golem

ETHBerlin 2019

# WebAssembly in Golem

ŁUKASZ GLEŃ

# What is gWASM?

01

- gWASM = WebAssembly on Golem
- new platform in Golem

02

Running distributed WebAssembly applications in decentralized Golem Network

03

Golem Network = users (requestors) rent out compute cycles of other users’ (providers) machines for a fee (Golem Network Tokens)

# Distributing WebAssembly computations

01

Heavy computation that can be executed in parallel

02

You need WASM binary to execute computations

03

With Golem you find computation nodes

04

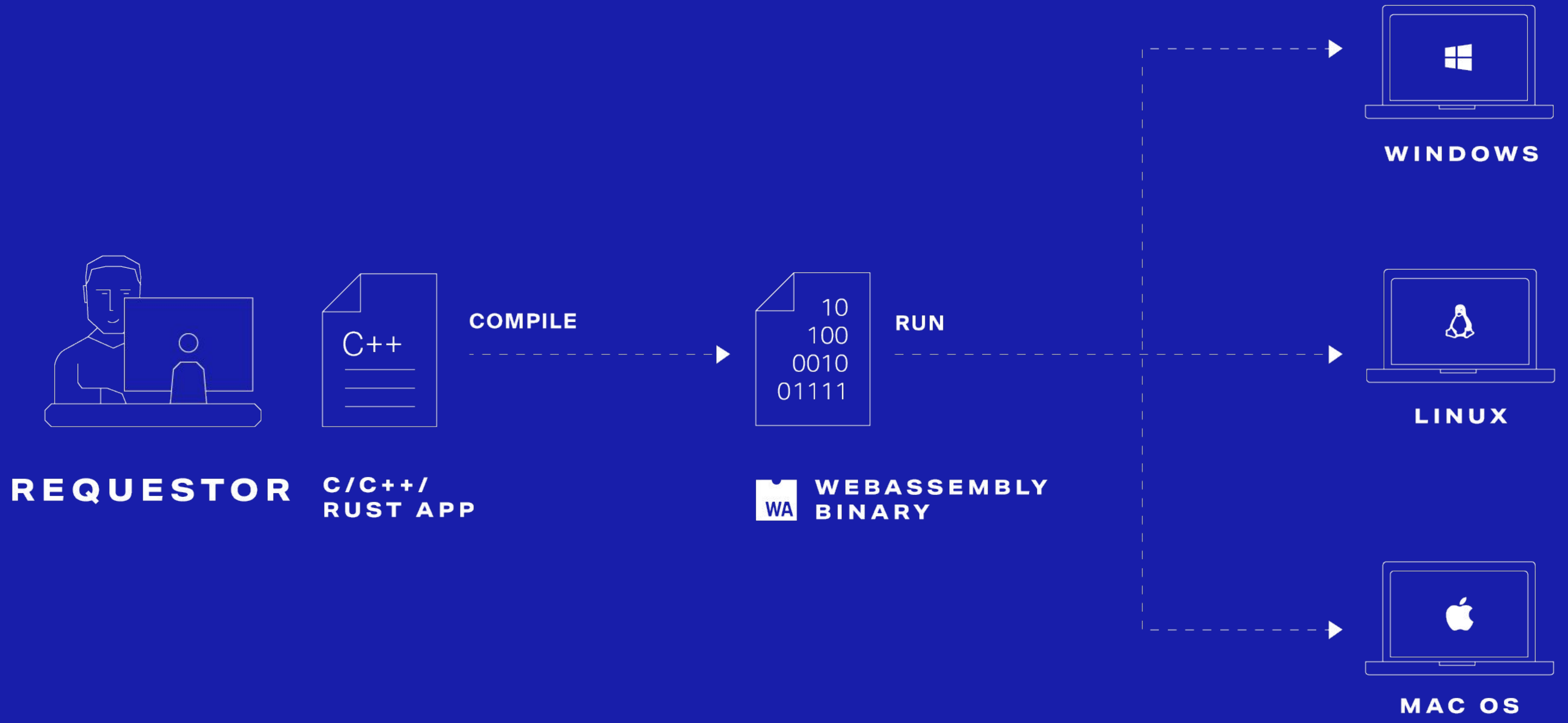
Every node receives WASM binary and input data

05

Nodes do computations

06

You collect results


REQUESTOR
C++
C/C++/
RUST APP
COMPILE
10
100
0010
01111
WA
WEBASSEMBLY
BINARY
RUN
WINDOWS
LINUX
MAC OS

# Bare gWASM tasks

01

You can send gWASM task directly to Golem

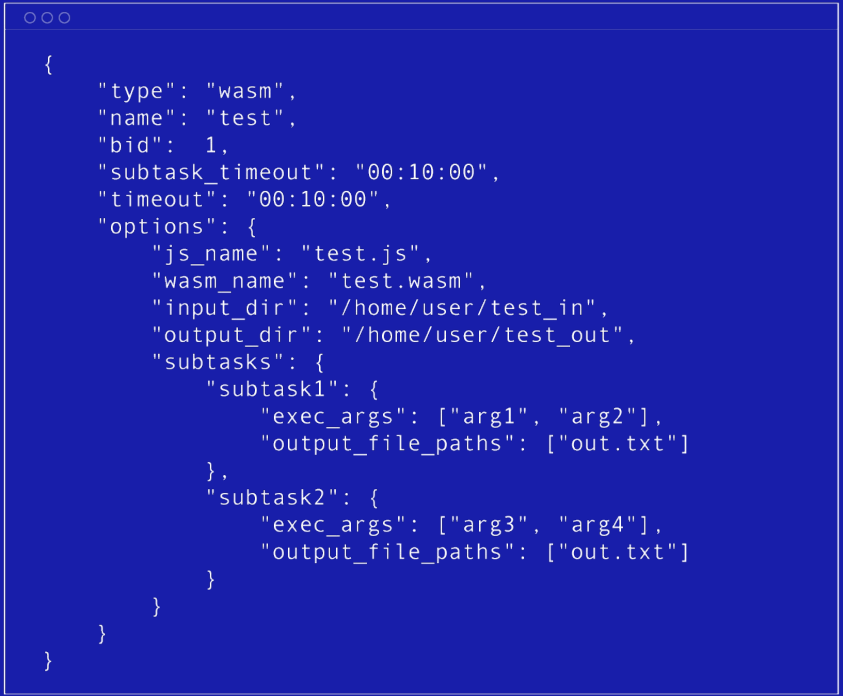
02

You need:

- WASM binary
- Task.json with Golem task configuration
- Structured folder with input data
- Connection to Golem node

```
.
|-- input_dir
|   |-- program.js
|   |-- program.wasm
|   |-- subtask1
|   |   |-- input_file_1_1
|   |   `-- input_file_1_2
|   `-- subtask2
|       |-- input_file_2_1
|       `-- input_file_2_2
```

**directory structure**



```
{
    "type": "wasm",
    "name": "test",
    "bid":  1,
    "subtask_timeout": "00:10:00",
    "timeout": "00:10:00",
    "options": {
        "js_name": "test.js",
        "wasm_name": "test.wasm",
        "input_dir": "/home/user/test_in",
        "output_dir": "/home/user/test_out",
        "subtasks": {
            "subtask1": {
                "exec_args": ["arg1", "arg2"],
                "output_file_paths": ["out.txt"]
            },
            "subtask2": {
                "exec_args": ["arg3", "arg4"],
                "output_file_paths": ["out.txt"]
            }
        }
    }
}
```

**gWASM task json** example

*Creating a task from file*

# g-flite

01

Bare gWASM tasks are not user friendly, it is better to automate this

02

Example of gWASM application: g-flite

03

Flite: text to speed

04

g-flite

- Splits text
- Executes flite cross-compiled to WASM on nodes
- Collects and combines results

05

- Distributing computations transparent for a user
- g-flite UI is similar as flite UI

# g-flite

```
testcase ) 
```

# Application architecture

01

g-flite as an example

02

Applications and services with distributed back-end

03

Golem should provide more solutions and services that simplify developing applications and services with distributed back-end, like clouds do

# WASM, WASM, WASM

01

Server-side WebAssembly

A client cross-compiles but the code is executed on remote node

It is different than web browser or blockchain approach

02

Requirements: performance, access to files vs security, determinism vs supporting various applications

03

Limitations:

- Cross-compilation
- Determinism
- Access to resources
- Single threaded
- etc

# Security

01

Secure execution on a remote machine

02

Access to files, code execution, error handling, etc

03

Sandboxed SpiderMonkey engine from Mozilla

04

Next step: WASI

# Results verification

01

Verification must comply with trustless and anonymity principles

02

Verification by Redundancy ( a la BOINC)

03

It is mandatory for computations to be deterministic

# Distributed Economy

01

Prices are set at each task separately between a requestor and a provider without any arbitration

02

Short term renting out of computational power  form a marketplace with demand, supply and rules of market competition

03

Distributed economies are a work in progress

# Next steps

01

Testnet -> Mainnet

02

WASI

03

How to reach out wider audience

- More demonstrative computations
- gWASM store
- Support and actively developed docs
- How users receive and understand gWASM

# Thank you!

Useful Links

docs https://docs.golem.network/#/Products/Brass-Beta/gWASM

g-flite https://github.com/golemfactory/g-flite

gWASM store https://github.com/golemfactory/wasm-store

dev survey https://forms.gle/f15EyzKnH24DGNMH7

WASI https://wasi.dev/