

Thomas Watson

@wa7son

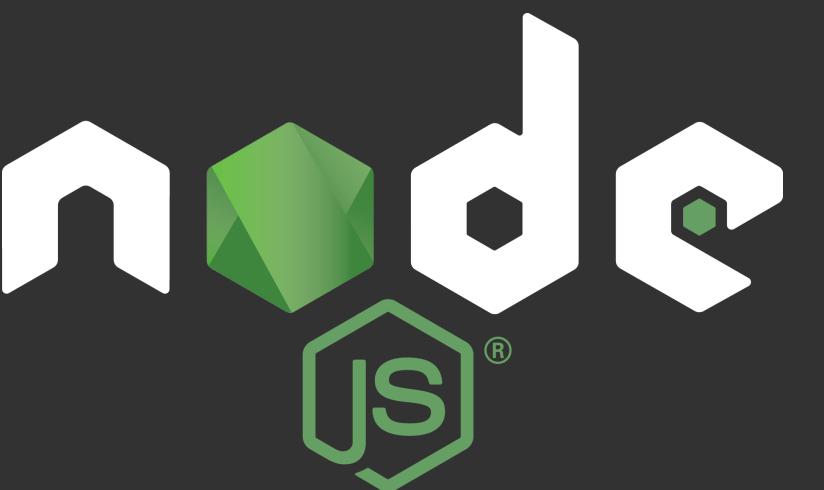
github.com/watson



The Trouble with Tracers

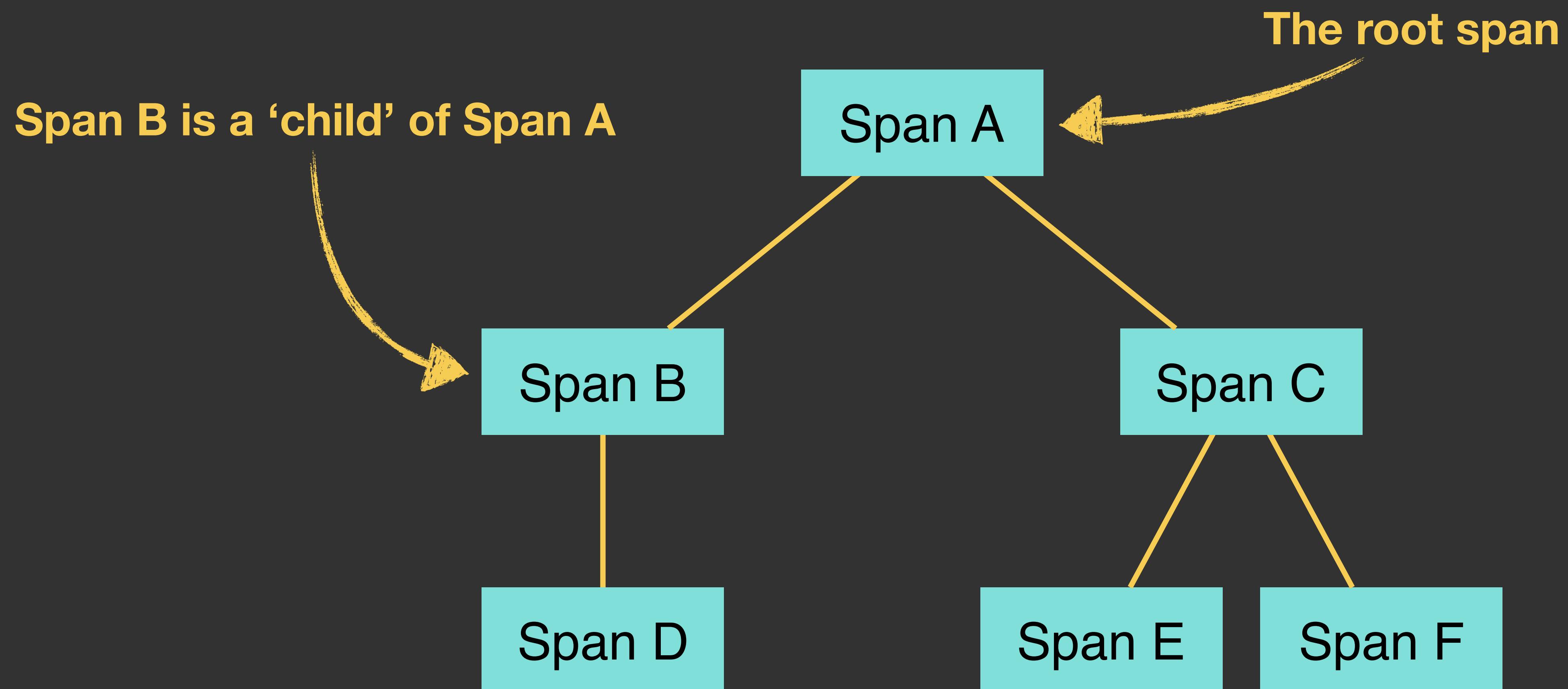
Who am I?

- Thomas Watson
- Open Source developer at github.com/watson
- Principal Software Engineer at Elastic
- Node.js Core Member
- Tweets as @wa7son
- Slides: github.com/watson/talks

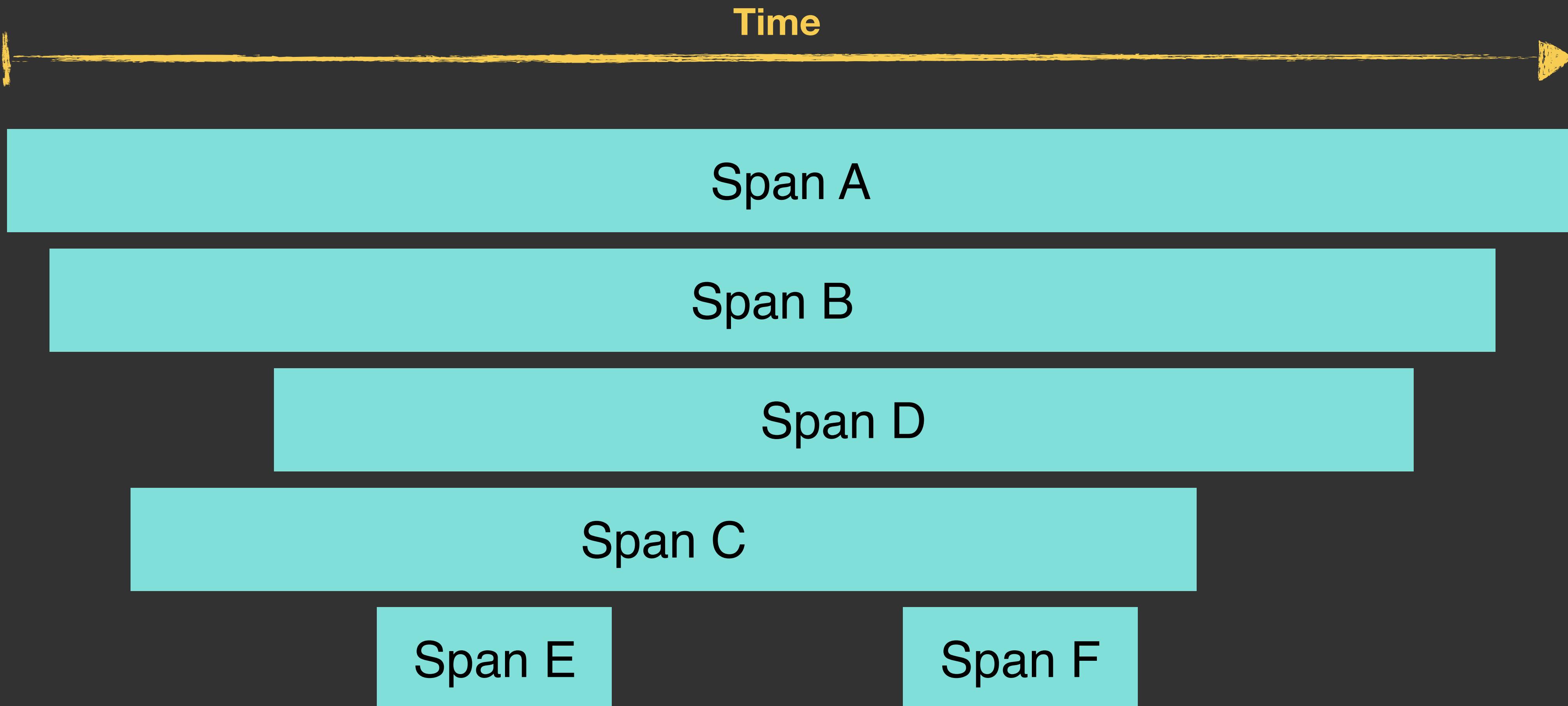


Tracing 101

Trace Graph



Trace Timeline



```
const span = tracer.startSpan('query', { childOf: otherSpan })  
  
mysql.query(sql, function (err, result) {  
  span.end()  
  // ...  
})
```

```
tracer.startSpan('query')

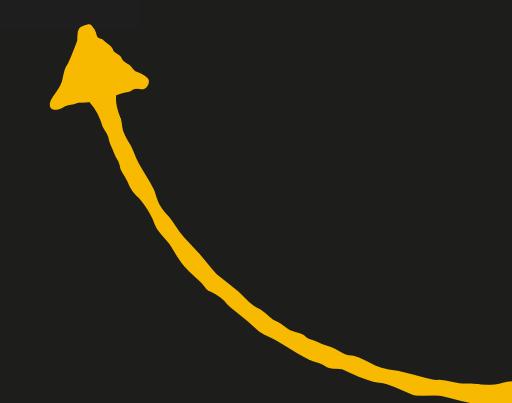
mysql.query(sql, processResult)

// ...

function processResult (err, result) {
  const span = ???  

  span.end()  

  // ...
}
```



TODO: Get the current span somehow

```
const span = tracer.startSpan('query')

mysql.query(sql, processResult.bind(null, { span }))

// ...

function processResult (ctx, err, result) {
  const span = ctx.span
  span.end()
  // ...
}
```

```
tracer.startSpan('query')

mysql.query(sql, processResult)

// ...

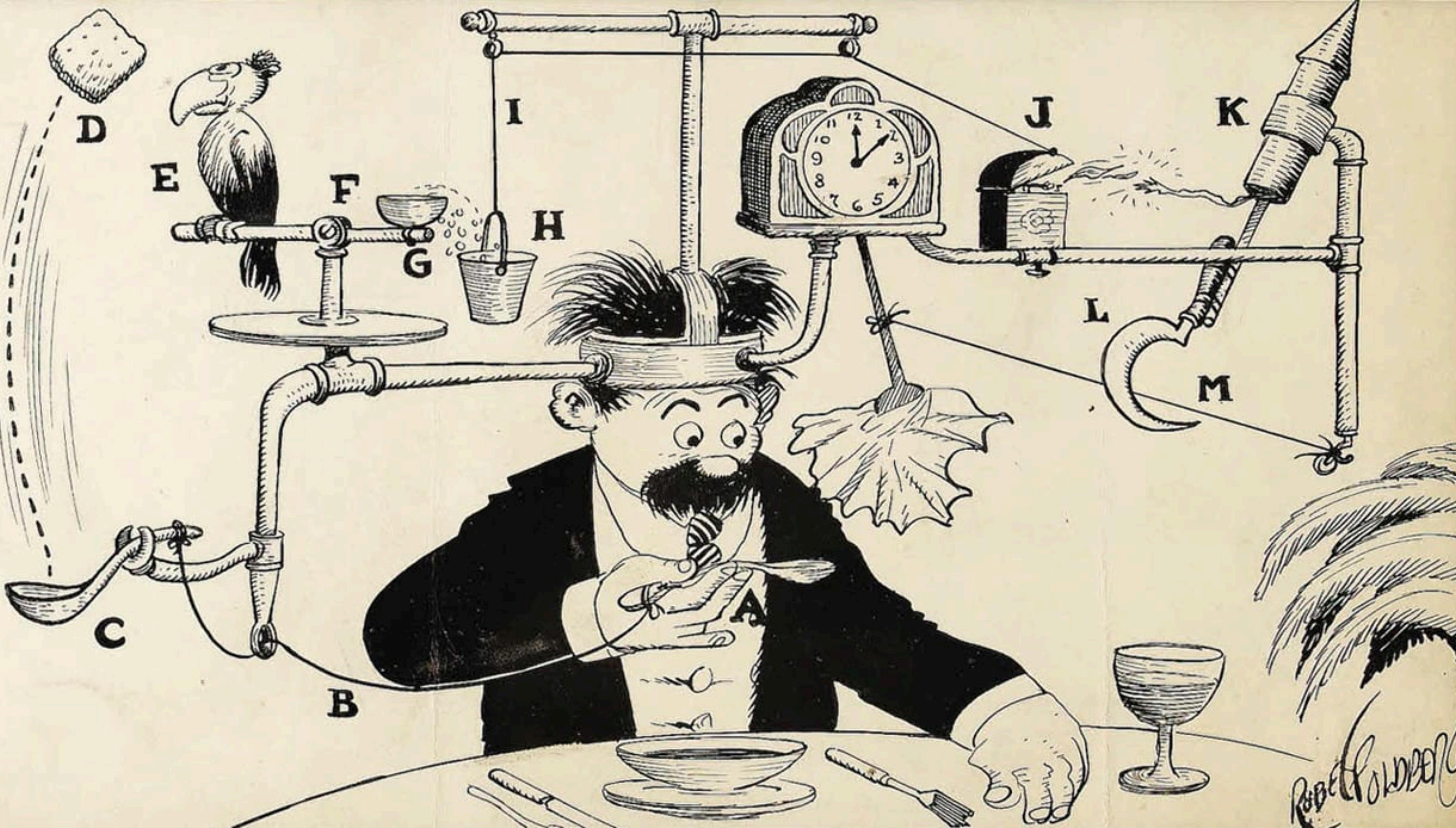
function processResult (err, result) {
  const span = tracer.getCurrentSpan()
  span.end()
  // ...
}
```

```
mysql.query(sql, function (err, result) {  
    // ...  
})
```

```
mysql.qu  
    // ...  
})
```



```
sult) {
```



Dirty Secrets

Patching require
No built-in context propagation
Getting stack traces
Patching every async call -> `async_hooks`

- `async await...` thenables broken
- Promise destroy hook triggered GC
- Uses numeric ids which doesn't play nice with weak maps => hacks in place to clean it up, but those come with risks
- What you want to patch isn't directly exported TAV

Resource management (memory, CPU etc)
Injecting headers into HTTP requests
Babel/TypeScript `defineProperty` hell
Object reuse pool
Metrics: Not a lot of API's to get data from => read things from the filesystem = brittle

How do we know if
mysql is used in the current project?

How do we know when
a mysql query is performed?

`mysql.query()`



1. Start a span when the ‘query’ function is called
2. Associate new span with current trace
3. Make current span child of current “active” span
4. End the span when the query finishes



Patching require()

```
const Module = require('module')
```

```
const Module = require('module')  
  
const module = new Module()
```

```
const Module = require('module')

const module = new Module()

module.require('http')
```

Module.prototype.require

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
    const exports = origRequire.apply(this, arguments)

    if (id === 'http') {
        const request = exports.request

        exports.request = function () {
            const req = request.apply(this, arguments)

            console.log('New outgoing request to', req.getHeader('Host'))

            return req
        }
    }

    return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
const Module = require('module')

const origRequire = Module.prototype.require

Module.prototype.require = function (id) {
  const exports = origRequire.apply(this, arguments)

  if (id === 'http') {
    const request = exports.request

    exports.request = function () {
      const req = request.apply(this, arguments)

      console.log('New outgoing request to', req.getHeader('Host'))

      return req
    }
  }

  return exports
}
```

```
npm install require-in-the-middle
```

```
const path = require('path')
const Hook = require('require-in-the-middle')

// Hook into the express and mongodb module
Hook(['express', 'mongodb'], function (exports, name, basedir) {
  const { version } = require(path.join(basedir, 'package.json'))

  console.log('loading %s@%s', name, version)

  // expose the module version as a property on its exports object
  exports._version = version

  // whatever you return will be returned by `require`
  return exports
})
```

But wait a second...

ECMAScript Modules 😕

Patching Transpiled Modules or:

How I Learned to Stop Worrying and Love TypeScript & Babel

```
23 * import { parse } from 'graphql';
24 * import { parse } from 'graphql/language';
25 */
26
27 // The GraphQL.js version info.
28 export { version, versionInfo } from './version';
29
30 // The primary entry point into fulfilling a GraphQL request.
31 export { GraphQLArgs, graphql, graphqlSync } from './graphql';
32
33 // Create and operate on GraphQL type definitions and schema.
34 export {
35   // Definitions
36   GraphQLSchema,
37   GraphQLDirective,
38   GraphQLScalarType,
39   GraphQLObjectType,
40   GraphQLInterfaceType,
41   GraphQLUnionType,
42   GraphQLEnumType,
43   GraphQLInputObjectType,
44   GraphQLList,
45   GraphQLNonNull,
46   // Standard GraphQL Scalars
```

```
1 "use strict";
2
3 Object.defineProperty(exports, "__esModule", {
4   value: true
5 });
6 Object.defineProperty(exports, "version", {
7   enumerable: true,
8   get: function get() {
9     return _version.version;
10  }
11 });
12 Object.defineProperty(exports, "versionInfo", {
13   enumerable: true,
14   get: function get() {
15     return _version.versionInfo;
16  }
17 });
18 Object.defineProperty(exports, "graphql", {
19   enumerable: true,
20   get: function get() {
21     return graphql.graphql;
```

```
1 "use strict";
2
3 Object.defineProperty(exports, "__esModule", {
4   value: true
5 });
6 Object.defineProperty(exports, "graphql", {
7   enumerable: true,
8   get: function() {
9     return _v
10    }
11 });
12 Object.defineProperty(exports, "GraphQL", {
13   enumerable: true,
14   get: function() {
15     return _v
16    }
17 });
18 Object.defineProperty(exports, "graphql", {
19   enumerable: true,
20   get: function() {
21     return graphql;
22   }
23 });
24
25 module.exports = exports;
26
27 // This is a test for the issue described in https://github.com/karesu/reflect-metadata/issues/10
28 // It shows that the 'reflect-metadata' package does not correctly handle the 'Object.defineProperty' method
29 // when it is used to define a property on a module-level variable.
30 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
31 // defined in the 'exports' object of the current module.
32 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
33 // defined in the 'exports' object of the current module.
34 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
35 // defined in the 'exports' object of the current module.
36 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
37 // defined in the 'exports' object of the current module.
38 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
39 // defined in the 'exports' object of the current module.
40 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
41 // defined in the 'exports' object of the current module.
42 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
43 // defined in the 'exports' object of the current module.
44 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
45 // defined in the 'exports' object of the current module.
46 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
47 // defined in the 'exports' object of the current module.
48 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
49 // defined in the 'exports' object of the current module.
50 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
51 // defined in the 'exports' object of the current module.
52 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
53 // defined in the 'exports' object of the current module.
54 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
55 // defined in the 'exports' object of the current module.
56 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
57 // defined in the 'exports' object of the current module.
58 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
59 // defined in the 'exports' object of the current module.
60 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
61 // defined in the 'exports' object of the current module.
62 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
63 // defined in the 'exports' object of the current module.
64 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
65 // defined in the 'exports' object of the current module.
66 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
67 // defined in the 'exports' object of the current module.
68 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
69 // defined in the 'exports' object of the current module.
70 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
71 // defined in the 'exports' object of the current module.
72 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
73 // defined in the 'exports' object of the current module.
74 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
75 // defined in the 'exports' object of the current module.
76 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
77 // defined in the 'exports' object of the current module.
78 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
79 // defined in the 'exports' object of the current module.
80 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
81 // defined in the 'exports' object of the current module.
82 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
83 // defined in the 'exports' object of the current module.
84 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
85 // defined in the 'exports' object of the current module.
86 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
87 // defined in the 'exports' object of the current module.
88 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
89 // defined in the 'exports' object of the current module.
90 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
91 // defined in the 'exports' object of the current module.
92 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
93 // defined in the 'exports' object of the current module.
94 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
95 // defined in the 'exports' object of the current module.
96 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
97 // defined in the 'exports' object of the current module.
98 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
99 // defined in the 'exports' object of the current module.
100 // The 'reflect-metadata' package is used to generate metadata for the 'graphql' variable
101 // defined in the 'exports' object of the current module.
```

```
npm install shallow-clone-shim
```

```
Object.defineProperty(exports, 'foo' {
  enumerable: true,
  get: function get () {
    return 'hello'
  }
})  
  
const clone = require('shallow-clone-shim')  
  
const newExports = clone({}, exports, {
  foo (descriptor) {
    // descriptor == Object.getOwnPropertyDescriptor(exports, 'foo')
    const getter = descriptor.get
    descriptor.get = function get () {
      return getter() + ' world'
    }
    return descriptor
  }
})
```

Patching Async Calls

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout

global.setTimeout = function (callback, ...args) {
  const origSpan = tracer.currentSpan

  return origSetTimeout(function () {
    const preCallbackSpan = tracer.currentSpan
    tracer.currentSpan = origSpan

    callback.apply(this, arguments)

    tracer.currentSpan = preCallbackSpan
  }, ...args)
}
```

```
const origSetTimeout = global.setTimeout  
  
global.setTimeout = function (callback, ...args) {  
    const origSpan = tracer.currentSpan  
  
    return origSetTimeout(function () {  
        const preCallbackSpan = tracer.currentSpan  
        tracer.currentSpan = origSpan  
  
        callback.apply(this, arguments)  
  
        tracer.currentSpan = preCallbackSpan  
    }, ...args)  
}
```

All async API's

- net
- http
- child_process
- timers
- dns
- fs
- zlib
- crypto
- global.Promise

```
require('async_hooks')
```

But wait a second...

Async/await + Thenables 😕

Resource Usage

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')

const currentSpans = new Map()

asyncHooks.createHook({
  init (asyncId) {
    const span = tracer.currentSpan

    if (span) {
      currentSpans.set(asyncId, span)
    }
  },
  destroy (asyncId) {
    currentSpans.delete(asyncId)
  }
})
```

```
const asyncHooks = require('async_hooks')
```

```
const currentSpans = new Map()
```

```
asyncHooks.createHook({  
  init (asyncId) {  
    const span = tracer.currentSpan
```

Can we trust (when)
destroy is called?

```
    if (span) {  
      currentSpans.set(asyncId, span)  
    }  
  },  
  
  destroy (asyncId) {  
    currentSpans.delete(asyncId)  
  }  
})
```

Not a WeakMap 😢

Userland Callback Queues

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

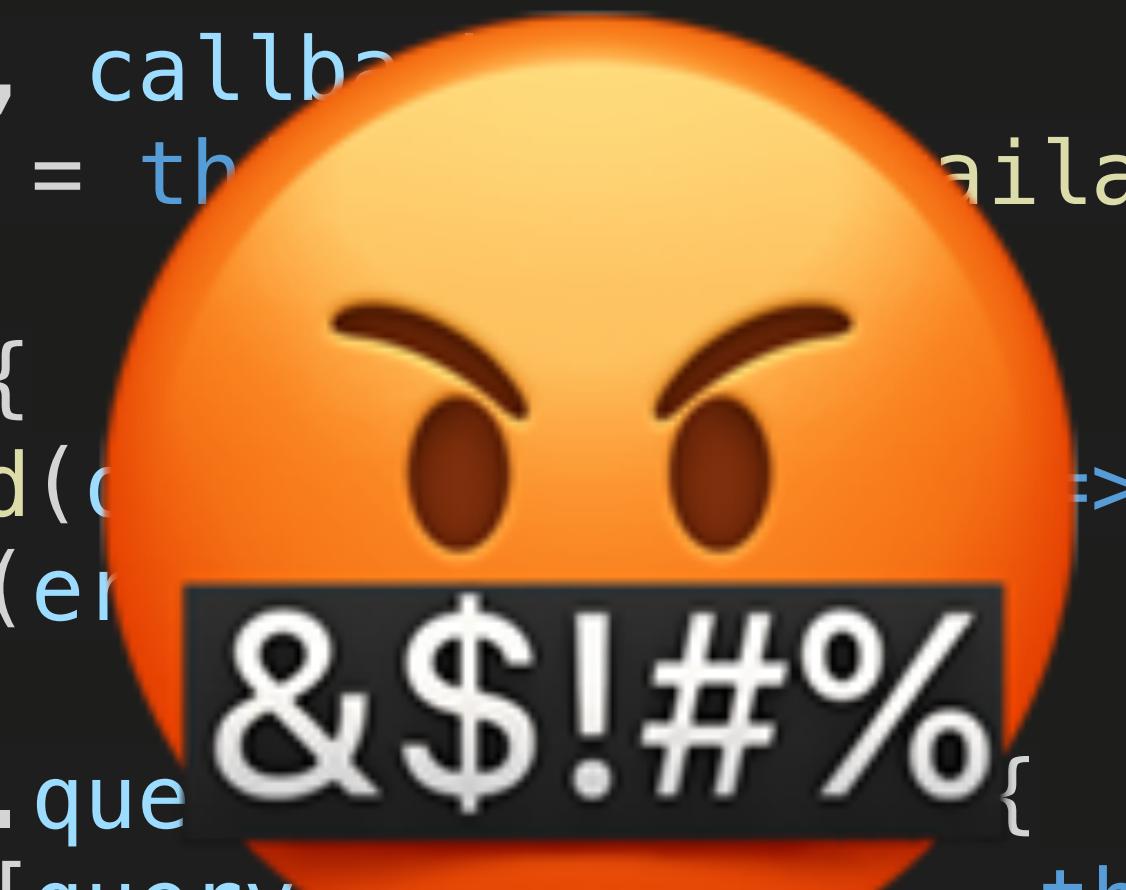
```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query, (err, res) => {  
        callback(err, res)  
  
        if (this.queue.length > 0) {  
          const [query, callback] = this.queue.shift()  
          this.request(query, callback)  
        }  
      })  
    } else {  
      this.queue.push([query, callback])  
    }  
  }  
}
```

Called in the context
of the previous
'socket.send()' call

```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
    this.queue = []  
  }  
  
  request (query, callback) {  
    const socket = this.pool.getAvailableSocket()  
  
    if (socket) {  
      socket.send(query)  
      return callback(null)  
    } else {  
      if (this.queue.length) {  
        const [query, callback] = this.queue.shift()  
        this.request(query, callback)  
      }  
    }  
  }  
}
```



```
class Client {  
  constructor () {  
    this.pool = new Pool({ size: 5 })  
  }
```

Which is called in
the context of the
'socket.send()' call

This.currentSpan turns to null after a POST request #515



rannygmx opened this issue on 15 Aug 2018 · 20 comments

Called in the context
of the previous
'socket.send()' call

```
const socket = this.pool.getAvailableSocket()  
  
if (socket) {  
  socket.send(query, (err, res) => {  
    callback(err, res)  
  
    if (this.queue.length > 0) {  
      const [query, callback] = this.queue.shift()  
      this.request(query, callback)  
    }  
  })  
} else {  
  this.queue.push([query, callback])  
}  
}
```



watson commented on 15 Aug 2018

Member

+ 😊 ...

Hi @rannygmx

Thanks for letting us know about this issue. The problem you're experiencing is most likely related to a context propagation issue. This happens when your application performs an async operation that we didn't anticipate. This is when a callback is scheduled to be called after some async operation finishes, but we fail to register it. If so, we don't know which transaction was the active transaction when the callback is finally called, and so `currentTransaction` is suddenly `null`.

90% of the time this is because of what's called a "user-land callback queue", where for instance a database-driver maintains an internal pool of connections to the databases. When you then make a database query, but all connections in the pool are in use, your query is put into a queue that waits until one of the connections in the pool becomes available. If the Node.js agent doesn't know about this pool, we'll also not know which transaction should be activated once the result of the query is returned.

Step one in figuring out what's going on in your particular case is to take a look at your dependencies. If you don't mind, could you please share with us the list of dependencies in your `package.json` file?

```
class Client {
```



rannygmx commented on 15 Aug 2018

Author + 😊 ...

Thanks for the explanation,
This is my dependencies list:

```
"dependencies": {  
  "async": "^2.4.1",  
  "body-parser": "^1.12.3",  
  "compression": "^1.4.3",  
  "cors": "^2.6.0",  
  "cron": "^1.2.1",  
  "dom-js": "^0.0.9",  
  "elastic-apm-node": "^1.7.0",  
  "elasticsearch": "^13.1.1",  
  "errorhandler": "^1.3.5",  
  "express": "^4.12.3",  
  "kafka-node": "^0.3.2",  
  "lodash": "^4.13.1",  
  "method-override": "^2.3.2",
```



Interservice Communication

GET /products HTTP/1.1
Host: www.example.com 
Date: Mon, 29 Oct 2018 16:11:05 GMT
Connection: keep-alive
Content-Length: 0

<magic header>

```
GET /products HTTP/1.1
Host: www.example.com
<magic header>
Date: Mon, 29 Oct 2018 16:11:05 GMT
Connection: keep-alive
Content-Length: 0
```

GET /products HTTP/1.1

Host: www.example.com

traceparent: 00-82c5500f40667e5500e9ae8e9711553c-992631f881f78c3b-01

Date: Mon, 29 Oct 2018 16:11:05 GMT

Connection: keep-alive

Content-Length: 0

GET /products HTTP/1.1
Host: www.example.com
traceparent: 00-82c550e0-992631f881f78c3b-01
Date: Mon, 29 Oct 2018
Connection: keep-alive
Content-Length: 0

Candidate
Recommendation



Trace Context Working Group

GET /products HTTP/1.1

Host: www.example.com

traceparent: 00-82c5500f40667e5500e9ae8e9711553c-992631f881f78c3b-01

Date: Mon, 29 Oct 2018 16:11:05 GMT

Connection: keep-alive

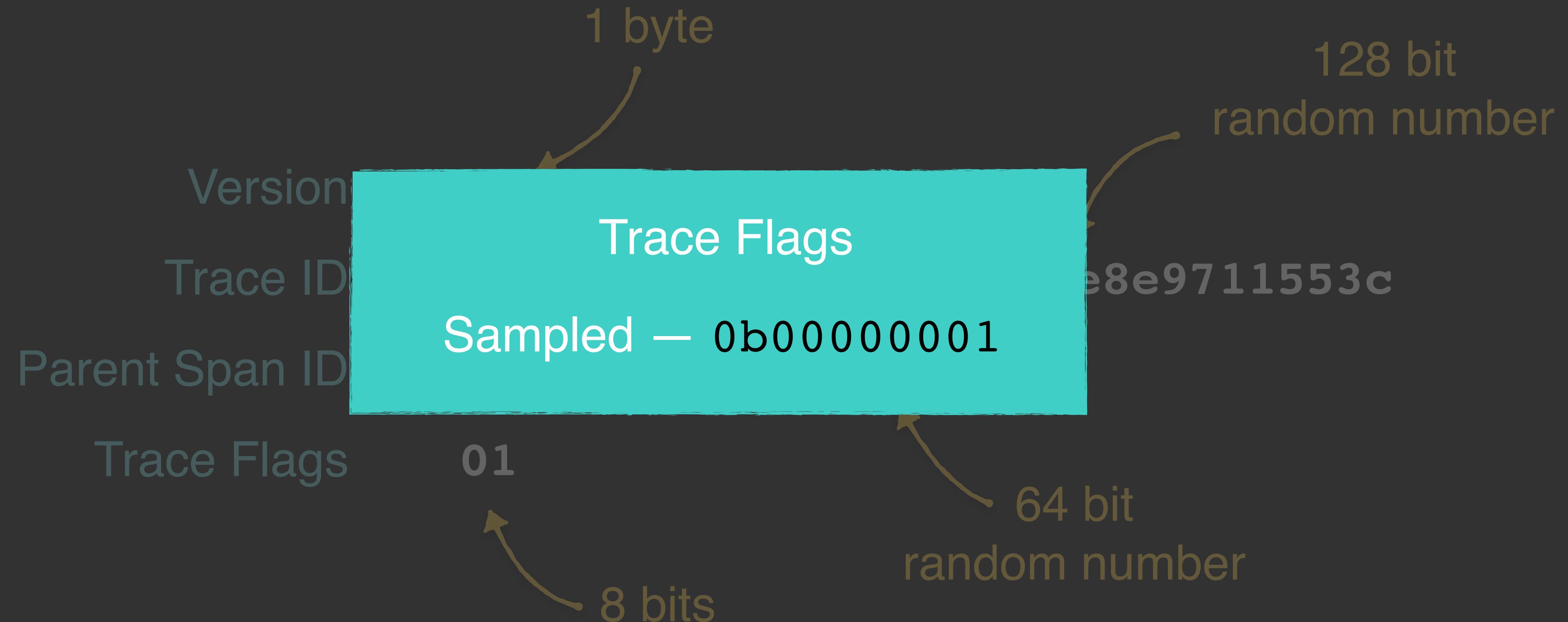
Content-Length: 0

traceparent

Version	00	1 byte
Trace ID	82c5500f40667e5500e9ae8e9711553c	128 bit random number
Parent Span ID	992631f881f78c3b	
Trace Flags	01	8 bits



traceparent



github.com / w3c / trace-context-binary

The screenshot shows a GitHub repository page for the 'w3c/trace-context-binary' project. The repository has 9 commits, 2 branches, 0 releases, and 2 contributors. The latest commit was made 22 days ago. The repository URL is <https://w3c.github.io/trace-context-b...>.

Key statistics:

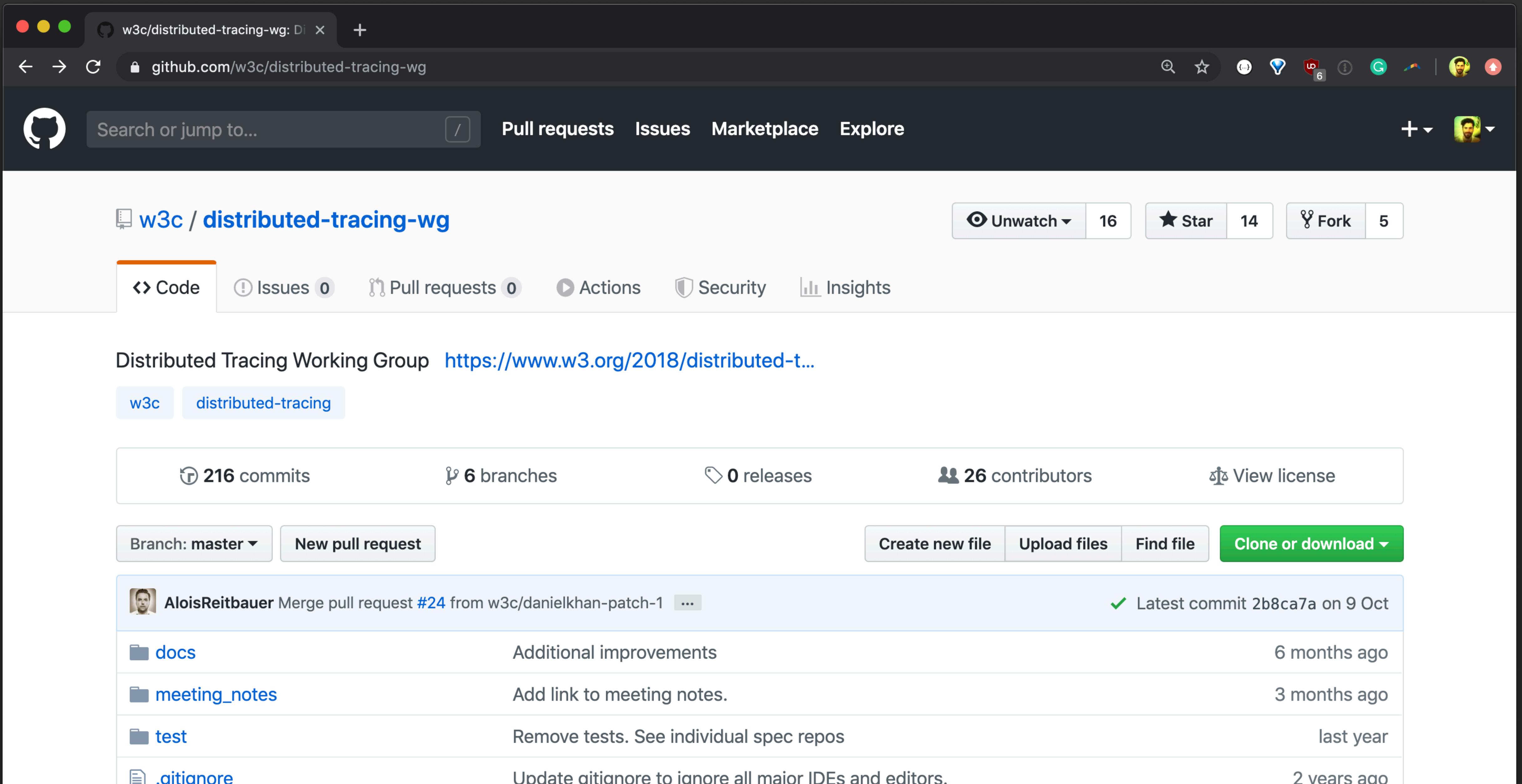
- 9 commits
- 2 branches
- 0 releases
- 2 contributors
- View license

Recent activity:

- SergeyKanzhelev Initial proposal for binary protocol (#2) ... Latest commit 571cafa 22 days ago
- spec Initial proposal for binary protocol (#2) 22 days ago
- CODE_OF_CONDUCT.md Adding baseline CODE_OF_CONDUCT.md 3 months ago
- CONTRIBUTING.md Moved from w3c/trace-context 3 months ago
- LICENSE.md Moved from w3c/trace-context 3 months ago
- README.md Moved from w3c/trace-context 3 months ago
- index.html Initial proposal for binary protocol (#2) 22 days ago

Standards

github.com / w3c / distributed-tracing-wg



The screenshot shows the GitHub repository page for `w3c/distributed-tracing-wg`. The repository name is displayed at the top left. The header includes a search bar, navigation links for Pull requests, Issues, Marketplace, and Explore, and user profile icons. Below the header, the repository name is shown again with a book icon. To the right are buttons for Unwatch (16), Star (14), and Fork (5). A navigation bar below the header includes tabs for Code (selected), Issues (0), Pull requests (0), Actions, Security, and Insights.

Distributed Tracing Working Group <https://www.w3.org/2018/distributed-t...>

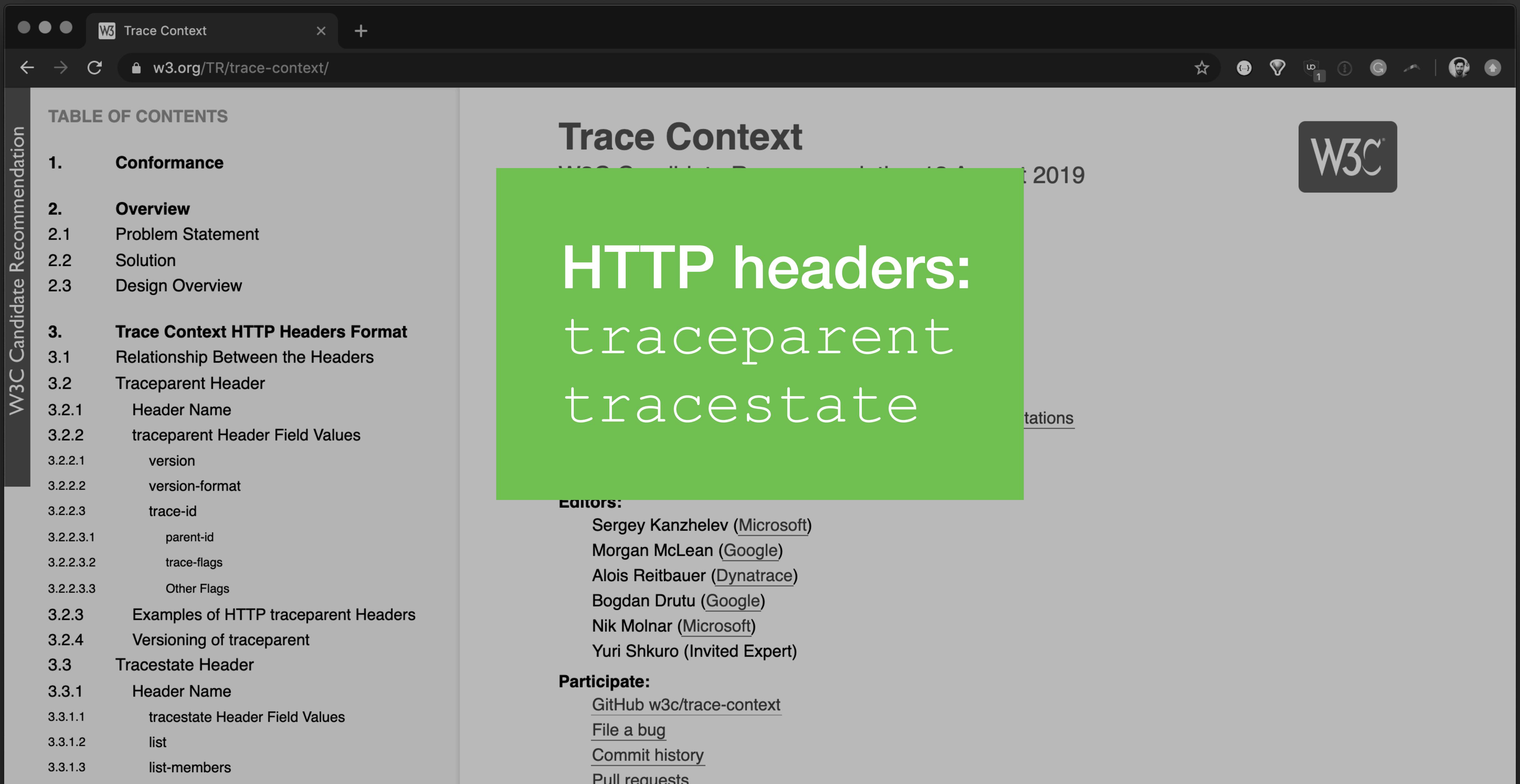
Branch: master ▾ New pull request Create new file Upload files Find file Clone or download ▾

Icon	Commit Details	Last Commit
	AloisReitbauer Merge pull request #24 from w3c/danielkhan-patch-1 ...	✓ Latest commit 2b8ca7a on 9 Oct
	docs Additional improvements	6 months ago
	meeting_notes Add link to meeting notes.	3 months ago
	test Remove tests. See individual spec repos	last year
	.gitignore Update gitignore to ignore all major IDEs and editors.	2 years ago

The screenshot shows a web browser window with the following details:

- Title Bar:** W3 Trace Context
- Address Bar:** w3.org/TR/trace-context/
- Toolbar:** Includes standard browser icons for back, forward, search, and refresh.
- W3C Candidate Recommendation Sidebar:** A vertical sidebar on the left with a blue header "W3C Candidate Recommendation". It contains the "TABLE OF CONTENTS" and a detailed list of sections and sub-sections for the Trace Context specification.
- Main Content Area:**
 - Section Header:** Trace Context
 - Text:** W3C Candidate Recommendation 13 August 2019
 - Information Links:**
 - This version: <https://www.w3.org/TR/2019/CR-trace-context-20190813/>
 - Latest published version: <https://www.w3.org/TR/trace-context/>
 - Latest editor's draft: <https://w3c.github.io/trace-context/>
 - Implementation report: <https://github.com/w3c/trace-context/#reference-implementations>
 - Previous version: <https://www.w3.org/TR/2019/CR-trace-context-20190509/>
 - Editor Information:**
 - Editors:
 - Sergey Kanzhelev ([Microsoft](#))
 - Morgan McLean ([Google](#))
 - Alois Reitbauer ([Dynatrace](#))
 - Bogdan Drutu ([Google](#))
 - Nik Molnar ([Microsoft](#))
 - Yuri Shkuro (Invited Expert)
 - Participate:
 - [GitHub w3c/trace-context](#)
 - [File a bug](#)
 - [Commit history](#)
 - [Pull requests](#)

w3.org / TR / trace-context



The screenshot shows a web browser displaying the W3C Trace Context specification page. The title bar reads "W3 Trace Context". The URL in the address bar is "w3.org/TR/trace-context/". The page content includes a "TABLE OF CONTENTS" on the left and a main "Trace Context" section on the right. The main section features a large green banner with the text "HTTP headers: traceparent tracestate". It also lists "Editors" and "Participate" links.

W3C Candidate Recommendation

TABLE OF CONTENTS

- 1. Conformance
- 2. Overview
 - 2.1 Problem Statement
 - 2.2 Solution
 - 2.3 Design Overview
- 3. Trace Context HTTP Headers Format
 - 3.1 Relationship Between the Headers
 - 3.2 Traceparent Header
 - 3.2.1 Header Name
 - 3.2.2 traceparent Header Field Values
 - 3.2.2.1 version
 - 3.2.2.2 version-format
 - 3.2.2.3 trace-id
 - 3.2.2.3.1 parent-id
 - 3.2.2.3.2 trace-flags
 - 3.2.2.3.3 Other Flags
 - 3.2.3 Examples of HTTP traceparent Headers
 - 3.2.4 Versioning of traceparent
 - 3.3 Tracestate Header
 - 3.3.1 Header Name
 - 3.3.1.1 tracestate Header Field Values
 - 3.3.1.2 list
 - 3.3.1.3 list-members

W3C Editor's Draft

Propagation format for distributed trace context: Trace Context headers

w3c.github.io/correlation-context/

TABLE OF CONTENTS

- 1. Overview
- 2. Correlation Context HTTP Header Format
 - 2.1 Format
 - 2.1.1 Header name
 - 2.1.2 Header value
 - 2.1.3 Name format
 - 2.1.4 Value format
 - 2.1.5 Properties
 - 2.2 Examples of HTTP headers
 - 2.2.1 Example use case

This version: <https://w3c.github.io/correlation-context/>

Latest published version: <https://www.w3.org/TR/correlation-context/>

Latest editor's draft: <https://w3c.github.io/correlation-context/>

Editors:

- Sergey Kanzhelev ([Microsoft](#))
- Morgan McLean ([Google](#))
- Alois Reitbauer ([Dynatrace](#))

Participate:

- [GitHub w3c/correlation-context](#)
- [File a bug](#)
- [Commit history](#)
- [Pull requests](#)

Discussions:

- [We are on Gitter.](#)

Copyright © 2019 W3C® ([MIT](#), [ERCIM](#), [Keio](#), [Beihang](#)). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

w3c.github.io / correlation-context

W3C Editors Draft

W3 Propagation format for distribu X +
w3c.github.io/correlation-context/ 

TABLE OF CONTENTS

- 1. Overview
- 2. Correlation Context HTTP Header Format
 - 2.1 Format
 - 2.1.1 Header name
 - 2.1.2 Header value
 - 2.1.3 Name format
 - 2.1.4 Value format
 - 2.1.5 Properties
 - 2.2 Examples of HTTP headers
 - 2.2.1 Example use case

Propagation format for distributed trace context: Trace Context headers  W3C Editor's Draft 6 MAY 2019 

HTTP headers:

correlation-context

Sergey Kanzhelev ([Microsoft](#))

Morgan McLean ([Google](#))

Alois Reitbauer ([Dynatrace](#))

Participate:

[GitHub w3c/correlation-context](#)

[File a bug](#)

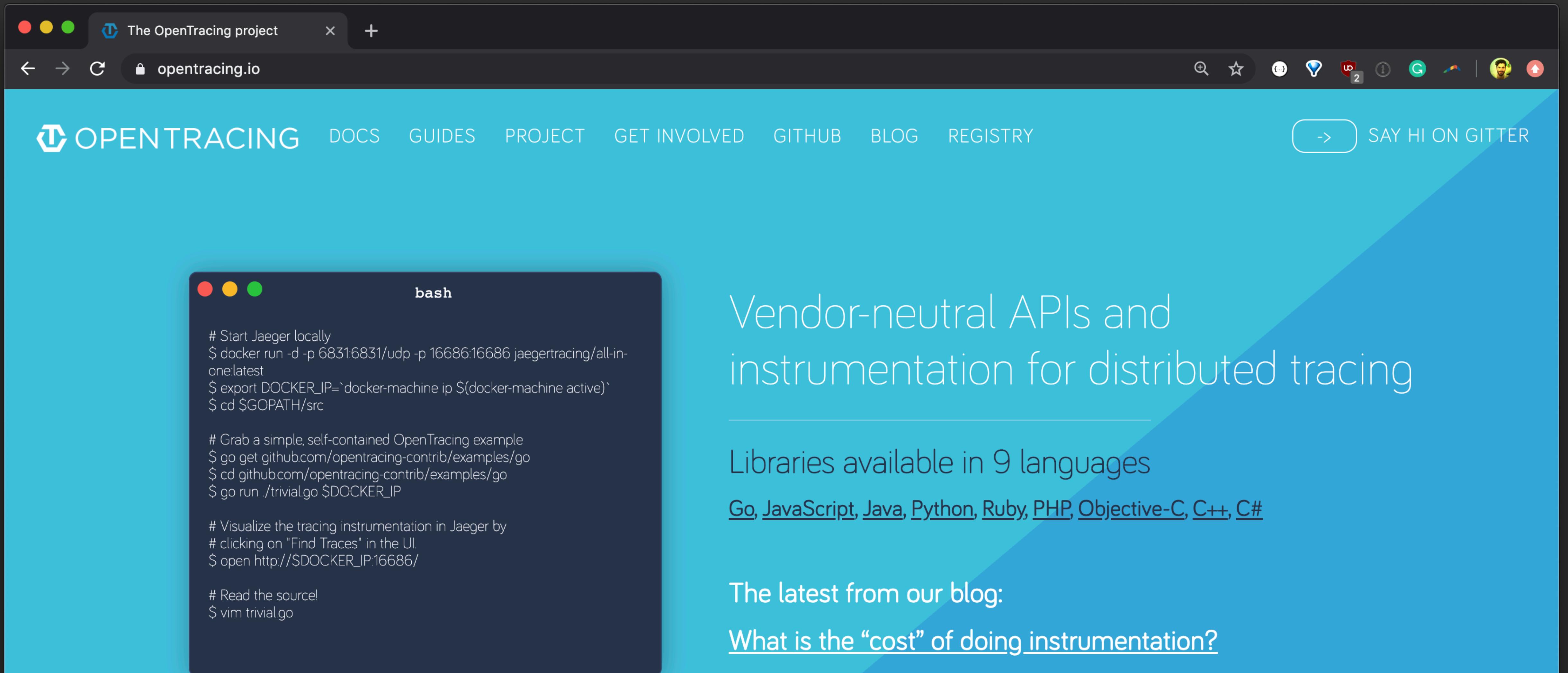
[Commit history](#)

[Pull requests](#)

Discussions:

[We are on Gitter.](#)

opentracing.io



The screenshot shows a web browser window with the title bar "The OpenTracing project" and the URL "opentracing.io". The page itself has a teal gradient background. At the top, there's a navigation bar with links for "OPENTRACING", "DOCS", "GUIDES", "PROJECT", "GET INVOLVED", "GITHUB", "BLOG", and "REGISTRY". On the right, there's a button "SAY HI ON GITTER" with a "->" icon. Below the navigation, there's a large text area with a dark background containing a terminal session titled "bash". The terminal session shows commands to start Jaeger locally, clone an OpenTracing example repository, run it, and open a browser to visualize the tracing instrumentation. To the right of this terminal session, there's a large headline "Vendor-neutral APIs and instrumentation for distributed tracing". Below that, there's a section about available libraries in 9 languages, listing Go, JavaScript, Java, Python, Ruby, PHP, Objective-C, C++, and C#. Further down, there's a section for the latest blog posts, with a link to "What is the “cost” of doing instrumentation?".

```
# Start Jaeger locally
$ docker run -d -p 6831:6831/udp -p 16686:16686 jaegertracing/all-in-one:latest
$ export DOCKER_IP=`docker-machine ip $(docker-machine active)`
$ cd $GOPATH/src

# Grab a simple, self-contained OpenTracing example
$ go get github.com/opentracing-contrib/examples/go
$ cd github.com/opentracing-contrib/examples/go
$ go run ./trivial.go $DOCKER_IP

# Visualize the tracing instrumentation in Jaeger by
# clicking on "Find Traces" in the UI.
$ open http://$DOCKER_IP:16686/

# Read the source!
$ vim trivial.go
```

Vendor-neutral APIs and instrumentation for distributed tracing

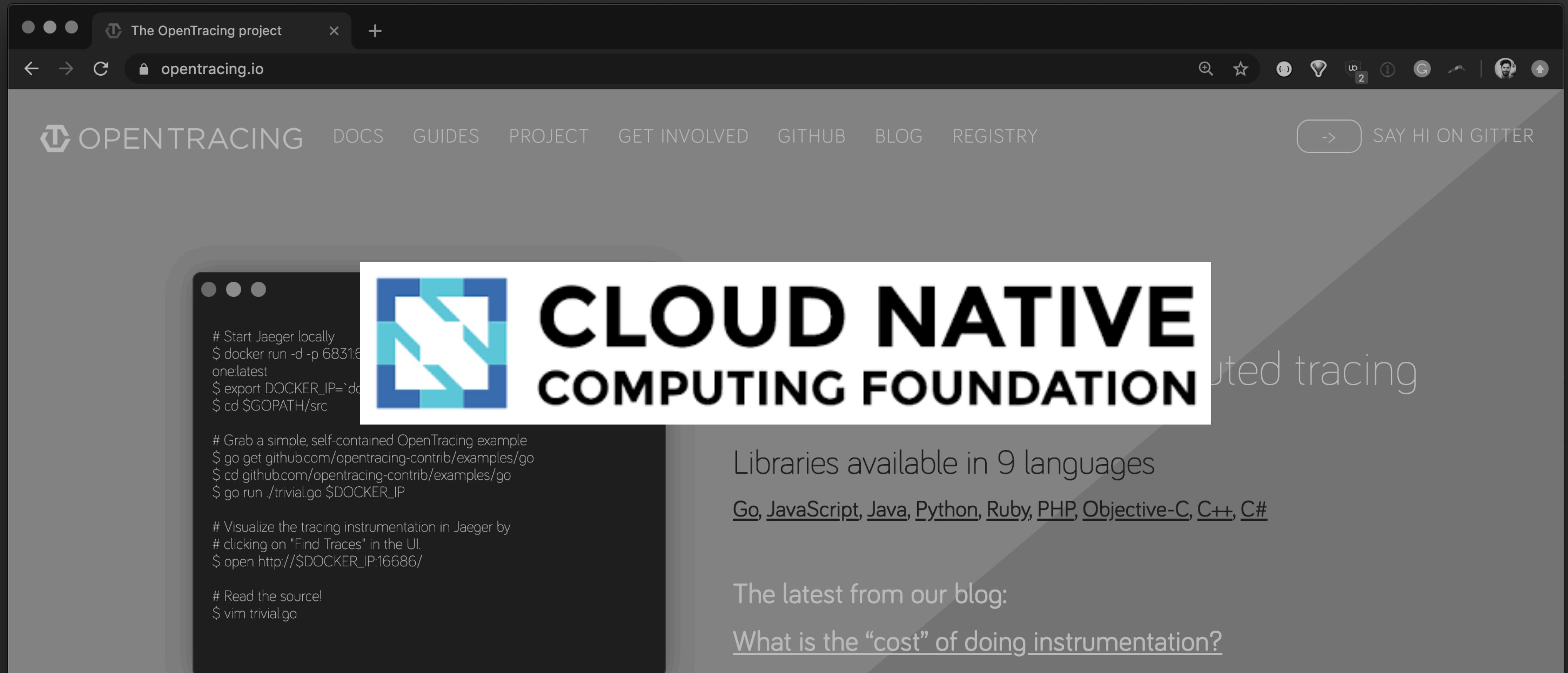
Libraries available in 9 languages

[Go](#), [JavaScript](#), [Java](#), [Python](#), [Ruby](#), [PHP](#), [Objective-C](#), [C++](#), [C#](#)

The latest from our blog:

[What is the “cost” of doing instrumentation?](#)

opentracing.io



The OpenTracing project

opentracing.io

OPEN TRACING DOCS GUIDES PROJECT GET INVOLVED GITHUB BLOG REGISTRY -> SAY HI ON GITTER

Start Jaeger locally
\$ docker run -d -p 6831:6831 jaeger/one:latest
\$ export DOCKER_IP=`docker inspect --format='{{.NetworkSettings.IPAddress}}' \$(docker ps -q)`
\$ cd \$GOPATH/src

Grab a simple, self-contained OpenTracing example
\$ go get github.com/opentracing-contrib/examples/go
\$ cd github.com/opentracing-contrib/examples/go
\$ go run ./trivial.go \$DOCKER_IP

Visualize the tracing instrumentation in Jaeger by
clicking on "Find Traces" in the UI.
\$ open http://\$DOCKER_IP:16686/

Read the source!
\$ vim trivial.go

CLOUD NATIVE COMPUTING FOUNDATION

Cloud Native Computing Foundation

Libraries available in 9 languages

Go, [JavaScript](#), [Java](#), [Python](#), [Ruby](#), [PHP](#), [Objective-C](#), [C++](#), [C#](#)

The latest from our blog:

[What is the “cost” of doing instrumentation?](#)

opentracing.io

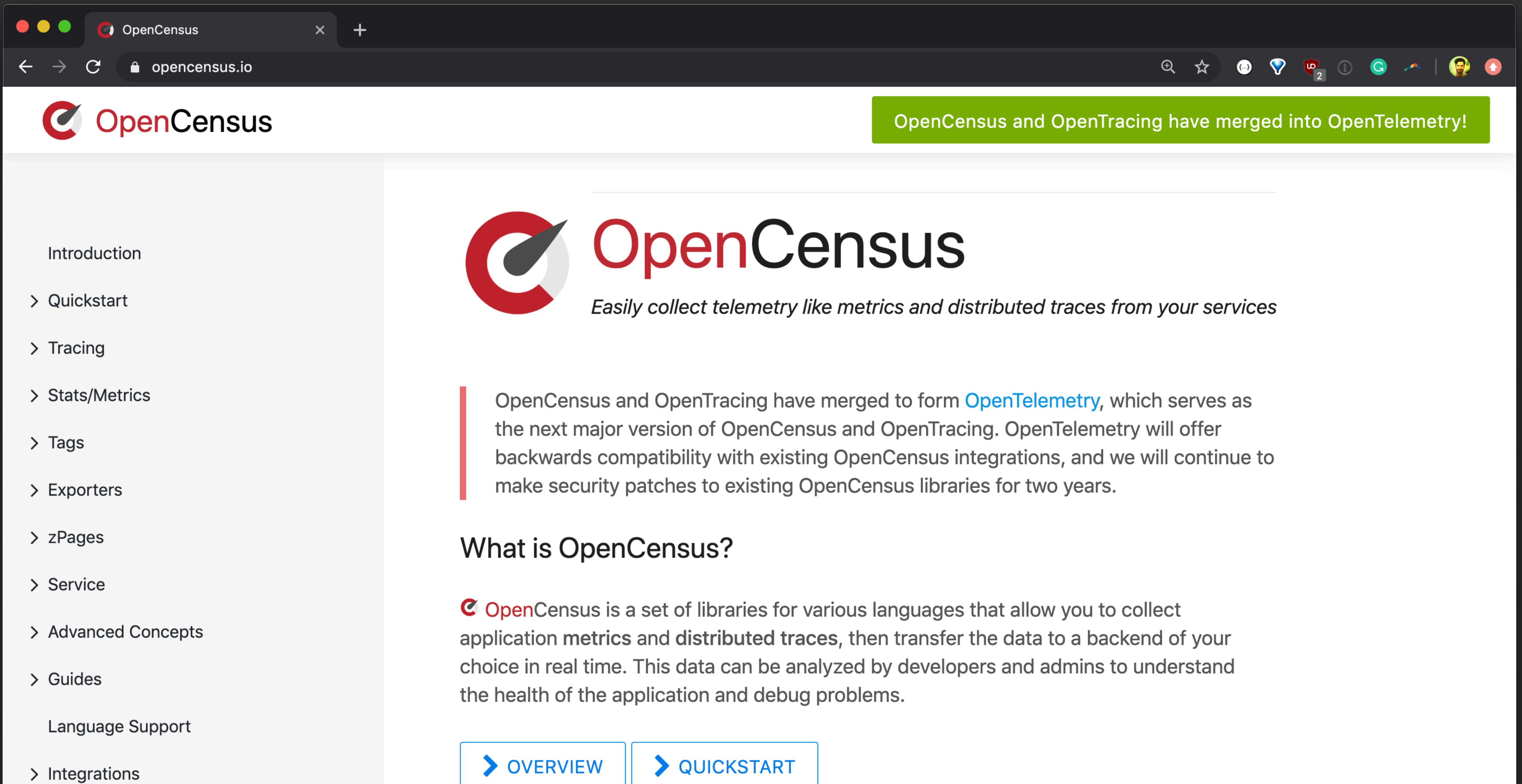
The screenshot shows a web browser window with the title bar "The OpenTracing project" and the URL "opentracing.io". The main content area displays a code snippet for handling errors in an HTTP request using the OpenTracing API. The code uses the tracer's startSpan method to create a span for the request, sets an error tag on it, logs the error details (event: 'error', error object, message, stack), and finally finishes the span.

```
const span = tracer.startSpan('http_request')

span.setTag(opentracing.Tags.ERROR, true)

span.log({
  event: 'error',
  'error.object': err,
  message: err.message,
  stack: err.stack
})

span.finish()
```



The screenshot shows a web browser displaying the opencensus.io homepage. The title bar reads "OpenCensus" and the address bar shows "opencensus.io". A green banner at the top right states "OpenCensus and OpenTracing have merged into OpenTelemetry!". The main content features the OpenCensus logo and the text "Easily collect telemetry like metrics and distributed traces from your services". A sidebar on the left lists various documentation sections: Introduction, Quickstart, Tracing, Stats/Metrics, Tags, Exporters, zPages, Service, Advanced Concepts, Guides, Language Support, and Integrations. At the bottom, there are two buttons: "OVERVIEW" and "QUICKSTART".

OpenCensus and OpenTracing have merged into OpenTelemetry!

OpenCensus

Easily collect telemetry like metrics and distributed traces from your services

Introduction

> Quickstart

> Tracing

> Stats/Metrics

> Tags

> Exporters

> zPages

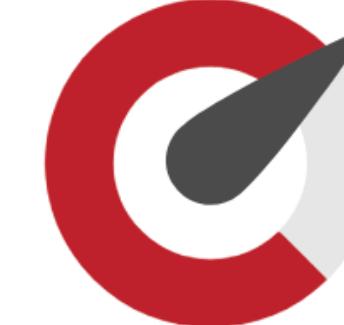
> Service

> Advanced Concepts

> Guides

Language Support

> Integrations

 OpenCensus

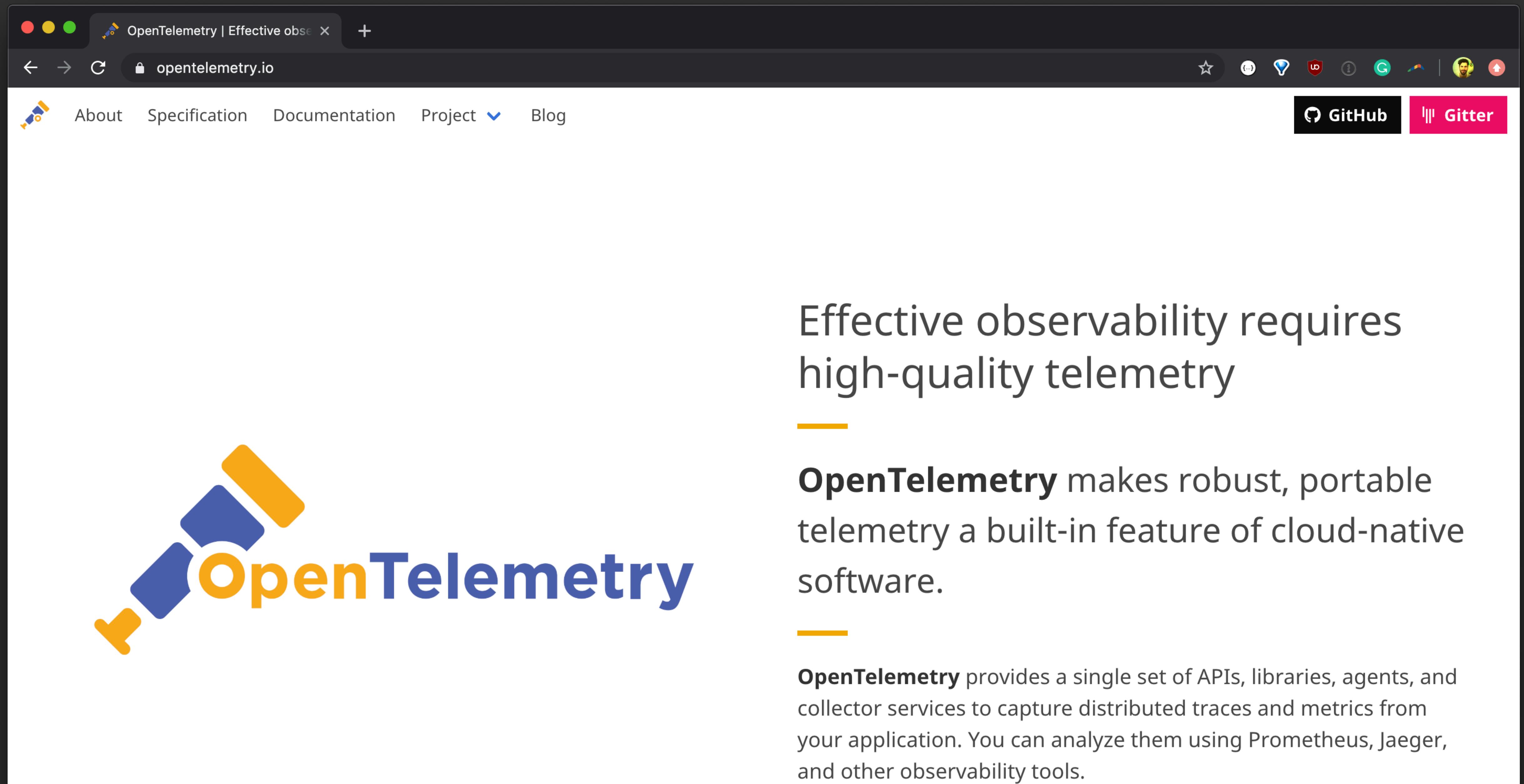
OpenCensus and OpenTracing have merged to form [OpenTelemetry](#), which serves as the next major version of OpenCensus and OpenTracing. OpenTelemetry will offer backwards compatibility with existing OpenCensus integrations, and we will continue to make security patches to existing OpenCensus libraries for two years.

What is OpenCensus?

 OpenCensus is a set of libraries for various languages that allow you to collect application metrics and distributed traces, then transfer the data to a backend of your choice in real time. This data can be analyzed by developers and admins to understand the health of the application and debug problems.

[OVERVIEW](#) [QUICKSTART](#)

opentelemetry.io



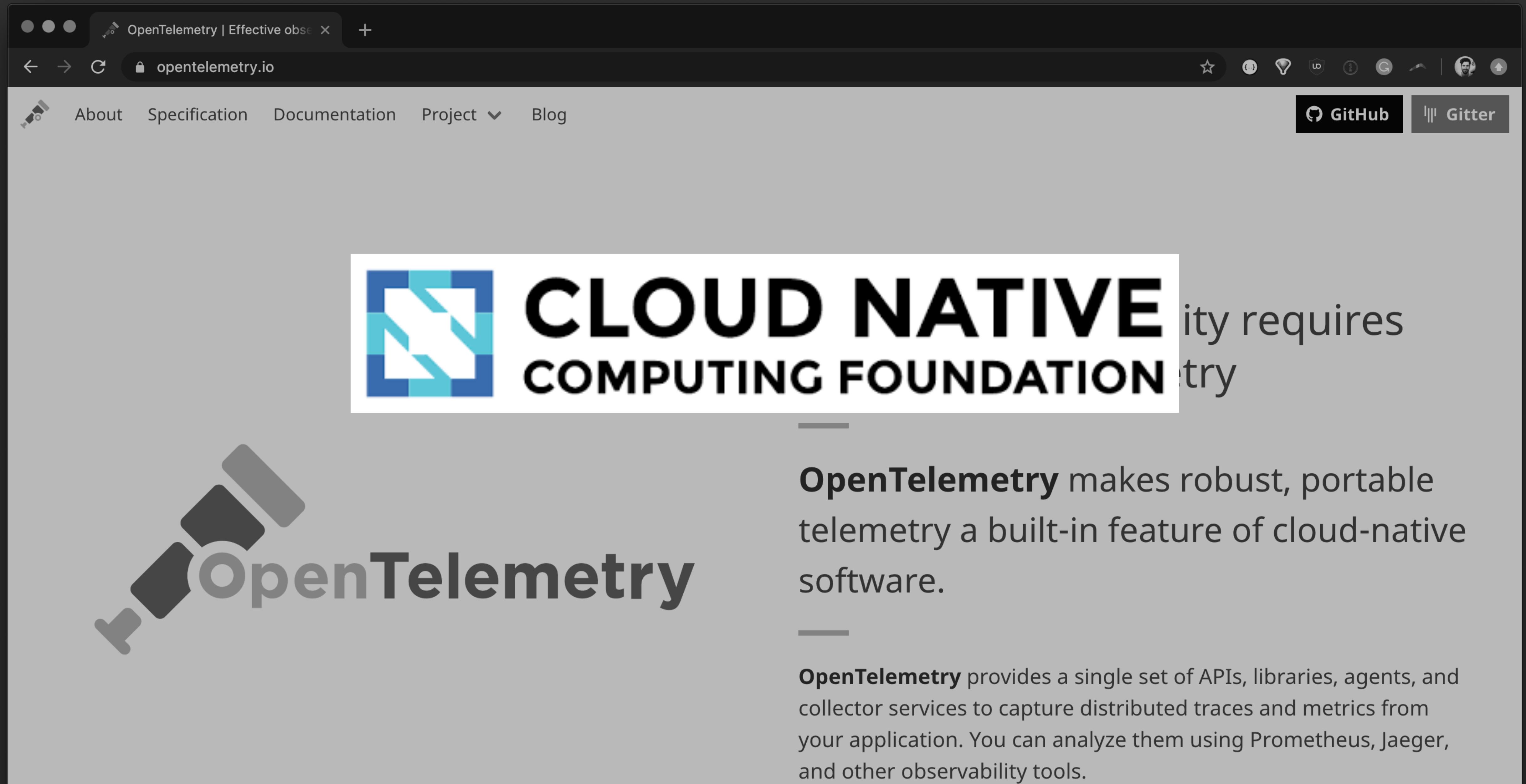
The screenshot shows the official OpenTelemetry website at opentelemetry.io. The page features a dark header with the site's logo and navigation links for About, Specification, Documentation, Project (with a dropdown menu), and Blog. On the right, there are GitHub and Gitter buttons. The main content area has a large heading "Effective observability requires high-quality telemetry" followed by three descriptive paragraphs about the project's goals and capabilities.

Effective observability requires high-quality telemetry

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

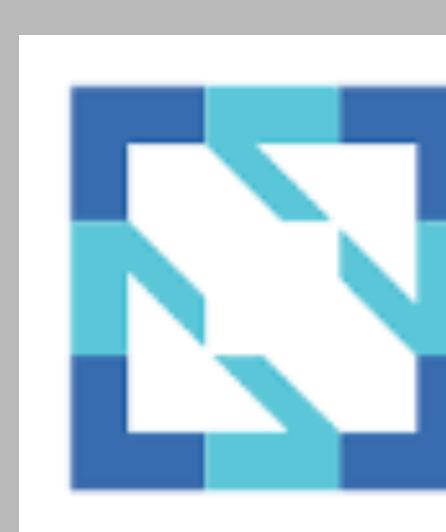
opentelemetry.io



The screenshot shows a web browser displaying the opentelemetry.io homepage. The address bar shows the URL "opentelemetry.io". The page features a navigation bar with links for "About", "Specification", "Documentation", "Project", and "Blog". On the right side of the header are buttons for "GitHub" and "Gitter". The main content area includes the Cloud Native Computing Foundation logo and a large "OpenTelemetry" logo with a stylized microphone icon. There are two sections of text describing the project's purpose and its API support.

OpenTelemetry | Effective observability

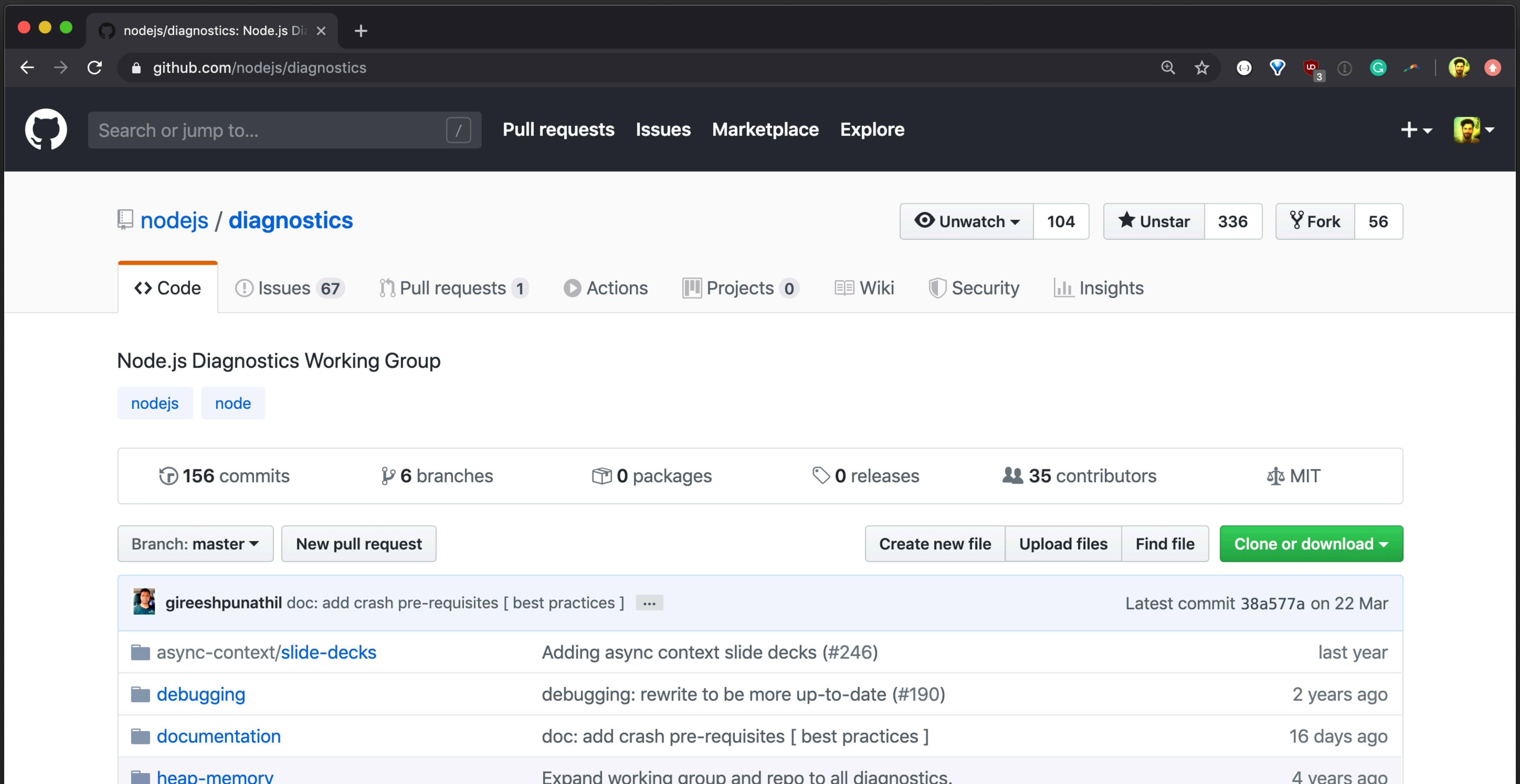
About Specification Documentation Project Blog

 CLOUD NATIVE COMPUTING FOUNDATION

OpenTelemetry makes robust, portable telemetry a built-in feature of cloud-native software.

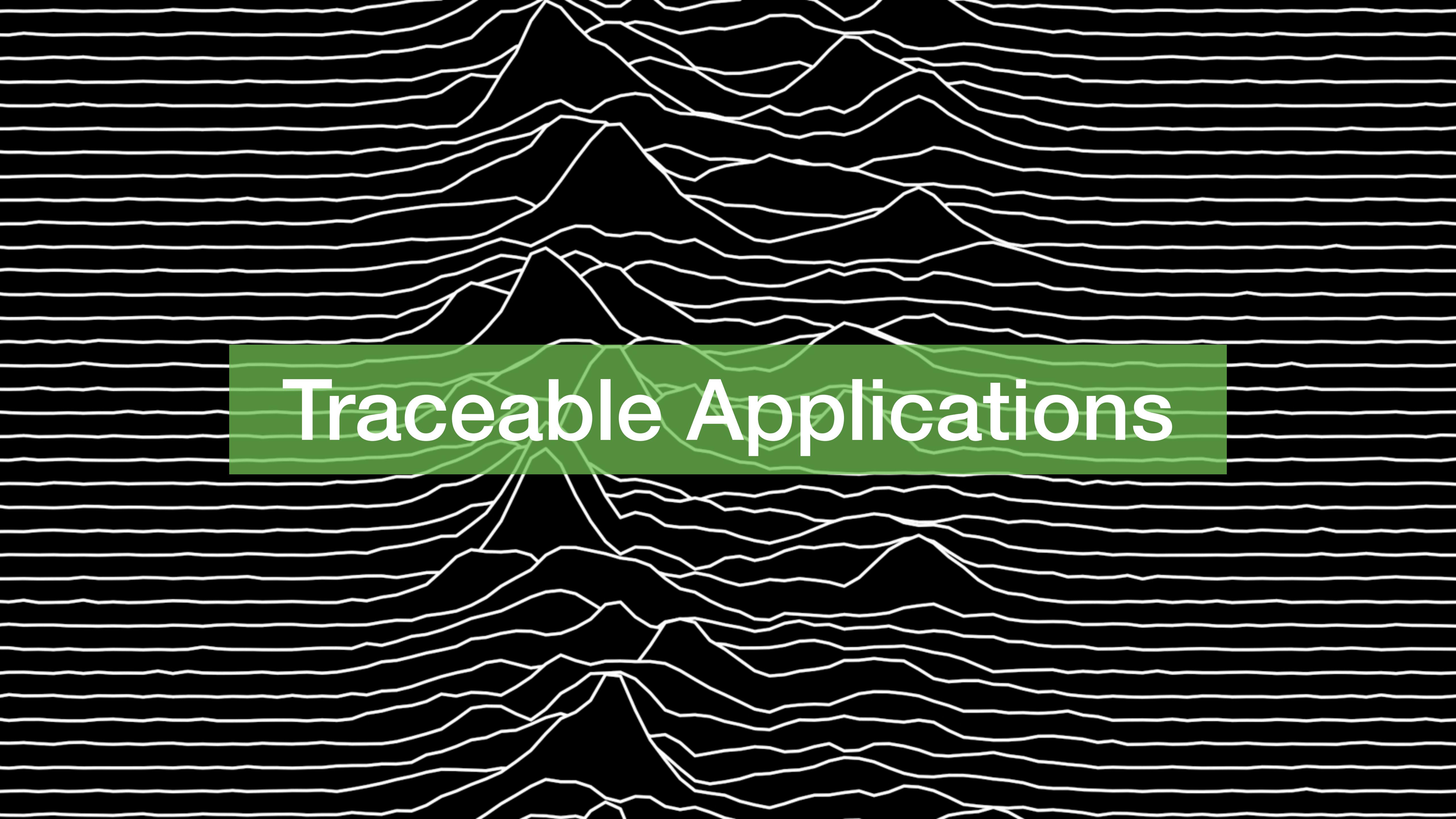
OpenTelemetry provides a single set of APIs, libraries, agents, and collector services to capture distributed traces and metrics from your application. You can analyze them using Prometheus, Jaeger, and other observability tools.

github.com / nodejs / diagnostics



The screenshot shows the GitHub repository page for `nodejs/diagnostics`. The repository name is displayed at the top left, along with a search bar, a pull requests button, an issues button, a marketplace button, and an explore button. To the right are buttons for unwatching (104), unstarring (336), and forking (56). Below this, a navigation bar includes links for Code, Issues (67), Pull requests (1), Actions, Projects (0), Wiki, Security, and Insights. The main content area is titled "Node.js Diagnostics Working Group" and features two primary buttons: "nodejs" and "node". Below these are summary statistics: 156 commits, 6 branches, 0 packages, 0 releases, 35 contributors, and an MIT license. A "Branch: master" dropdown and a "New pull request" button are located on the left, while "Create new file", "Upload files", "Find file", and a "Clone or download" button are on the right. The main feed displays recent commits, including:

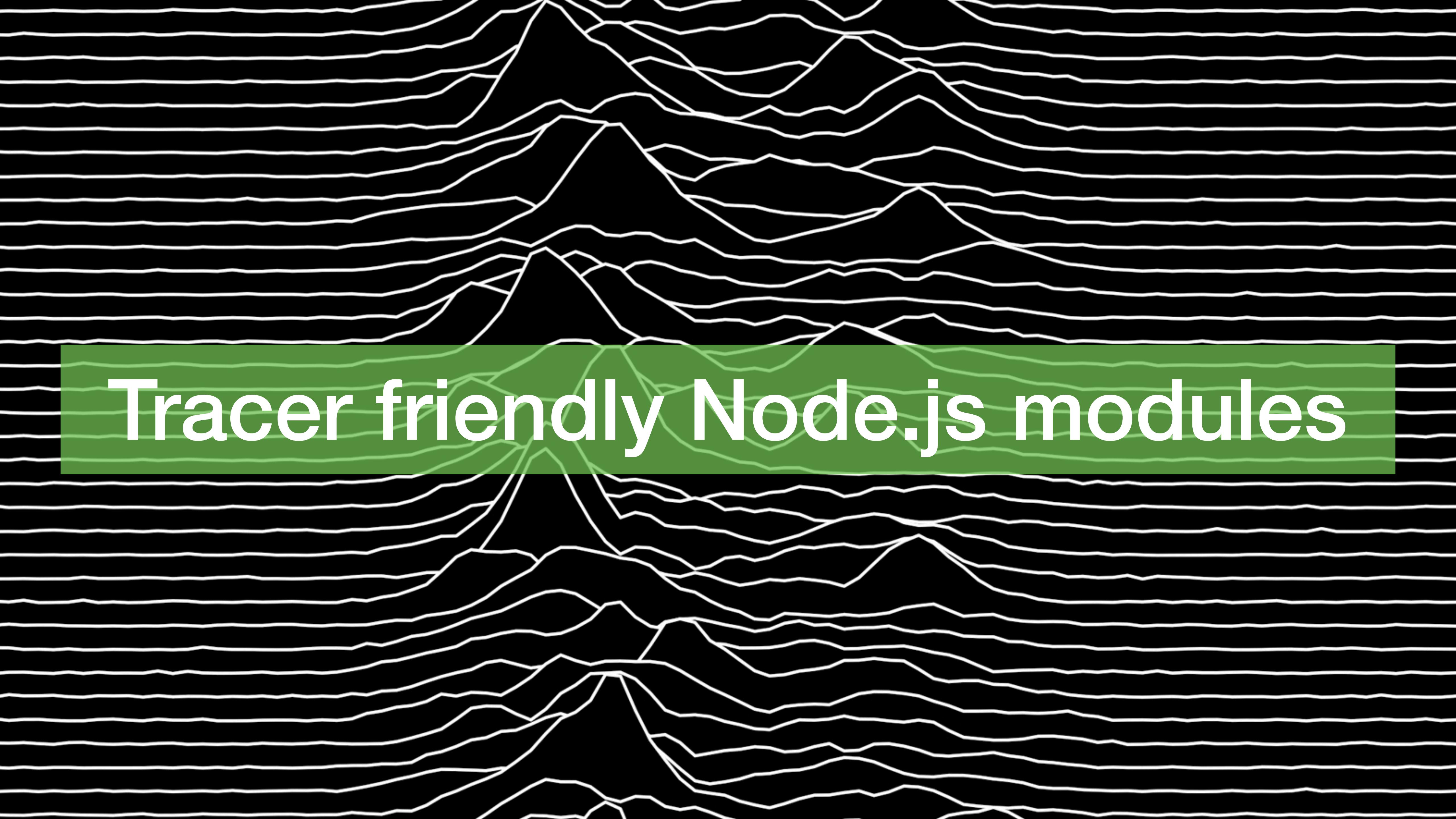
- gireeshpunathil doc: add crash pre-requisites [best practices] ... Latest commit 38a577a on 22 Mar
- async-context/slides-decks Adding async context slide decks (#246) last year
- debugging debugging: rewrite to be more up-to-date (#190) 2 years ago
- documentation doc: add crash pre-requisites [best practices] 16 days ago
- heap-memory Expand working group and repo to all diagnostics. 4 years ago



Traceable Applications

Traceable Applications

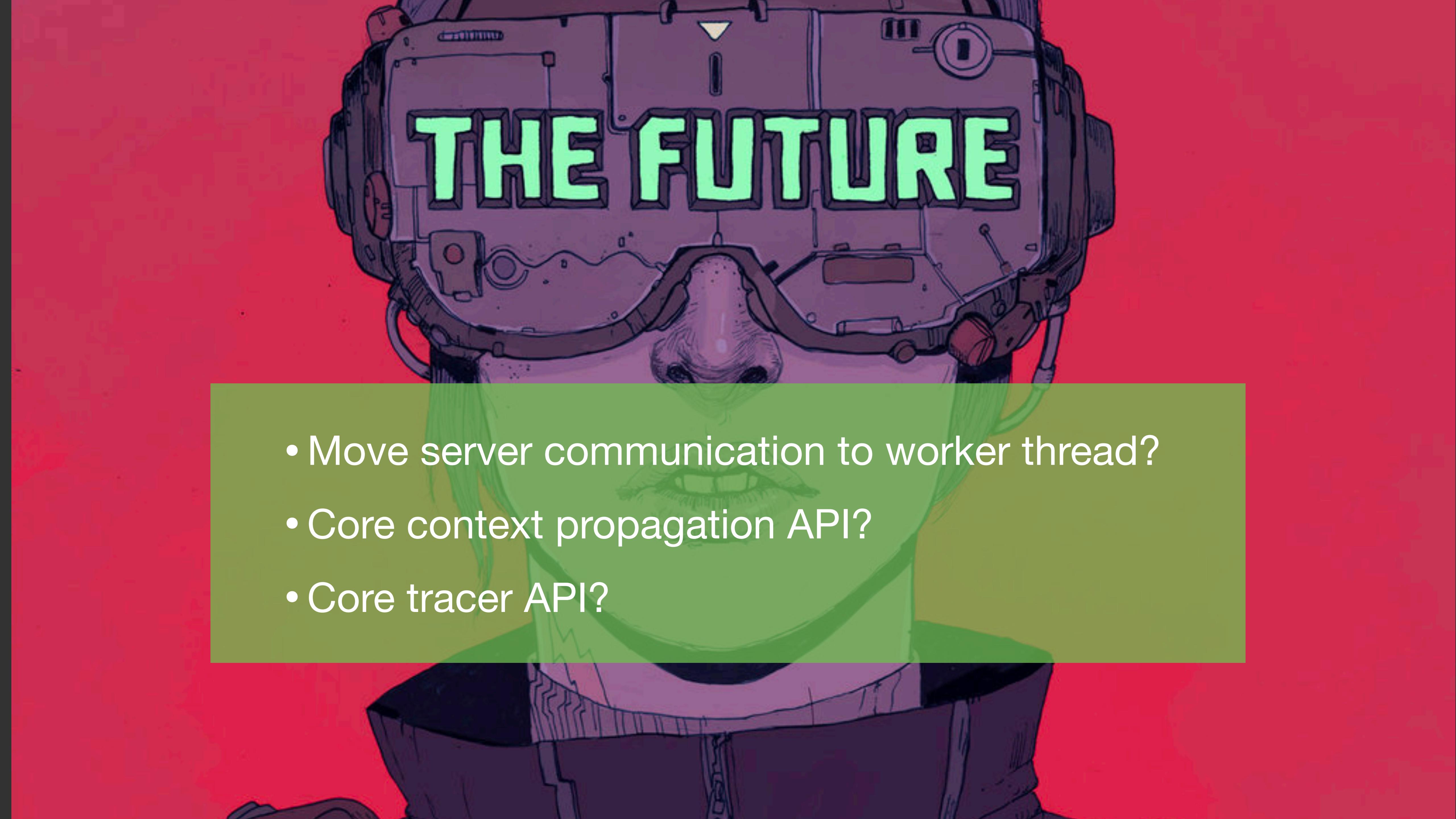
- Bad:
 - Async/await + thenables
- Challenges:
 - Userland callback queue
 - Custom protocols, message queues etc
 - Storing all data



Tracer friendly Node.js modules

Tracer friendly Node.js modules

- Bad:
 - Userland callback queue
 - Async/await + thenables
- Want:
 - Hooks

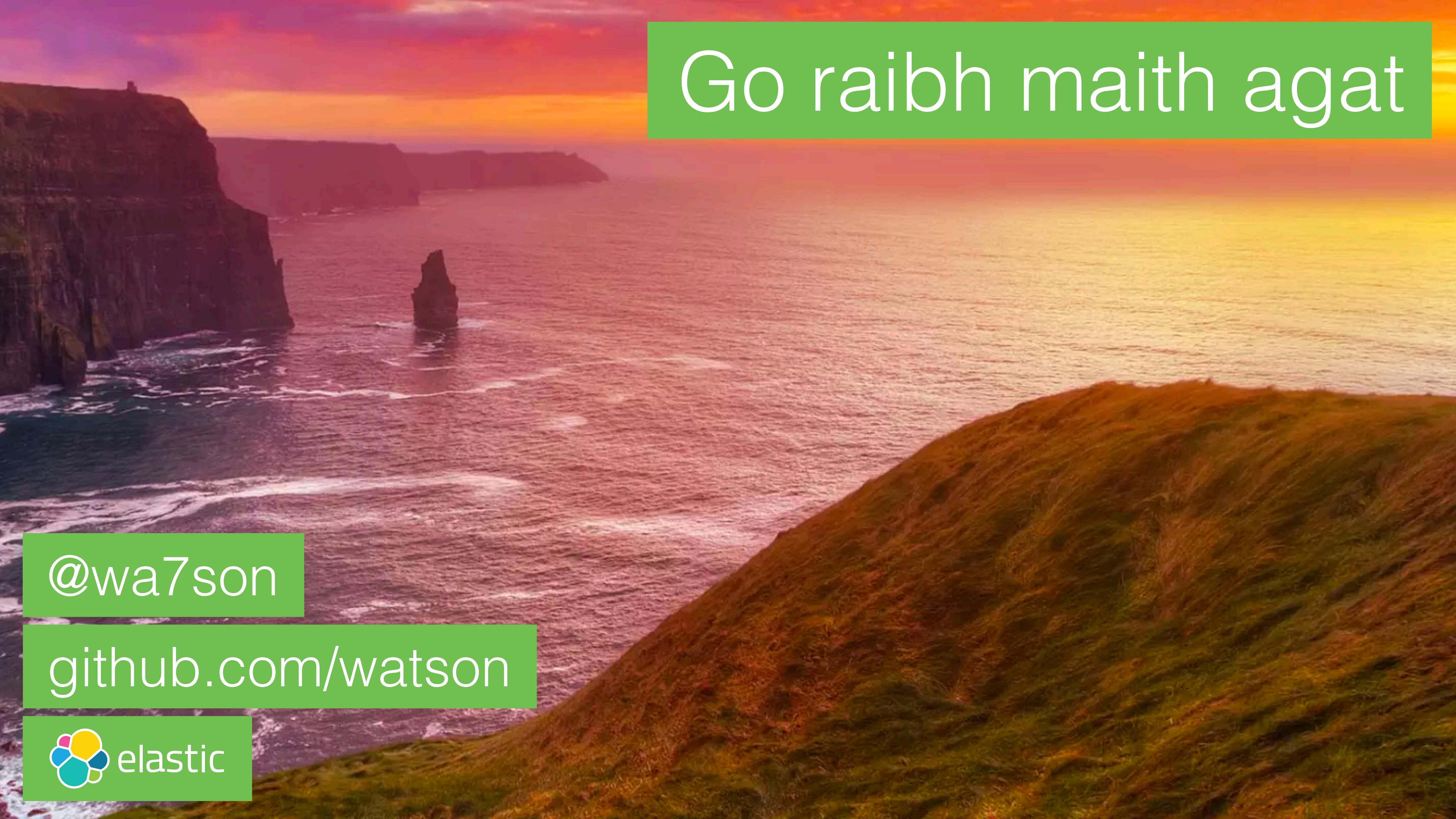


THE FUTURE

- Move server communication to worker thread?
- Core context propagation API?
- Core tracer API?



mmm

A wide-angle photograph of a coastal landscape at sunset. The sky is filled with warm orange and yellow hues. On the left, dark, rugged cliffs rise from the ocean. In the center, a tall, thin rock formation stands in the water. On the right, a grassy hillside slopes down towards the sea. A small white lighthouse is visible on top of one of the cliffs on the far left.

Go raibh maith agat

@wa7son

github.com/watson

