FERMYON

WebAssembly and Containers

Nov 14, 2023

Objective and Agenda

- First-hand experience with server-side WebAssembly and Kubernetes.

- 10 min. introduction to server-side WebAssembly
- Follow along the tutorial

- GitHub repository with all the content you need to complete the tutorial:
- https://github.com/fermyon/workshops/blob/main/wasm-and-containers/

FERMYON

What is WebAssembly?

- It is a specification of a binary instruction format, designed as a portable compilation target
- Originates from the browser, now also available outside
- Language support is emerging and stabilizing
- Wasm is just another name for it

4 things making WebAssembly great

Binary Size	Startup Time	Portability	Security
Rust hello-world ~2MB AOT compiled ~300KB Basic Spin http api ~2.3MB JIT ~1.1MB AOT	Startup times comparable with natively compiled code Only 2.3x slower than native*	Build once, run anywhere! Same build (JIT) works across OS and platform arc	Sandboxed execution Capability based security model

*https://00f.net/2023/01/04/webassembly-benchmark-2023/

FERMYON

What are good use-case for WASI?

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

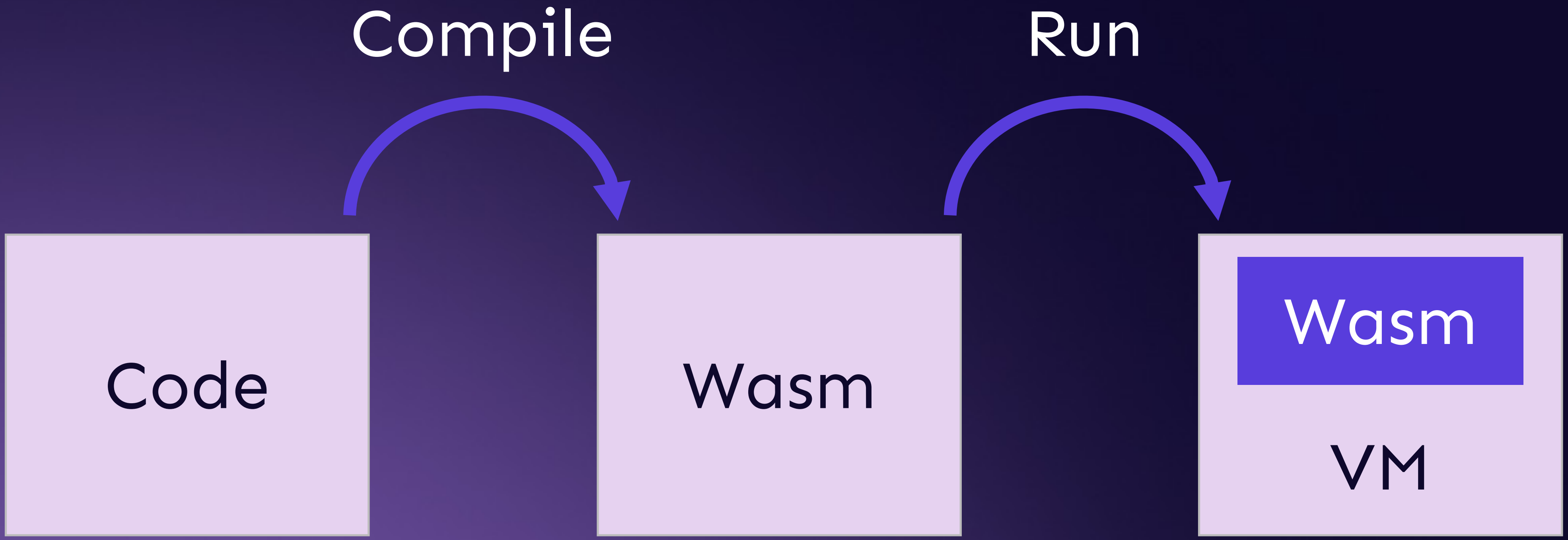
No dependencies to carry along

Developer and Operator experiences

Quick start-up time
Size of workload
Security model
Portability

https://wasi.dev/

Compile and Run



Runtimes and VMs

JavaScript runtimes

Designed to complement and run alongside JavaScript

- V8 (Chromium browsers)
- SpiderMonkey (Firefox)
- Nitro (WebKit)

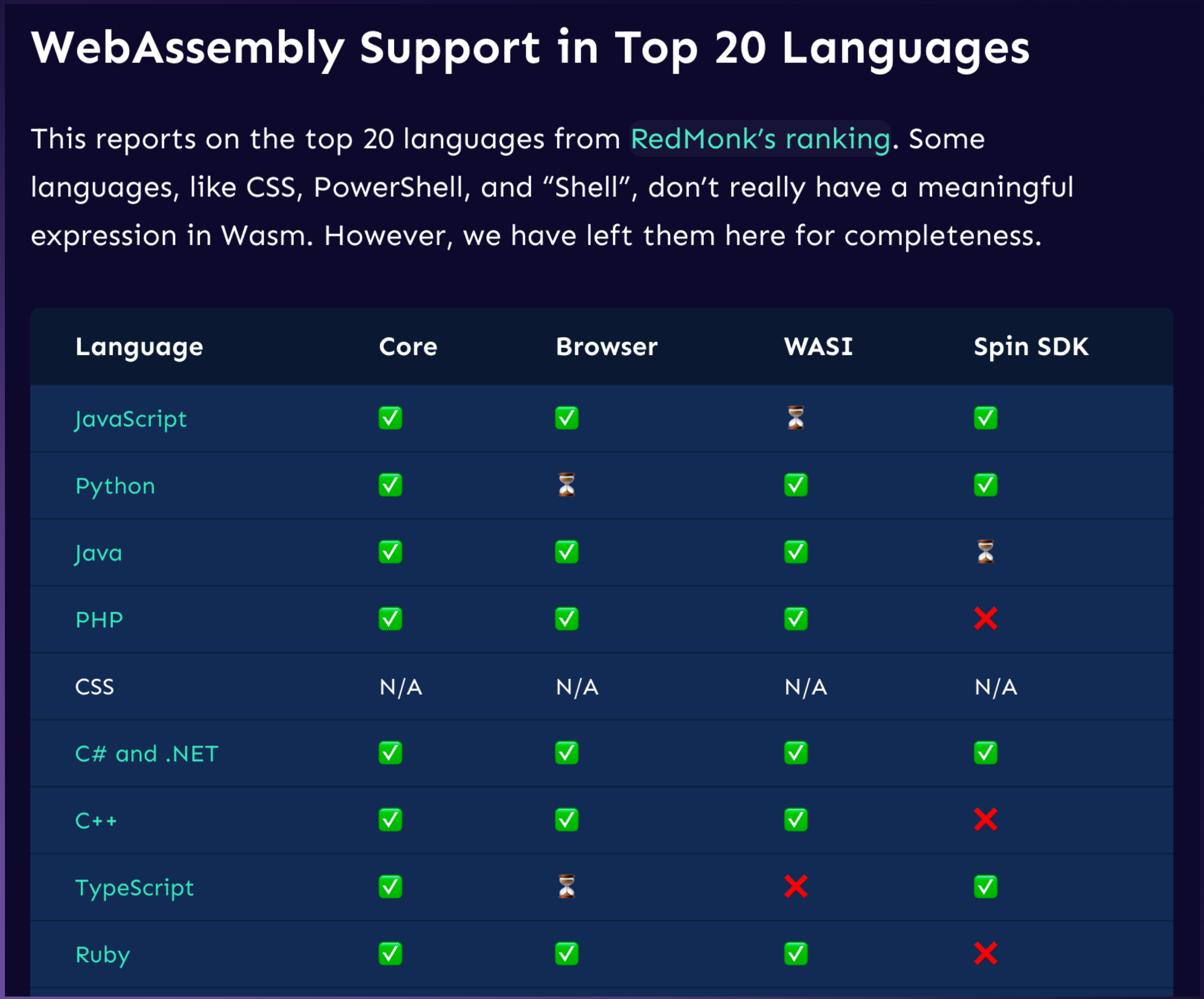
WASI runtimes

Designed to be independent of browsers

- Wasmtime
- WasmEdge

Experimental:
Node.js and Bun

Compilation and Language Support



WebAssembly Support in Top 20 Languages

This reports on the top 20 languages from RedMonk's ranking. Some languages, like CSS, PowerShell, and "Shell", don't really have a meaningful expression in Wasm. However, we have left them here for completeness.

Language	Core	Browser	WASI	Spin SDK
JavaScript	✅	✅	⏳	✅
Python	✅	⏳	✅	✅
Java	✅	✅	✅	⏳
PHP	✅	✅	✅	❌
CSS	N/A	N/A	N/A	N/A
C# and .NET	✅	✅	✅	✅
C++	✅	✅	✅	❌
TypeScript	✅	⏳	❌	✅
Ruby	✅	✅	✅	❌

https://www.fermyon.com/wasm-languages/webassembly-language-support/

3 easy options for running WebAssembly

Use a runtime

```
> cargo build --target wasm32-wasi --release
> wasmtime target/wasm32-wasi/release/my_app.wasm
```

Use a framework

```
> spin build –f my_app/spin.toml
> spin up –f my_app/spin.toml
```

Use runwasi with Kubernetes

```
> docker build --platform wasi/wasm –t my_app .
> docker push ghcr.io/my_name/my_app
> kubectl apply –f ./runtimeclass.yaml
> kubectl apply –f ./my_app.yaml
```

Dockerfile comparison

~/spin_webassembly/Dockerfile

```
FROM scratch

COPY spin.toml .
COPY target/wasm32-wasi/release/hello_world.wasm
target/wasm32-wasi/release/hello_world.wasm
```

~/python_flask/Dockerfile

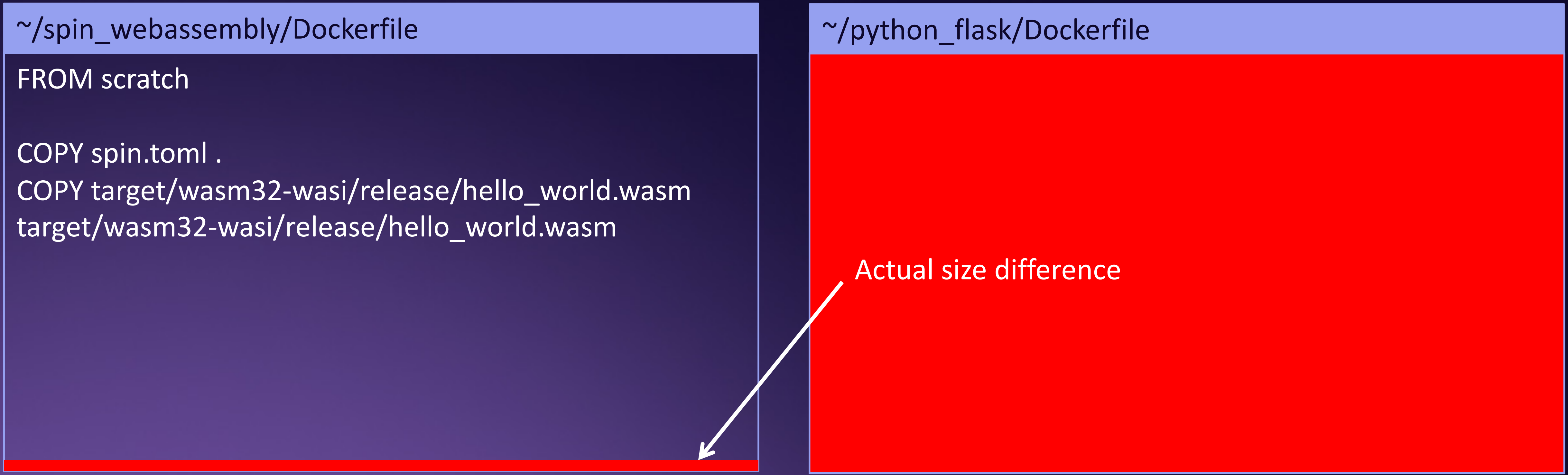
```
FROM python:3.10-alpine
WORKDIR /app

COPY requirements.txt /app

RUN --mount=type=cache,target=/root/.cache/pip \ pip3
install -r requirements.txt
COPY . /app

ENTRYPOINT ["python3"]
CMD ["app.py"]
```

Dockerfile comparison



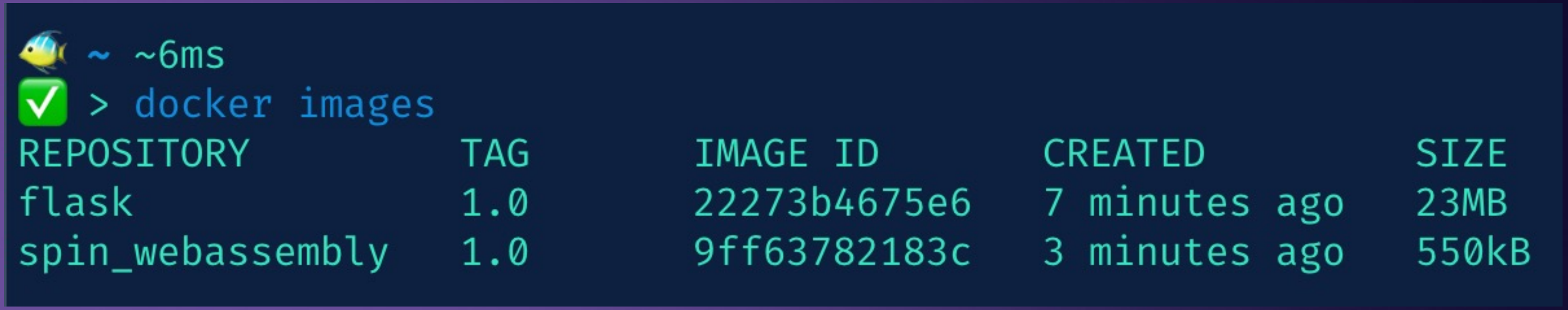
~/spin_webassembly/Dockerfile

```
FROM scratch

COPY spin.toml .
COPY target/wasm32-wasi/release/hello_world.wasm
target/wasm32-wasi/release/hello_world.wasm
```

~/python_flask/Dockerfile

Actual size difference



```
🐠 ~ ~6ms
✅ > docker images
REPOSITORY         TAG        IMAGE ID       CREATED          SIZE
flask              1.0        22273b4675e6   7 minutes ago    23MB
spin_webassembly   1.0        9ff63782183c   3 minutes ago    550kB
```

Modules

→ Part 1: Build an WebAssembly application using Spin

Part 2: Run your Spin app in a container

Part 3: Deploy to Kubernetes

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

-> wasm-and-containers

SPIN

+

FERMYON

CLOUD

Spin is the developer tool for building WebAssembly microservices and web applications

Single binary tool

One tool for development and runtime

Many languages

Go, Rust, C#, Java, JavaScript, or any with Wasm+WASI support

Easy to get started

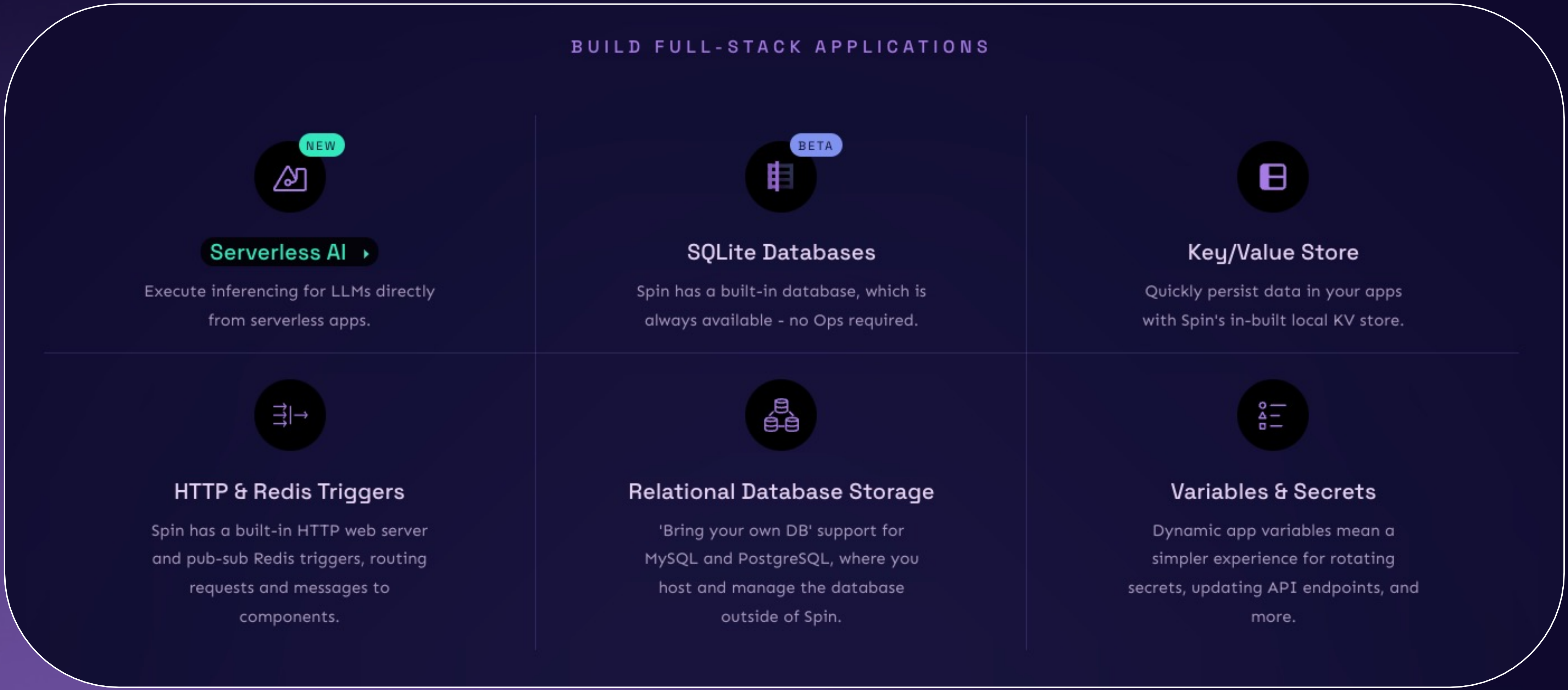
Use the template library, or provide your own and share components

No boilerplate required

Spin triggers get you right to the important part

Work with data

Use Postgres, Redis and file storage to persist your data



BUILD FULL-STACK APPLICATIONS

NEW

Serverless AI ▸

Execute inferencing for LLMs directly from serverless apps.

BETA

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.

FERMYON

Modules

Part 1: Build an WebAssembly application using Spin

→ Part 2: Run your Spin app in a container

Part 3: Deploy to Kubernetes

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

-> wasm-and-containers

Modules

Part 1: Build an WebAssembly application using Spin

Part 2: Run your Spin app in a container

→ Part 3: Deploy to Kubernetes

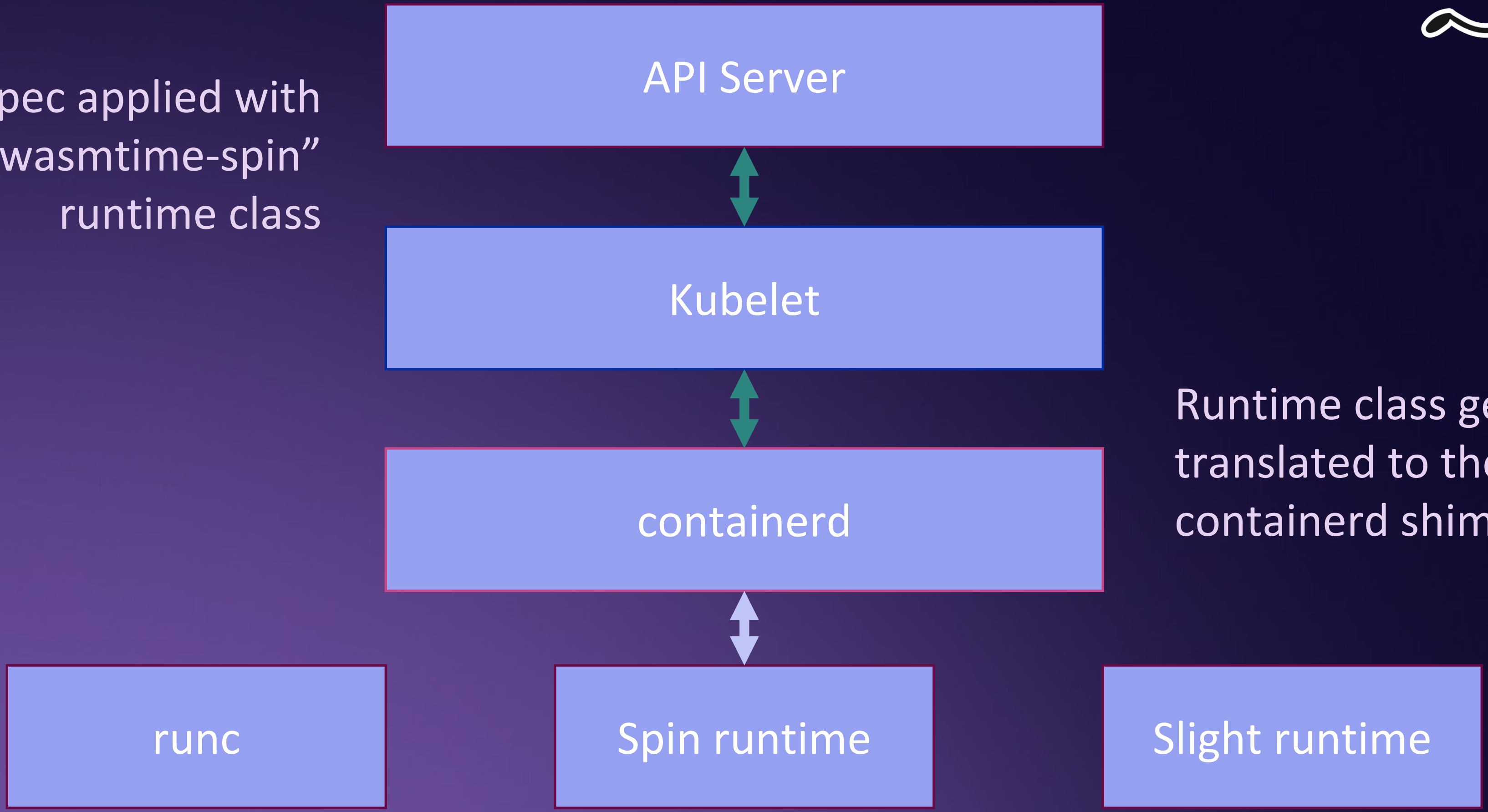
https://github.com/fermyon/workshops/

-> wasm-and-containers

FERMYON

Running Wasm in Kubernetes

Pod spec applied with "wasmtime-spin" runtime class



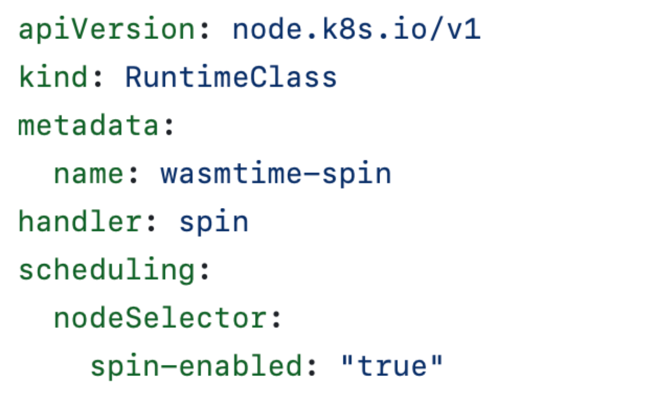
Runtime class gets translated to the containerd shim handler

Spin handler

These containerd shims are just binaries on the $PATH

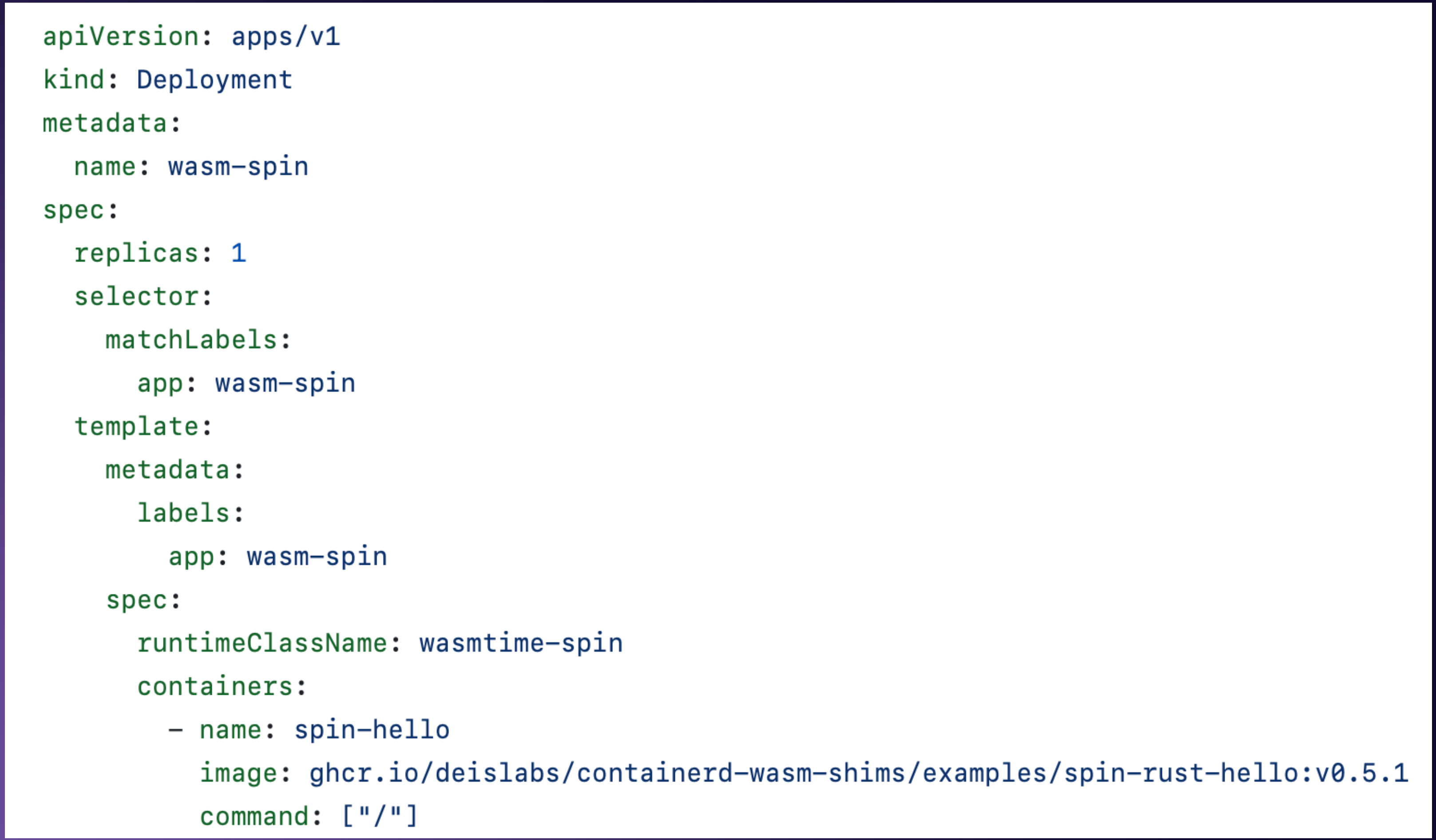
Running Wasm in Kubernetes

Runtime Class



```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: wasmtime-spin
handler: spin
scheduling:
  nodeSelector:
    spin-enabled: "true"
```

Spin pod deployment



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wasm-spin
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wasm-spin
  template:
    metadata:
      labels:
        app: wasm-spin
    spec:
      runtimeClassName: wasmtime-spin
      containers:
        - name: spin-hello
          image: ghcr.io/deislabs/containerd-wasm-shims/examples/spin-rust-hello:v0.5.1
          command: ["/"]
```

Thank You!

@mikkelhegn

mikkel@fermyon.com

Join our Discord server!

Check out the hub!

FERMYON