

Fastify

Fast and low overhead framework, for Node.js

<https://github.com/davidedantonio/wetalk-fastify>



Lead Maintainers

weBeetle



Matteo Collina

Nearform



Tomas della Vedova

Elastic

Contributors

weBeetle



Tommaso Allevi
github npm twitter



Ethan Arrowood
github npm twitter



Cemre Mengu
github npm twitter



David Clements
github npm twitter



Dustin Deus
github npm twitter



Luciano Mammino
github npm twitter



Evan Shortiss
github npm twitter



Manuel Spigolon
github npm twitter



James Sumners
github npm twitter



Nathan Woltman
github npm twitter

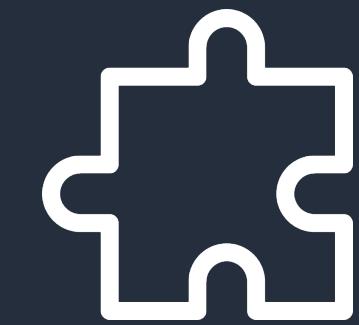




+120 release



+2100 commit



+120 plugins

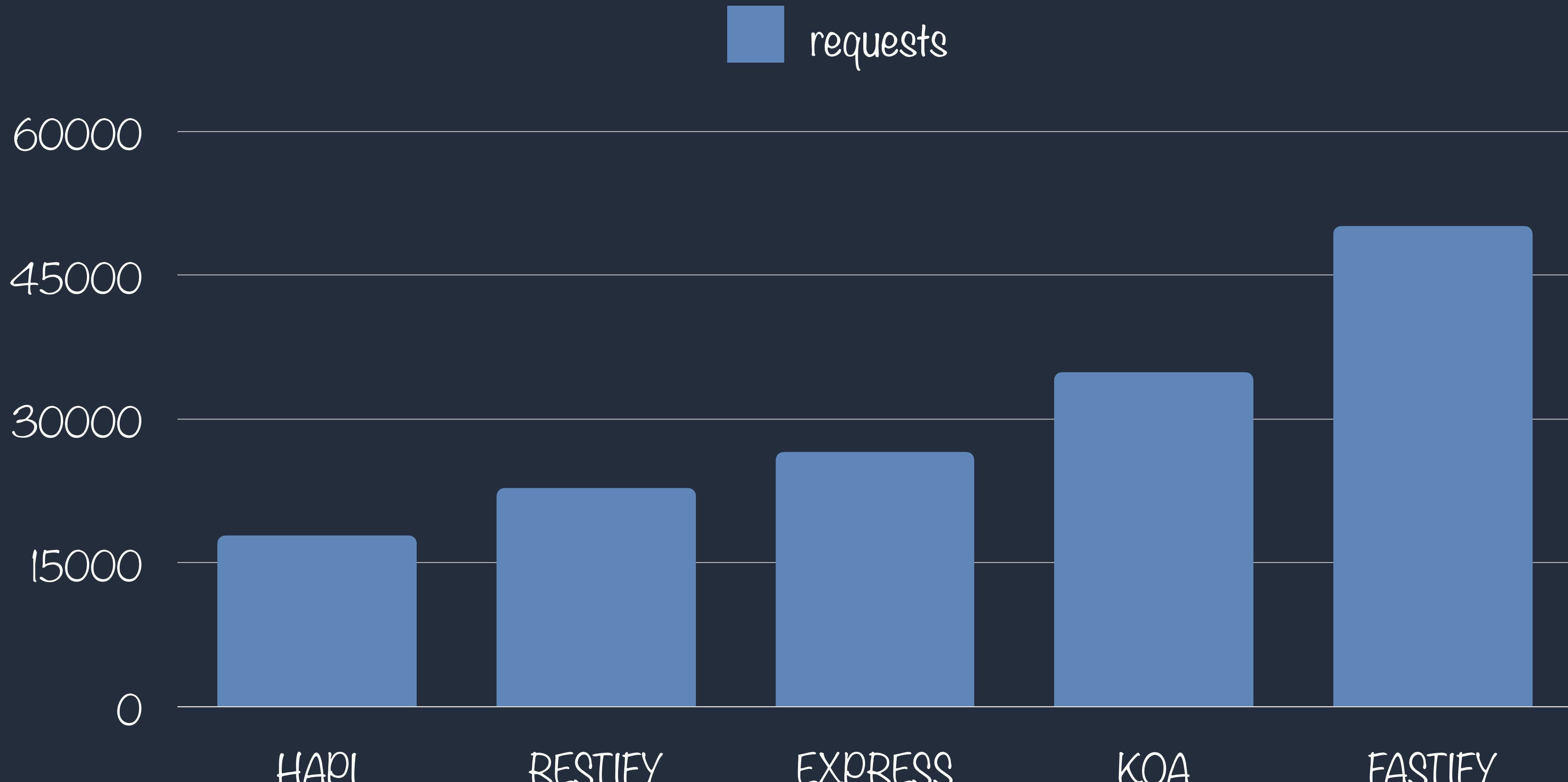


+200 collaborators



Why Fastify?





A brief example

weBeetle

```
const fastify = require('fastify')({ logger: true })

// Declare a route for our application
fastify.get('/', async (req, res) => {
  return { hello: 'world' }
})

const startServer = async port => {
  try {
    await fastify.listen(port)
    fastify.log.info(`Server started on http://localhost:${port}`)
  } catch (e) {
    fastify.log.error(e)
    process.exit(1)
  }
}

startServer(3000)
```



// demo #1 and #2



Extendible via plugins



```
fastify.register(  
  require('my-plugin'),  
  { options }  
)
```



Fastify plugins

weBeetle

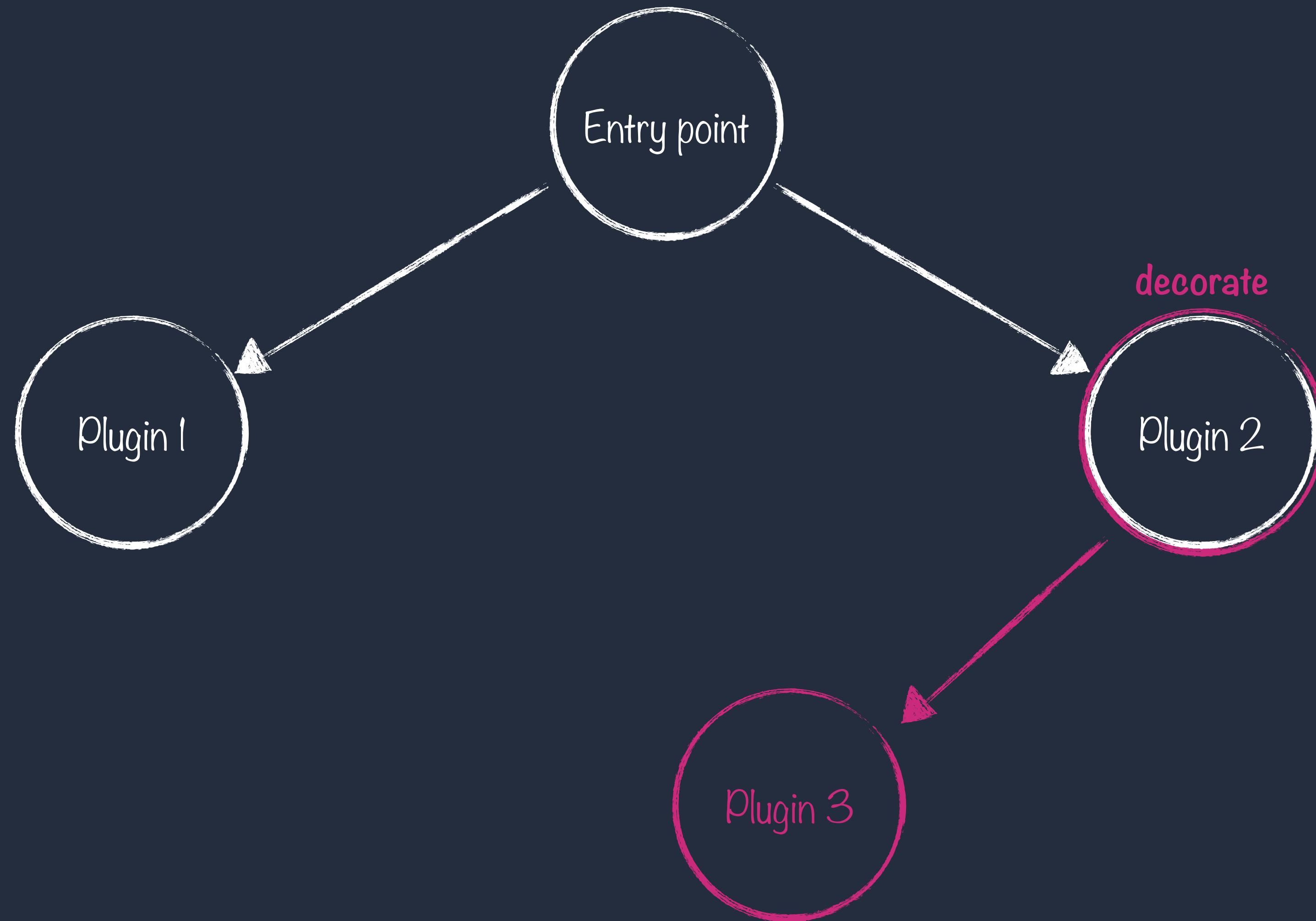
```
function myPlugin (fastify, opts, next) {  
  // add another plugin  
  fastify.register (...)  
  
  // add an Hook  
  fastify.addHook(...)  
  
  // decorate fastify instance  
  fastify.decorate(...)  
  
  // add routes  
  fastify.route(...)  
  
  next()  
}  
  
module.exports = myPlugin
```

async await
is supported as well



Plugins encapsulation

weBeetle



Exposing functionality to parents

weBeetle

```
const fp = require('fastify-plugin')
const plugin2 = require('plugin-2')

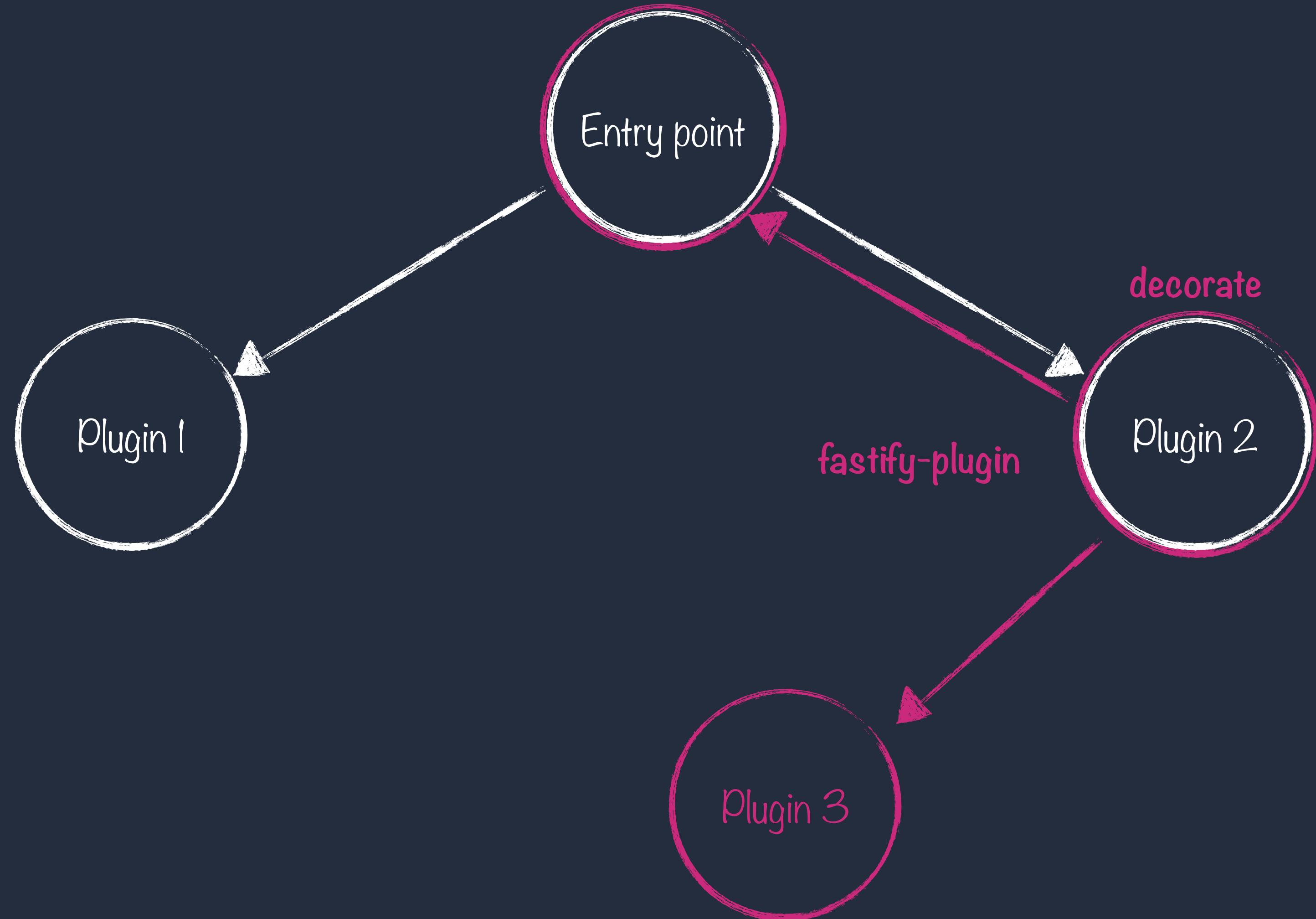
function myPlugin (fastify, opts, next) {
  fastify.decorate('plugin2', plugin2)
  next()
}

module.exports = fp(myPlugin)
```



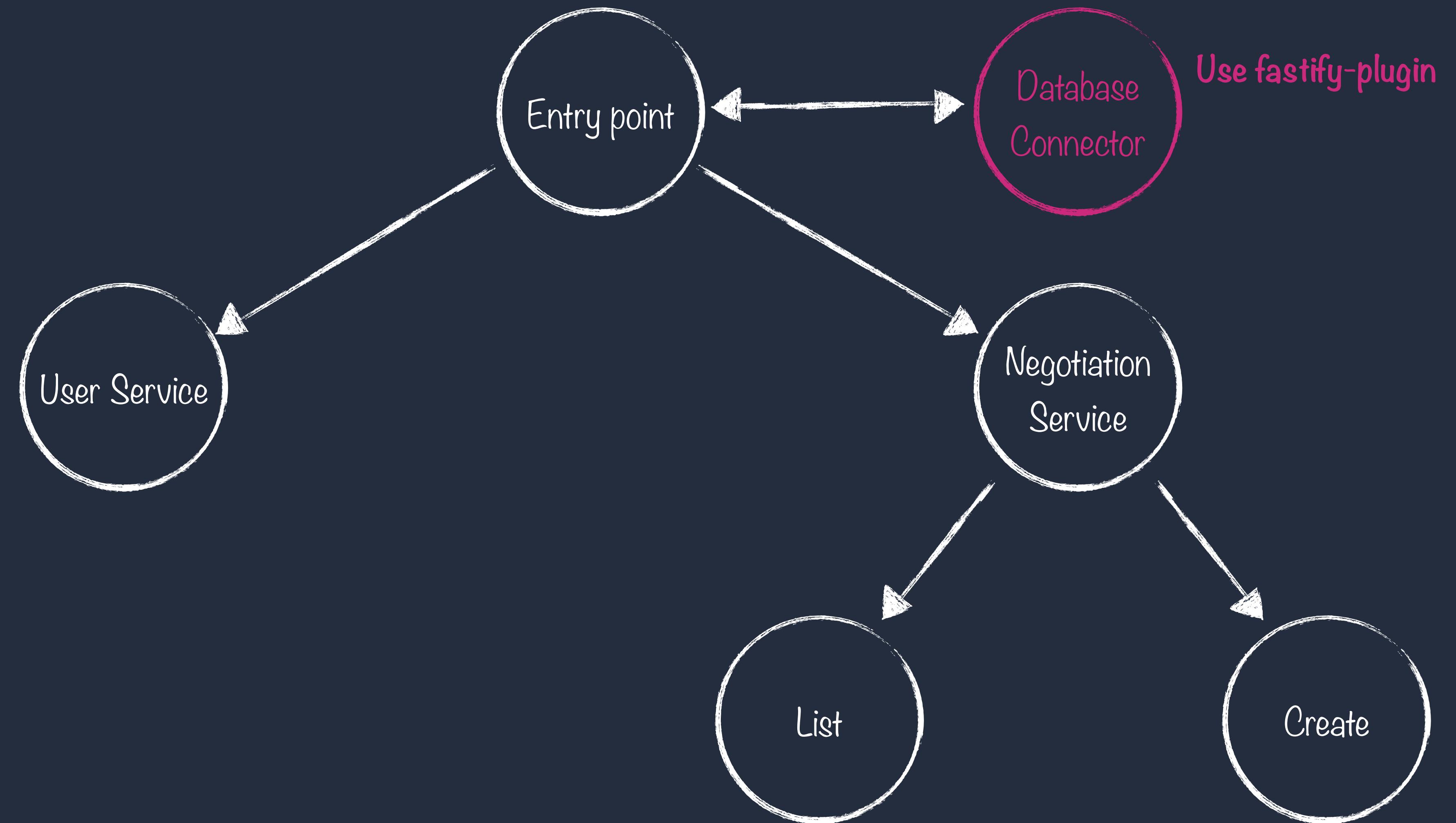
Plugins encapsulation

weBeetle



Real world example

weBeetle



Encapsulation is the keystone of fastify

weBeetle

```
const fastify = require('fastify')()
fastify.register(require('./api/v1'), {
  prefix: '/v1',
  logLevel: 'error'
})

fastify.register(require('./api/v2'), {
  prefix: '/v2',
  logLevel: 'debug'
})
```



EVERYTHING
is a plugin



// demo #3



Hooks



Request/Response Hooks

weBeetle

By using the hooks you can interact directly inside the lifecycle of Fastify. There are seven different Hooks that you can use (in order of execution):

- ✓ onRequest
- ✓ preParsing
- ✓ preValidation
- ✓ preHandler
- ✓ preSerialization
- ✓ onError
- ✓ onSend
- ✓ onResponse



Request/Response Hooks - Example

weBeetle

```
fastify.addHook('onError', (request, reply, error, next) => {  
    // apm stands for Application Performance Monitoring  
    apm.sendError(error)  
    next()  
})
```



You are able to hook into the application-lifecycle as well. It's important to note that these hooks aren't fully encapsulated. The **this** inside the hooks are encapsulated but the handlers can respond to an event outside the encapsulation boundaries.

✓ onClose

✓ onRoute



Application Hooks - onClose

weBeetle

Triggered when **fastify.close()** is invoked to stop the server

```
fastify.addHook('onClose', (instance, done) => {
  // some code
  done()
})
```



Application Hooks - onRoute

weBeetle

Triggered when a new route is registered

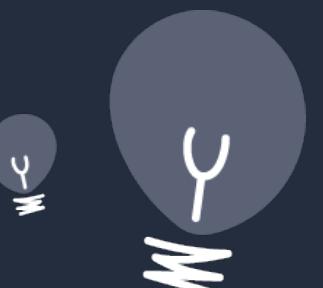
```
fastify.addHook('onRoute', (routeOptions) => {
  // some code
  routeOptions.method
  routeOptions.schema
  routeOptions.url
  routeOptions.bodyLimit
  routeOptions.logLevel
  routeOptions.prefix
})
```



Route level Hooks

weBeetle

```
fastify.route({
  method: 'GET',
  url: '/',
  schema: { ... },
  preValidation: function (request, reply, done) {
    // this hook will always be executed after the shared `preValidation` hooks
    done()
  },
  preHandler: function (request, reply, done) {
    // this hook will always be executed after the shared `preHandler` hooks
    done()
  },
  handler: function (request, reply) {
    reply.send({ hello: 'world' })
  }
})
```



// demo #4



Decorators



The API allows you to add new properties to the Fastify instance. Possible values are not restricted by type and could be functions, objects or strings, for example.

```
fastify.decorate('sum', (a, b) => {
  return a + b
})
```

```
fastify.decorate('conf', {
  db: 'some.db',
  port: 3000
})
```

```
console.log(fastify.sum(5, 3))
console.log(fastify.conf.db)
```



JSON schema



- Standard
- Fast
- You can use <http://www.jsonschema.net> to generate all schemas you want



JSON schema ... Example

weBeetle

```
fastify.post('/login', {
  schema: {
    body: S.object()
      .prop('username', S.string().required())
      .prop('password', S.string().required()),
    response: {
      200: S.object()
        .prop('token', S.string()),
      400: S.object()
        .prop('message', S.string()),
      404: S.object()
        .prop('message', S.string()),
      500: S.object()
        .prop('message', S.string())
    }
  }
}, async function (req, reply) {...}
```



Create Fastify App

<https://github.com/davidedantonio/create-fastify-app>



Create an app with create-fastify-app

weBeetle

```
$ npm install -g create-fastify-app  
$ npx create-fastify-app generate:project -d ./your-project-name  
$ cd ./your-project-name  
$ npm install  
$ npm run dev
```



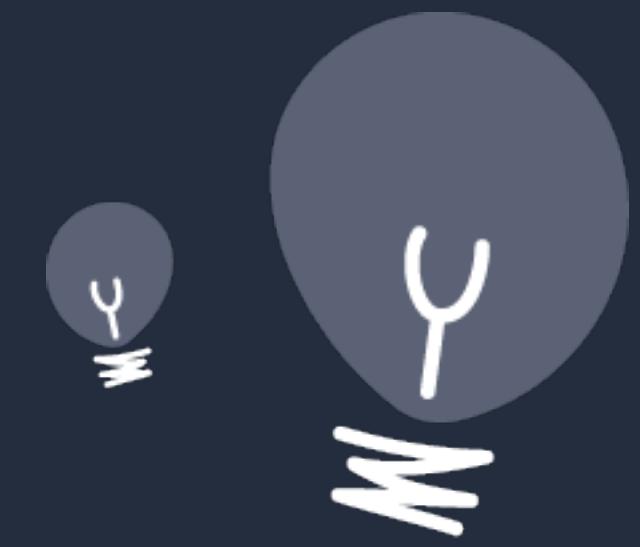
create-fastify-app project folder

weBeetle

```
./services/service-name
├── docker-compose.yml
├── Dockerfile
├── package.json
└── README.md

src
├── hooks
│   ├── onError.js
│   ├── onRequest.js
│   ├── onResponse.js
│   ├── onSend.js
│   ├── preHandler.js
│   ├── preParsing.js
│   ├── preSerialization.js
│   └── preValidation.js
├── index.js
├── plugins
│   └── README.md
└── services
    ├── hello
    │   └── index.js
    ├── README.md
    └── root.js

test
```



Our application

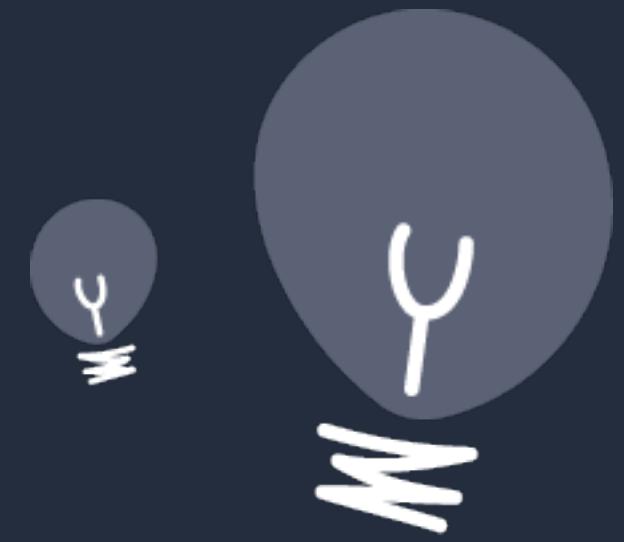
Ticketing System



Step 1 - Let's add a Database

weBeetle

```
$ fastify-app add:mongo  
? Host: localhost  
? Port: 27017  
? Collection: wetalk  
? User:  
? Password:  
$ npm install
```



Step 1 - Let's add a Database

weBeetle

```
'use strict'

const fp = require('fastify-plugin')
const MongoDB = require('fastify-mongodb')

module.exports = fp(async (fastify, opts) => {
  let mongoOpts = Object.assign({}, {
    useNewUrlParser: true,
    url: process.env.MONGODB_URL || 'mongodb://localhost:27017/wetalk',
  }, opts.mongodb)

  if (process.env.MONGODB_USER) {
    mongoOpts = Object.assign(mongoOpts, {
      auth: {
        user: process.env.MONGODB_USER || 'root',
        password: process.env.MONGODB_PASSWORD || ''
      }
    })
  }

  fastify.register(MongoDB, mongoOpts)
})
```



Step 2 - Let's create ticket endpoints

weBeetle

```
$ fastify-app generate:service
```

```
? Service Name: ticket
? What methods do you want to generate
  ⚡ DELETE
  ⚡ GET
  ⚡ HEAD
  ⚡ PATCH
  ⚡ POST
  ⚡ PUT
  > ⚡ OPTIONS
? Route Prefix /api
File /ticket/index.js generated successfully with DELETE, GET, POST, PUT methods
File /ticket/README.md generated successfully
File /test/services/ticket.test.js generated successfully
Service ticket generated successfully
```



Step 2 - Let's create ticket endpoints

weBeetle

```
module.exports = async function(fastify, opts) {
  fastify.delete('/ticket', async function (request, reply) {
    // Implement your business logic here...

    return reply
      .code(200)
      .send({data: 'DELETE ok!'})
  })

  fastify.get('/ticket', async function (request, reply) {
    // Implement your business logic here...

    return reply
      .code(200)
      .send({data: 'GET ok!'})
  })

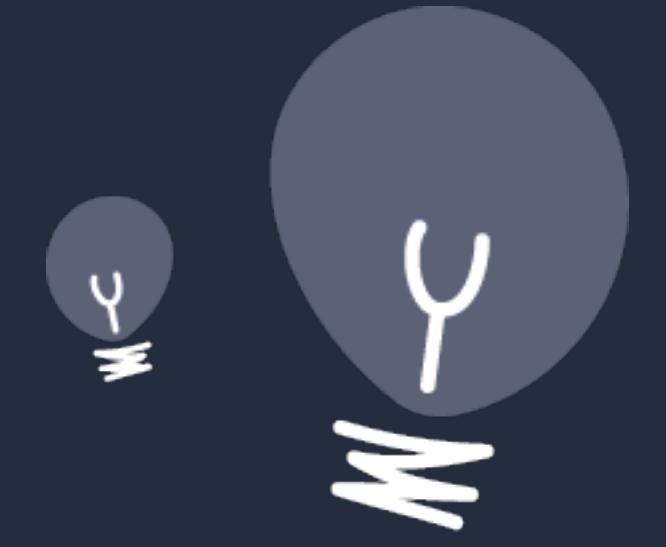
  fastify.post('/ticket', async function (request, reply) {
    // Implement your business logic here...

    return reply
      .code(200)
      .send({data: 'POST ok!'})
  })

  fastify.put('/ticket', async function (request, reply) {
    // Implement your business logic here...

    return reply
      .code(200)
      .send({data: 'PUT ok!'})
  })
}

module.exports.autoPrefix = '/api'
```



// demo #5



Authentication

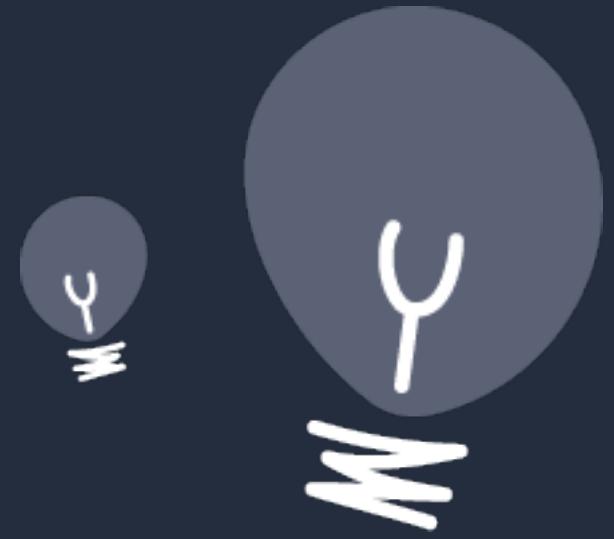
JWT



Step 3 - Add fastify-jwt

weBeetle

```
$ npm install --save fastify-jwt  
$ npm install --save secure-password
```



Step 3 - Add fastify-jwt

weBeetle

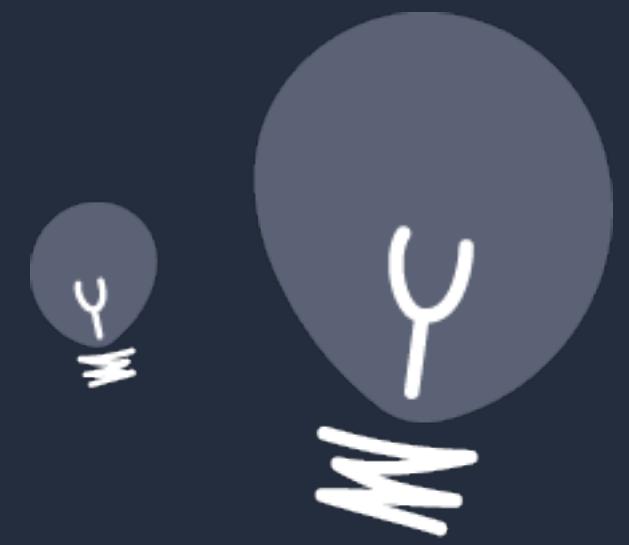
```
'use strict'

const fp = require('fastify-plugin')
const JWT = require('fastify-jwt')

module.exports = fp(async (fastify, opts) => {

  let jwt0pts = Object.assign({}, {
    secret: process.env.JWT_SECRET || 'ThisIsSecret'
  }, opts.jwt)

  fastify.register(JWT, jwt0pts)
})
```

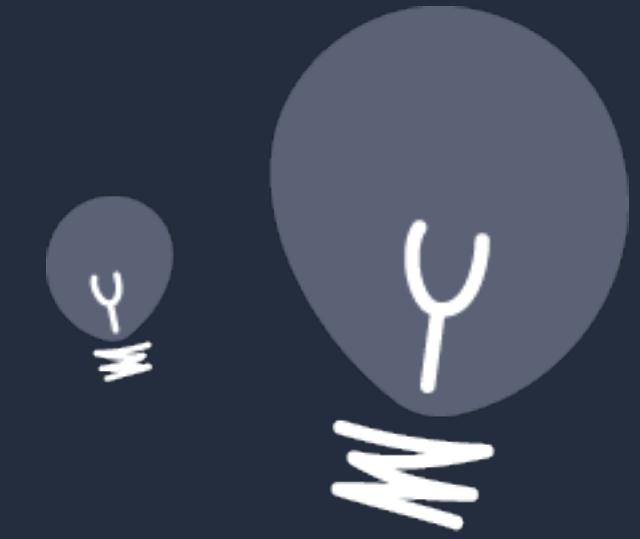


Step 3 - Add fastify-jwt

weBeetle

```
fastify.post('/signup', { ... }, async function (request, reply) {
  ....
  ....
  ....
})

fastify.post('/signin', { ... }, async function (request, reply) {
  ....
  ....
  ....
})
```

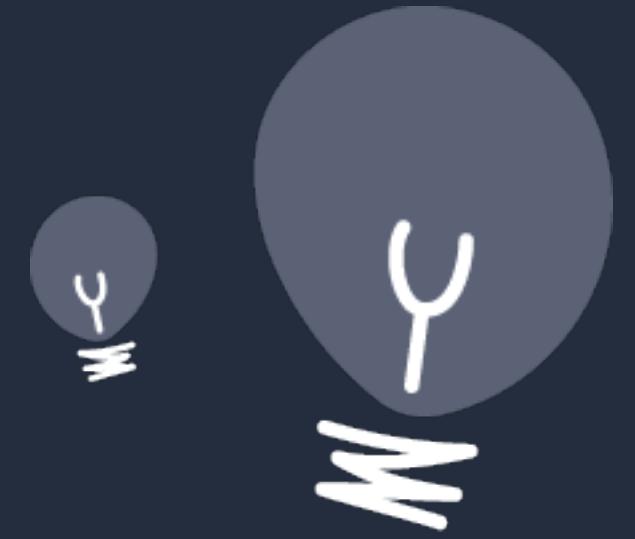


Step 3 - Add fastify-jwt

weBeetle

```
module.exports = fp(async (fastify, opts) => {
  fastify.addHook('preHandler', async (request, reply) => {
    const regex = new RegExp('/api/*')
    const url = request.raw.originalUrl

    if (regex.test(url))
      return request.jwtVerify()
  })
})
```



Step 4 - make tickets specific for users

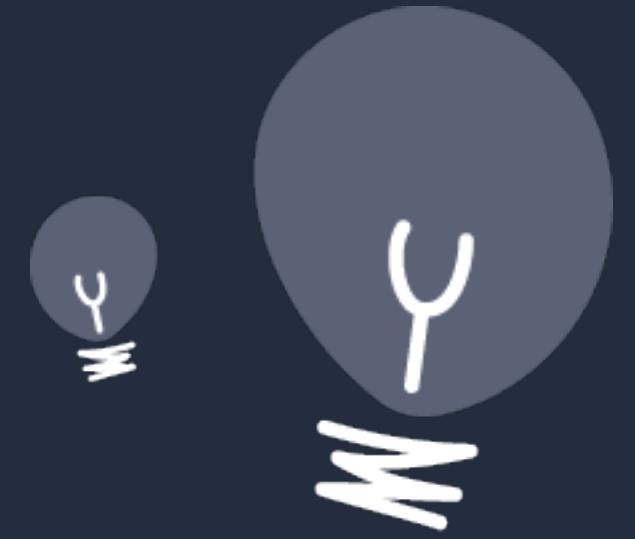
weBeetle

```
fastify.post('/ticket', {
  schema: { ... }
}, async function (request, reply) {

  const ticket = request.body
  Object.assign(ticket, {
    username: request.user.username
  })

  const data = await ticketsCollection.insertOne(ticket)
  const _id = data.ops[0]._id

  return Object.assign({
    _id
  }, ticket)
})
```



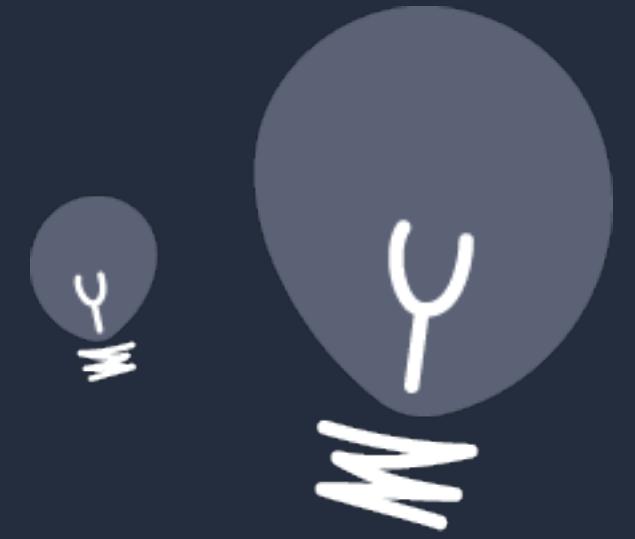
Step 4 - make tickets specific for users

weBeetle

```
fastify.get('/tickets', {
  schema: { ... }
}, async function (request, reply) {

  const tickets = await ticketsCollection.find({
    username: request.user.username
  }).sort({
    _id: -1 // new tickets first
  }).toArray()

  return tickets
})
```

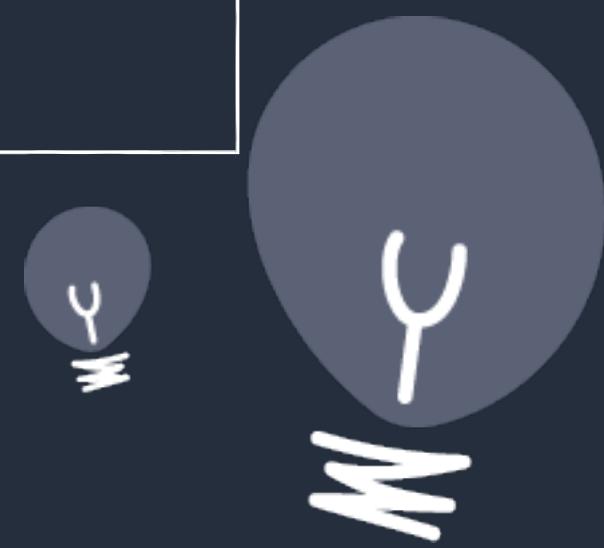
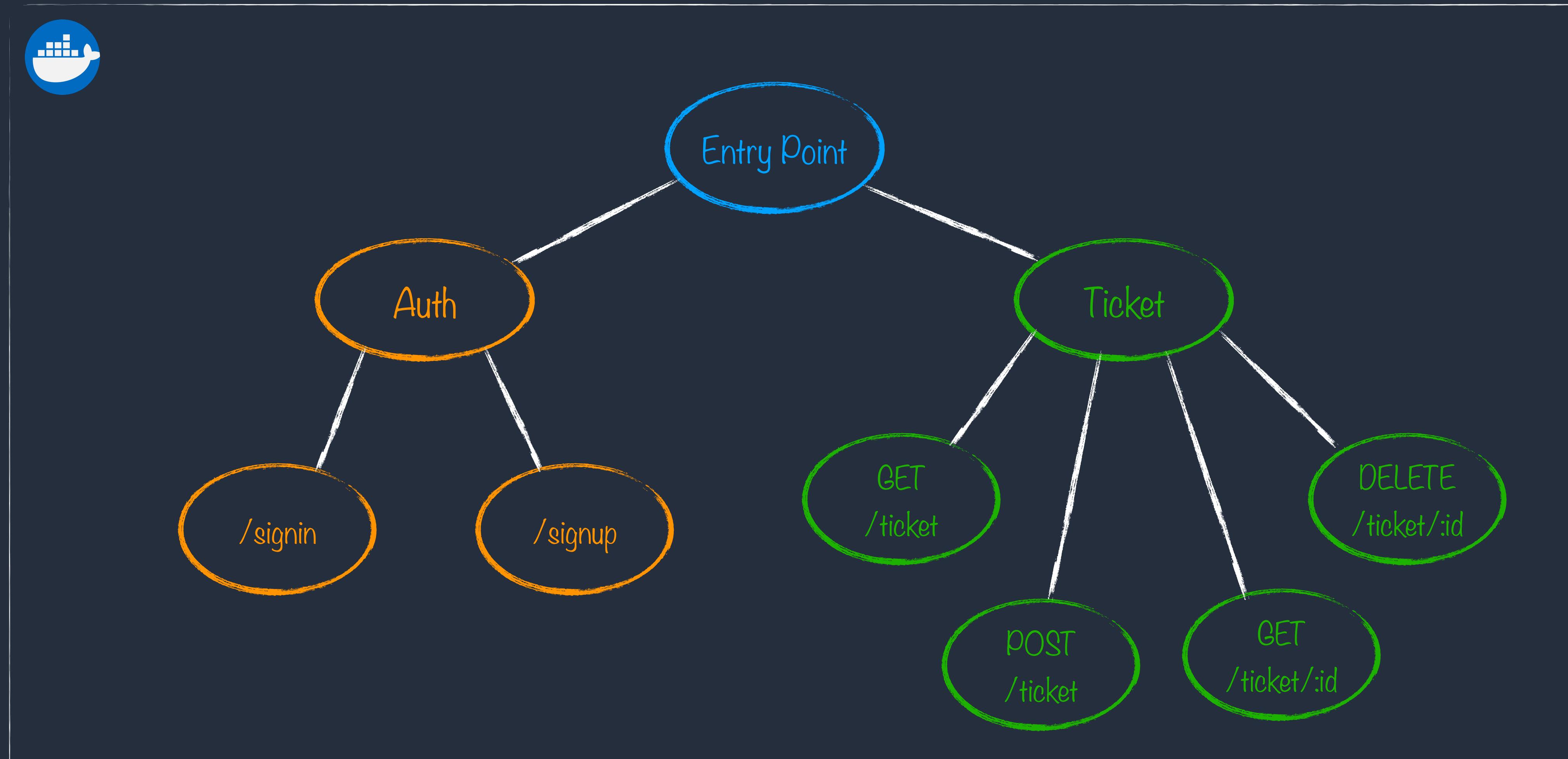


// demo #6



Our application

weBeetle

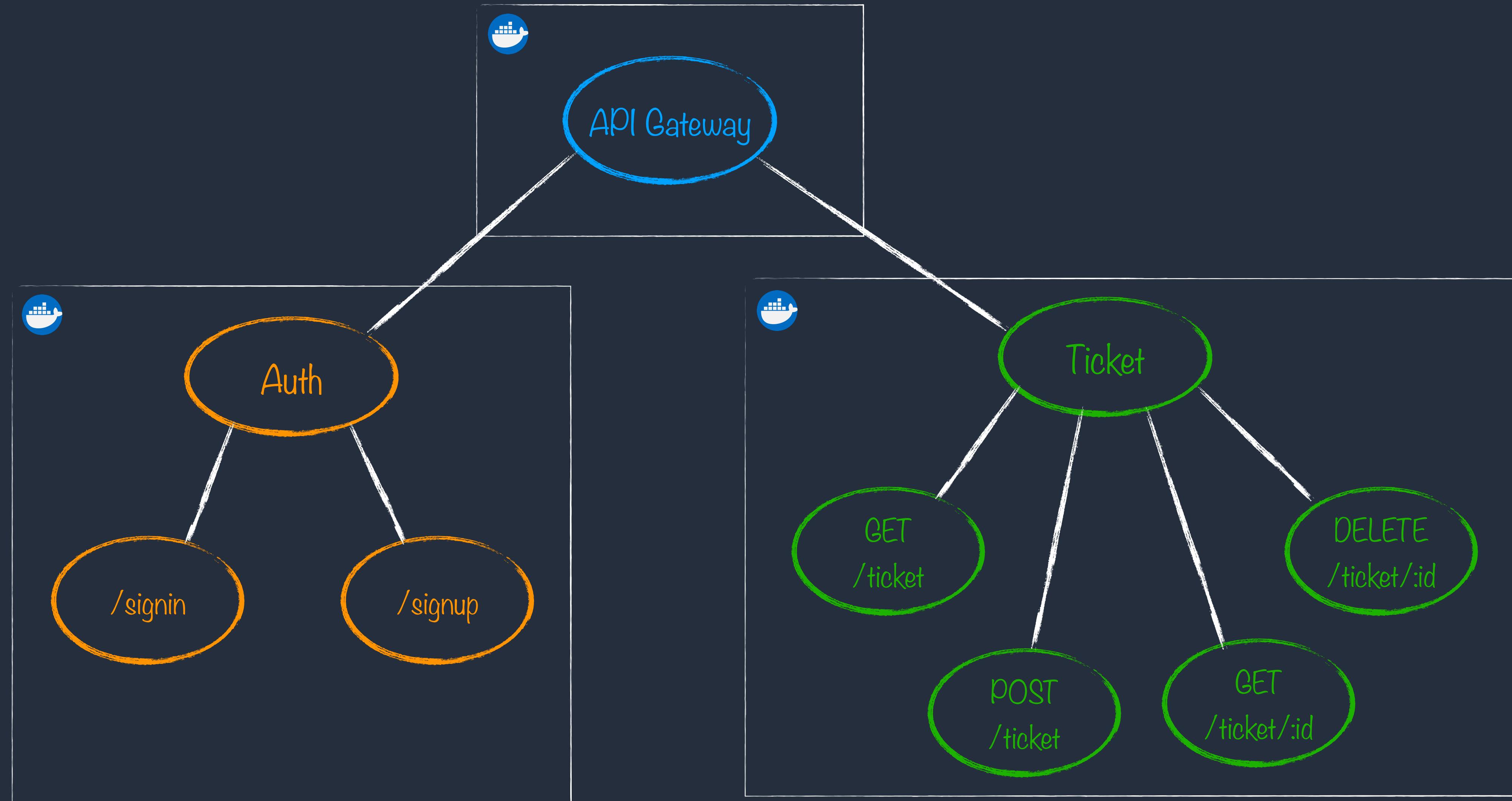


Towards microservices



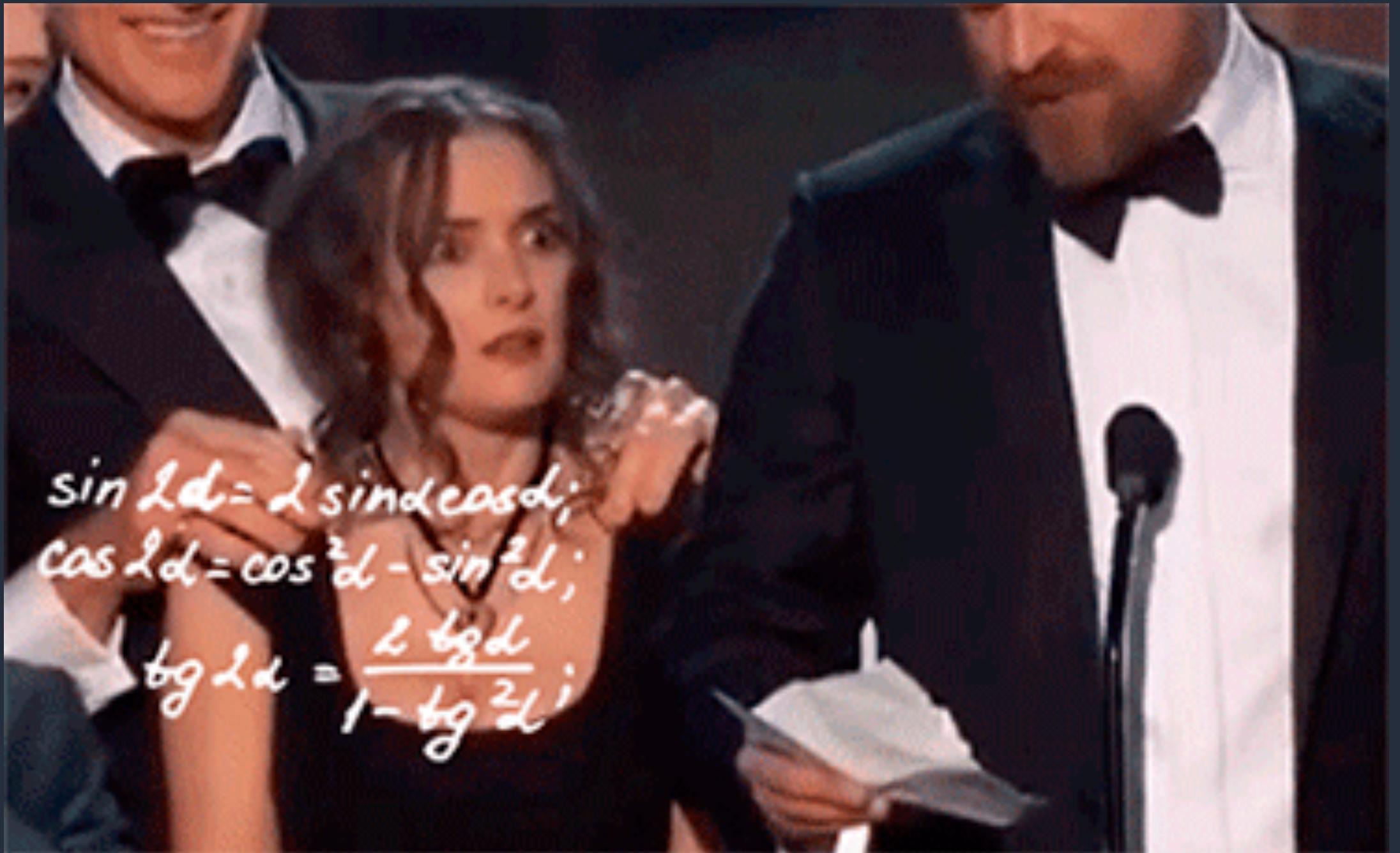
How we change our application

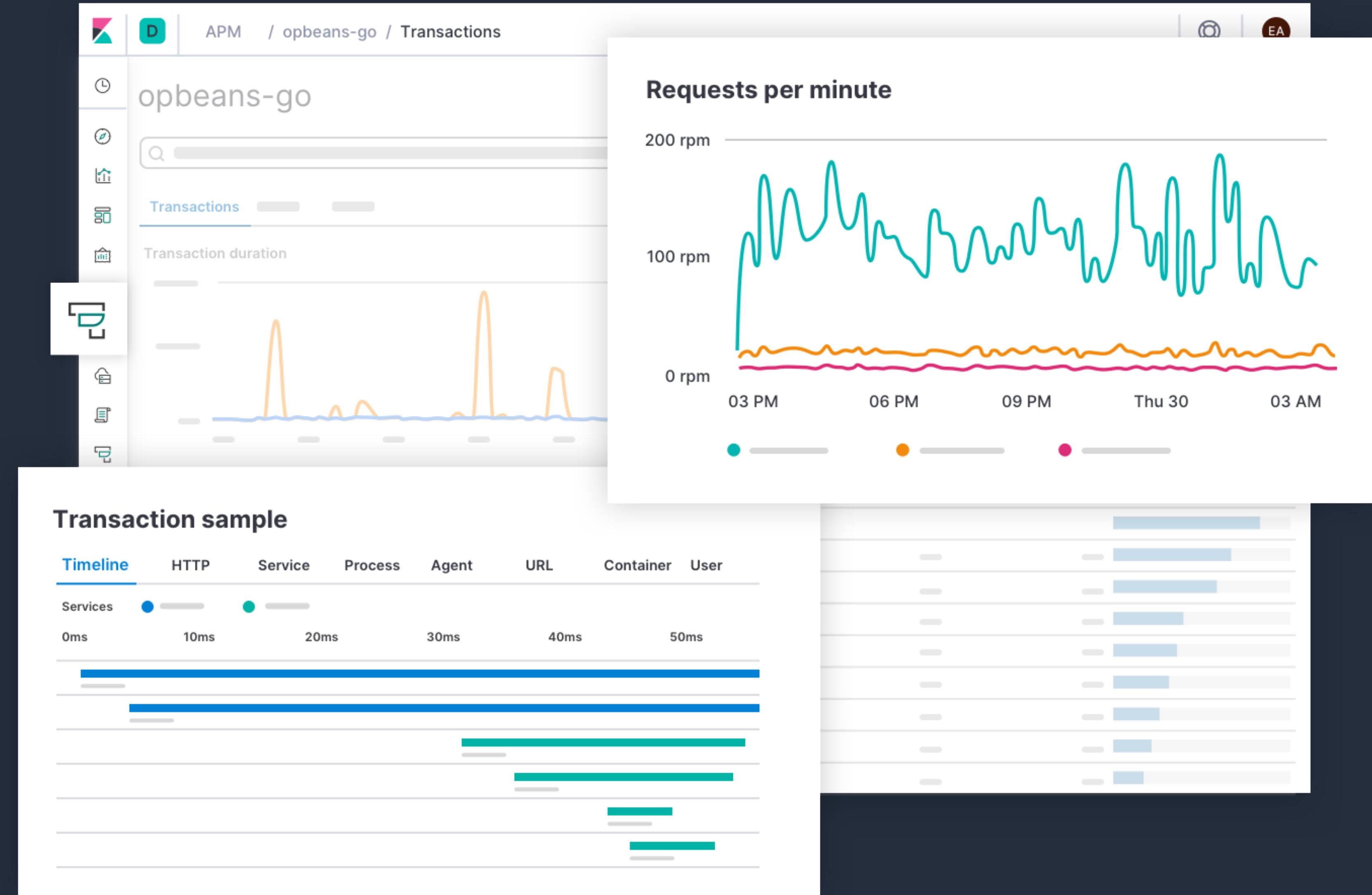
weBeetle



How we monitor our application?







// demo #7



L

[.]



So long and thanks

but I am the fish

