golem

FOSDEM 2020

Facilitating deterministic distributed computation with WASI

JAKUB KONKA

Who am I?

“Who Are You”

My name is Jakub Konka

R&D Researcher at Golem Factory

Regular contributor to Wasmtime and WASI, and one of the authors of wasi-common library

Member of WebAssembly CG

@kubkon

✉ kubkon@golem.network

✉ kubkon@jakubkonka.com

@kubkon

What is WASI?

What is WASI?

01

WASI - **WebAssembly System Interface**

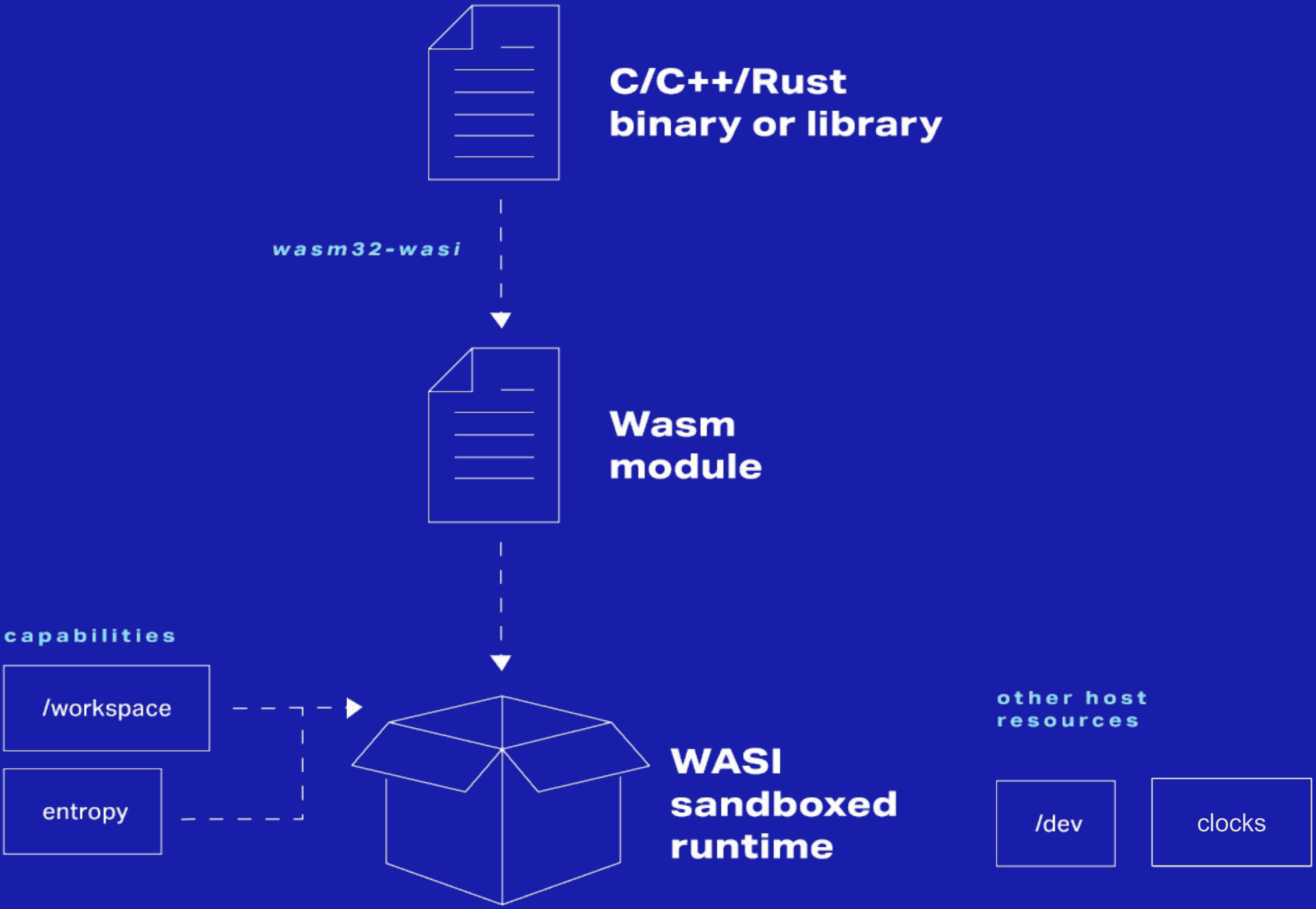
02

Standardisation led by **Bytecode Alliance**

03

Capability-based security - safe and portable access to host’s resources

Source: https://wasi.dev



C/C++/Rust binary or library

wasm32-wasi

Wasm module

capabilities

/workspace

entropy

WASI sandboxed runtime

other host resources

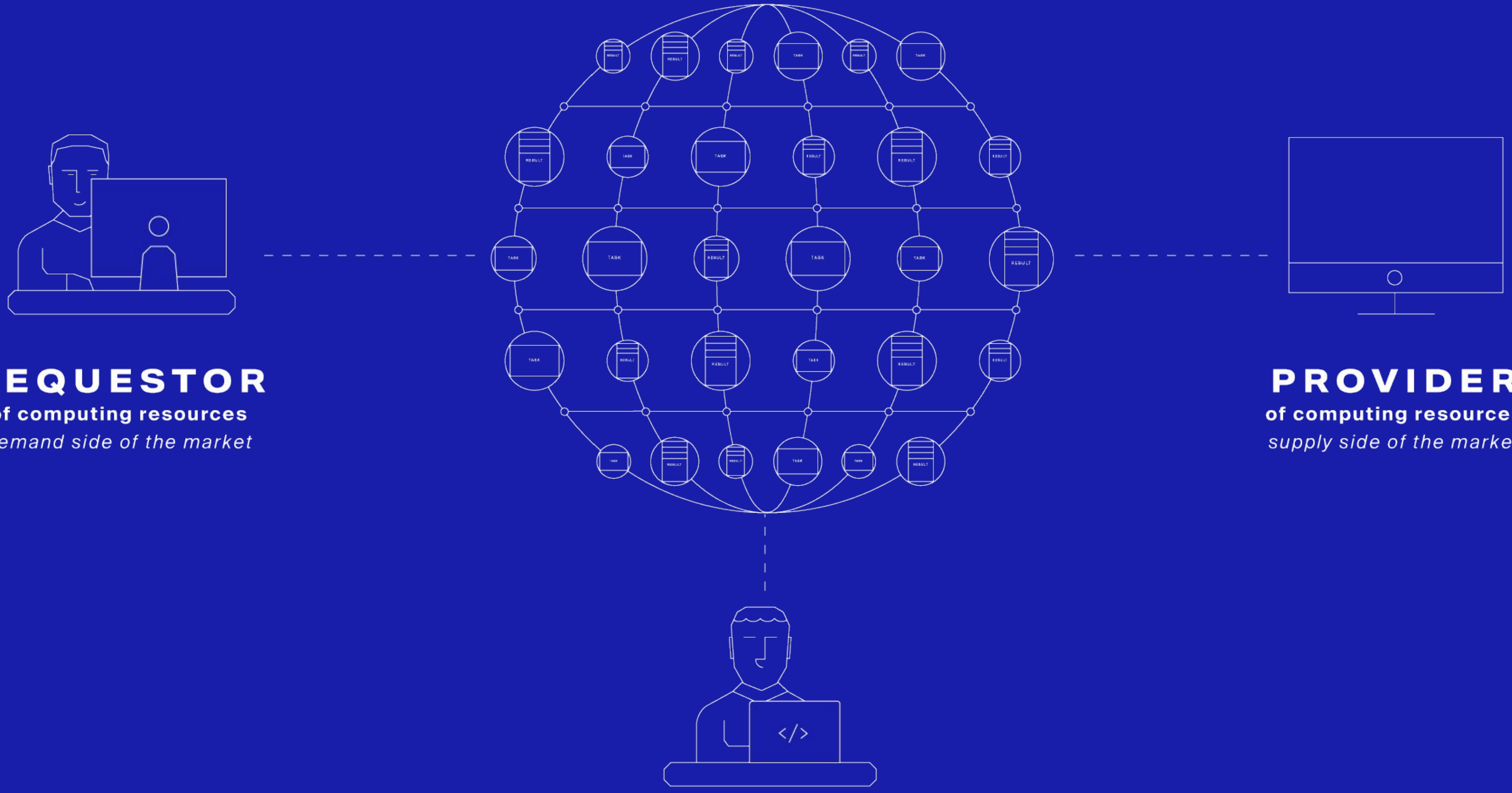
/dev

clocks

Allowed ✓	Forbidden ✗
`File::create(“/workspace/new”)?;`	`File::open(“/dev/null”)?;`
. . .	. . .
`rand::thread_rng();`	`let now = SystemTime::now();`
. . .	. . .

What is the setting?

Meet the Golem Network



REQUESTOR

of computing resources

demand side of the market

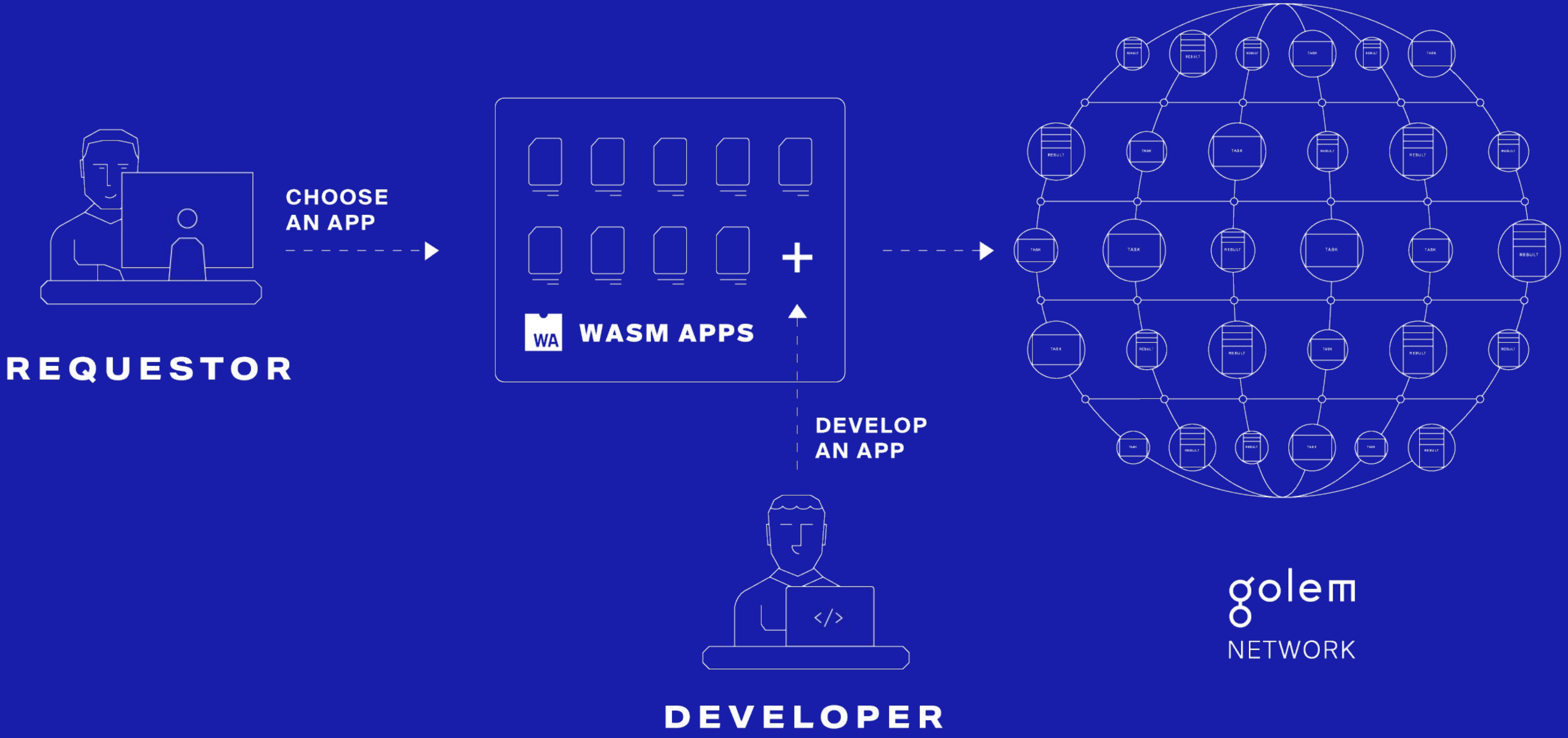
PROVIDER

of computing resources

supply side of the market

DEVELOPER

Meet the Golem Network



CHOOSE AN APP

REQUESTOR

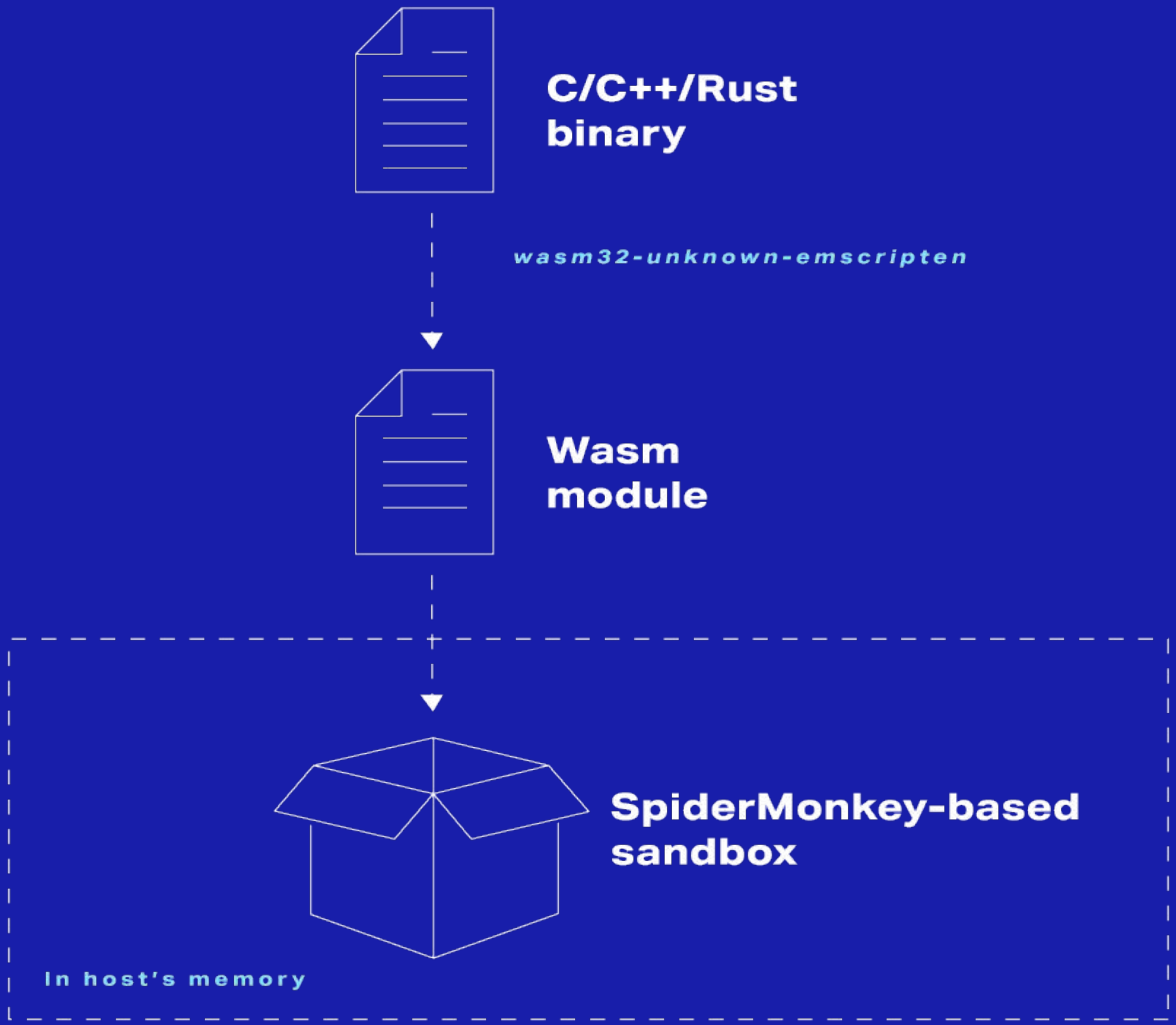
WASM APPS

DEVELOP AN APP

DEVELOPER

golem
NETWORK

Wasm sandbox in Golem



C/C++/Rust binary

wasm32-unknown-emscripten

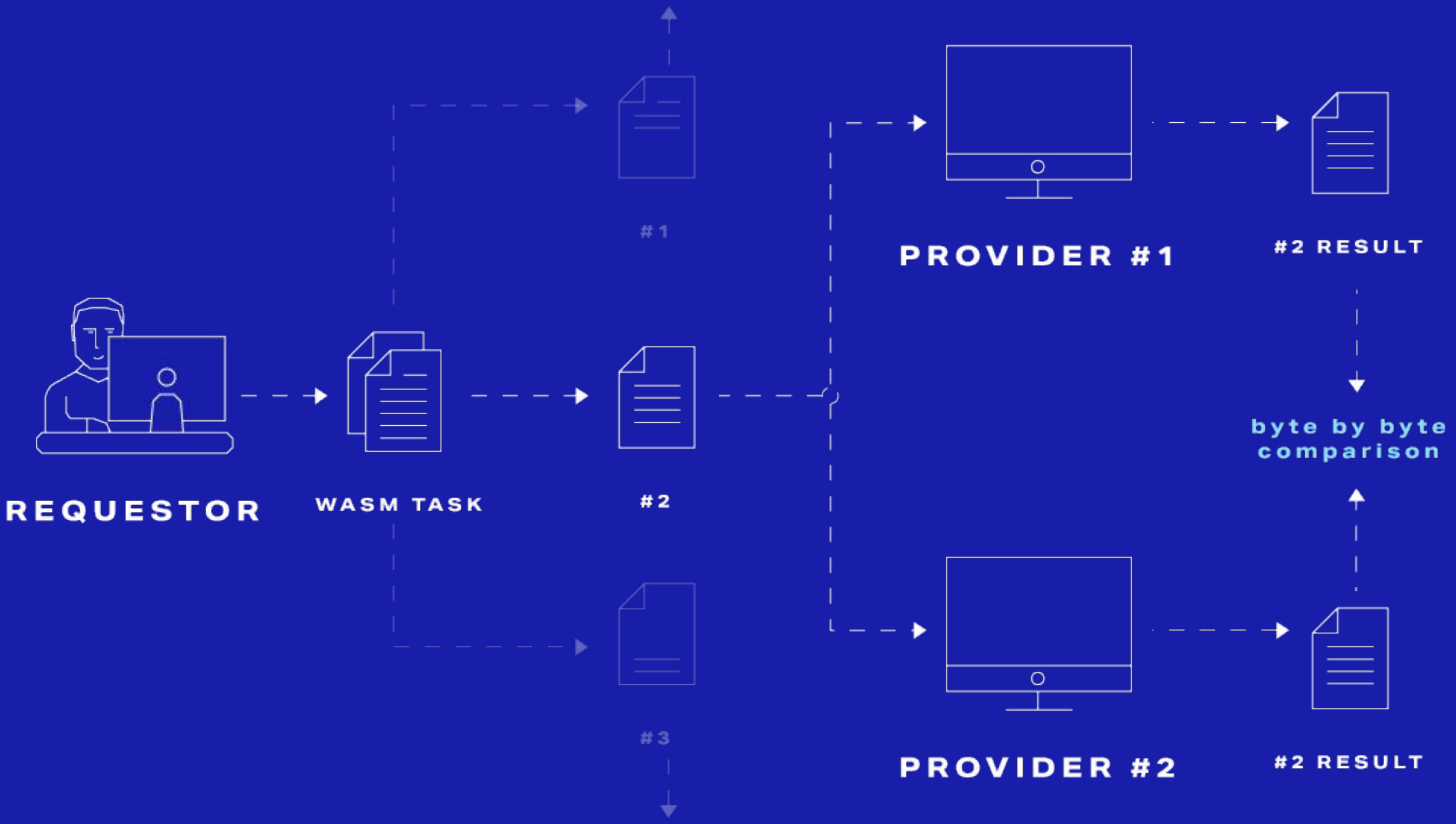
Wasm module

SpiderMonkey-based sandbox

In host's memory

golem

Verification by redundancy



Is WASI deterministic?

Sources of nondeterminism in WASI

01

Access to random device

- Provided by `random_get`
- Will get its own module
- Will require a capability

```
unsafe fn random_get(
    buf: *mut u8,
    buf_len: Size,
) -> Result<(), Errno> {
    // call `getrandom` to access
    // host's entropy source, and
    // populate input `buf`
}
```

Sources of nondeterminism in WASI

02

Access to system clocks

- Provided by `clock_time_get`
- Will get its own module
- Will require a capability

```
unsafe fn clock_time_get(
    id: Clockid,
    precision: Timestamp
) -> Result<Timestamp, Errno> {
    // call `clock_gettime` to
    // get current host’s time
    // etc.
}
```

Sources of nondeterminism in WASI

03

File atim/mtim/ctim stats

- Part of `Filestat` struct
- Inherently set by the host when file is created/modified
- Can be read by a module via `fd_filestat_get` or `path_filestat_get`

```
unsafe fn fd_filestat_get(
    fd: Fd
) -> Result<Filestat, Errno> {
    // call `fstat` to
    // get info on the underlying
    // host’s fd
}
```

```
struct Filestat {
    dev: Device,
    ino: Inode,
    filetype: Filetype,
    nlink: Linkcount,
    size: Filesize,
    atim: Timestamp,
    mtim: Timestamp,
    ctim: Timestamp,
}
```

Sources of nondeterminism in WASI

04

Listing contents of a directory

- Provided by `fd_readdir`
- Order of entries dependent on the host *and* the filesystem used

```
unsafe fn fd_readdir(
    fd: Fd,
    buf: *mut u8,
    buf_len: Size,
    cookie: Dircookie,
) -> Result<Size, Errno> {
    // call `readdir` iteratively
    // to get enough dir entries
    // starting from `cookie` to
    // fully populate `buf`
}
```

golem

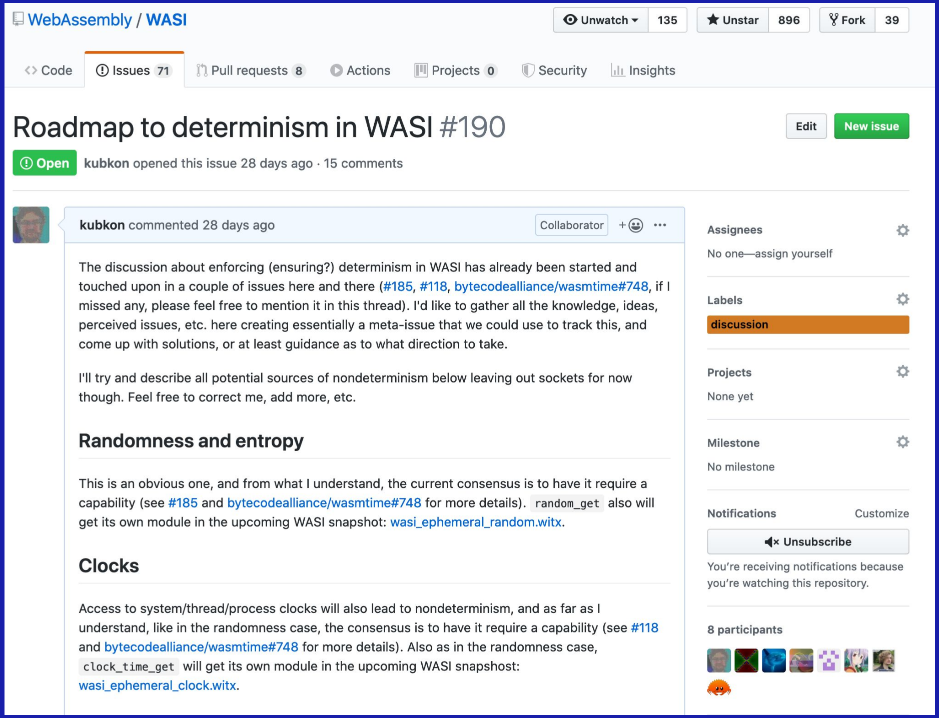
Sources of nondeterminism in WASI

05

And the list goes on!

Encourage you to join the ongoing discussion here:

WebAssembly/WASI/issues/190



Roadmap to determinism in WASI #190

Open kubkon opened this issue 28 days ago · 15 comments

kubkon commented 28 days ago

The discussion about enforcing (ensuring?) determinism in WASI has already been started and touched upon in a couple of issues here and there (#185, #118, bytecodealliance/wasmtime#748, if I missed any, please feel free to mention it in this thread). I'd like to gather all the knowledge, ideas, perceived issues, etc. here creating essentially a meta-issue that we could use to track this, and come up with solutions, or at least guidance as to what direction to take.

I'll try and describe all potential sources of nondeterminism below leaving out sockets for now though. Feel free to correct me, add more, etc.

Randomness and entropy

This is an obvious one, and from what I understand, the current consensus is to have it require a capability (see #185 and bytecodealliance/wasmtime#748 for more details). `random_get` also will get its own module in the upcoming WASI snapshot: wasi_ephemeral_random.witx.

Clocks

Access to system/thread/process clocks will also lead to nondeterminism, and as far as I understand, like in the randomness case, the consensus is to have it require a capability (see #118 and bytecodealliance/wasmtime#748 for more details). Also as in the randomness case, `clock_time_get` will get its own module in the upcoming WASI snapshot: wasi_ephemeral_clock.witx.

Assignees
No one—assign yourself

Labels
discussion

Projects
None yet

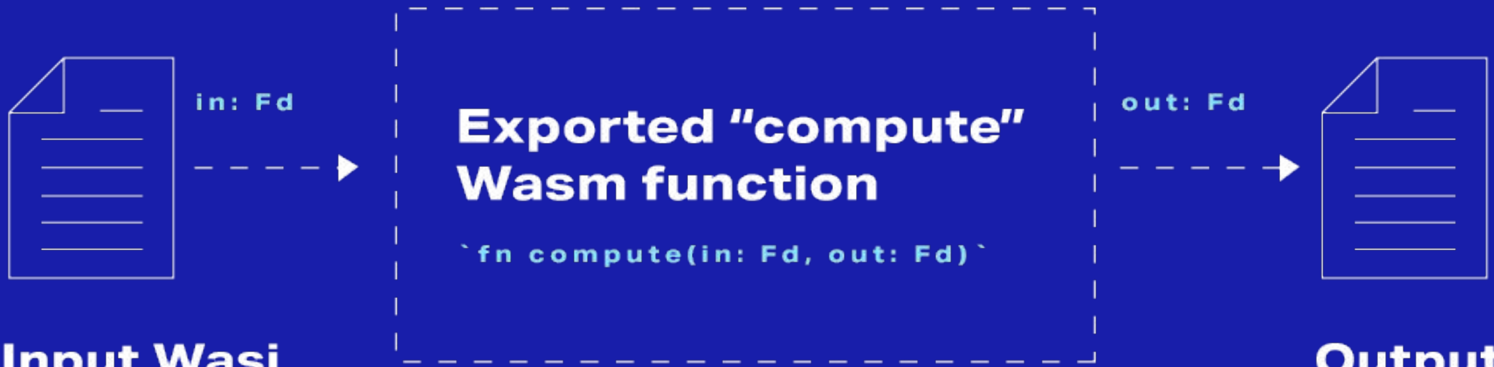
Milestone
No milestone

Notifications Customize
Unsubscribe
You're receiving notifications because you're watching this repository.

8 participants

Can WASI be made deterministic though?

The model



in: Fd

Exported “compute” Wasm function

`fn compute(in: Fd, out: Fd)`

out: Fd

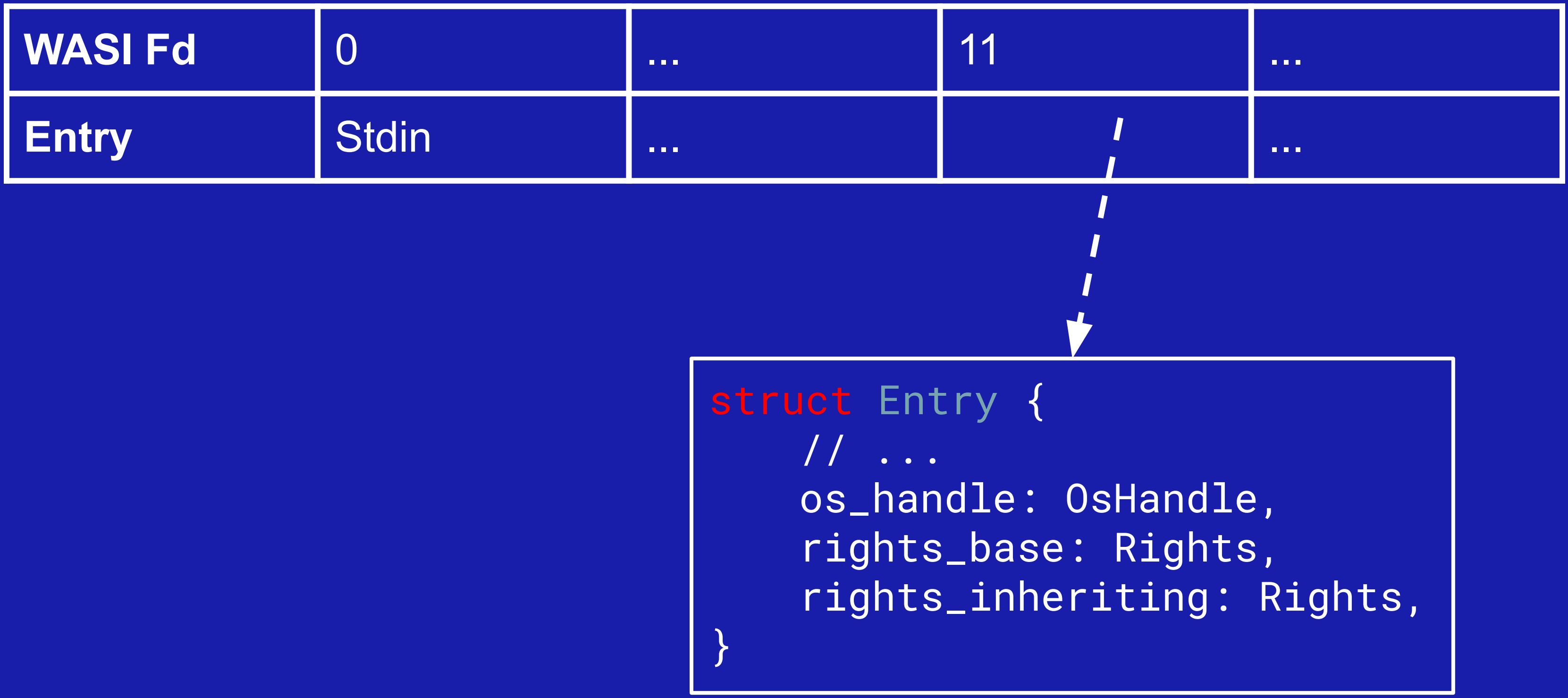
Input Wasi file descriptor

The only rights we provide is reading or in WASI terms: `rights::fd_read`

Output Wasi file descriptor

The only rights we provide is writing or in WASI terms: `rights::fd_write`

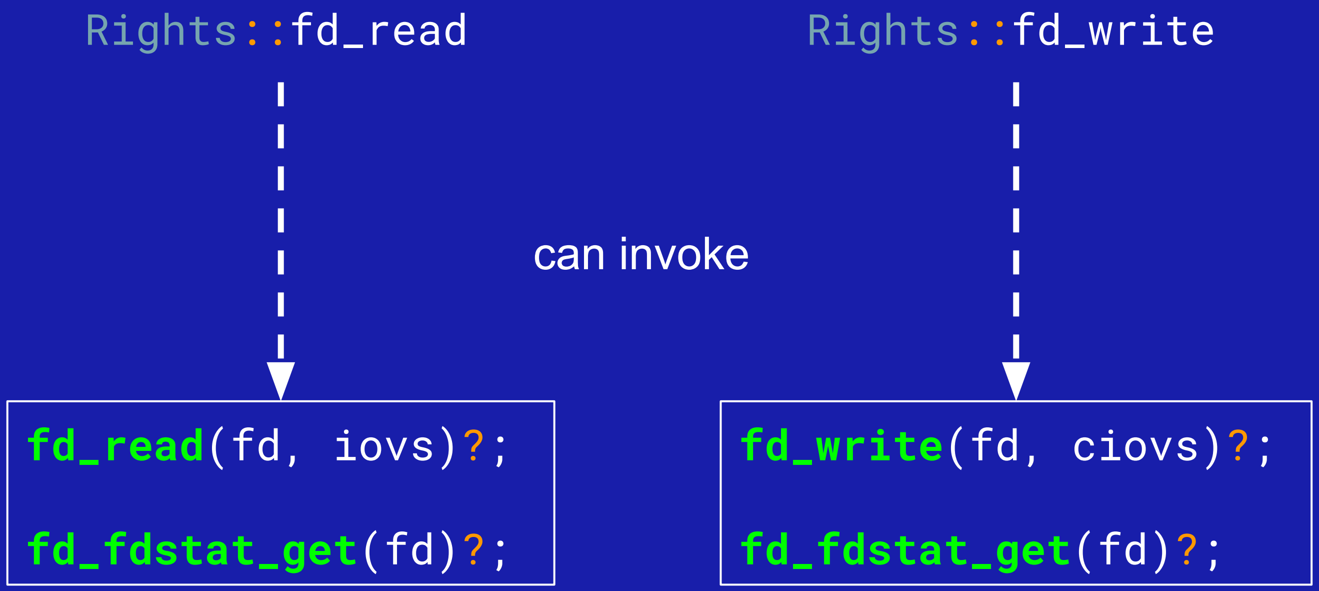
What is WASI file descriptor?



WASI Fd	0	...	11	...
Entry	Stdin	...		...

```
struct Entry {
    // ...
    os_handle: OsHandle,
    rights_base: Rights,
    rights_inheriting: Rights,
}
```

WASI Fd rights?



```
Rights::fd_read
```

```
Rights::fd_write
```

can invoke

```
fd_read(fd, iovs)?;

fd_fdstat_get(fd)?;
```

```
fd_write(fd, ciovs)?;

fd_fdstat_get(fd)?;
```

But nothing else!

Have we just achieved determinism?

Almost! But not quite there yet...

You can still invoke these, since
they are `Fd` independent

```
poll_oneoff(...)?;
```

```
random_get(...)?;
```

```
environ_get(...)?;
```

```
clock_time_get(...)?;
```

Good news is, they will all get their own
module and require a capability

Time for examples!

Everything’s on Github!

01

Examples + description on Github:

kubkon/wasi-compute

02

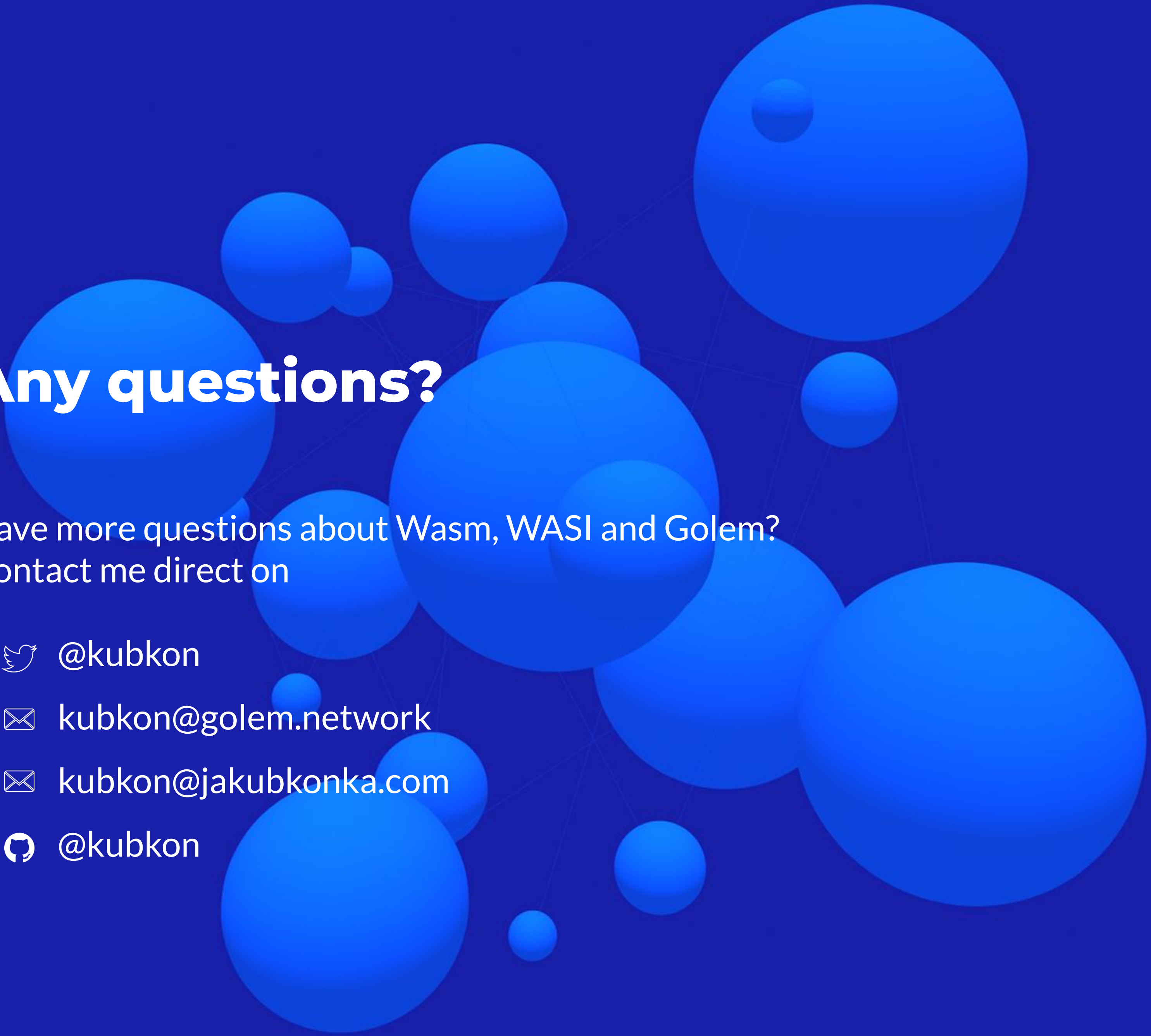
3 examples to play with:

1. hello-compute – read from `in`, uppercase, write to `out`
2. test-compute – verify that `in` and `out` have only `fd_read` and `fd_write` respectively
3. flite-compute – plug in a text-to-speech `flite` engine into model

03

Fork, play with, break, extend…
In general, have fun!

Any questions?



Have more questions about Wasm, WASI and Golem?
Contact me direct on

@kubkon

kubkon@golem.network

kubkon@jakubkonka.com

@kubkon