



GOTO Copenhagen 2018
Conference Nov. 19 - 21

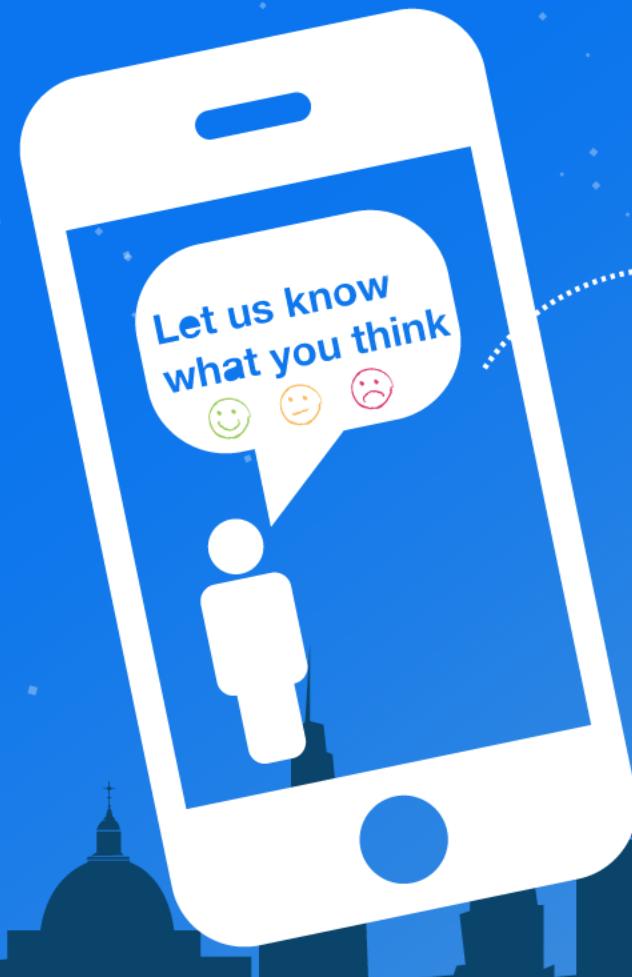
Bringing Magic To Microservice Architecture Development

Ellen Körbes

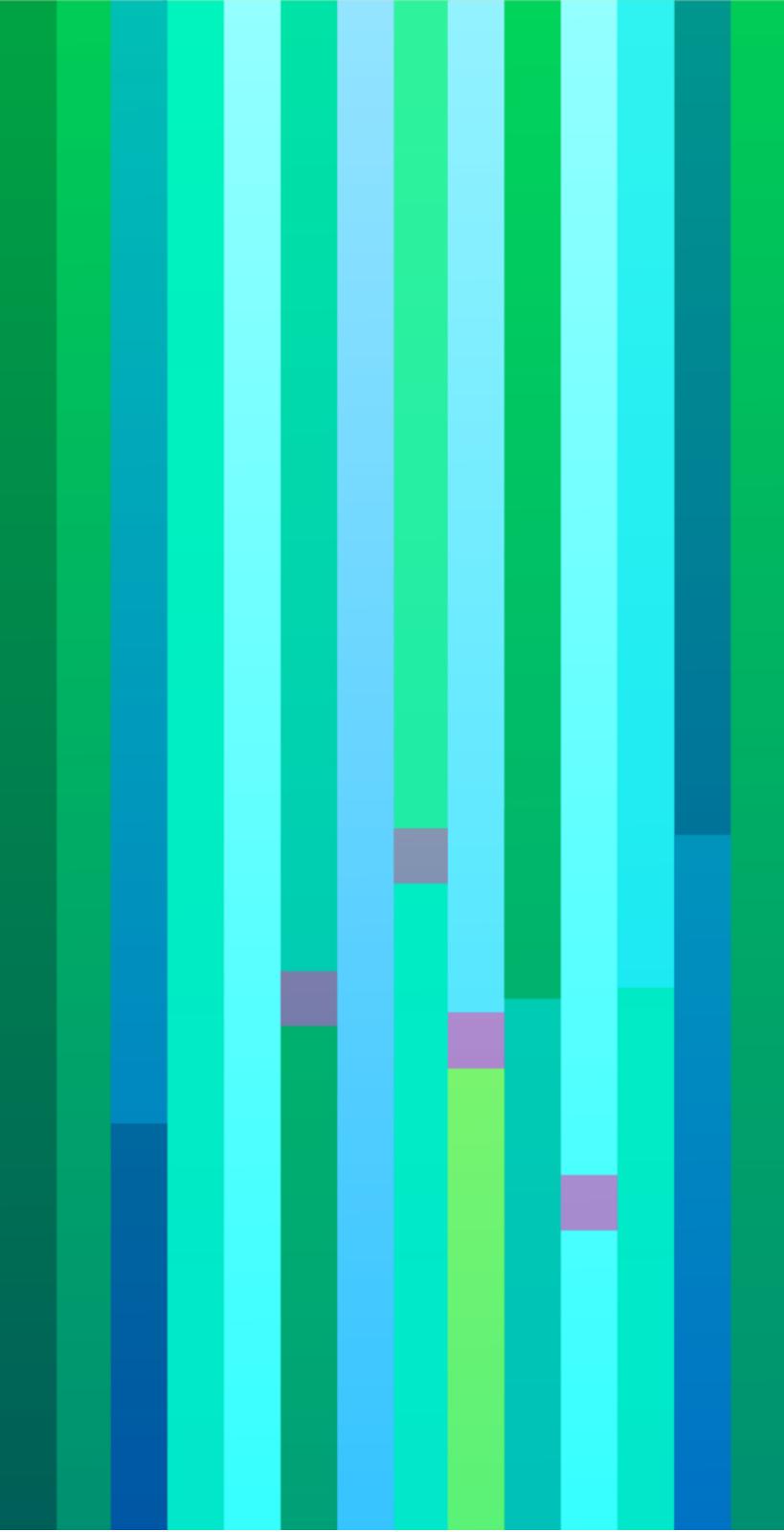
- Microservice architectures
- Distributed systems
- Cloud-native applications
- Service-oriented architectures
- Multi-service systems



GOTO Copenhagen 2018
Conference Nov. 19 - 21



**Click ‘Rate Session’
to rate session
and ask questions.**



ellen körbes

developer relations

ellen@garden.io

@ellenkorbes

they/them

Remember monoliths?

Monoliths:

- Service dependencies aren't a problem.
- Your dependency graph is managed by your compiler.
- Writing application-wide tests is easy.

Monoliths:

- It's easy to learn the system for the first time.
- The required mental context is manageable.
- Good feedback loops.

They were fun.

(Some of them, anyway.)

The magic of programming was there.

Then came multi-service systems.

An amazing new world.

But!

Let's see it.

OFFER OF THE DAY

Buy 1000 socks, get a shoe for free!

Login | Register



HOME

CATALOGUE ▾

0 items in cart



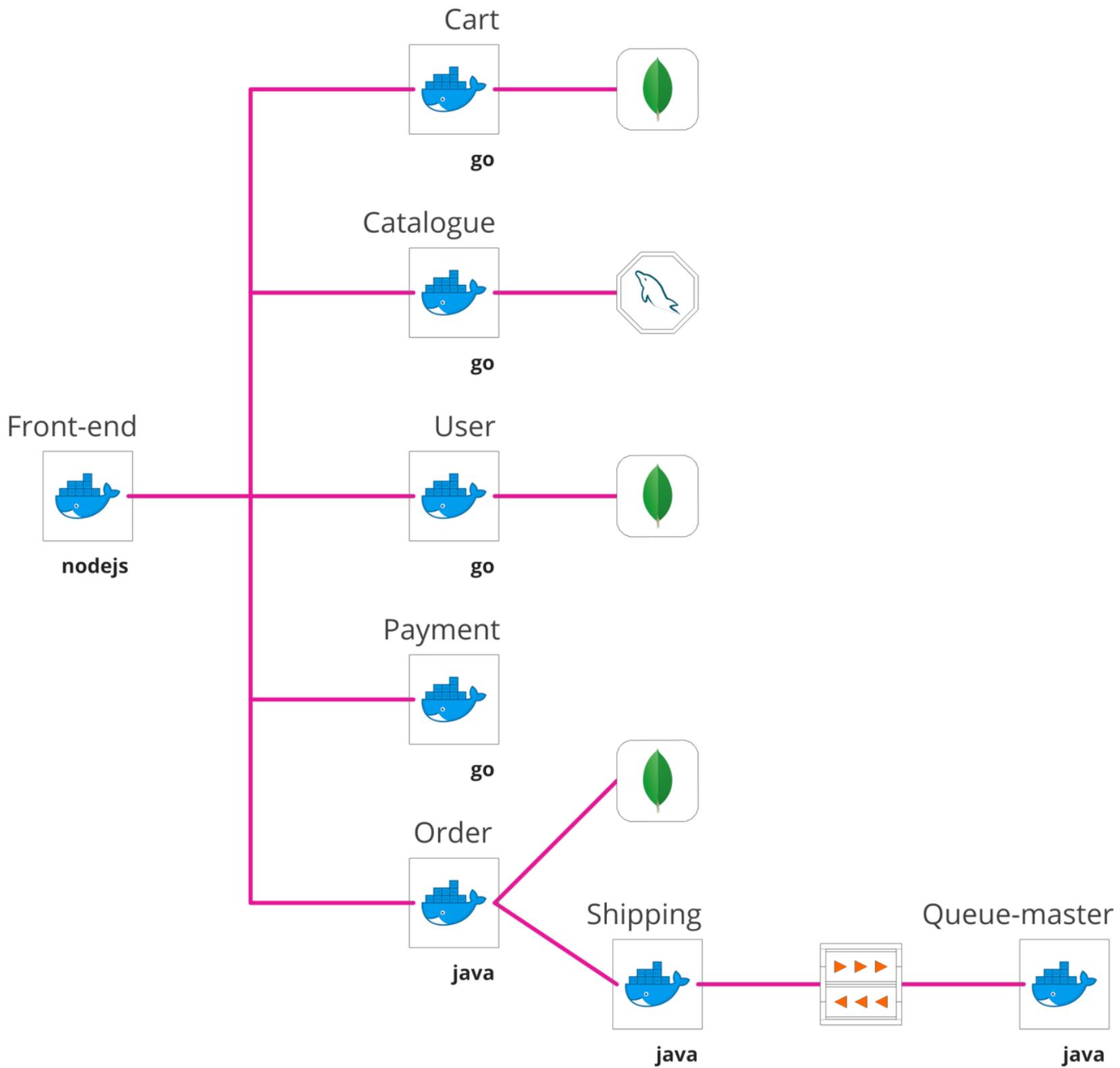
WE LOVE SOCKS!

Fun fact: Socks were invented by woolly

BEST PRICES

We price check our socks with trained monkeys

100% SATISFACTION
GUARANTEED

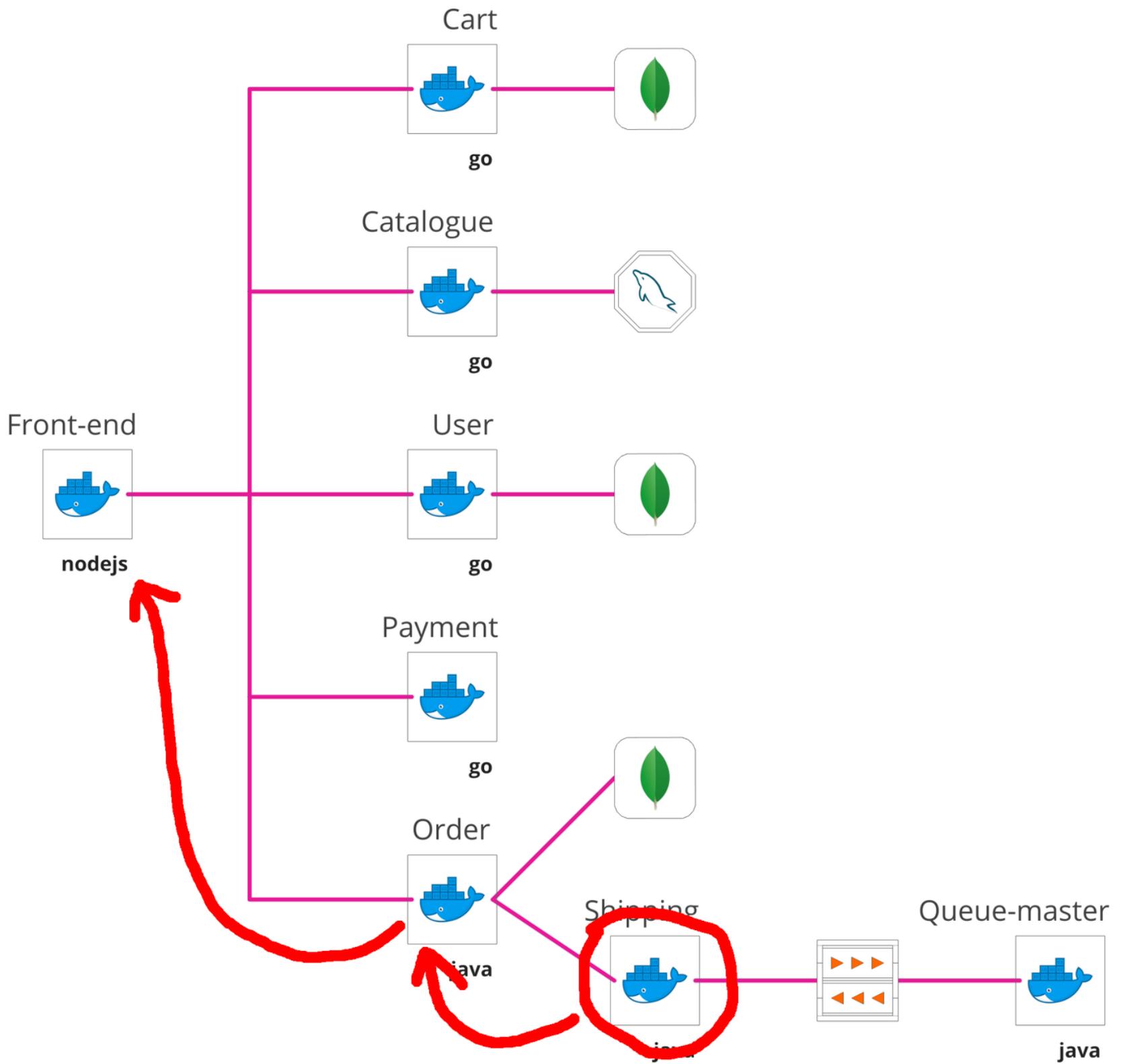


How do you develop on a system like this in practice?

Painfully.

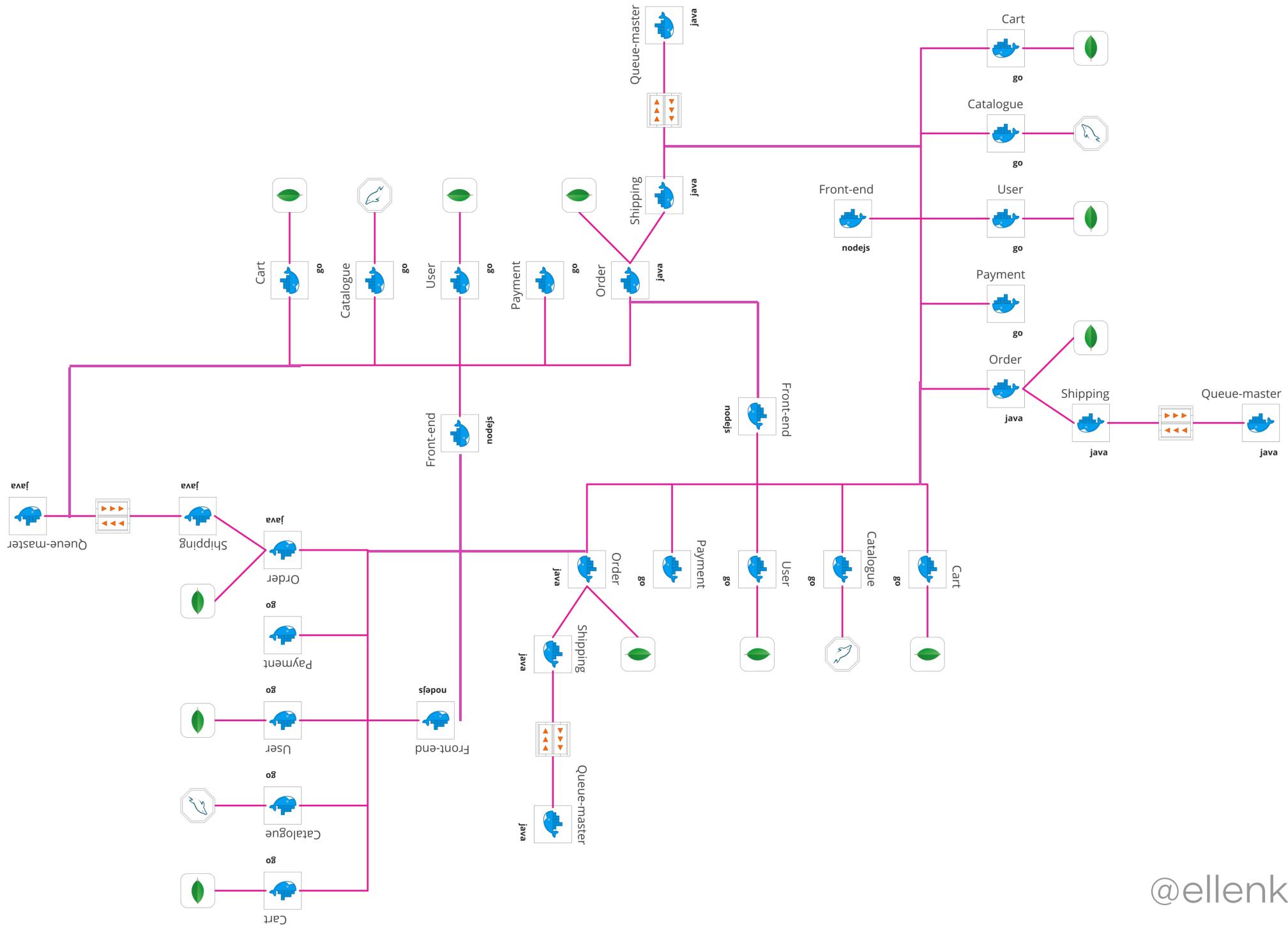
Why?

- Service dependencies are a pain to manage.



Why?

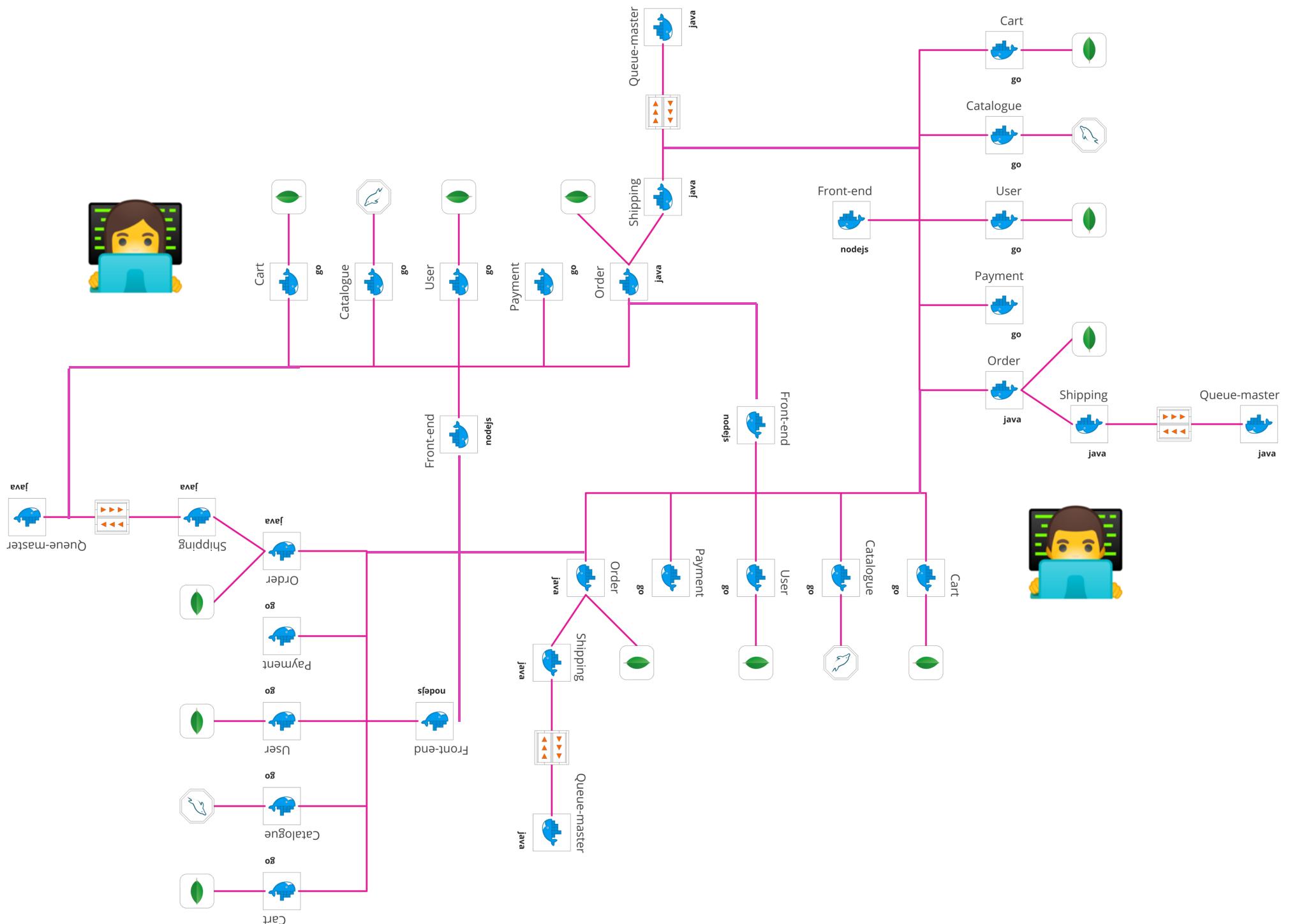
- Service dependencies are a pain to manage.
- Human brains aren't made to be dependency graphs.



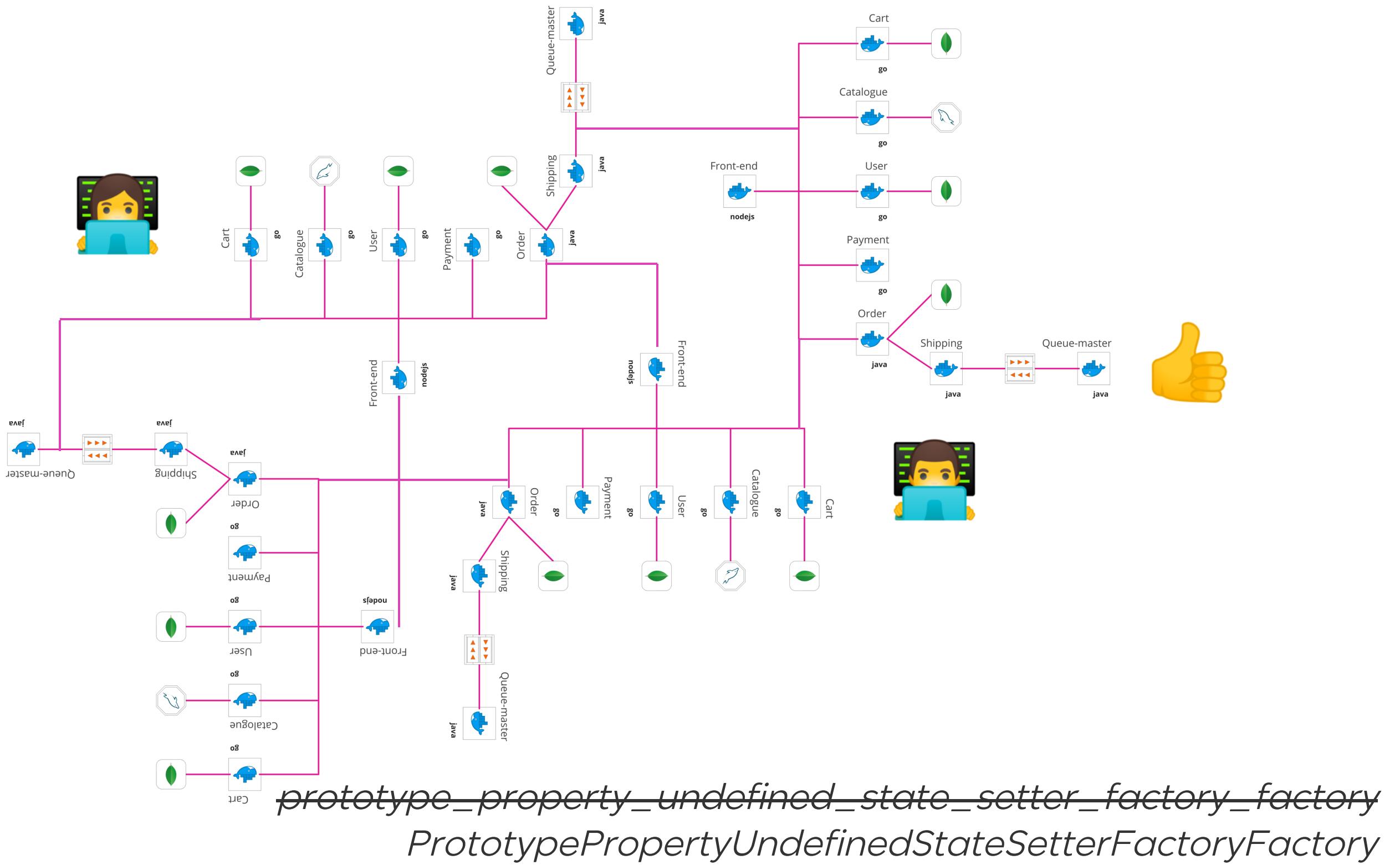
@ellenkorbes

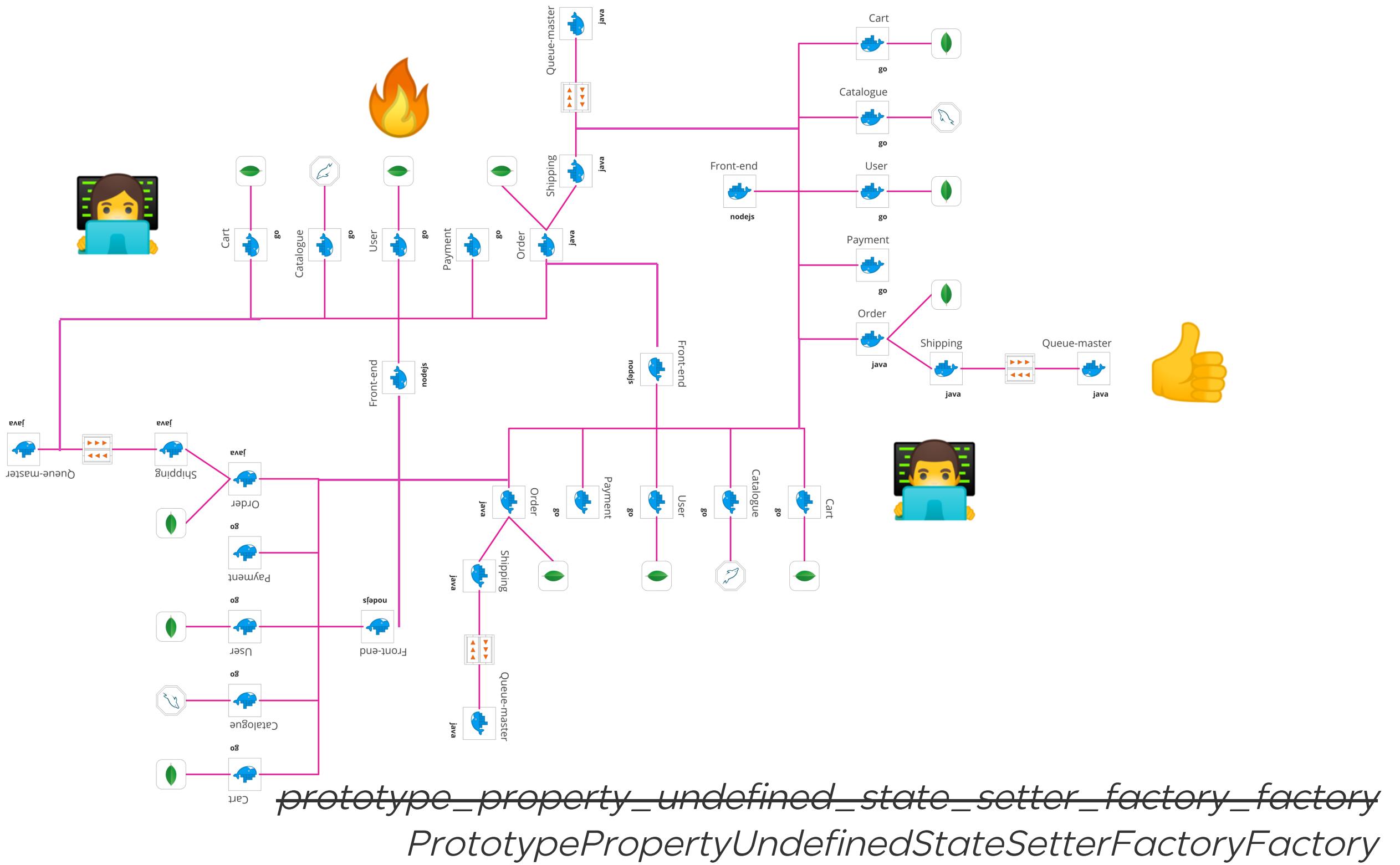
Why?

- Service dependencies are a pain to manage.
- Human brains aren't made to be dependency graphs.
- Testing is not trivial.



@ellenkorbes





Why?

- On-boarding is difficult.

microservices-demo/catalogue - Google Chrome

Secure | https://github.com/microservices-demo/catalogue

Catalogue

A microservices-demo service that provides catalogue/product information. This service is built, tested and released by travis.

Bugs, Feature Requests and Contributing

We'd love to see community contributions. We like to keep it simple and use Github issues to track bugs and feature requests and pull requests to manage contributions.

API Spec

Checkout the API Spec [here](#)

To build this service

Dependencies

```
go get -u github.com/FiloSottile/gvt
gvt restore
```

microservices-demo/shipping: Shipping service for microservices-demo application - Google Chrome

Secure | https://github.com/microservices-demo/shipping

shipping

A microservices-demo service that provides shipping capabilities.

This build is built, tested and released by travis.

Build

Java

```
mvn -DskipTests package
```

Docker

```
GROUP=weaveworksdemos COMMIT=test ./scripts/build.sh
```

Test

microservices-demo/front-end: Front-end application for ALL the microservices - Google Chrome

Secure | https://github.com/microservices-demo/front-end

Build

Dependencies

Name	Version
Docker	>= 1.12
Docker Compose	>= 1.8.0
Make (optional)	>= 4.1

Node

```
npm install
```

Docker

```
make test-image
```

How to Install Apache Maven on Ubuntu 16.04 - Vultr.com - Google Chrome

Secure | https://www.vultr.com/docs/how-to-install-apache-maven-on-ubuntu-16-04

Table of Contents

- Introduction
- Prerequisites
- Step 1: Update your server
- Step 2: Install Java
- Step 3: Install Apache Maven
- Step 4: Setup environment variables
- Step 5: Verify installation

Next, rename the extracted directory.

```
sudo mv apache-maven-3.3.9 maven
```

Step 4: Setup environment variables

Next, you will need to setup the environment variables such as `M2_HOME`, `M2`, `MAVEN_OPTS`, and `PATH`. You can do this by creating a `mavenenv.sh` file inside of the `/etc/profile.d/` directory.

```
sudo nano /etc/profile.d/mavenenv.sh
```

Add the following lines:

```
export M2_HOME=/opt/maven
export PATH=${M2_HOME}/bin:${PATH}
```

Save and close the file, update its permissions, then load the environment variables with the following command:

```
sudo chmod +x /etc/profile.d/mavenenv.sh
sudo source /etc/profile.d/mavenenv.sh
```

Step 5: Verify installation

```
→ ~ cd ~/go/src/github.com/microservices-demo/catalogue
→ catalogue git:(master) ✘ go test
# github.com/microservices-demo/catalogue
./service_test.go:132: T.Errorf call needs 2 args but has 3 args
FAIL    github.com/microservices-demo/catalogue [build failed]
→ catalogue git:(master) ✘ go build -o catalogue
→ catalogue git:(master) ✘ chmod +x catalogue
→ catalogue git:(master) ✘ ./catalogue
./catalogue: 2: ./catalogue: Syntax error: newline unexpected
→ catalogue git:(master) ✘
```

```
---> f7ae3bc3a33b
Successfully built f7ae3bc3a33b
Successfully tagged front-end:latest
env: can't execute 'bash': No such file or directory
Makefile:55: recipe for target 'e2e' failed
make: *** [e2e] Error 127
→ front-end git:(master) ✘ npm start
```

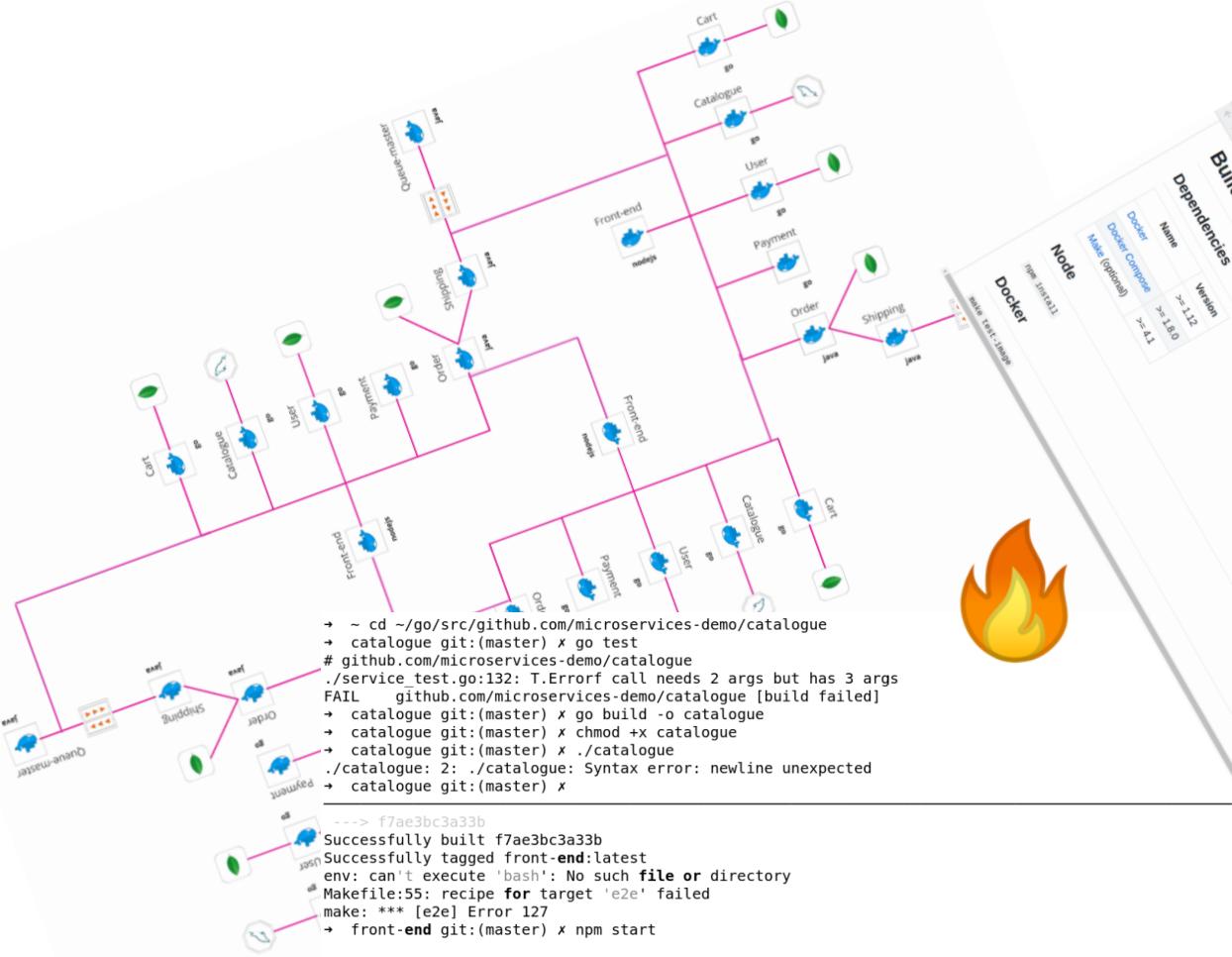
```
> microservices-demo-front-end@0.0.1 start /home/ellen/code/microservices-demo/front-end
> node server.js
```

```
Using local session manager
App now running in development mode on port 8079
```

```
→ shipping git:(master) mvn -DskipTests package [15/446]
[INFO] Scanning for projects...
Downloading: https://repo.spring.io/libs-release/org/springframework/boot/spring-boot-starter-parent/1.4.0.RELEASE/spring-boot-starter-parent-1.4.0.RELEASE.pom
Downloading: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/1.4.0.RELEASE/spring-boot-starter-parent-1.4.0.RELEASE.pom
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[FATAL] Non-resolvable parent POM for works.weave.microservices-demo:shipping:[unknown-version]: Could not transfer artifact org.springframework.boot:spring-boot-starter-parent:pom:1.4.0.RELEASE from/to spring-releases (https://repo.spring.io/libs-release): java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty and 'parent.relativePath' points at wrong local POM @ line 14, column 13
@
[ERROR] The build could not read 1 project -> [Help 1]
[0] 0:zsh* "ellen@t480: ~/go/src/" 15:05 25-Okt-18
```

Why?

- Onboarding is difficult.
- The amount of mental overhead is exhausting.



A stylized flame emoji, consisting of three overlapping orange and yellow shapes.

```
→ ~ cd ~/go/src/github.com/microservices-demo/catalogue
→ catalogue git:(master) ✘ go test
# github.com/microservices-demo/catalogue
./service_test.go:132: T.Errorf call needs 2 args but has 3 arguments
FAIL    github.com/microservices-demo/catalogue [build failed]
→ catalogue git:(master) ✘ go build -o catalogue
→ catalogue git:(master) ✘ chmod +x catalogue
→ catalogue git:(master) ✘ ./catalogue
./catalogue: 2: ./catalogue: Syntax error: newline unexpected
→ catalogue git:(master) ✘
```

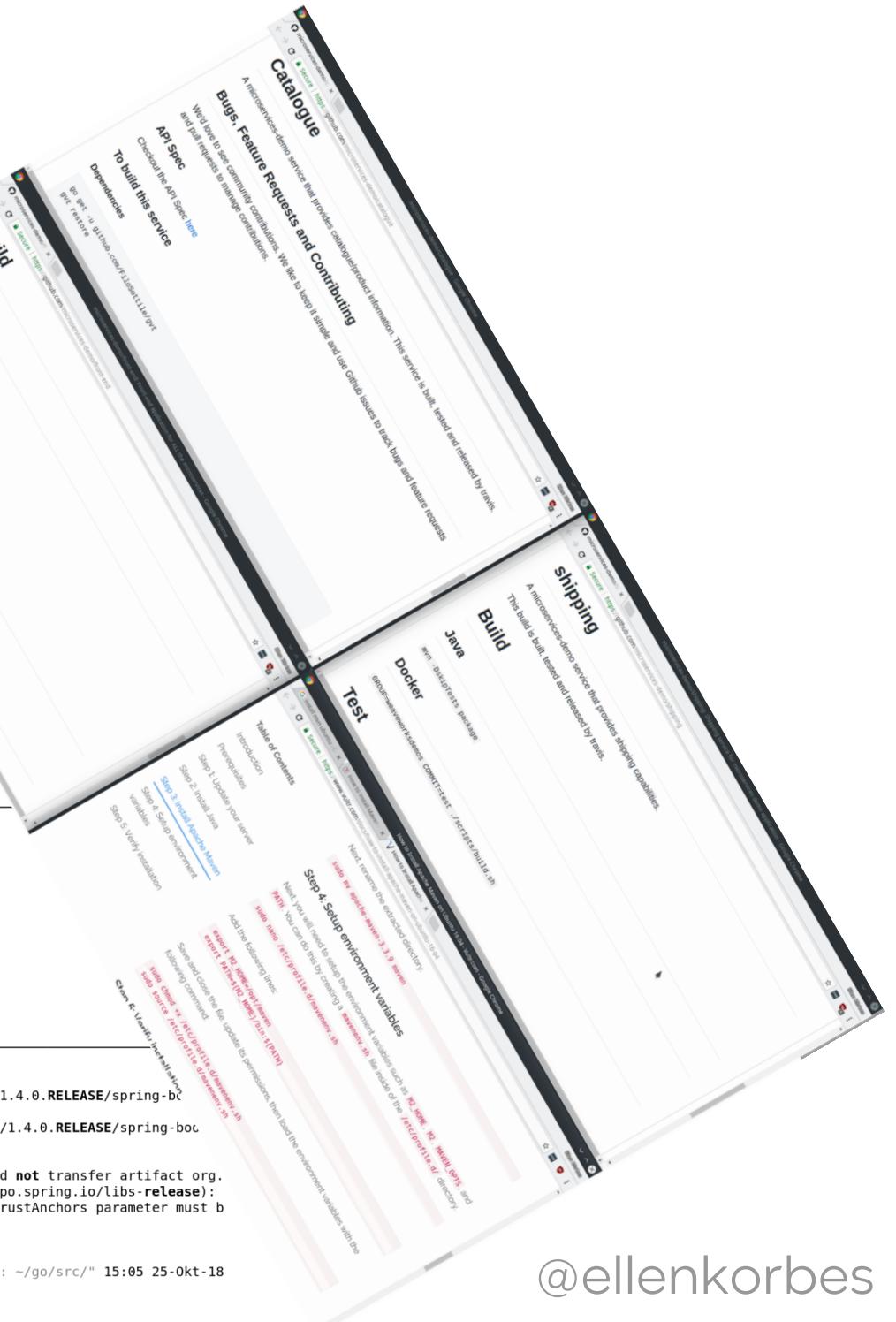
```
--> f7ae3bc3a33b
Successfully built f7ae3bc3a33b
Successfully tagged front-end:latest
env: can't execute 'bash': No such file or directory
Makefile:55: recipe for target 'e2e' failed
make: *** [e2e] Error 127
→ front-end git:(master) ✘ npm start
```

```
> microservices-demo-front-end@0.0.1 start /home  
> node server.js  
  
Using local session manager  
App now running in development mode on port 8079
```

```
→ shipping git:(master) mvn -DskipTests package  
[INFO] Scanning for projects
```

```
→ shipping git:(master) mvn -DskipTests package
[INFO] Scanning for projects...
Downloading: https://repo.spring.io/libs-release/org/springframework/boot/spring-boot-starter-parent/1.4.0.RELEASE/spring-boot-starter-parent-1.4.0.RELEASE.pom
Downloading: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-boot-starter-parent/1.4.0.RELEASE/spring-boot-starter-parent-1.4.0.RELEASE.pom
[ERROR] [ERROR] Some problems were encountered while processing the POMs:
[FATAL] Non-resolvable parent POM for works.weave.microservices-demo:shapecraft:[unknown-version]: Could not transfer artifact org.springframework.boot:spring-boot-starter-parent:pom:1.4.0.RELEASE from/to spring-releases (https://repo.spring.io/libs-release): java.lang.RuntimeException: Unexpected error: java.security.InvalidAlgorithmParameterException: the trustAnchors parameter must be non-empty and 'parent.relativePath' points at wrong local POM @ line 14, column 13
@
```

"ellen@t480: ~/go/src/" 15:05 25-0kt-18

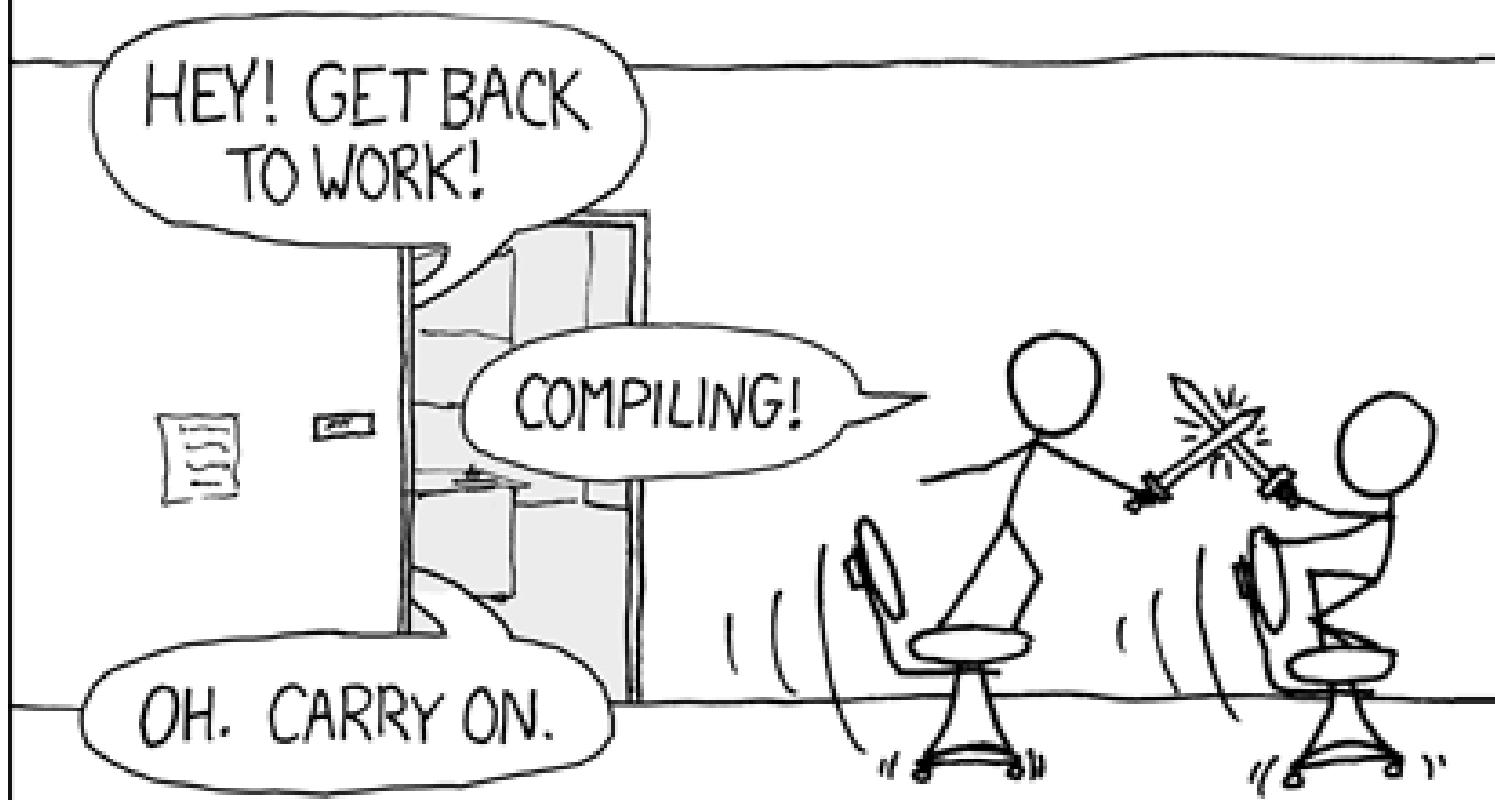


@ellenkorbes

Why?

- Onboarding is difficult.
- The amount of mental overhead is exhausting.
- And the feedback loops?

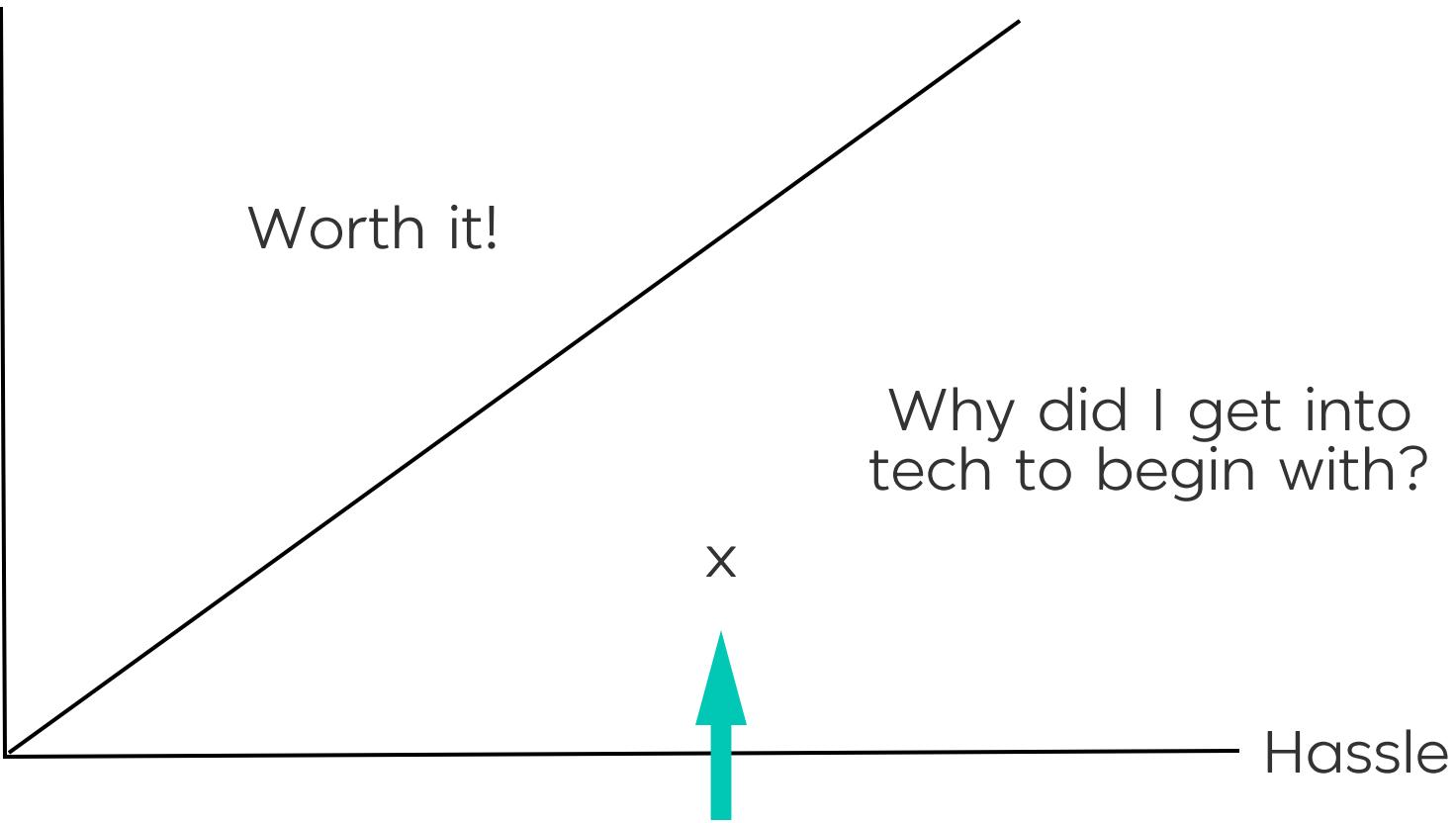
THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF: "MY CODE'S COMPILING."



i.e. not fun.

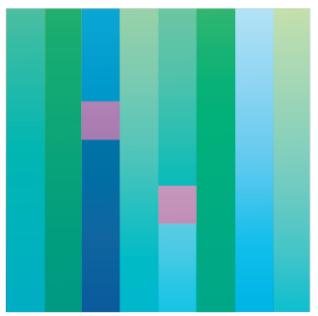
The amount of hassle
takes all the magic away.

Fun



Multi-container systems development

Let's solve this problem.



garden

“ Garden is a development engine for Kubernetes that makes developing, building and testing multi-service systems fast, easy, and fun. ”

"A what, now?"

That whole pile of hassle
we just talked about?

Gone.

1. eysi@Eysis-MacBook-Pro: ~/code/garden-io/sock-shop-mono (zsh)

→ **sock-shop-mono** (master) garden d█

Link.

```

✓ shipping → Building shipping:54a7ef3e59-1542621319... → Done (took 2.1 sec)
✓ shipping → Deploying → Ready
✓ shipping → Running shipping tests → Success
✓ orders → Deploying → Ready
✓ front-end → Deploying → Ready
✓ orders → orders tests → Already passed
✓ front-end → front-end tests → Already passed
✓ shipping → Building shipping:54a7ef3e59-1542621348... → Done (took 1.5 sec)
✓ shipping → Deploying → Ready
✓ shipping → Running shipping tests → Success
✓ orders → Deploying → Ready
✓ front-end → Deploying → Ready

```

```

25
26     @RequestMapping(value = "/shipping", method = RequestMethod.GET)
27     public String getShipping() {
28         return "GET ALL Shipping Resource.";
29     }
30
31     // █
32     @RequestMapping(value = "/shipping/{id}", method = RequestMethod.GET)
33     public String getShippingById(@PathVariable String id) {
34         return "GET Shipping Resource with id: " + id;
35     }
36
37     @ResponseStatus(HttpStatus.CREATED)
38     @RequestMapping(value = "/shipping", method = RequestMethod.POST)

```

Link.

Hot reload demo

We're using Garden's hot reload functionality in combination with [Gatsby](#) (running inside a container) to live-update this page.

```
import Layout from '../components/layout'

const IndexPage = () => (
  <Layout>
    <h1>Hot reload demo</h1>
    <p>We're using Garden's hot reload functionality in combination with <a href="https://github.com/gatsbyjs/gatsby">Gatsby</a> (running inside a container) to live-update this page.</p>
  </Layout>
)

export default IndexPage
```

minimal/src/pages/index.js 7,7 All
"minimal/src/pages/index.js" 12L, 364C

Link.

```
✓ shipping → Building shipping:54a7ef3e59-1542621348... → Done (took 1.5 sec)
✓ shipping → Deploying → Ready
✓ shipping → Running shipping tests → Success
✓ orders → Deploying → Ready
✓ front-end → Deploying → Ready
✓ front-end → front-end tests → Already passed
✓ orders → orders tests → Already passed
✓ shipping → Building shipping:54a7ef3e59-1542621445... → Done (took 1.8 sec)
✓ shipping → Deploying → Ready
✓ shipping → Running shipping tests → Success
✓ orders → Deploying → Ready
✓ front-end → Deploying → Ready
```

□

```
15 });
16
17 module.exports = {
18   catalogueUrl: util.format("http://catalogue%s", domain),
19   tagsUrl: util.format("http://catalogue%s/tags", domain),
20   cartsUrl: util.format("http://carts%s/carts", domain),
21   ordersUrl: util.format("http://orders%s", domain),
22   customersUrl: util.format("http://user%s/customers", domain),
23   addressUrl: util.format("http://user%s/addresses", domain),
24   cardsUrl: util.format("http://user%s/cards", domain),
25   loginUrl: util.format("http://user%s/login", domain),
26   registerUrl: util.format("http://user%s/register", domain),
27 };
28 }());
```

Link.

How does it work?

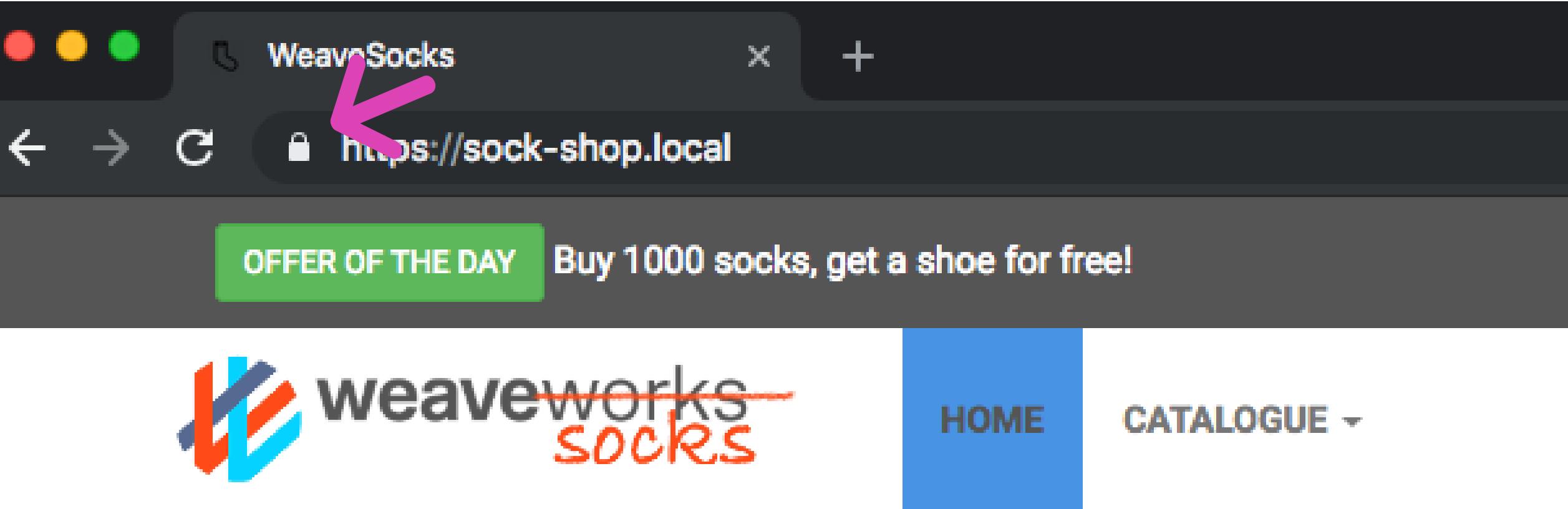
```
→ sock-shop tree .
```

```
•
└── garden.yml
└── services
    ├── carts
    │   └── garden.yml
    ├── carts-db
    │   └── garden.yml
    ├── catalogue
    │   └── garden.yml
    ├── catalogue-db
    │   └── garden.yml
    ...
    └── user-db
        └── garden.yml
```

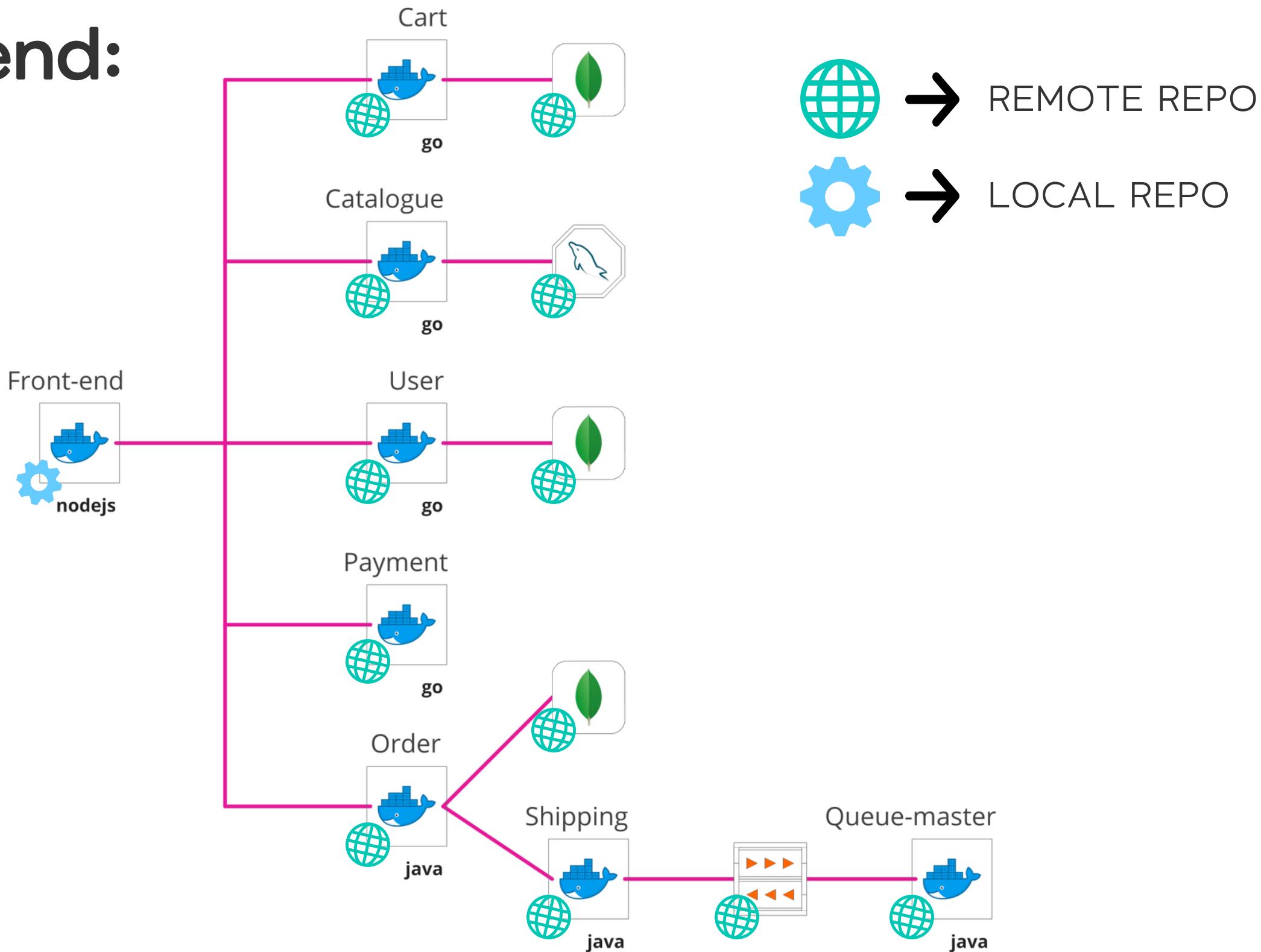
```
project:  
  name: sock-shop  
environments:  
  - name: local  
    providers:  
      - name: local-kubernetes
```

```
module:  
  description: Catalogue service  
  type: container  
  name: catalogue  
  dockerfile: /docker/catalogue/Dockerfile  
  services:  
    - name: catalogue  
      ports:  
        - name: http  
          containerPort: 80  
  dependencies:  
    - catalogue-db
```

```
module:  
  description: Front end service  
  type: container  
  name: front-end  
  services:  
    - name: front-end  
      command: [/usr/local/bin/npm, start]  
      ports:  
        - name: http  
          containerPort: 8080  
      ingresses:  
        - path: /  
          port: http  
  dependencies:  
    - orders  
    - payment  
    - user  
    - catalogue  
    - carts  
  tests:  
    - name: unit  
      command: [npm, test]  
    - name: e2e  
      command: [npm, test-e2e]  
  dependencies:  
    - orders  
    - payment  
    - user
```



Front-end:



Front-end:

```
module:  
  description: Orders service  
  type: container  
  name: orders  
  repositoryUrl: https://github.com/microservices-demo/orders.git#0.4.7  
  dockerfile: /docker/orders/Dockerfile  
serv module:  
  - description: Front end service  
    type: container  
    name: front-end  
    services:  
      - name: front-end  
        command: [/usr/local/bin/npm, start]  
        ports:  
          - name: http  
          ...  
  
  dockerfile: /docker/carts/Dockerfile  
  services:  
    - name: carts  
    ...
```



Garden:

- Dependency management is built in and virtually free for all build, test, and deploy tasks.
- The workflow is consistent, reliable, and low maintenance.
- Dramatically shorter feedback loops.

Our vision.

A layer between
orchestration and
development.

Currently, you declare:

- Your infrastructure/orchestration.
- How your stack looks when done.

Now, you can declare:

- Your development workflow.
- How your stack *gets to* done.

The context humans
used to have to carry
becomes codified.

A system that knows
how your stack works.

Remote development environment.

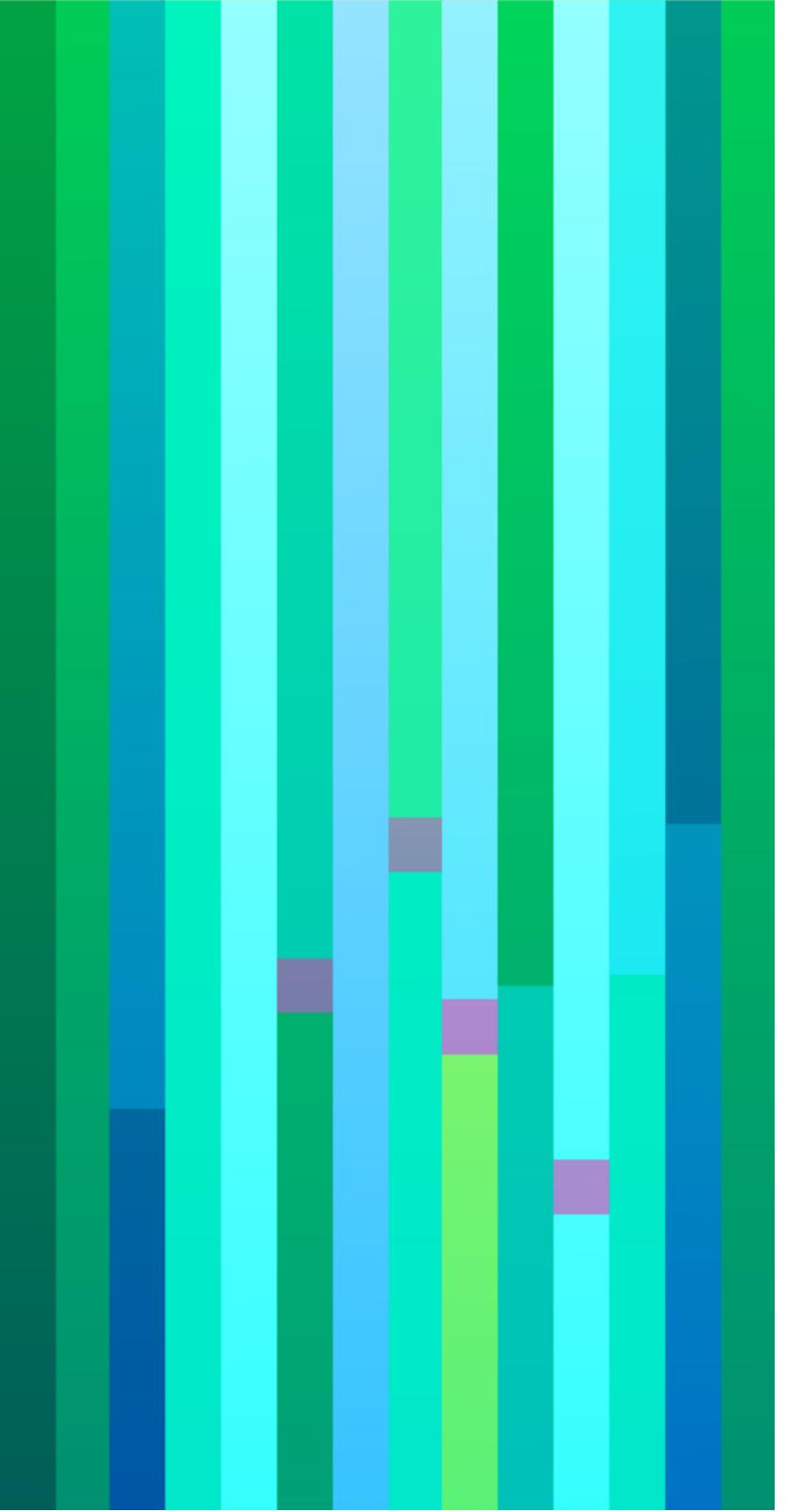
How about, as you code:

- Machine learning code analysis
- Security test plugins
- Performance analysis

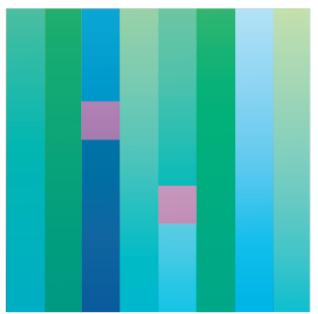
The development environment
of the future is **not** an IDE.

The development environment
of the future is a system that
knows how your stack works.

Open source. Alpha stage.
Eager to hear from you.

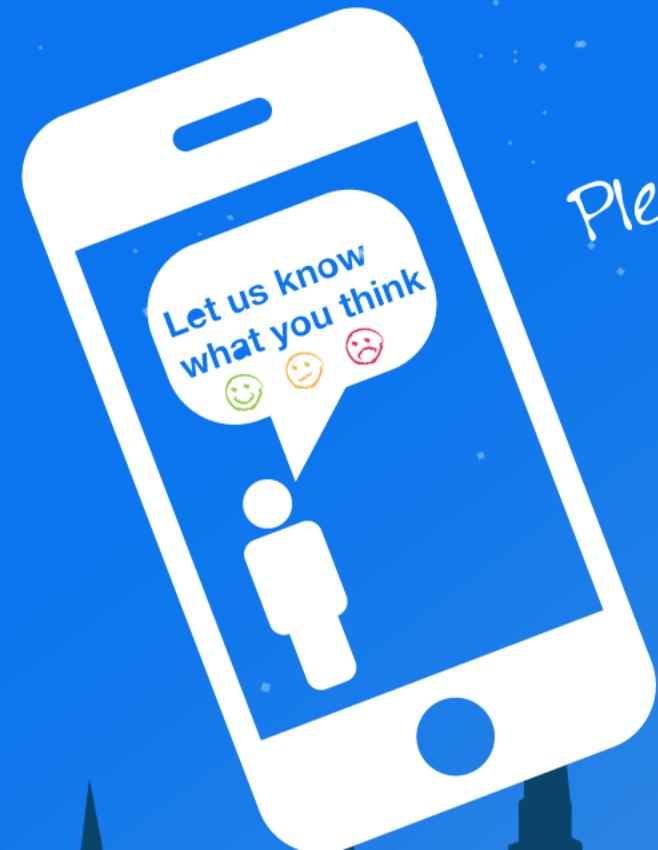


Focus on the
fun part and let
Garden do the rest.



garden

ellen@garden.io | @ellenkorbes



Please

**Remember to
rate this session**

Thank you!