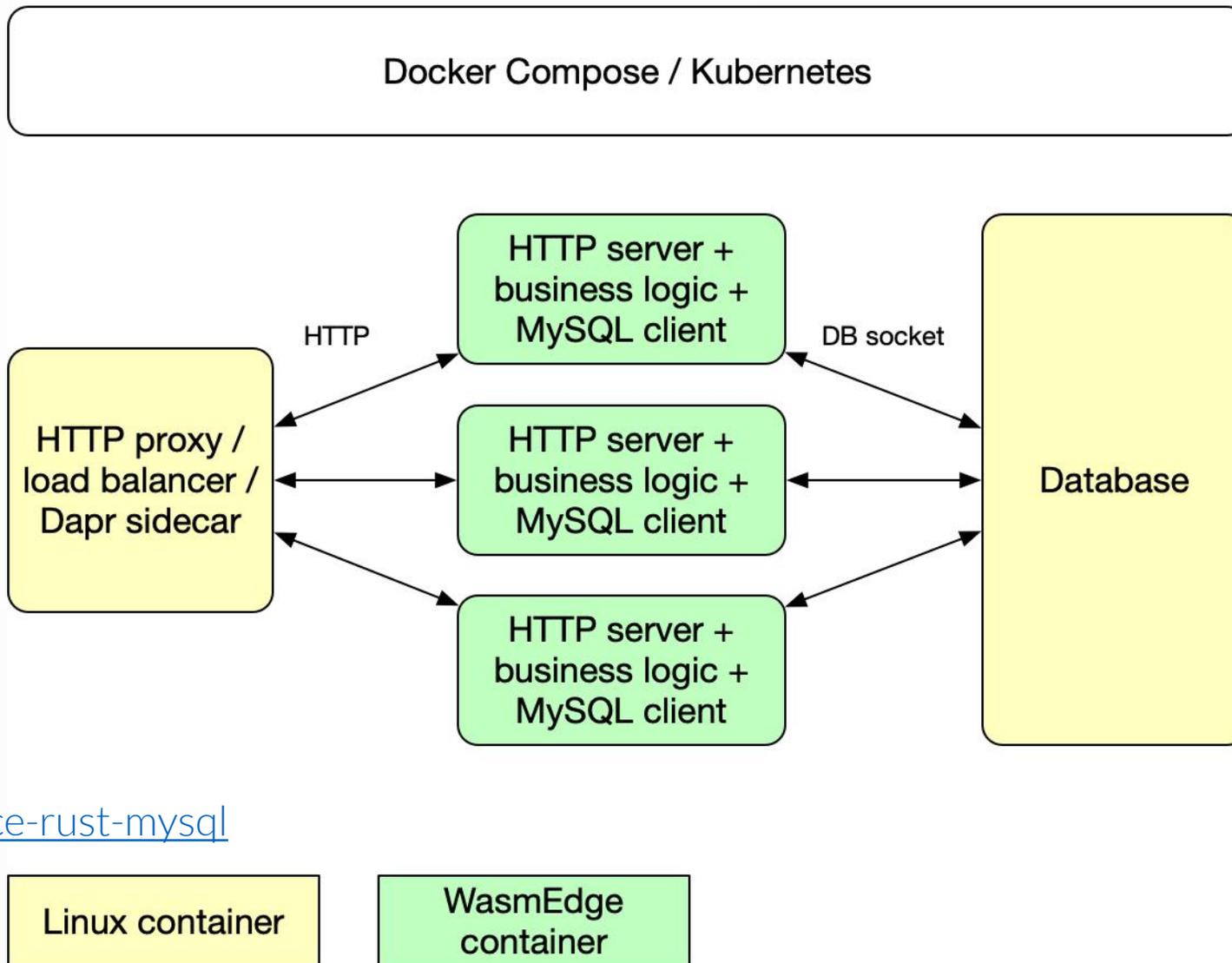
Demo: A database backed microservice

https://wasmedge.org/docs/develop/build-and-run/docker_wasm#create-and-deploy-a-database-driven-microservice-in-rust

How it started



How it's going



<https://github.com/second-state/microservice-rust-mysql>

Your main branch isn't protected Protect this branch

Table of commit history with columns for author, file name, description, and time ago.

About section: A template project for building a database-driven microservice in Rust and run it in the WasmEdge sandbox. Includes Readme, Apache-2.0 license, 98 stars, 4 watching, 12 forks.

Contributors section: 4 contributors listed: juntao Michael Yuan, mikesir87 Michael Irwin.

Star here: https://github.com/WasmEdge/WasmEdge

Light, fast and secure

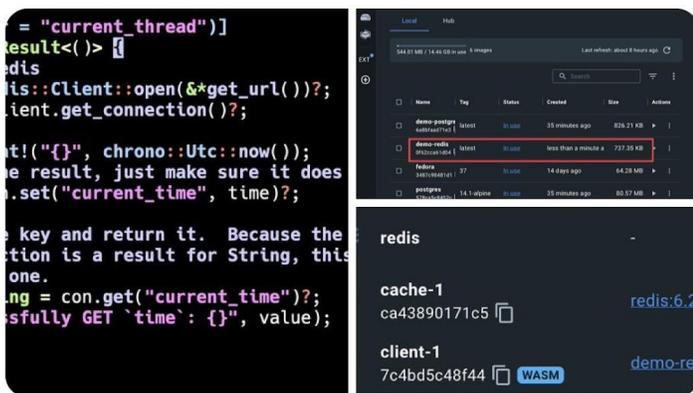


wasmedge
@realwasmedge

A complete Redis app running inside a secure Wasm container managed by Docker + #Wasm. Total app size is 0.7MB and starts in milliseconds. (A comparable Linux container app for #redis is easily 50+MB).

github.com/WasmEdge/wasme...

@Redisinc @Docker



11:55 PM · 2/11/23 from Austin, TX · 12.2K Views

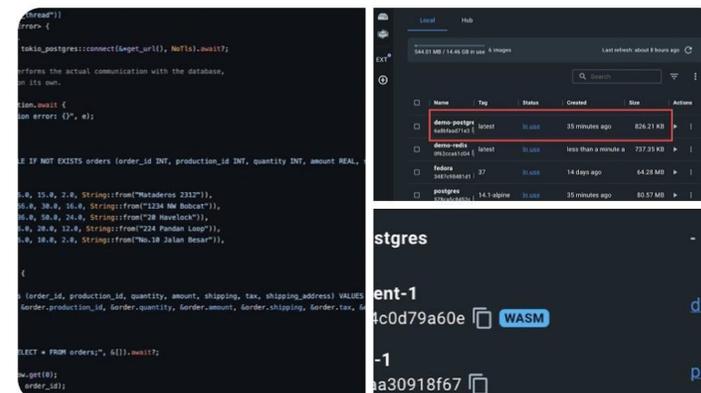


wasmedge
@realwasmedge

A #PostgreSQL client app running inside a secure Wasm container managed by Docker + #Wasm. Total app size is 0.8MB. It runs anywhere and starts in milliseconds. (A comparable Linux container is easily 50MB).

github.com/WasmEdge/wasme...

@PostgreSQL @planetpostgres @Docker



6:50 PM · 2/14/23 from San Francisco, CA · 16K Views

全球开源技术峰会

THE GLOBAL OPEN SOURCE TECHNOLOGY CONFERENCE

Example: Log processing

<https://github.com/second-state/wasmedge-mysql-binlog-kafka>



全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

▶ Wasm use cases

- Microservices
- Runtime for serverless functions
- UDF for databases
- Embedded functions for data streaming frameworks
- **AI and LLM automation**
- Native API for SaaS
- Smart contracts on blockchains

- Plugins provide crucial infrastructure for LLM applications
 - memory (embeddings db)
 - eyes, ears (external event triggers)
 - hands (external API for actions)
- They are “inefficient” serverless functions
 - Simple logic in the function
 - Lots of OAUTH complexity and external connections
 - Extended I/O waiting time
- WasmEdge also works with AI models
 - Supports inference with TF, TFLite, PyTorch, and OpenVINO
 - Google Mediapipe models: <https://github.com/WasmEdge/WasmEdge/issues/2355>
 - Document AI models: https://github.com/sarrah-basta/wasmedge_ai_testing

Merged

[Rust] Optimize host function #2394

apekuss merged 8 commits into `WasmEdge:master` from `apekuss:rust/refactor-function` on Apr 2



alabulei1 commented on Apr 5

Member

Hello, I am a [serverless review bot](#) on [flows.network](#). Here are my reviews of code commits in this PR.

Overall Summary:

This set of patches involves modifications primarily to the Rust bindings for the `wasmedge-sys` component, the related CI workflow, and the test files. The primary focus of these changes is to improve memory management, remove limits on the number of host functions, and update assertions in tests.

Potential issues and errors:

1. Removing the check for host function capacity and the 'MAX_HOST_FUNC_LENGTH' environment variable could lead to excessive resource consumption, affecting system stability and performance.
2. The `test_crazy_function_creation` function may consume a significant amount of system resources and take a long time to finish. Additionally, increasing the loop count from 10 to 1,000,000 in `function.rs` might lead to performance concerns and resource exhaustion.
3. Changing the assertions in tests to use `assert!` instead of `assert_eq!` (and vice versa) could lead to altered test outcomes and potential false test failures.
4. Mutexes and RwLocks in the code can potentially cause deadlocks if not implemented carefully.
5. Updating the Rust CI workflow to restrict the number of test threads to 1 across all platforms could slow down the CI process.

Important Findings:

1. Memory management has been improved through the addition of support for removing real functions in Rust, and the creation of a footprint for host functions.
2. Changes have been made to the Rust bindings CI workflow, including updating the operating systems matrix and Rust compiler versions matrix, as well as modifying the cargo test commands to set the test-threads to 1.
3. The various changes to the test files involve updating assertions and modifying test scenarios, affecting the conditions that are being checked and the performance of the tests.

Given these changes, it's important to ensure that potential issues such as deadlocks, resource exhaustion, and false test failures are mitigated. Also, be cautious when removing limitations on host functions, as it could impact overall system stability and performance.

Open

[WIP] plugin opencvmini #2403

dannysnl wants to merge 11 commits into `master` from `plugin-opencvmini`



juntao commented last month

Member

Hello, I am a [serverless review bot](#) on [flows.network](#). Here are my reviews of changed source code files in this PR.

plugins/wasmedge_opencvmini/opencvmini_base.h

The code snippet provided is a C++ code for a `WasmEdgeOpenCVMini` class. Here are some observations:

1. `#pragma once` is used to prevent multiple inclusions of the same file which is fine.
2. `SPDX-License-Identifier` and `SPDX-FileCopyrightText` are used to indicate the license.
3. The `opencvmini_env.h` header file and two other header files are included.
4. A template class named `WasmEdgeOpenCVMini` that inherits from the `Runtime::HostFunction<T>` class is defined.
5. The constructor initializes the inherited class `HostFunction<T>` with a value zero and an instance of `WasmEdgeOpenCVMiniEnvironment`.
6. The class declaration is enclosed within two namespace blocks, `WasmEdge` and `Host`.

There are no visible problems with the provided code snippet. This seems to be a class definition and more code may be added later.

The patch doesn't introduce any changes to the existing code. It is the same as the original code snippet.

plugins/wasmedge_opencvmini/opencvmini_env.cpp

This code snippet defines a plugin for the OpenCVMini library in the context of the WasmEdge project. The code sets up a `WasmEdgeOpenCVMiniEnvironment` class that derives from `Plugin::HostEnvironment`. There is a module `WasmEdgeOpenCVMiniModule` that is created and stored in the `Plugin::PluginModule::ModuleDescriptor` object `Descriptor`. This object includes the name of the plugin, its version, and other related information. The plugin descriptor contains only one module, namely `WasmEdgeOpenCVMiniModule`.

The code snippet seems to be correct and without any obvious problems. However, there may be issues with this code depending on the context in which it is used, which cannot be reviewed from this snippet alone.

The patch has no substantive changes. It is simply reformatting the code, reorganizing the newlines between some of the statements, and adjusting the indentation. Overall, it looks like style and formatting updates, and it does not introduce any functional changes to the code logic.

全球开源技术峰会

THE GLOBAL OPENSOURCE TECHNOLOGY CONFERENCE

✕ Add function 'check_prime' for node's crypto API by Aviii06 · Pull Request #82

📄 Changes from all commits ▾ File filter ▾ Conversations ▾ ⚙️

🔍 Filter changed files

▾ example_js/node

📄 main.mjs +

📄 package.json +

📄 rollup.config.js +

▾ src

▾ 49 █████ src/internal_module/crypto.rs 📄

```
16 +   if n <= 1 {
17 +       return JsValue::Bool(false);
18 +   }
19 +   let limit = (n as f64).sqrt() as i32;
20 +   for a in 2..limit {
21 +       if n % a == 0 {
22 +           return JsValue::Bool(false);
23 +       }
24 +   }
25 +   JsValue::Bool(true)
26 + }
27 +
```

< second-state/wasmedge-quickjs Add function 'check_prime' for node's crypto API



Potential problems:

1. The `check_prime` function can be optimized further, as it checks for divisibility with even numbers after 2, which isn't necessary.



<https://github.com/flows-network/github-pr-summary>

1. **Load the code review bot template in flows.network.** The template contains the source code for the bot itself. We will clone the source code to your own GitHub account so that you can modify and customize it later. Click on Create and Deploy.
2. **Authorize bot access to GitHub.** The `github_owner` and `github_repo` point to the target GitHub repo where the bot will review PRs. Click on Authorize to give the repo the necessary permissions in GitHub.
3. **Give the bot your OpenAI API key.** If you have saved API keys in the past, you can skip this step and reuse these keys.

THANKS