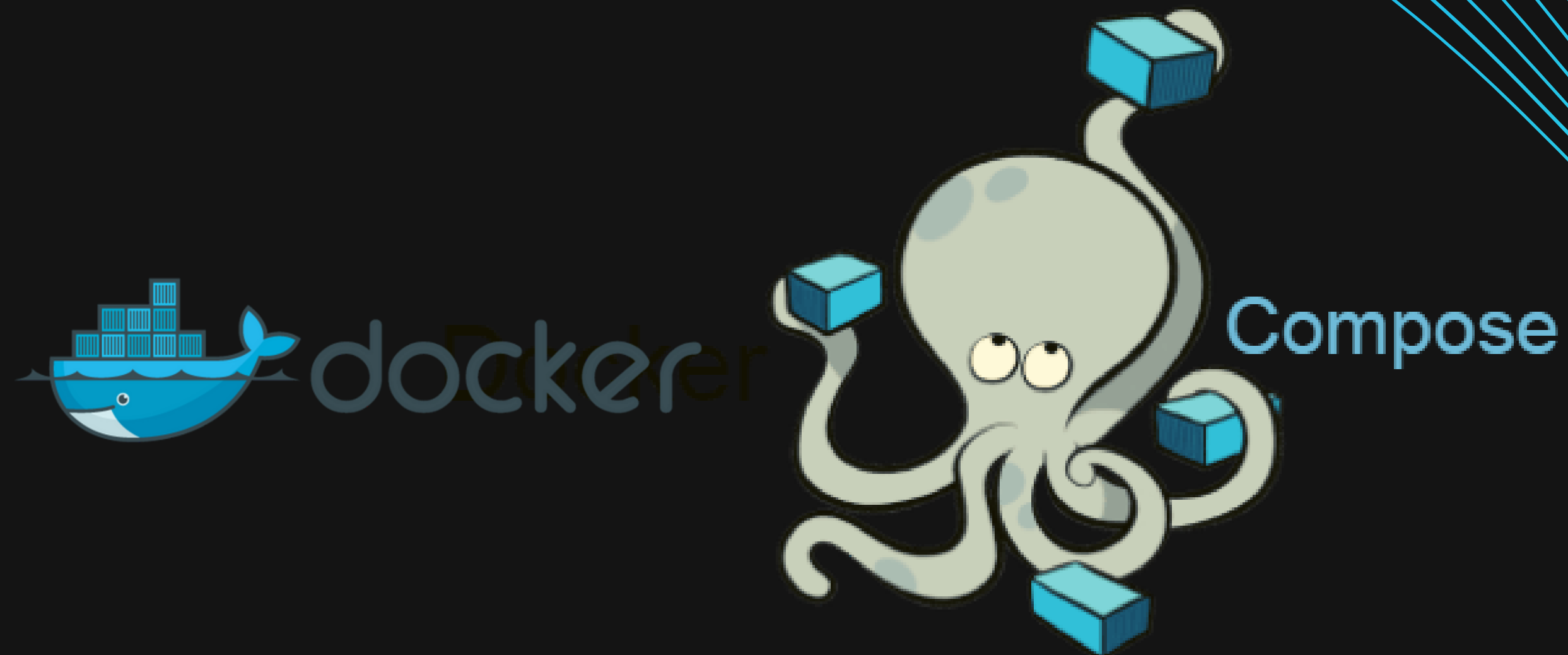


# Linguagem de Programação Visual

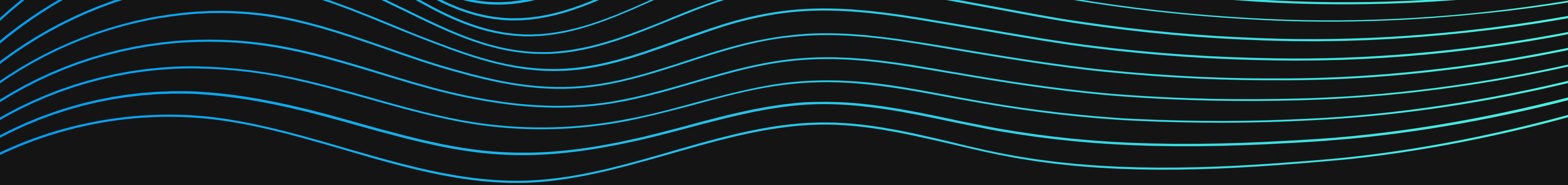
RICARDO HENDGES





Docker Compose e Docker são coisas diferentes!

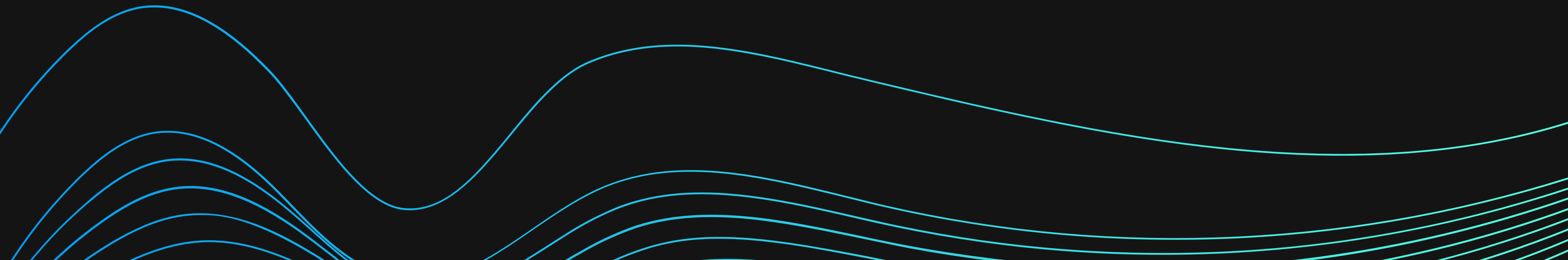
**Docker Compose é uma ferramenta de administração de containers.**



No mundo de aplicações descentralizadas, encontramos os Microservices e é comum que seu sistema esteja dividido em aplicações diferentes, as vezes até mesmo em linguagens de programação diferentes. Por causa dessa separação, você precisaria de mais de uma imagem e mais de um container para que tudo funcione.

Até mesmo se seu sistema for um monolito, a chance de você utilizar um banco de dados entre outras tecnologias fora da sua aplicação é grande!

Isso também fará com que você vá dividir cada vez mais em containers e imagem menores e que sejam responsáveis somente por uma tarefa.





# Opções para aumento de complexidade

Containers independentes!

`docker run ...`, `docker run ...`, `docker run ...`, `docker run ...`



# Opções para aumento de complexidade

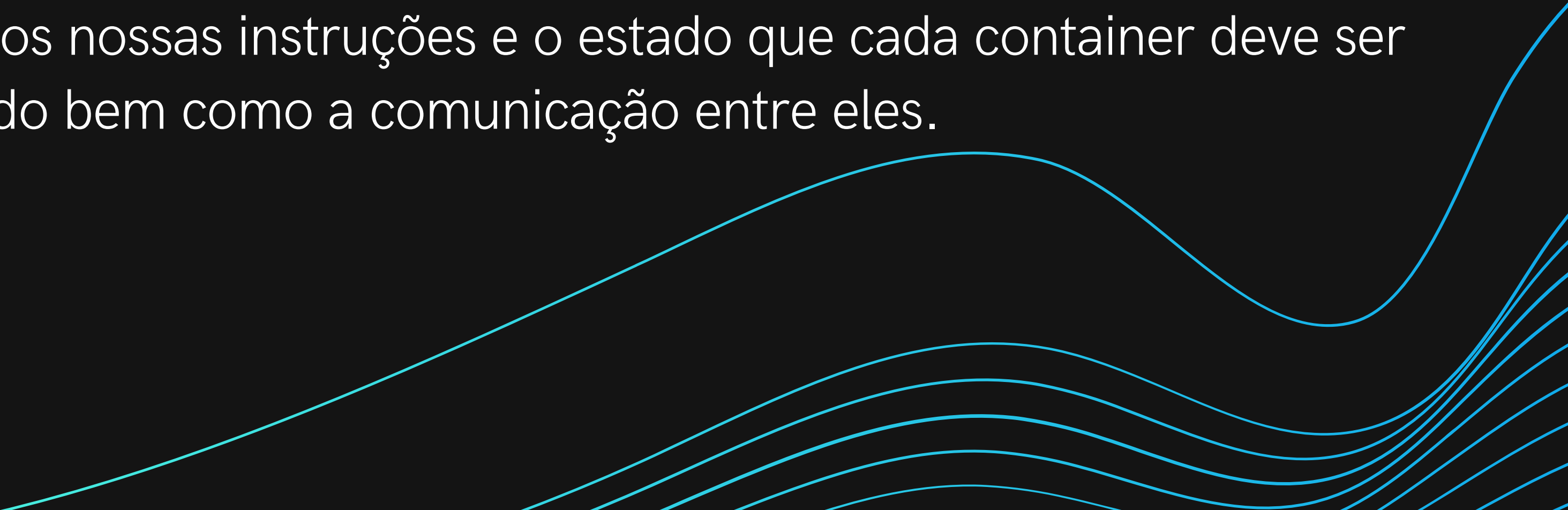
Agregar!

Imagine o tanto de configuração necessária no seu Dockerfile para que você rode sua aplicação, seu banco de dados, seu web server e etc. Você acabaria tendo um Dockerfile mega extenso e difícil de manter. Seria praticamente fazer o setup disso tudo no seu próprio computador.

# Opções para aumento de complexidade

# compose

Gerenciar vários containers dependentes. O arquivo docker-compose.yml é onde declaramos nossas instruções e o estado que cada container deve ser criado e operado bem como a comunicação entre eles.





# O arquivo YAML

## version

A primeira coisa que devemos informar no arquivo é a versão do Docker Compose que estamos utilizando.

```
version: "3"
```



# O arquivo YAML

## services

Quais containers vamos criar?

```
version: "3"
services:
  api-horus:
```

# O arquivo YAML

## image / container\_name / ports

Imagem base/origem para subir container.

Nome do container.

Portas que nosso container irá utilizar.

```
api-horus:  
  image: api-horus  
  container_name: api-horus  
  ports:  
    - "3000:3000"
```

# O arquivo YAML

## depends\_on

Imagens que são dependências desse container.

```
api-horus:  
  image: api-horus  
  container_name: api-horus  
  ports:  
    - "3000:3000"  
  depends_on:  
    - db-horus
```

# O arquivo YAML

## services

Adicionamos o segundo service.  
db-horus.

Imagem origem (nome do build)

Nome do container

Variáveis de ambiente

Portas

```
version: "3"
services:
  api-horus:
    image: api-horus
    container_name: api-horus
    ports:
      - "3000:3000"
    depends_on:
      - db-horus
  db-horus:
    image: db-horus
    container_name: db-horus
    environment:
      ENV POSTGRES_USER: admin
      ENV POSTGRES_PASSWORD: 123456
      ENV POSTGRES_DB: horus
    ports:
      - "5432:5432"
```

# Mudança na API

HOST - não é mais local. Agora nossa API se comunica pelo NOME do container.

JS pg.js M X

src > configs > JS pg.js > [e] pool

You, 3 hours ago | 1 author (You)

```
1  const { Pool } = require('pg')
2
3  const pool = new Pool({
4    user: 'admin',
5    host: 'db-horus',
6    database: 'horus',
7    password: '123456',
8    port: 5432,
9  })
```

PORT - Mesmo rodando o container em uma porta diferente da padrão 5432, você deve apontar para essa porta.

# Comandos compose

`docker-compose` = base para os comandos

`docker-compose down` = derruba imagens

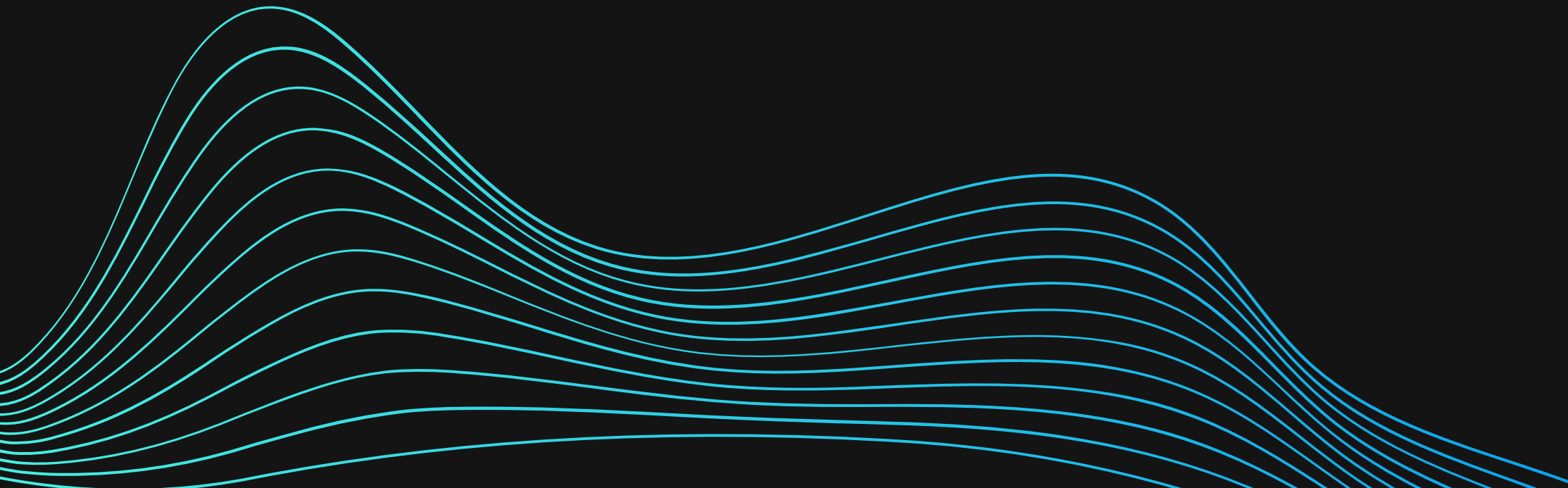
`docker-compose pull` = atualiza imagens

`docker-compose up -d` = sobe imagens

`-f` = especifica arquivo para executar qualquer opção acima.

ex: `docker-compose -f docker-composeAlterado.yaml up -d`

Temos ainda:  
build, logs, start, stop



# Mensões honrosas

## restart

Ação que deve ser tomada caso o processo responsável pelo contêiner morra.

- no: compose não tentará executá-lo novamente;
  - always: independentemente da causa, o Compose execute-o novamente;
  - on-failure: executa o contêiner somente caso resulte de uma falha;
  - unless-stopped: sempre irá garantir que o estado do contêiner seja em execução, ao menos que este seja colocado em estado stopped manualmente ou por algum outro motivo;
- 