Serverless Everywhere

WebAssembly Makes Servers Great Again

contact@secondstate.io
http://SecondState.io/

déjà vu

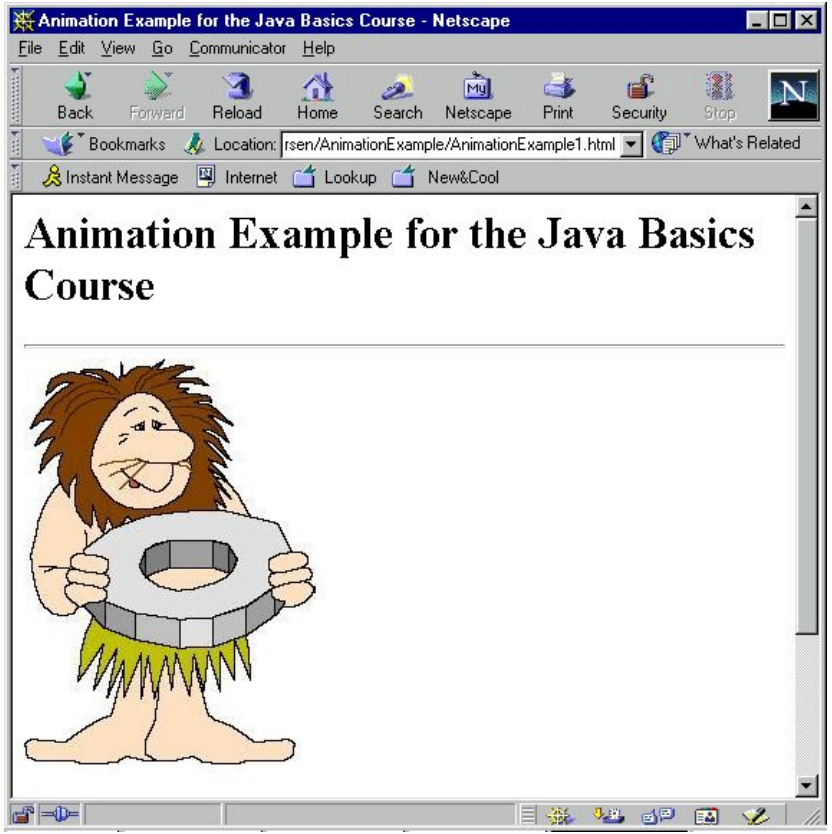
noun [U]

UK /ˌdeɪ.ʒɑː ˈvuː/ **US** /ˌdeɪ.ʒɑː ˈvuː/

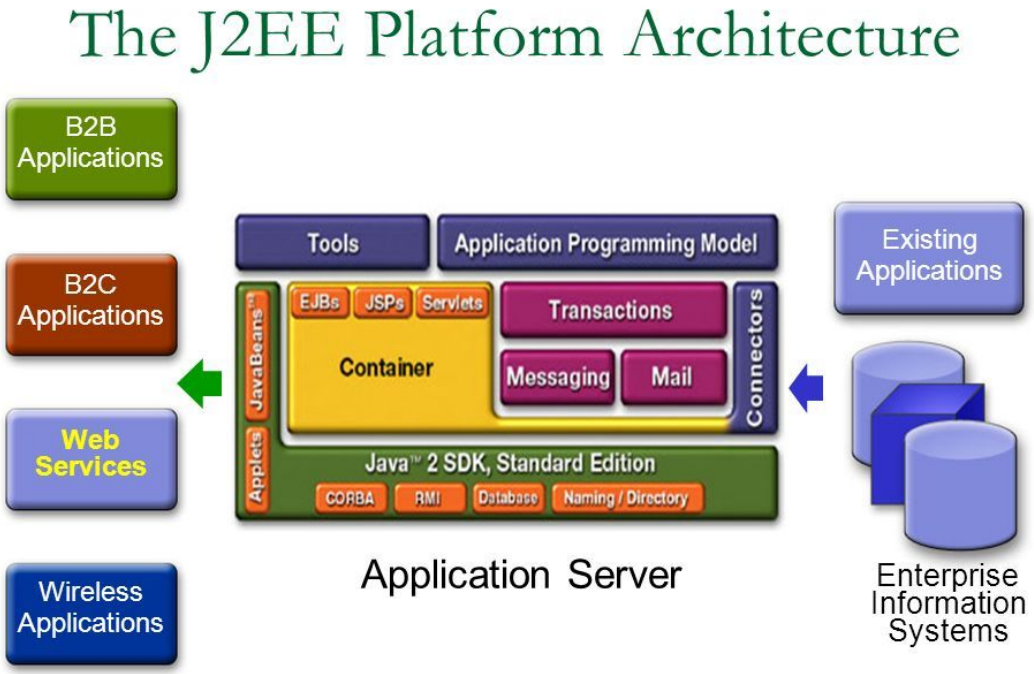
the strange feeling that in some way you have already experienced what is happening now:

Java in 1995


Applet Viewer: JavaRocksApplet.class
Applet
Java Rocks!!
Applet started.

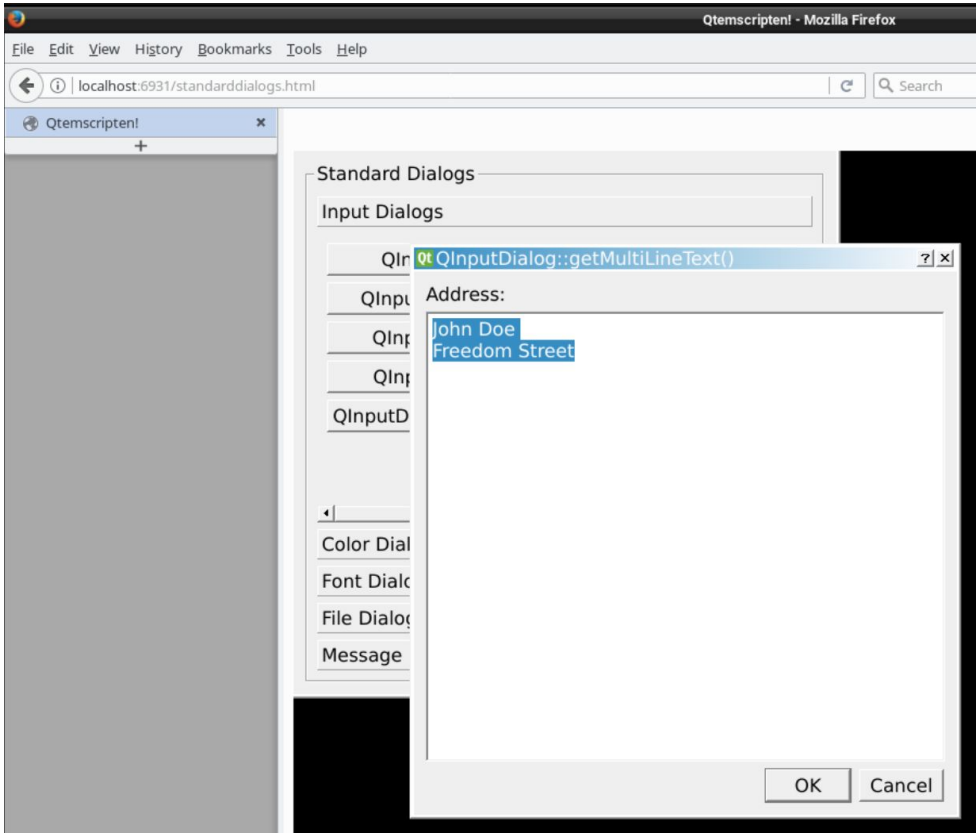

Animation Example for the Java Basics Course - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: rsen/AnimationExample/AnimationExample1.html What's Related
Instant Message Internet Lookup New&Cool
Animation Example for the Java Basics Course

Java in 1999

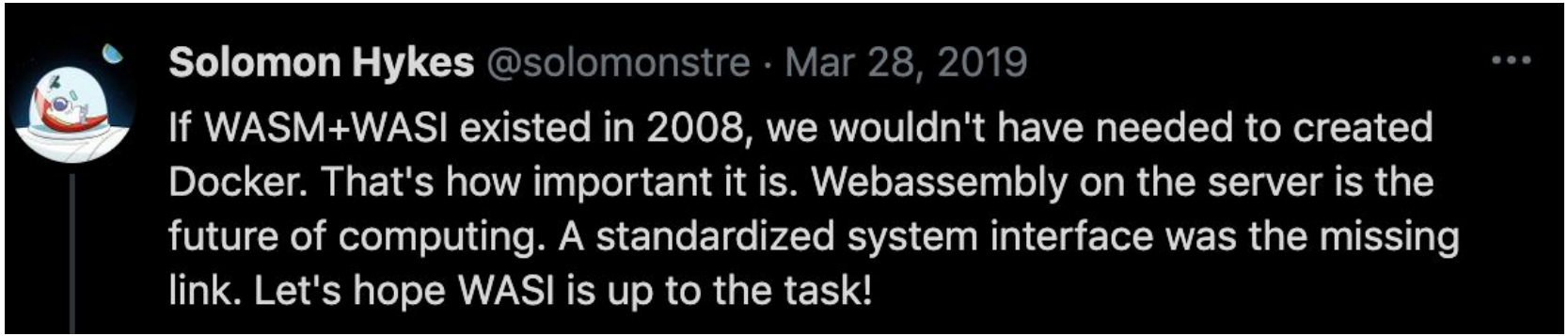

The J2EE Platform Architecture
B2B Applications
B2C Applications
Web Services
Wireless Applications
Tools
Application Programming Model
JavaBeans™
EJBs
JSPs
Servlets
Container
Transactions
Messaging
Mail
Connectors
Applets
Java™ 2 SDK, Standard Edition
CORBA
RMI
Database
Naming / Directory
Application Server
Existing Applications
Enterprise Information Systems

SECOND STATE

WebAssembly in 2017



WebAssembly in 2019



SECOND STATE

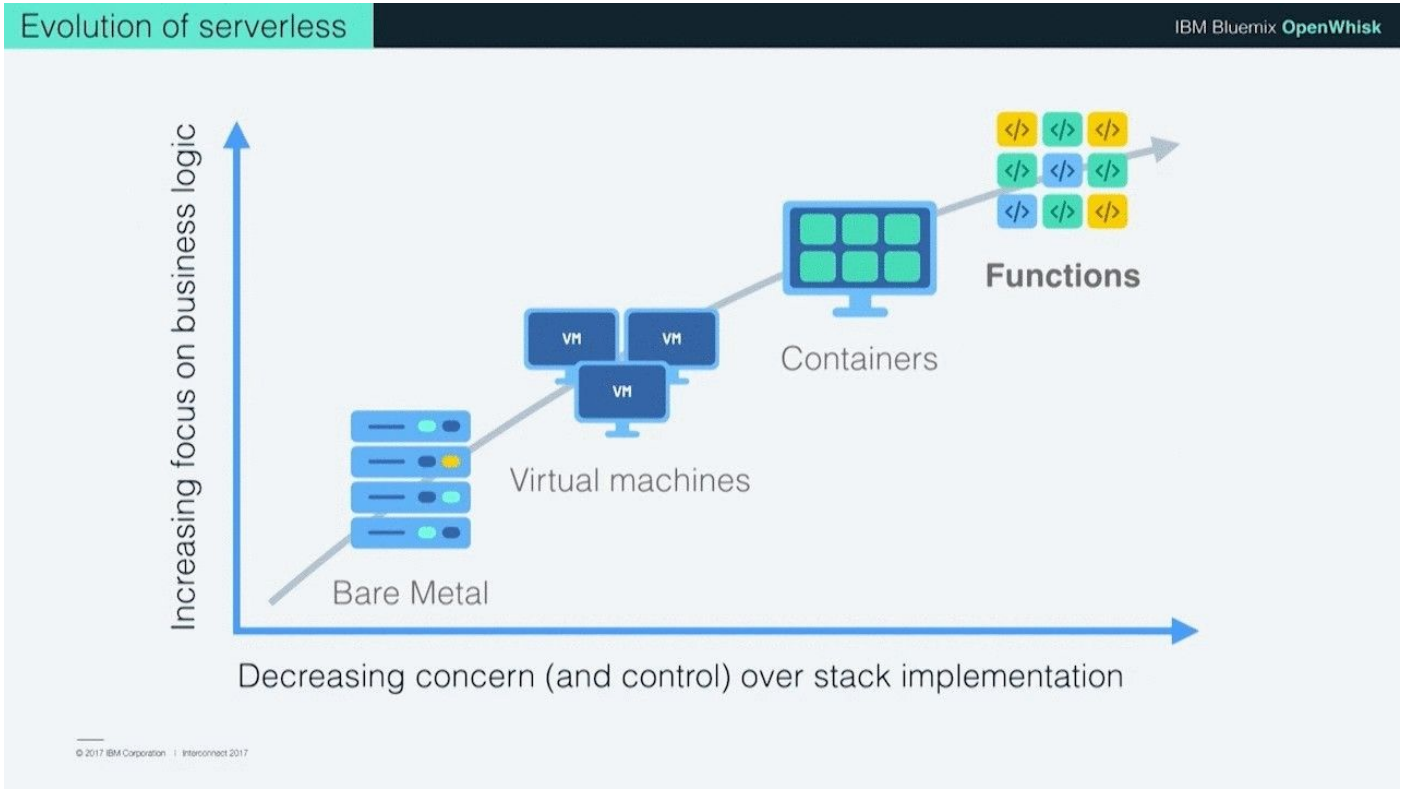
History doesn't repeat itself, but it often rhymes.

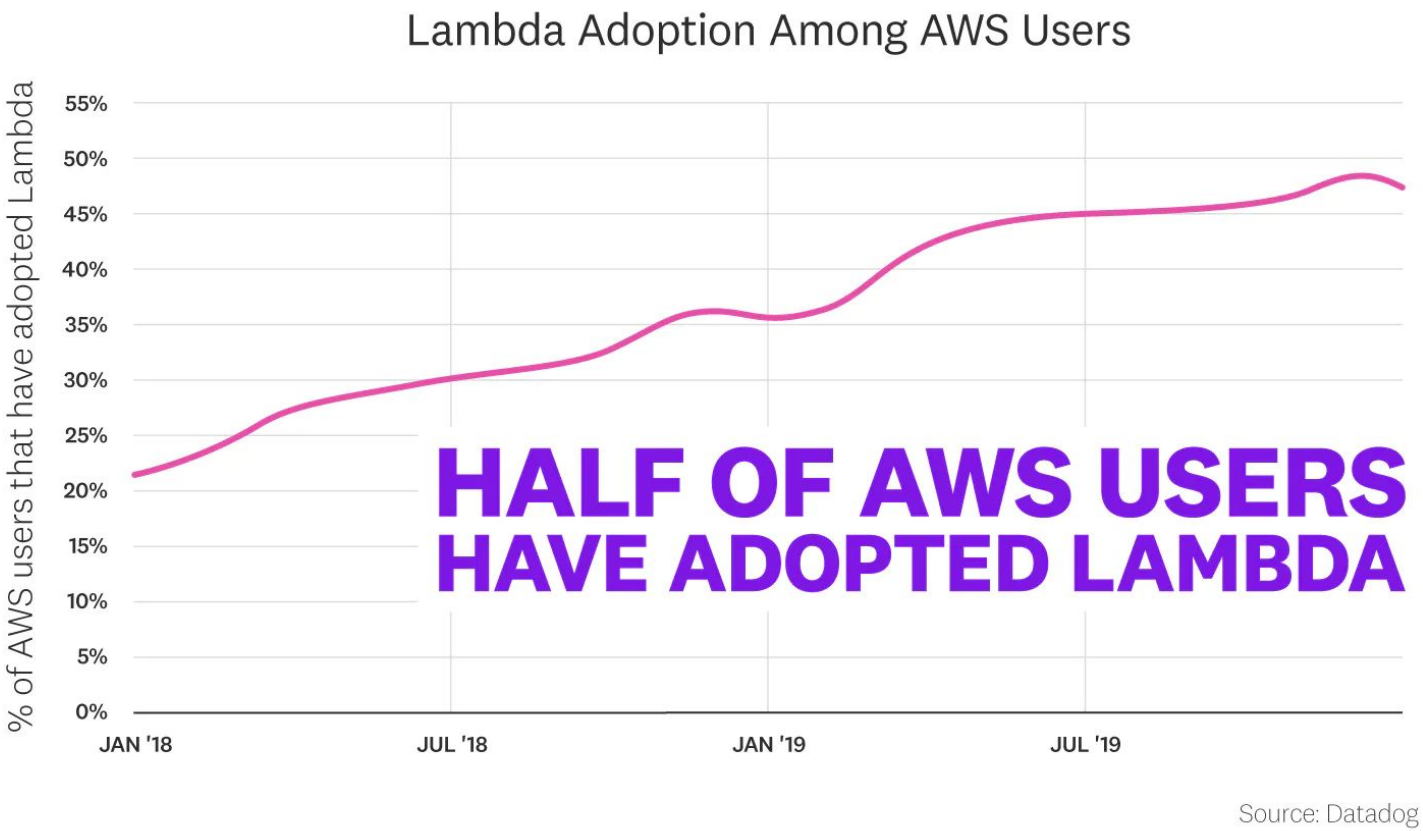
SECOND STATE

Application servers are OUT. Cloud native is IN.

SECOND STATE

The goal of cloud native is serverless





Source: Datadog

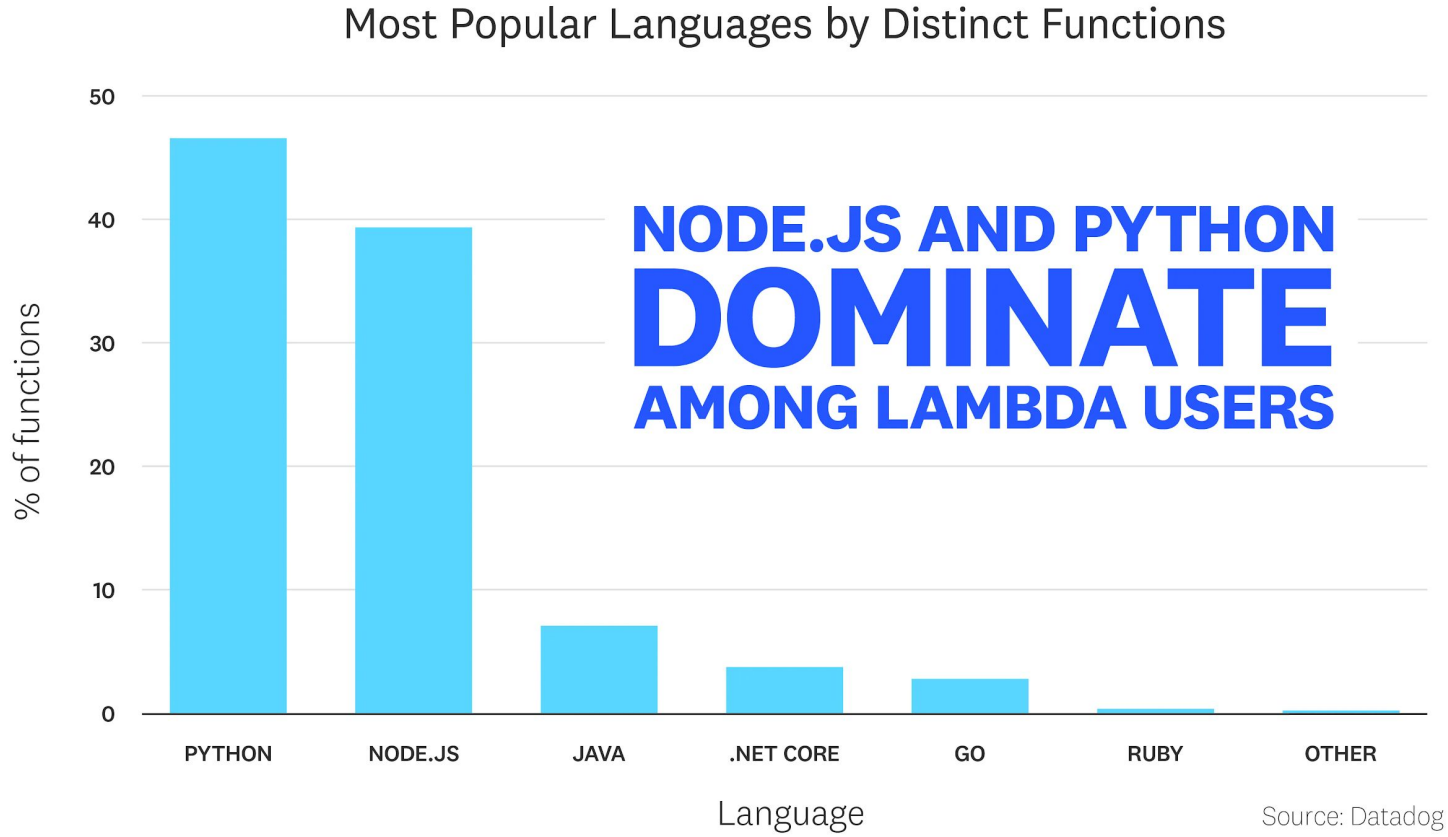
SECOND STATE

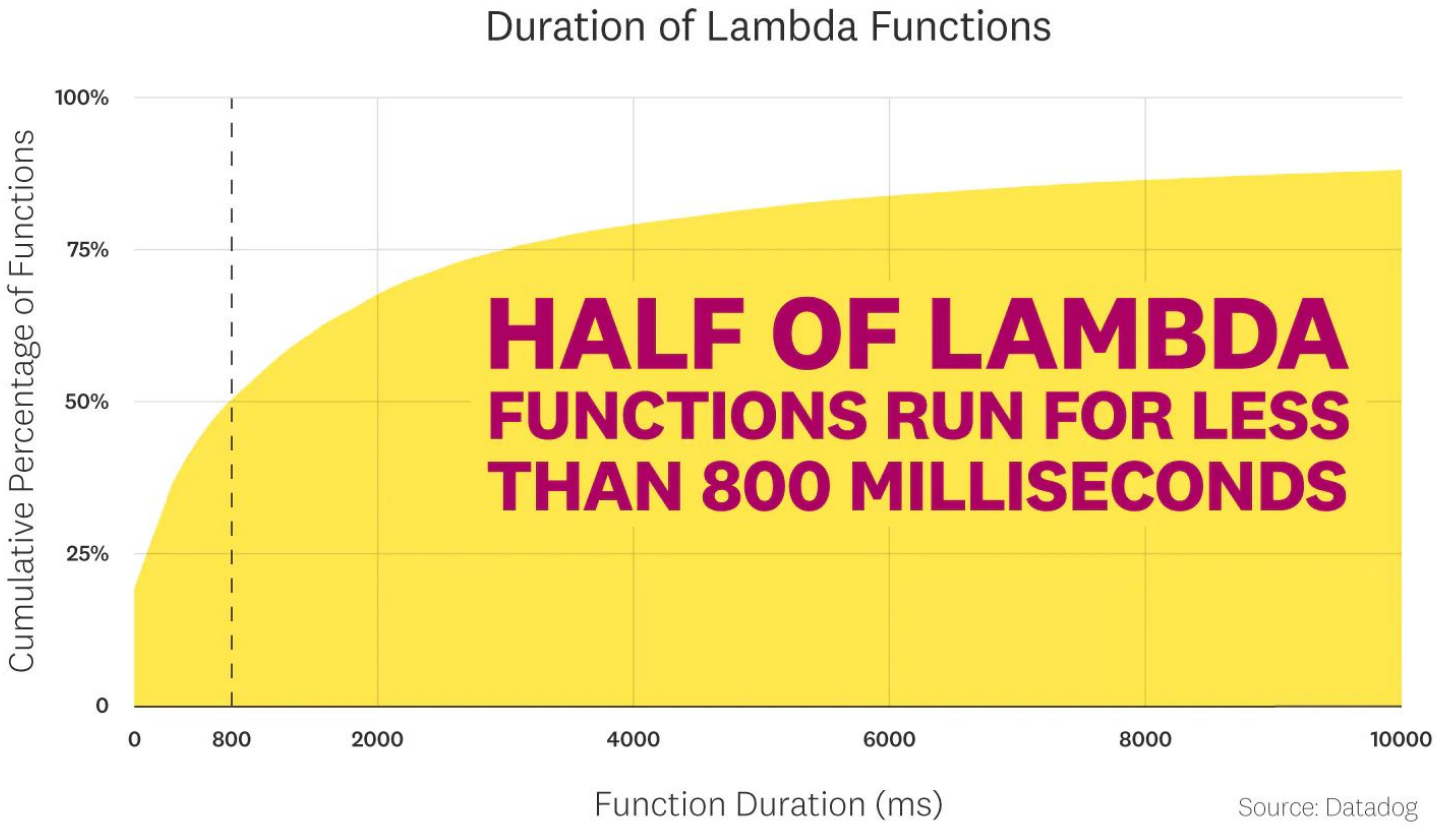
在用户规模上，腾讯云 Serverless 开发者总数已经突破100万，服务了超过10000家企业，其中大型企业客户超过500家。

在计算规模上，腾讯云 Serverless日调用次数超过了100亿次，成为国内服务规模最大的 Serverless 厂商。

Tencent Cloud Serverless has over 1 million developers, providing services to 10,000 businesses including over 500 large enterprises.

Tencent Cloud Serverless functions are called over 10 BILLION times per DAY.


Most Popular Languages by Distinct Functions
50
40
30
20
10
0
% of functions
NODE.JS AND PYTHON
DOMINATE
AMONG LAMBDA USERS
PYTHON
NODE.JS
JAVA
.NET CORE
GO
RUBY
OTHER
Language
Source: Datadog


Duration of Lambda Functions
Cumulative Percentage of Functions
100%
75%
50%
25%
0
0
800
2000
4000
6000
8000
10000
Function Duration (ms)
HALF OF LAMBDA
FUNCTIONS RUN FOR LESS
THAN 800 MILLISECONDS
Source: Datadog

SECOND STATE

The most common serverless function use case is to run a simple function on a heavy stack

SECOND STATE

But, there are much more to serverless than glue code!

SECOND STATE

The Jamstack!

SECOND STATE

What is Jamstack?

Jamstack is an architecture designed to make the web faster, more secure, and easier to scale. It builds on many of the tools and workflows which developers love, and which bring maximum productivity.

The core principles of pre-rendering, and decoupling, enable sites and applications to be delivered with greater confidence and resilience than ever before.

Pre-rendering

With Jamstack, the entire front end is prebuilt into highly optimized static pages and assets during a build process. This process of pre-rendering results in sites which can be served directly from a CDN, reducing the cost, complexity and risk, of dynamic servers as critical infrastructure.

With so many popular tools for generating sites, like Gatsby, Hugo, Jekyll, Eleventy, NextJS, and very many more, many web developers are already familiar with the tools needed to become productive Jamstack developers.

Enhancing with JavaScript

With the markup and other user interface assets of Jamstack sites served directly from a CDN, they can be delivered very quickly and securely. On this foundation, Jamstack sites can use JavaScript and APIs to talk to backend services, allowing experiences to be enhanced and personalized.

Supercharging with services

The thriving API economy has become a significant enabler for Jamstack sites. The ability to leverage domain experts who offer their products and service via APIs has allowed teams to build far more complex applications than if they were to take on the risk and burden of such capabilities themselves. Now we can outsource things like authentication and identity, payments, content management, data services, search, and much more.

Jamstack sites might utilise such services at build time, and also at run time directly from the browser via JavaScript. And the clean decoupling of these services allows for greater portability and flexibility, as well as significantly reduced risk.

SECOND STATE

Serverless runtimes

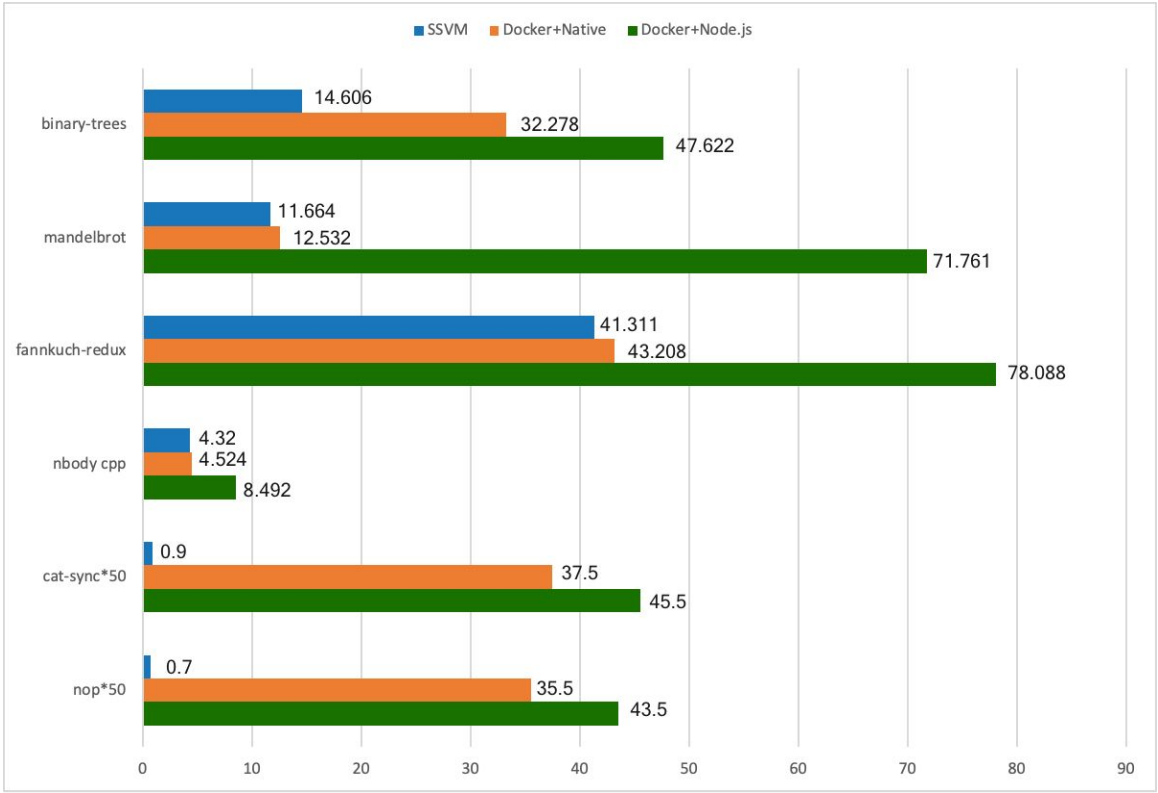
1. Hypervisor VM and microVMs (e.g., AWS Firecracker)

2. Application containers (e.g., Docker)

3. High level language VMs (e.g., JVM, Ruby / Python runtimes, v8, WebAssembly)

SECOND STATE

WASM vs Docker

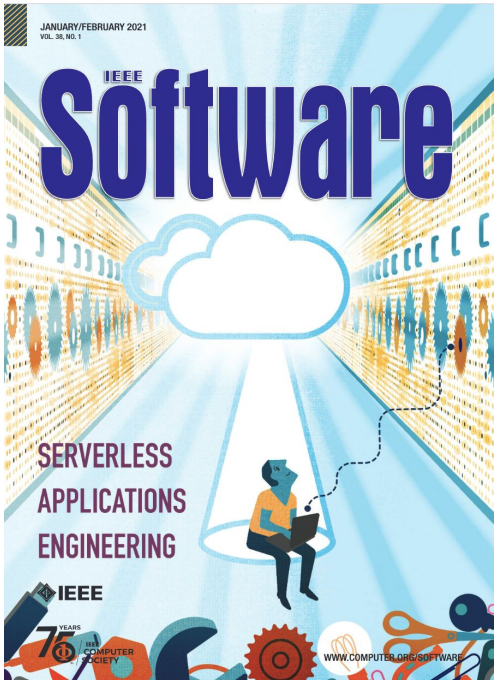


The benefits of WebAssembly VM

- Security
  - Especially important for untrusted code
  - Code that needs to access hardware
- Very efficient and lightweight
  - Near or exceed native performance (due to dynamic compiler optimizations, such as AOT)
- Runtime safety
- Portability / platform independence
- Manageability
  - Package management and storage
  - Automated deployment and ops
  - Hot swapping with zero down time

SECOND STATE

High-performance



The WasmEdge is already one of the fastest WebAssembly VMs.

	Binary-tree	Fannkuch-redux	Mandelbrot	nbody
Azure				
Docker	16.9 ± 0.9	26.1 ± 0.1	16.2 ± 0.9	4.1 ± 0.05
Lucet	Failed	56.6 ± 0.1	Failed	4.67 ± 0.03
SSVM	12.2 ± 0.1	32.8 ± 0.2	12.7 ± 0.1	3.75 ± 0.03
V8	17.4 ± 0.1	30.0 ± 0.2	10.45 ± 0.04	3.38 ± 0.02
WAVM	13.4 ± 0.1	29.4 ± 0.1	11.8 ± 0.07	3.74 ± 0.02
AWS				
Docker	29.9 ± 0.1	39.2 ± 0.1	11.29 ± 0.09	4.14 ± 0.06
Lucet	Failed	67.8 ± 0.1	Failed	5.52 ± 0.08
SSVM	13.4 ± 0.1	38 ± 0.1	10.8 ± 0.2	3.92 ± 0.08
V8	19.2 ± 0.2	40.±0.8	10.6 ± 0.1	3.69 ± 0.08
WAVM	15.1 ± 0.1	36.5 ± 0.1	9.96 ± 0.08	3.95 ± 0.08

SECOND STATE

Live Demos

Adding custom watermark

source code | live demo

OCR

source code | live demo

Image Flipping

source code | live demo

Bird Classification

source code | live demo

Plant Classification

source code | live demo

Food Classification

source code | live demo

Insect Classification

source code | live demo

SECOND STATE

Tutorials

Getting Started

Input and Output

Machine Learning

WASI: System Access

WASI: TensorFlow

WASI: Native Interface

SECOND STATE

Serverless Everywhere

SECOND STATE

Example: 飞书机器人

SECOND STATE

API vs Serverless

The tired

- Remote callback servers
- Manage and pay for servers
- Manage and pay for certificates
- Slow round trip
- High cost to try new ideas

The wired

- Reactive serverless functions
- Just upload a function
- Extremely fast
- No cost, no worries

Serverless Reactor

开发文档 快速开始 用例 登录 免费注册

Serverless Reactor

助你快速上线飞书上的机器人应用

免搭服务器、免运维、无需购买域名、零成本，

免费创建 开发文档

http://reactor.secondstate.info/

How it works

1. 开发者把飞书平台所需要的每个 callback 写成一个函数
2. 开发者把函数代码上传到 Serverless Reactor 平台，收到一个 callback URL
3. 开发者把这个 callback URL 填在飞书平台需要 callback 的地方

http://reactor.secondstate.info/docs/user-create-a-bot.html

SECOND STATE

编写机器人逻辑

请 fork 这个代码仓库。默认的函数是一个计算器机器人，向它发一个"2+2"的消息，它就会回答"4"。

```
use wasm_bindgen::prelude::*;
use meval;

#[wasm_bindgen]
pub fn text_received(msg: String, _username: String, _step_data: String) -> String {
  let x = meval::eval_str(&msg).unwrap();
  return format!("{}", x);
}
```

改动 src/lib.rs 这个文件，将它改为你的机器人逻辑。具体函数的写法请参见文档。

编译

将机器人的函数编译成可以部署的 WebAssembly 文件。

```
$ rustwasmc build
```

部署

参见文档创建一个飞书企业应用与 Serverless Reactor 的对应 app，将编译成功的 `pkg/calculator_lib_bg.wasm` 文件上传到 Serverless Reactor，并把生成的 service URL 提交给飞书。

https://github.com/second-state/serverless-reactor-starter

What's next

- Deep integration into 飞书
- Support more messaging apps from our hub
  - Slack
  - 钉钉
  - WeChat

SECOND STATE

Example: Stream data processing

https://github.com/second-state/yomo-flow-ssvm-example

SECOND STATE

Serverless function

```
#[no_mangle]
pub extern fn triple(x: f64) -> f64 {
  return f64::from(3) * x;
}
```

Call the function from the data flow processor

```
func (s *quicServerHandler) Read(st quic.Stream) error {
        // decode the data via Y3 Codec.
        ch := y3.
                FromStream(st).
                Subscribe(0x10).
                OnObserve(onObserve)

        go func() {
                for item := range ch {
                        // Invoke wasm
                        val := triple(item.(noiseData).Noise)
                        println(val)
                }
        }()

        return nil
}
```

```
func triple(i float64) float64 {
        // Reads the WebAssembly module as bytes.
        bytes, _ := wasm.ReadBytes("triple/pkg/triple_lib_bg.wasm")
        // Instantiates the WebAssembly module.
        instance, _ := wasm.NewInstance(bytes)
        defer instance.Close()
        // Gets the `sum` exported function from the WebAssembly instance.
        sum := instance.Exports["triple"]
        // Calls that exported function with Go standard values. The WebAssembly
        // types are inferred and values are casted automatically.
        result, _ := sum(i)
        return result.ToF64()
}
```

SECOND STATE

In action

```
Ξ _wrk/yomo-sink-ssvm → go run main.go
2021/01/26 00:14:20 Starting sink server...
2021/01/26 00:14:20 ✅ Listening on 0.0.0.0:4141
+6.913980e+015
+9.036794e+015
^Csignal: interrupt
```

SECOND STATE

Example: Automobiles

SECOND STATE

Software is eating cars

2.5M lines of code

14M lines of code

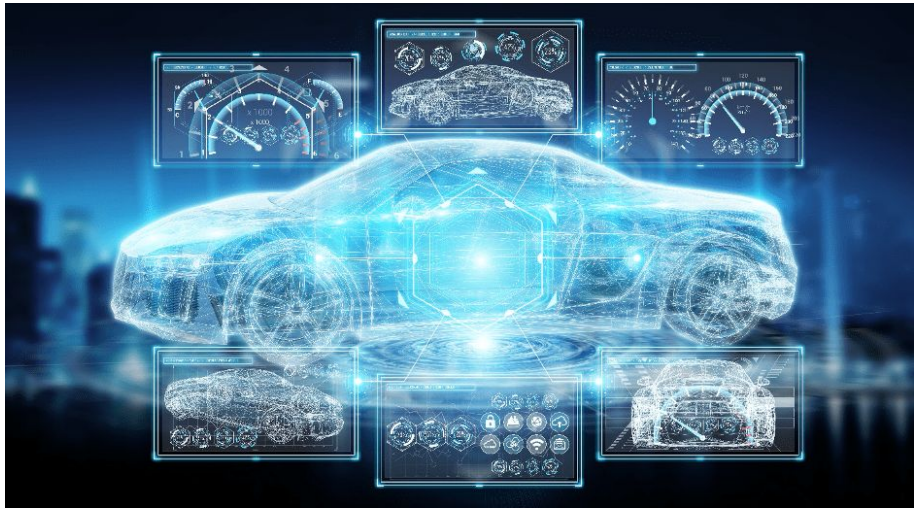
150M lines of code

500M lines of code

Cars tomorrow: Software Defined Vehicles (SDVs)

A central large computer with software applications such as

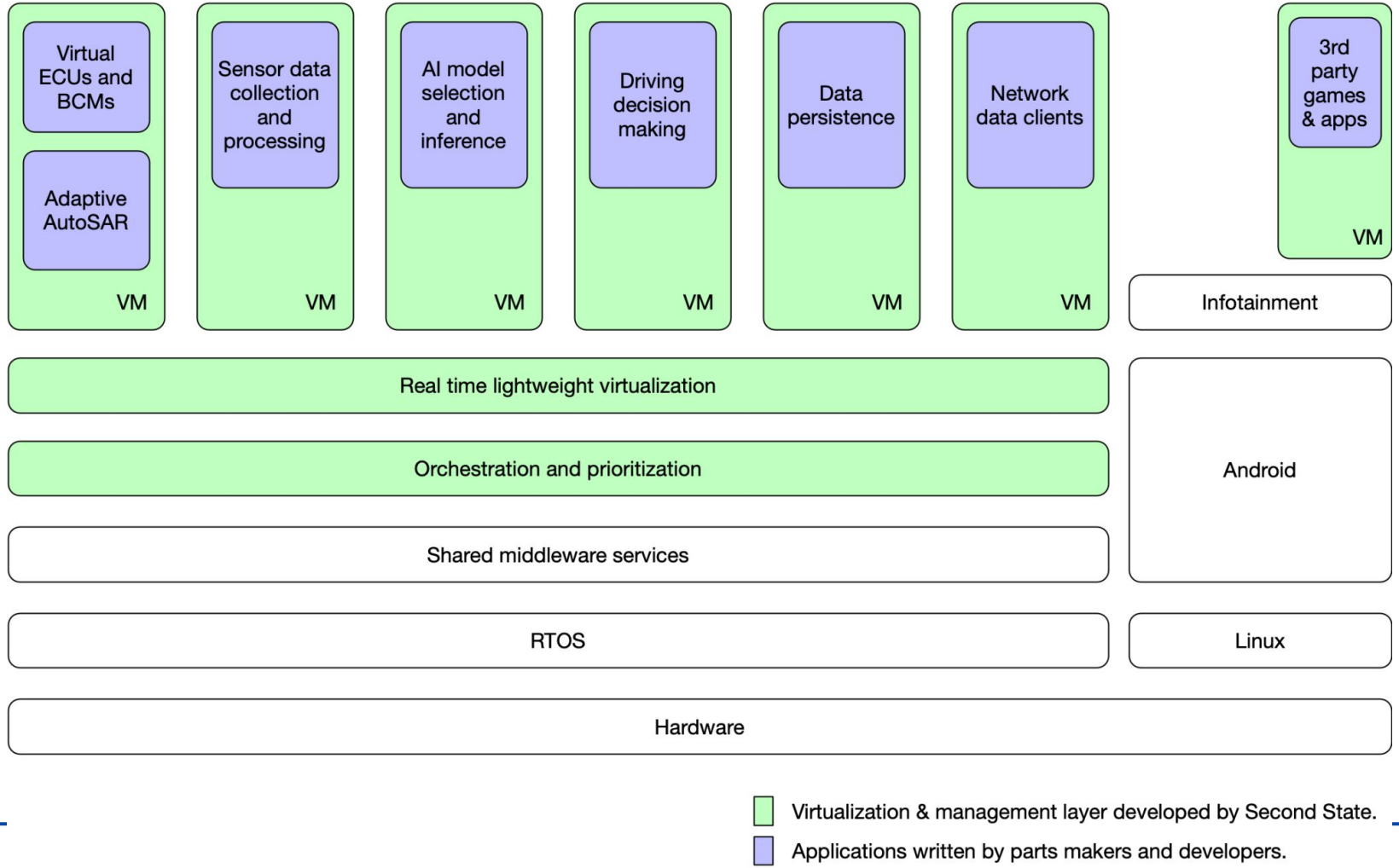
- Software-based ECUs
- Autonomous driving apps
  a. Sensor data processing
  b. AI inference
  c. Map apps
  d. Vehicle control
- Networked V2X apps
- Entertainment systems and dashboard apps



Must support OTA for bug fixes, new features, new hardware, and changing driving conditions.

A large central computer running apps written by many vendors and developers.

That is the classic use case for virtualization.


Virtual ECUs and BCMs
Adaptive AutoSAR
VM
Sensor data collection and processing
VM
AI model selection and inference
VM
Driving decision making
VM
Data persistence
VM
Network data clients
VM
3rd party games & apps
VM
Infotainment
Real time lightweight virtualization
Orchestration and prioritization
Shared middleware services
Android
RTOS
Linux
Hardware
Virtualization & management layer developed by Second State.
Applications written by parts makers and developers.


FUTUREWEI
Technologies


HUAWEI

SECOND STATE

WasmEdge

A popular WebAssembly VM optimized for

high-performance applications

Like us 👍https://github.com/WasmEdge/WasmEdge

SECOND
STATE

Thanks !