CLOUDFLARE®

Exploring Cloudflare Workers

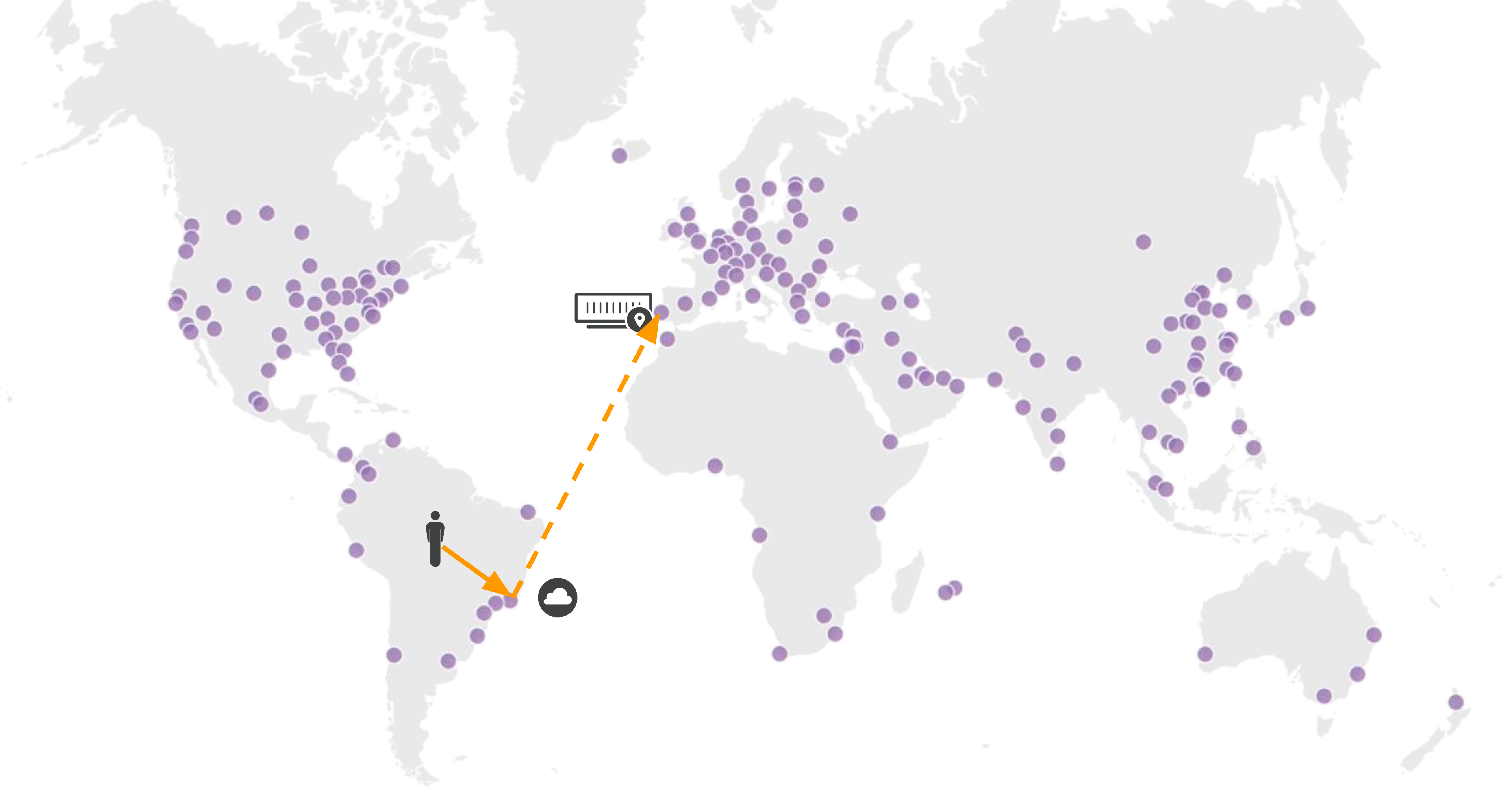
Harris Hancock - Systems Engineer, Cloudflare Workers

CLOUDFLARE®

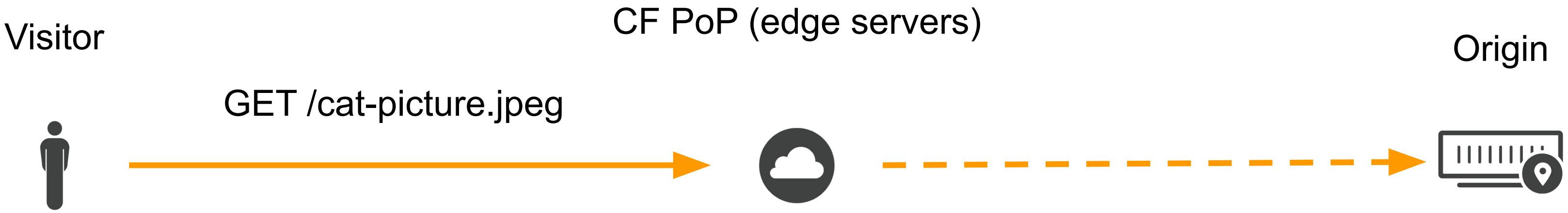
Mission: To help build a better internet.

We provide performance, security, reliability, and insight for web properties.

We optimize for latency.



180+ points of presence



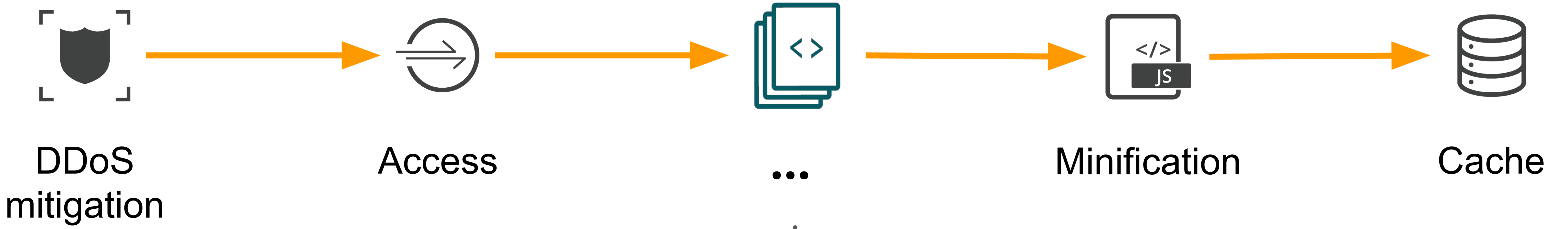
Visitor

CF PoP (edge servers)

Origin

GET /cat-picture.jpeg

Conceptually, Cloudflare is a pipeline of (lots of) features



DDoS mitigation

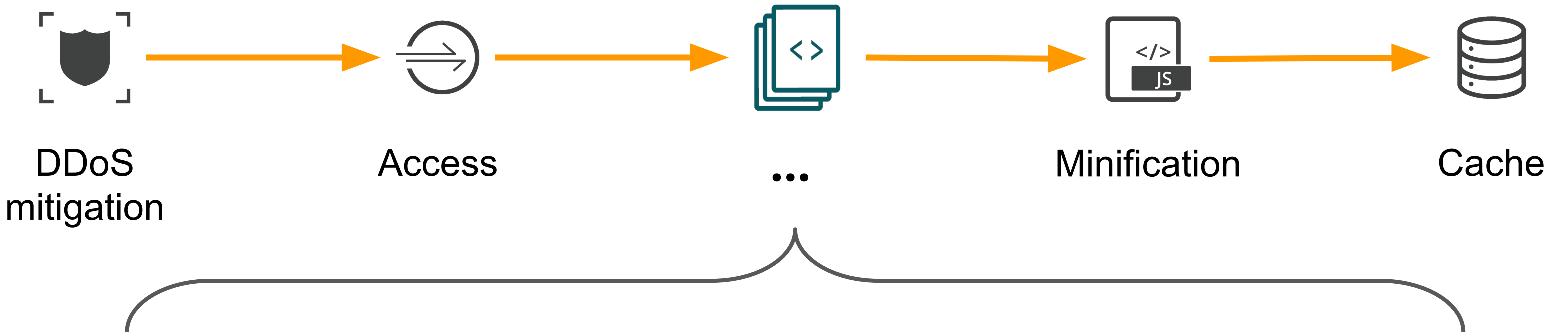
Access

...

Minification

Cache

How do we extend the pipeline?



How do we extend the pipeline?

Previously...

Cloudflare implemented all features

A fixed-function edge network

Ideally...

Customers can write their own features

A programmable edge network

Customers provide the logic,
we provide the platform.

That’s Serverless!

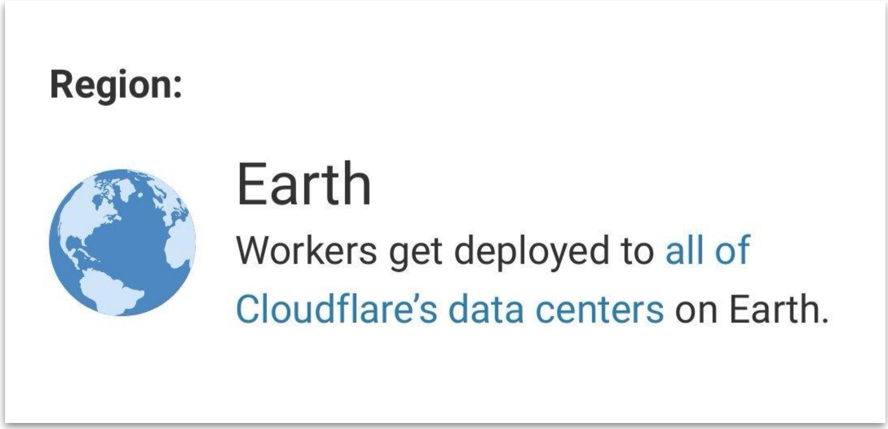
A misnomer: servers are involved. You just don’t worry about them.

Serverless infrastructure abstracts servers away from dev perspective.

Includes functions, storage, database services, and more.

Serverless infrastructure should ...

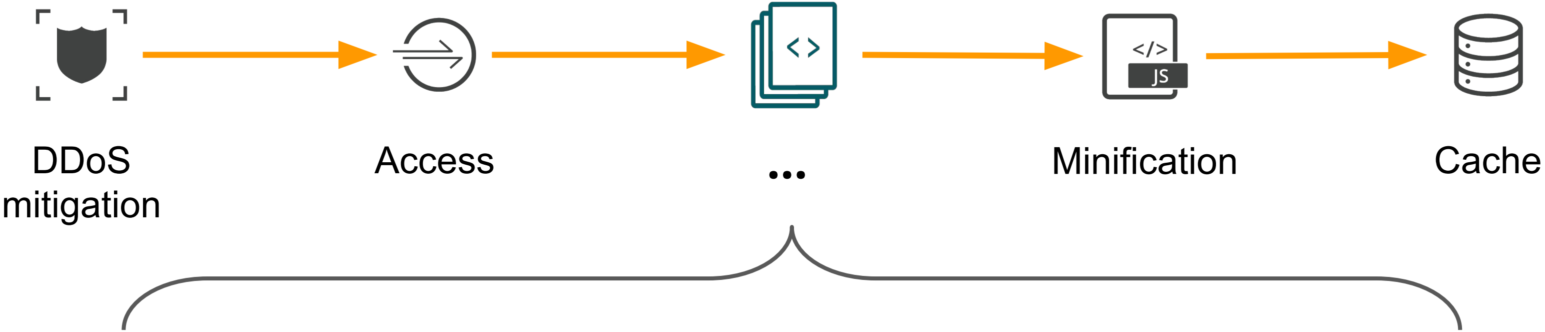
- Require no maintenance
- Bill on usage, not idle time
- Be globally distributed (no regions!)



Region:

Earth

Workers get deployed to all of Cloudflare’s data centers on Earth.



DDoS mitigation

Access

...

Minification

Cache

So these are containers, am I right?

No. In this case, containers don’t scale.

Scalability can mean...

Traffic (requests)

Easy: More locations = more capacity

Tenants (apps)

Hard: Every tenant in every location.

Some locations are small!

Needed:

> 100x Efficiency

Efficiency…

App Code Footprint

VM: 10GB
Container: 100MB
Needed: < 1MB

Baseline Memory Usage

VM: 1GB
Container: 100MB
Needed: < 5MB

Context Switching

VM: low
Container: medium
Needed: extreme

Startup Time

VM: 10s
Container: 500ms
Needed: < 5ms

This technology already exists!

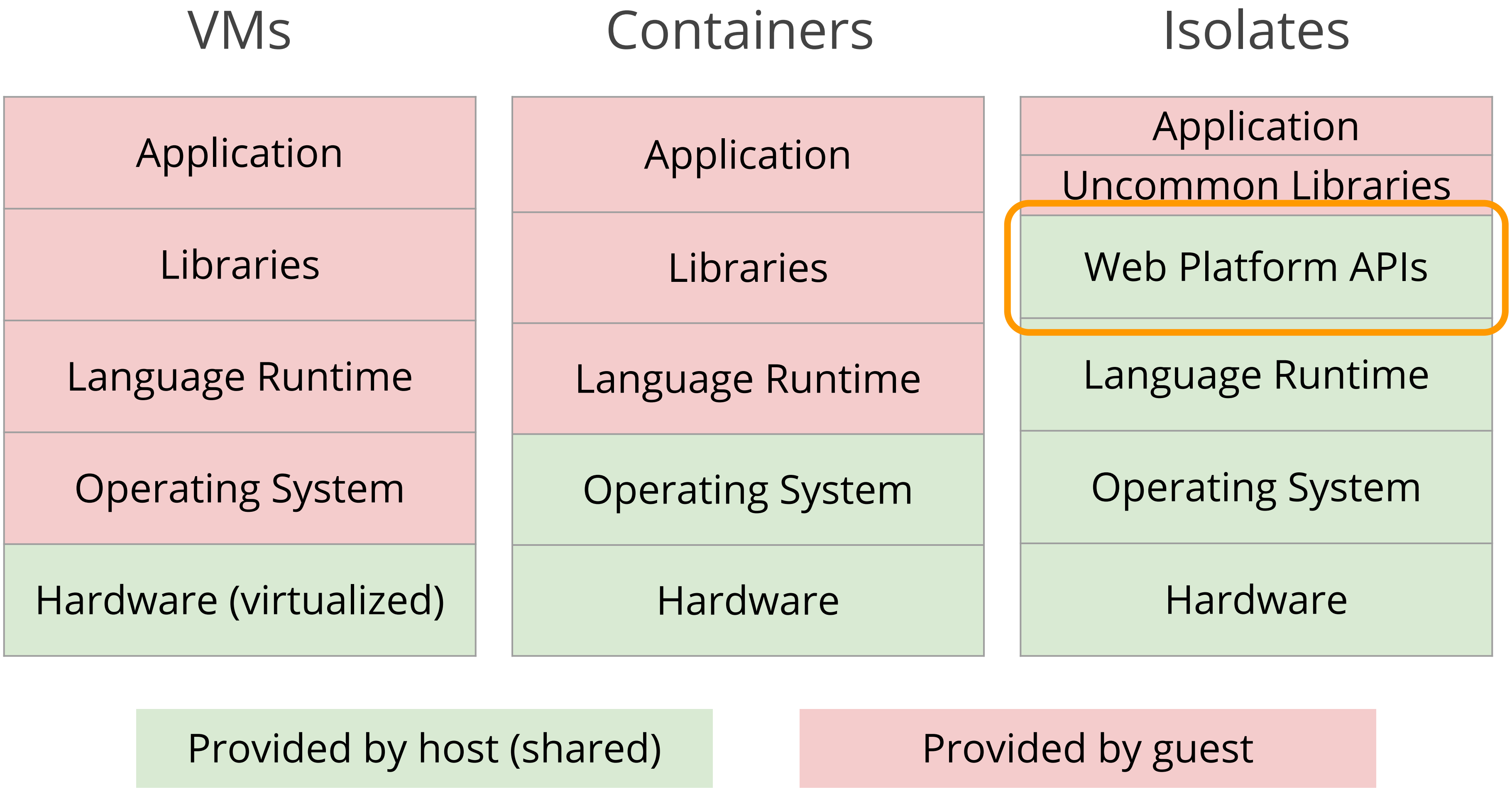
Browsers are optimized for...

- **Small downloads**
- **Fast startup**
- **Many tabs and frames**
- **Secure Isolation**

V8 gives us...

Isolates: tiny sandboxes in a single process
Named after V8 type: `class v8::Isolate`

Standard JavaScript builtins:
async/await, Date, RegExp classes, much more



VMs

- 🟥 Application
- 🟥 Libraries
- 🟥 Language Runtime
- 🟥 Operating System
- 🟩 Hardware (virtualized)

Containers

- 🟥 Application
- 🟥 Libraries
- 🟥 Language Runtime
- 🟩 Operating System
- 🟩 Hardware

Isolates

- 🟥 Application
- 🟥 Uncommon Libraries
- 🟩 Web Platform APIs
- 🟩 Language Runtime
- 🟩 Operating System
- 🟩 Hardware

🟩 Provided by host (shared)

🟥 Provided by guest

Web Platform APIs

W3C/WHATWG standards for:

- HTTP servers (Service Worker)
- HTTP clients (Fetch)
- URL manipulation
- Text encoding
- Stream processing
- Cryptography
- and more

```
addEventListener("fetch", event => {
  event.respondWith(handle(event.request))
}

async function handle(request) {
  // Redirect .jpeg requests to static file
server.
  let url = new URL(request.url)
  if (url.pathname.endsWith(".jpeg")) {
    url.host = "static.example.com"
    return fetch(url, request)
  } else {
    return fetch(request)
  }
}
```

Not Only JavaScript

Any language that compiles to JavaScript:

TypeScript is often a better choice for large projects.

V8 supports WebAssembly, too:

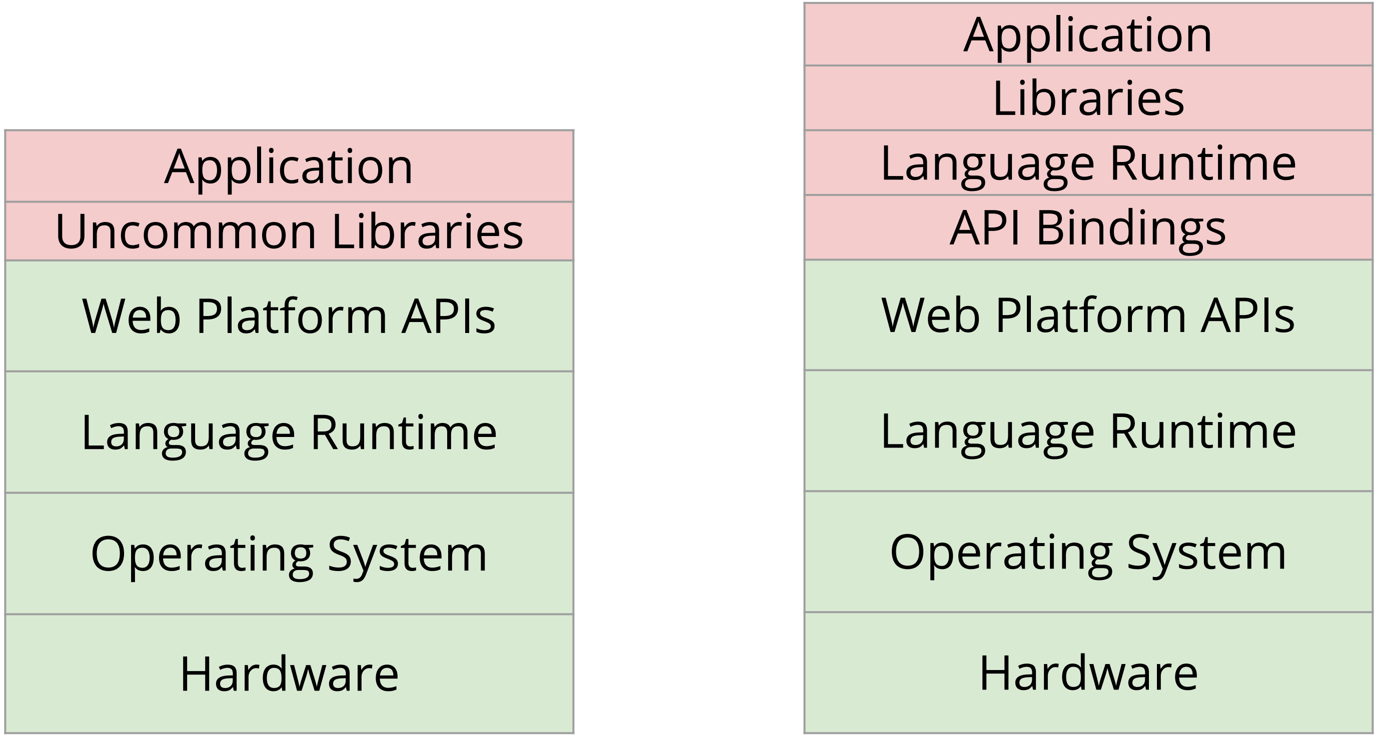
Useful for computationally expensive tasks.

Or codebases you can’t/won’t rewrite in JS.

Or just because you want to.

Isolates

WebAssembly



Isolates:

- Application
- Uncommon Libraries
- Web Platform APIs
- Language Runtime
- Operating System
- Hardware

WebAssembly:

- Application
- Libraries
- Language Runtime
- API Bindings
- Web Platform APIs
- Language Runtime
- Operating System
- Hardware

Missing a way to share common runtimes.

Is V8 secure enough for servers?

Relatively more bugs than VMs.

Reasons:

- Larger attack surface (Bad)
- More research (Good)
  - Bug bounty
  - Fuzzing
  - Import target

Nothing is “secure”

Security is **Risk Management**

Browser

Install updates fast.

Use separate profiles for trusted vs “suspicious” sites.

VS

Server

Install updates faster.

Use separate processes for trusted vs “suspicious” tenants.

Server

VS

Browser

Store all scripts ever loaded for forensic purposes. No eval().

Watch for segfaults, inspect scripts that cause them.

...can’t, privacy violation.

Worker Functions are Stateless

Pros:

Resilient. Can be reloaded with no noticeable effects.

Horizontally scalable.

Cons:

Difficult to implement rate limiting. Need global counter.

Also queues, locks.

Present: Workers KV: a distributed key/value store. Last write wins.

Future: Stateful “object” Workers: code + state that migrates.

Summary

Cloudflare Workers…

Makes Cloudflare’s edge network user-programmable.

Optimized for latency: < 5ms cold starts, anywhere on Earth.

Resources

We have a free tier, go have fun! https://workers.cloudflare.com/

Also, documentation: https://workers.cloudflare.com/docs/

And you can email me: harris at cloudflare