Getting Started with Serverless Wasm with Spin
What will you take away?

First-hand experience using server-side Wasm

- Familiarity with Spin framework features including
  - HTTP Routing
  - Serverless AI
  - Key Value Store
  - SQLite / NoOps DB

- Deployment of Spin application to multiple targets including
  - Locally
  - Fermyon Cloud
Agenda

• Introduction to WebAssembly
• Introduction to Spin
• Overview of the workshop
• Live-coding walkthrough of the workshop
Setting up a codespace

Workshop: github.com/fermyon/workshops
Poll: Have you used WebAssembly before?

- Yes, a lot!
- I’ve tried it out, but that’s about it.
- Not at all.
What are the three most important things to know about WebAssembly?

• It is a specification
• Of a binary instruction format, designed as a portable compilation target, and...
• Wasm is an abbreviation for it (also it’s not an acronym 😊)
Compile and Run

Code → Wasm → VM
Four reasons to care about Wasm

- Security — Sandboxed execution environment.
- Performance — Near native execution speed.
- Polyglot — Supports a wide array of languages.
- Portability — Cross platform and cross architecture.
WASI: a system interface for the Wasm platform

Make it portable
- Code outside of a browser needs a way to talk to a conceptual operating system.
- E.g., files, sockets, clocks, random numbers and many more higher-level types of resources

Make it secure
- Preserve in-browser security model through WASI’s Capability based security model
Language Support

- Guide tracks support for compiling a language to WebAssembly.

- Three sections:
  - Support for the top 20 languages
  - WebAssembly-specific languages
  - Other notable languages.

<table>
<thead>
<tr>
<th>Language</th>
<th>Core</th>
<th>Browser</th>
<th>WASI</th>
<th>Spin SDK</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Python</td>
<td>✔</td>
<td>✗</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Java</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>PHP</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>CSS</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>C# and .NET</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>C++</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>TypeScript</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
<tr>
<td>Ruby</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>C</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>Swift</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✗</td>
</tr>
<tr>
<td>R</td>
<td>✗</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

https://developer.fermyon.com/wasm-languages/webassembly-language-support/
### JavaScript runtimes

Designed to complement and run alongside JS to share functionality between JS and Wasm

<table>
<thead>
<tr>
<th>Browser</th>
<th>Server-side</th>
</tr>
</thead>
<tbody>
<tr>
<td>V8</td>
<td>node</td>
</tr>
<tr>
<td>Spidermonkey</td>
<td>Bermudez</td>
</tr>
</tbody>
</table>

### WASI runtimes

Designed to be independent of browsers, so it doesn’t depend on Web APIs or JS, and isn’t limited by the need to be compatible with JS

- Wasmtime
What are good use-cases for Wasm?

**Cloud**
- Functions-as-a-Service Frameworks
- Extensibility with the component-model

**Plug-ins**
- User-Defined Functions for databases
- Bring-your-own-code in SaaS platforms

**IoT**
- System resource usage
- No dependencies to carry along

**Developer and Operator experiences**
- Quick start-up time
- Size of workload
- Security model
- Portability
Spin is the developer tool for building serverless Wasm applications
SPIN 1.0

The Developer Tool for Serverless WebAssembly

Bring Your Code
- JS
- Python
- R
- Go
- TypeScript
- .NET
- C#

Supports many of the most popular programming languages

spin new

Serverless Made Simple
- Docker Hub
- GHCR
- Sigstore
- Redis
- Key/Value Storage
- HashiCorp Vault

Works with developer tools & registries. Adds internal & external storage

spin build

spin up

Deploy & Run
- Kubernetes Cluster with Containerd
- FERMYON CLOUD
- http://localhost:3000

spin cloud deploy
What's new in Spin?

**Serverless AI**
Execute inferencing for LLMs directly from serverless apps.

**SQLite Databases**
Spin has a built-in database, which is always available - no Ops required.

**Key/Value Store**
Quickly persist data in your apps with Spin's in-built local KV store.

**HTTP & Redis Triggers**
Spin has a built-in HTTP web server and pub-sub Redis triggers, routing requests and messages to components.

**Relational Database Storage**
Bring your own DB support for MySQL and PostgreSQL, where you host and manage the database outside of Spin.

**Variables & Secrets**
Dynamic app variables mean a simpler experience for rotating secrets, updating API endpoints, and more.
Discover and share content on Spin Up Hub
The quickest way to deploy and manage your serverless WebAssembly apps.

Architected for sub-millisecond cold starts and high-volume time-slicing of instances.
Fermyon Serverless AI

- **Fermyon**: 0.01s infra startup to execution, 3.54s execution (varies depending on the code being run)
- **OctoML**
- **Banana.dev**
- **Replicate**
- **Beam**
- **RunPod**

- Cold start time: (time from request to start of inferencing)
- Inference time: (depends on the code)
FERMYON

Let’s get hands-on
Spinning a Magic 8 Ball

A Magic 8 Ball
*Returns a random response to a question*

A MAGIC 8 Ball
*Remembers the responses to questions*

A MAGIC AI-GHT Ball
*Returns a sophisticated response to a question, backed by a LLM*
Workshop

https://github.com/fermyon/workshops/blob/main/magic-8-ball/README.md

*Enhanced with Serverless AI*
Workshop Flow

- **00** - Setup
- **01** - Getting Started with Spin
- **02** - Building a JSON API with Spin
- **03** - Enhancing JSON API with Fermyon Serverless AI
- **04** - Running your front-end application with Spin
- **05** - Persisting non-relational data with Spin's key/value store
- **06** - Deploying to Fermyon Cloud
- **07** - Bonus Exercise: Using an external database
- **08** - Embeddings and SQLite

https://github.com/fermyon/workshops
FERMYON

What did you think?

bit.ly/fermyondiscord

Follow along on Twitter

@fermyontech
@spinframework