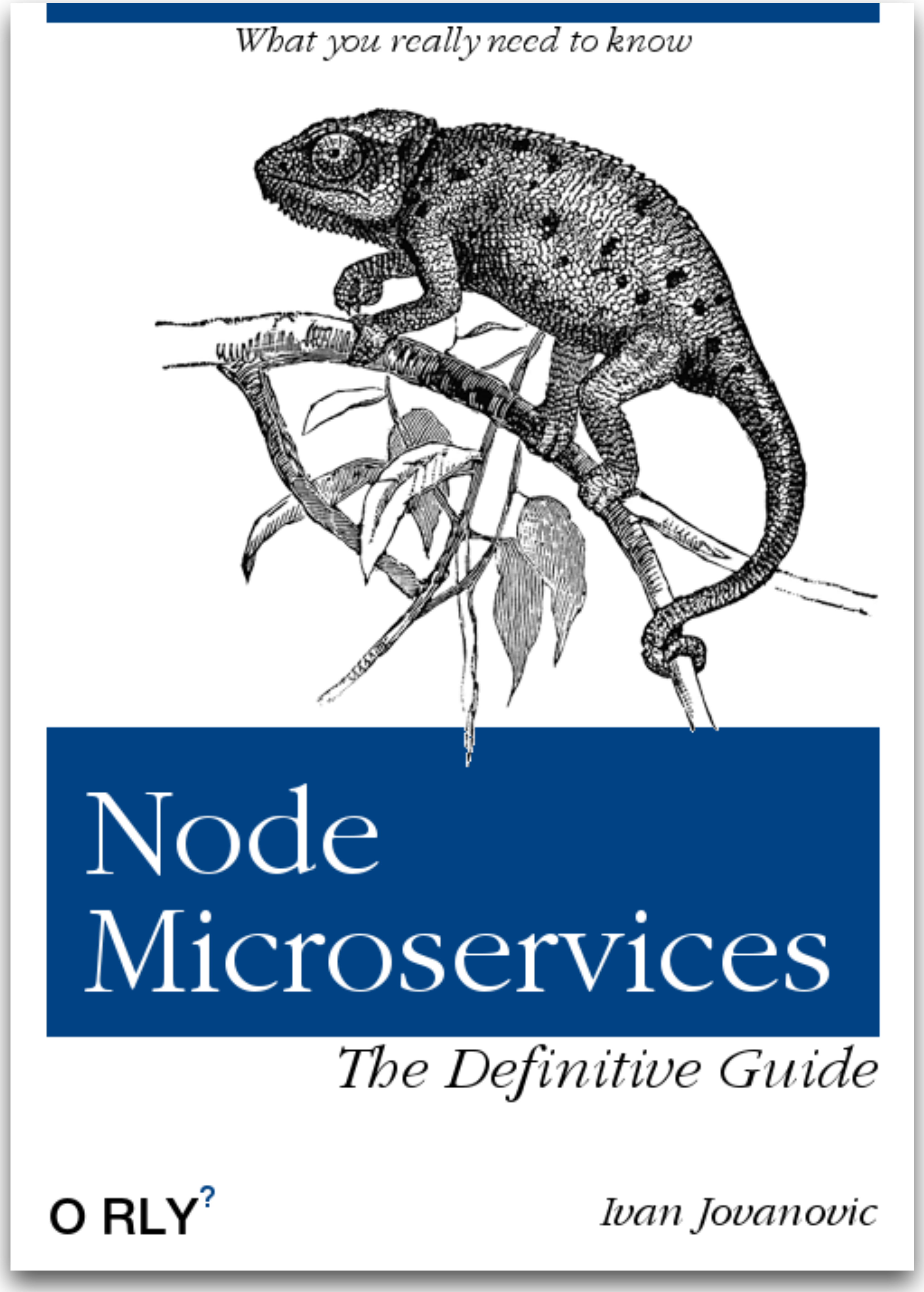
Node Microservices

The Definitive Guide

@ivanjov96

https://ivanjov.com



What you really need to know

Node Microservices

The Definitive Guide

O RLY?

Ivan Jovanovic

Ivan Jovanovic

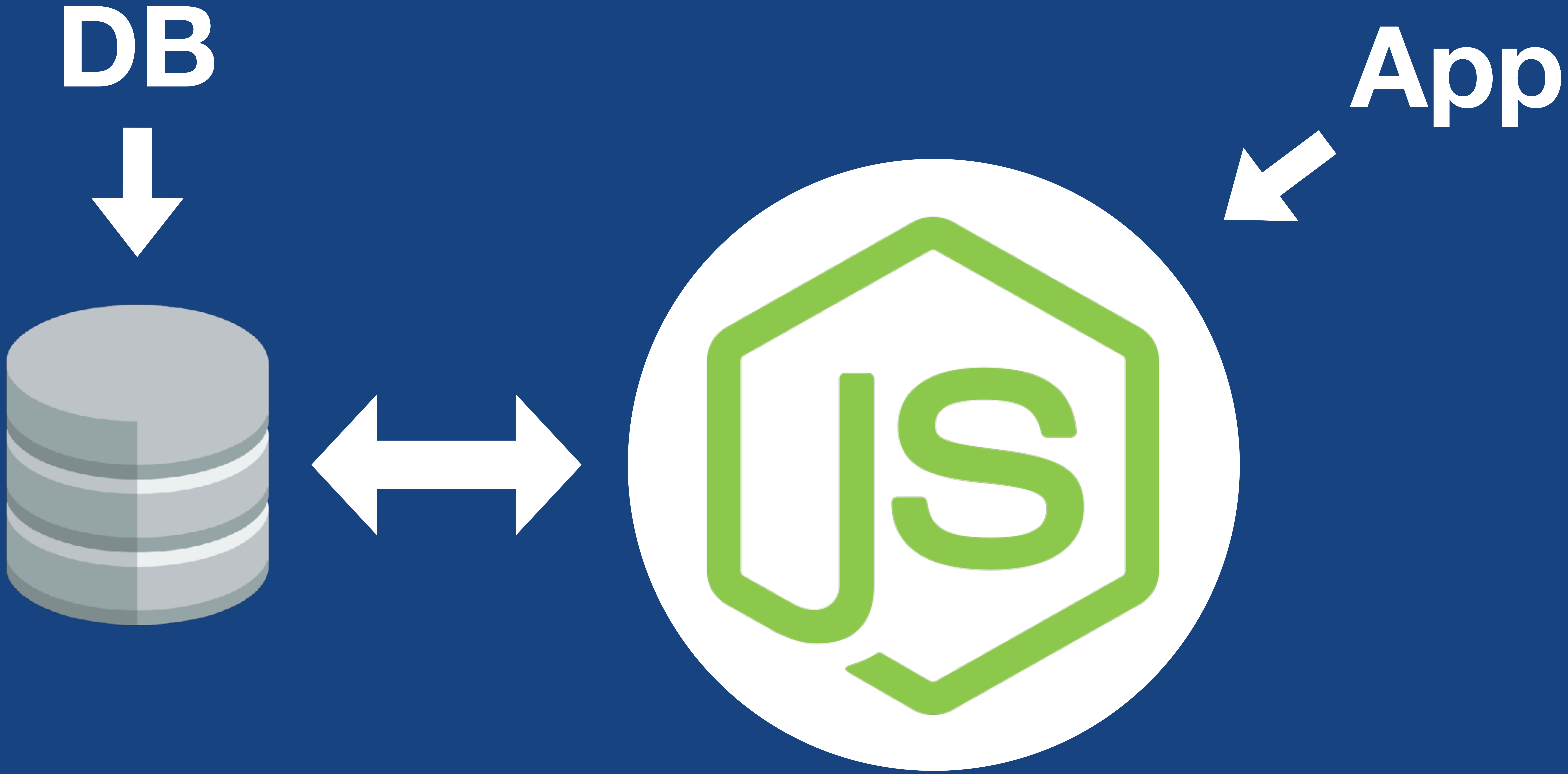
Senior software engineer @ nearForm

Founder @ JS Remotely

@ivanjov96 *https://ivanjov.com*

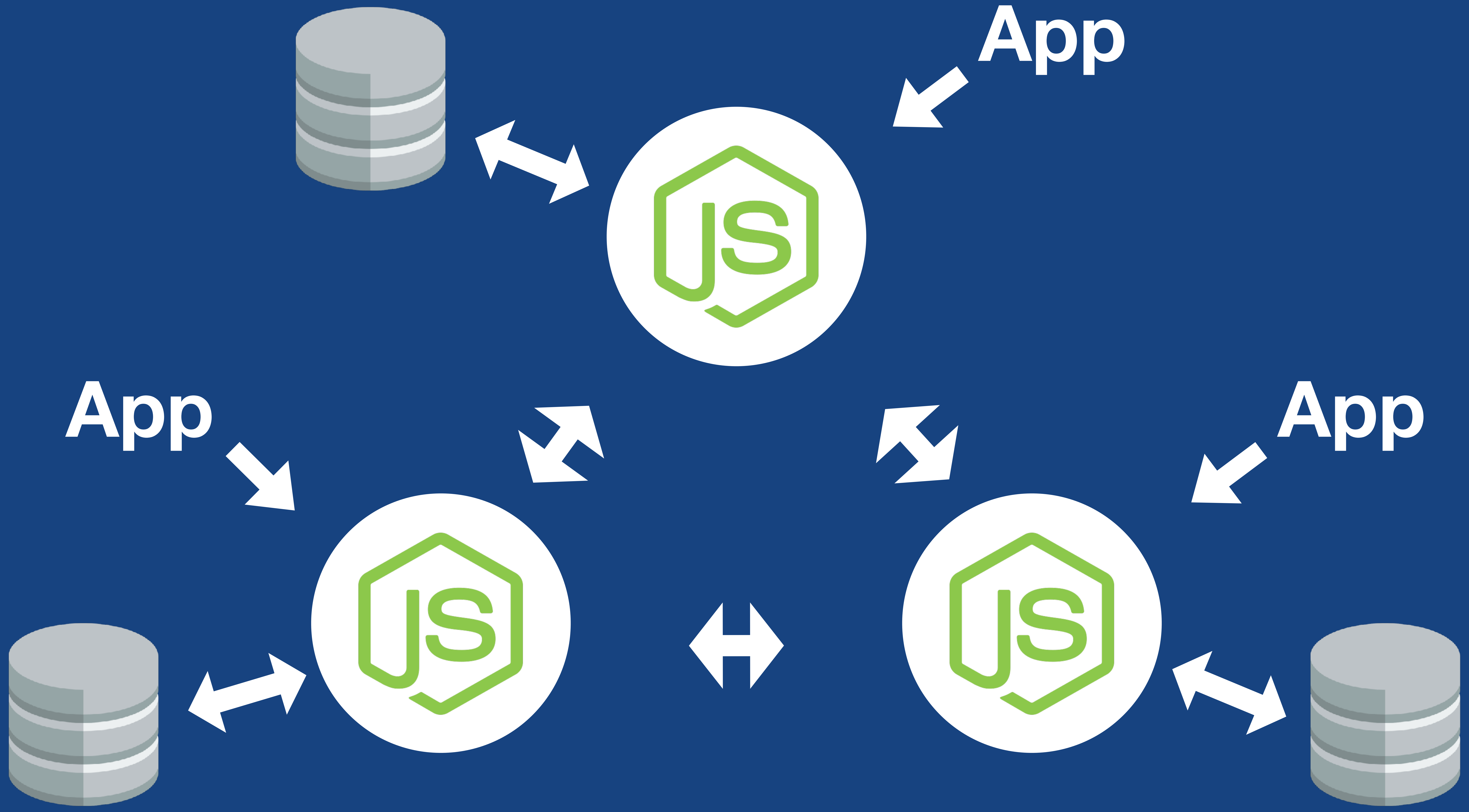
What are Microservices?

Microservices is an architectural style that structures an application as a collection of loosely coupled services



DB

App


App
App
App

Event bus

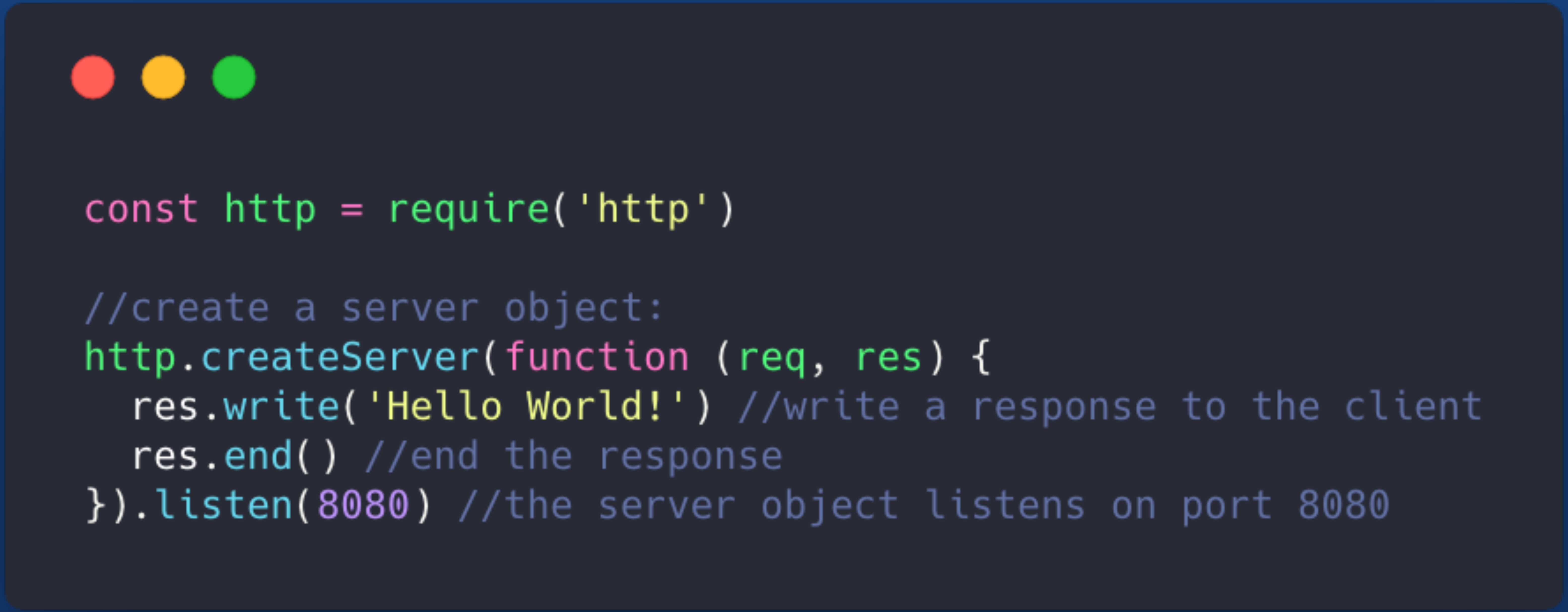
Event bus allows publish-subscribe-style communication between components without requiring the components to explicitly register with one another

Microservice principles

- Lightweight protocol between services
- Small services, one job per service
- Service independence
- Easier to understand, develop and test
- Speeds up development
- Enables continues delivery and deployment

Why Node.js?

- Small and lightweight!
- No boilerplate code
- Non-blocking I/O
- Speed!
- Freedom



```
const http = require('http')

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!') //write a response to the client
  res.end() //end the response
}).listen(8080) //the server object listens on port 8080
```

Tools

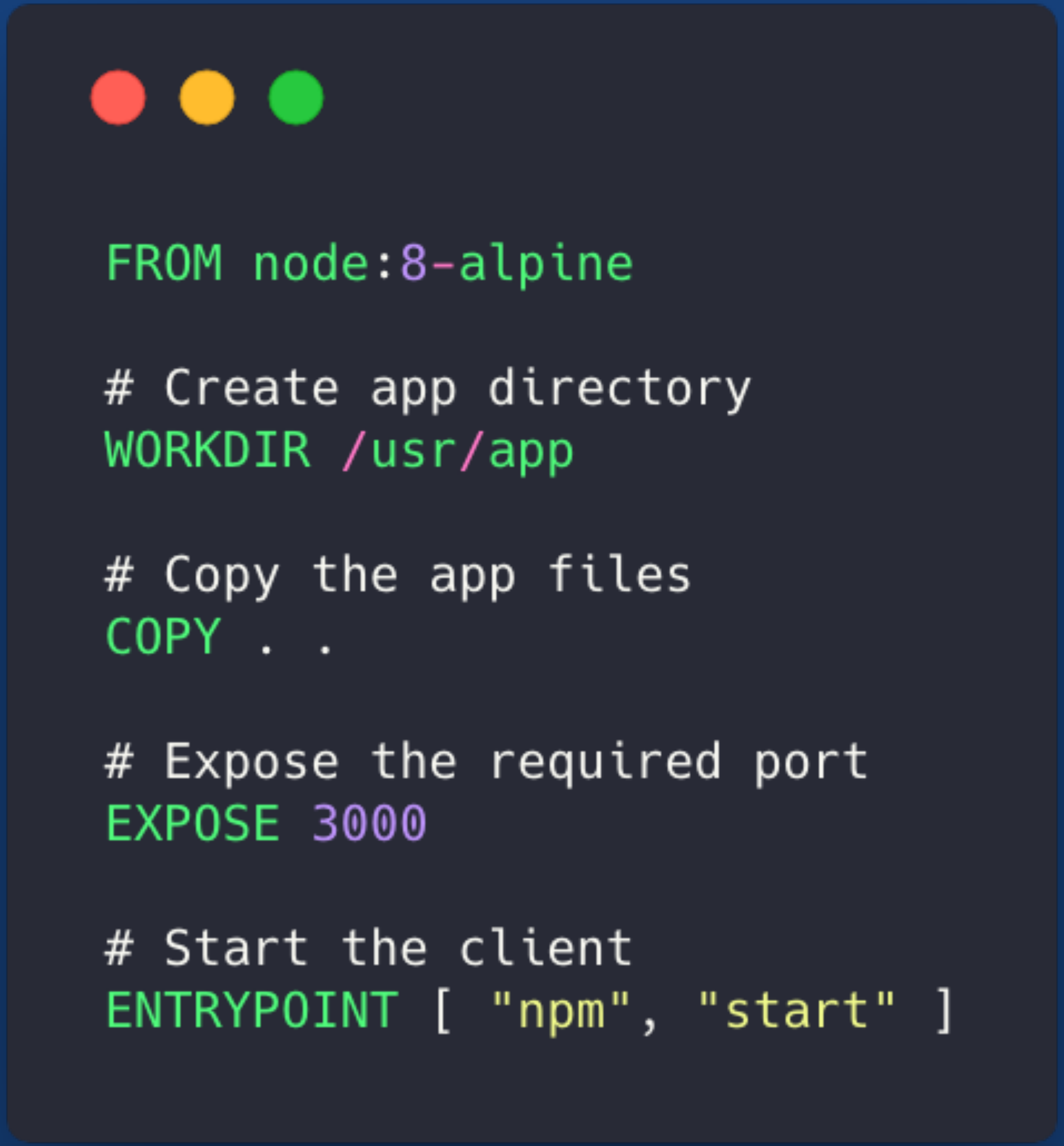
- **Docker** - containers
- **Kubernetes** - container orchestration
- **Kafka** - messaging
- **Elastic Stack** - logging

docker

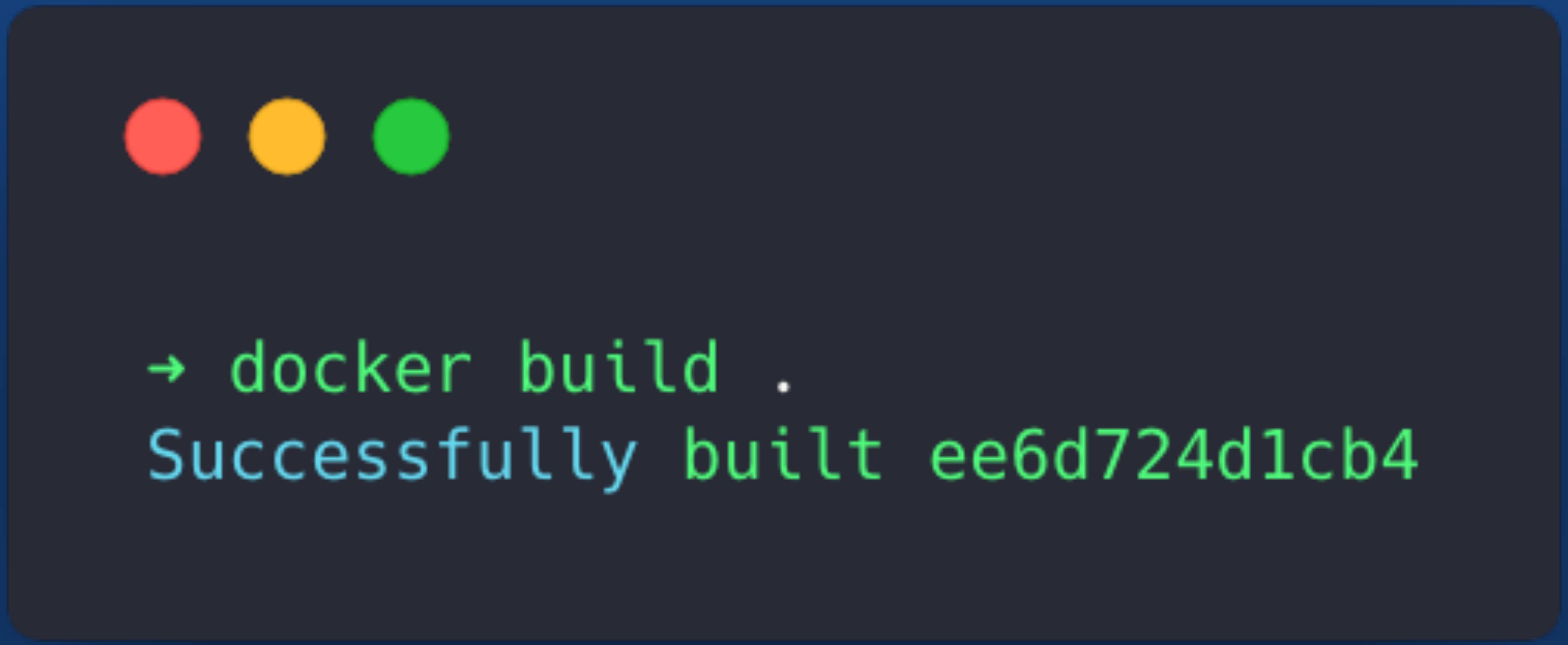
Docker is a tool designed to make it easier to create, deploy, and run applications using containers

Containers allow developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package

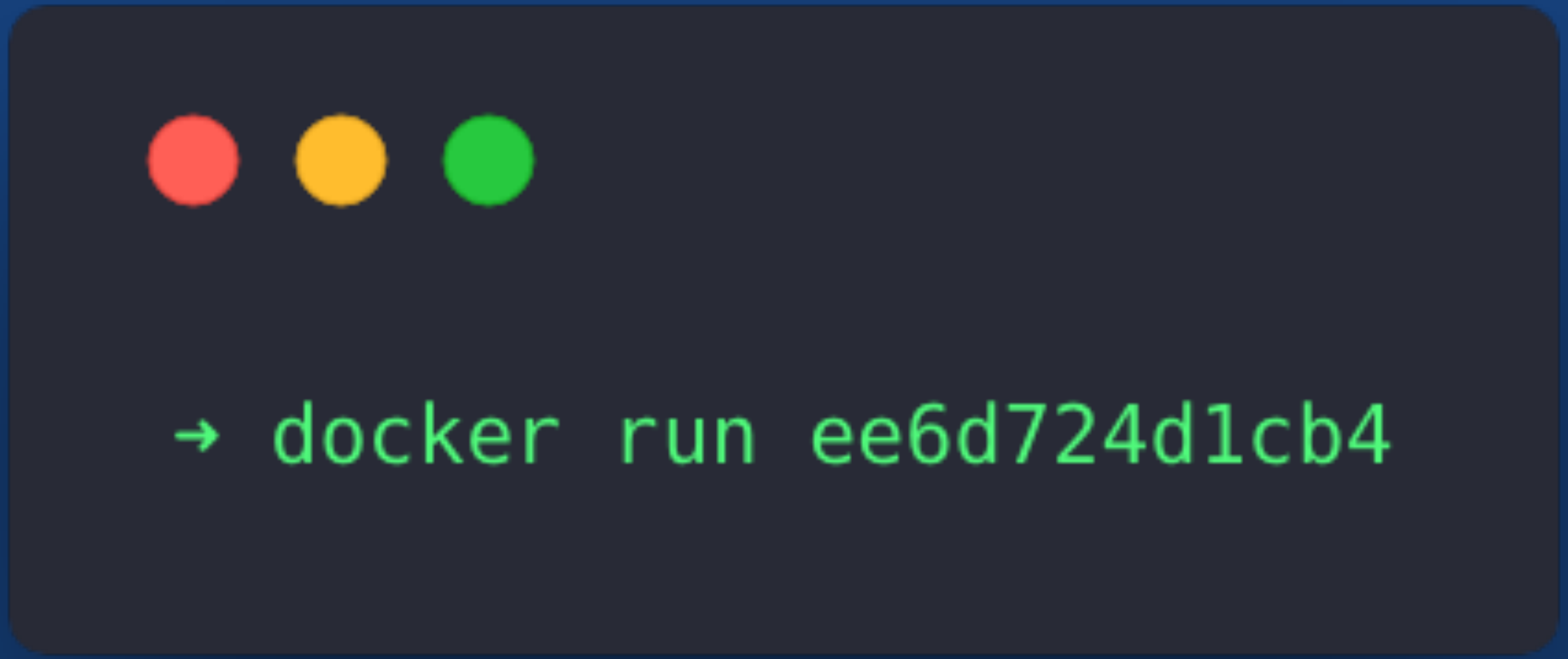
Dockerfile



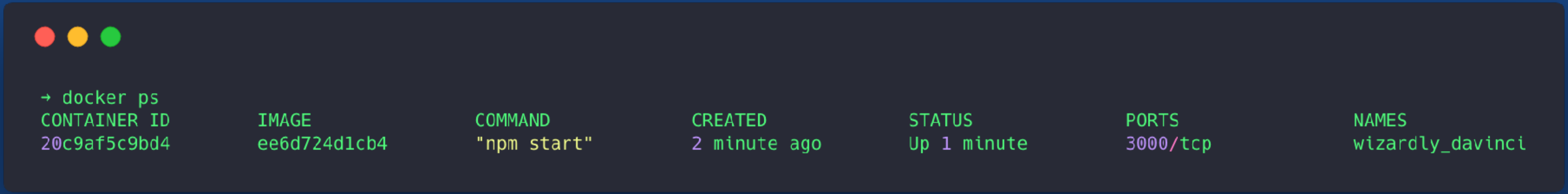
```
FROM node:8-alpine

# Create app directory
WORKDIR /usr/app

# Copy the app files
COPY . .

# Expose the required port
EXPOSE 3000

# Start the client
ENTRYPOINT [ "npm", "start" ]
```



```
➜ docker build .
Successfully built ee6d724d1cb4
```



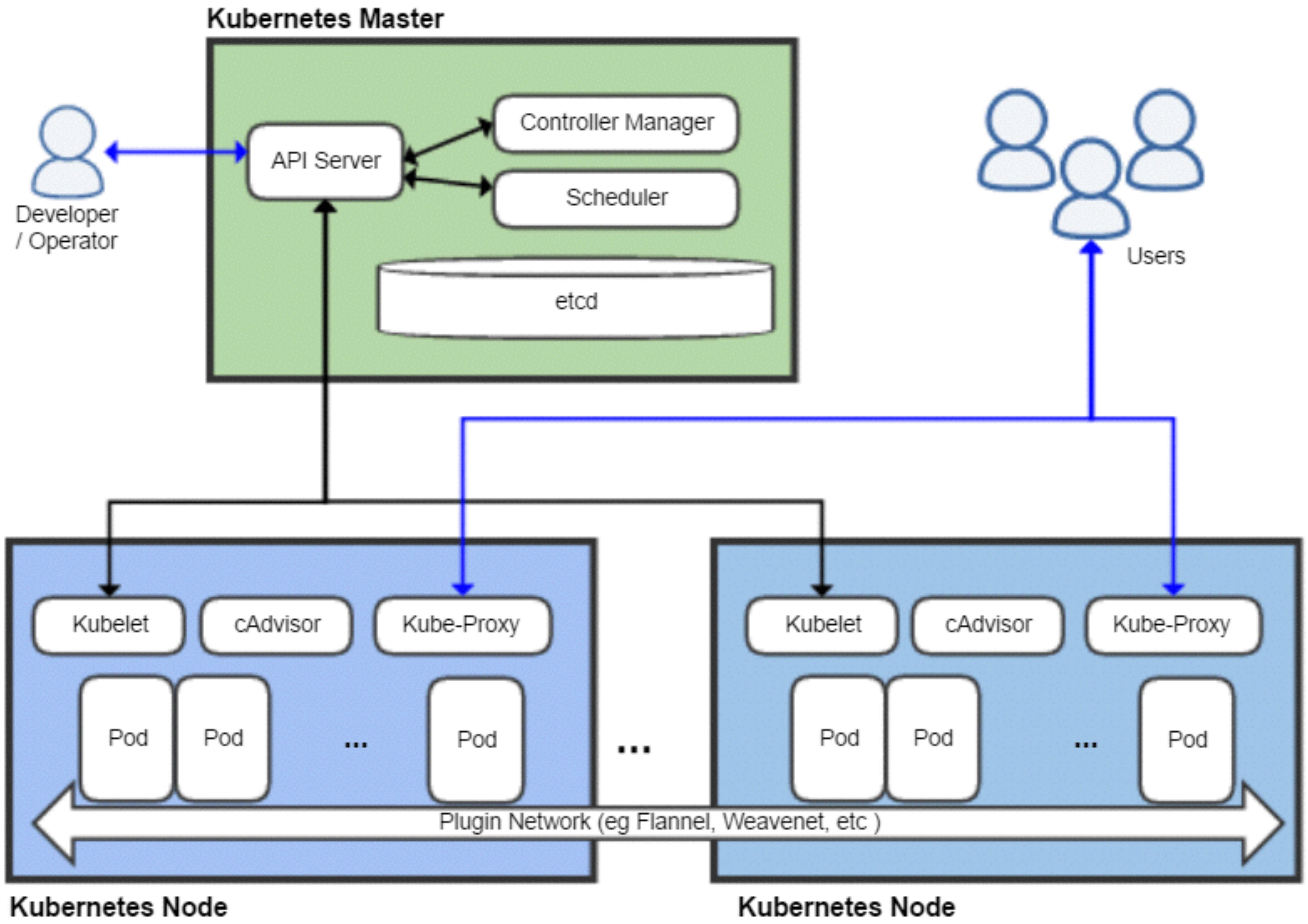
```
➜ docker run ee6d724d1cb4
```



```
→ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS              PORTS                NAMES
20c9af5c9bd4        ee6d724d1cb4        "npm start"         2 minute ago        Up 1 minute         3000/tcp             wizardly_davinci
```

kubernetes

Kubernetes is an open-source system for automating deployment, scaling and management of containerized applications

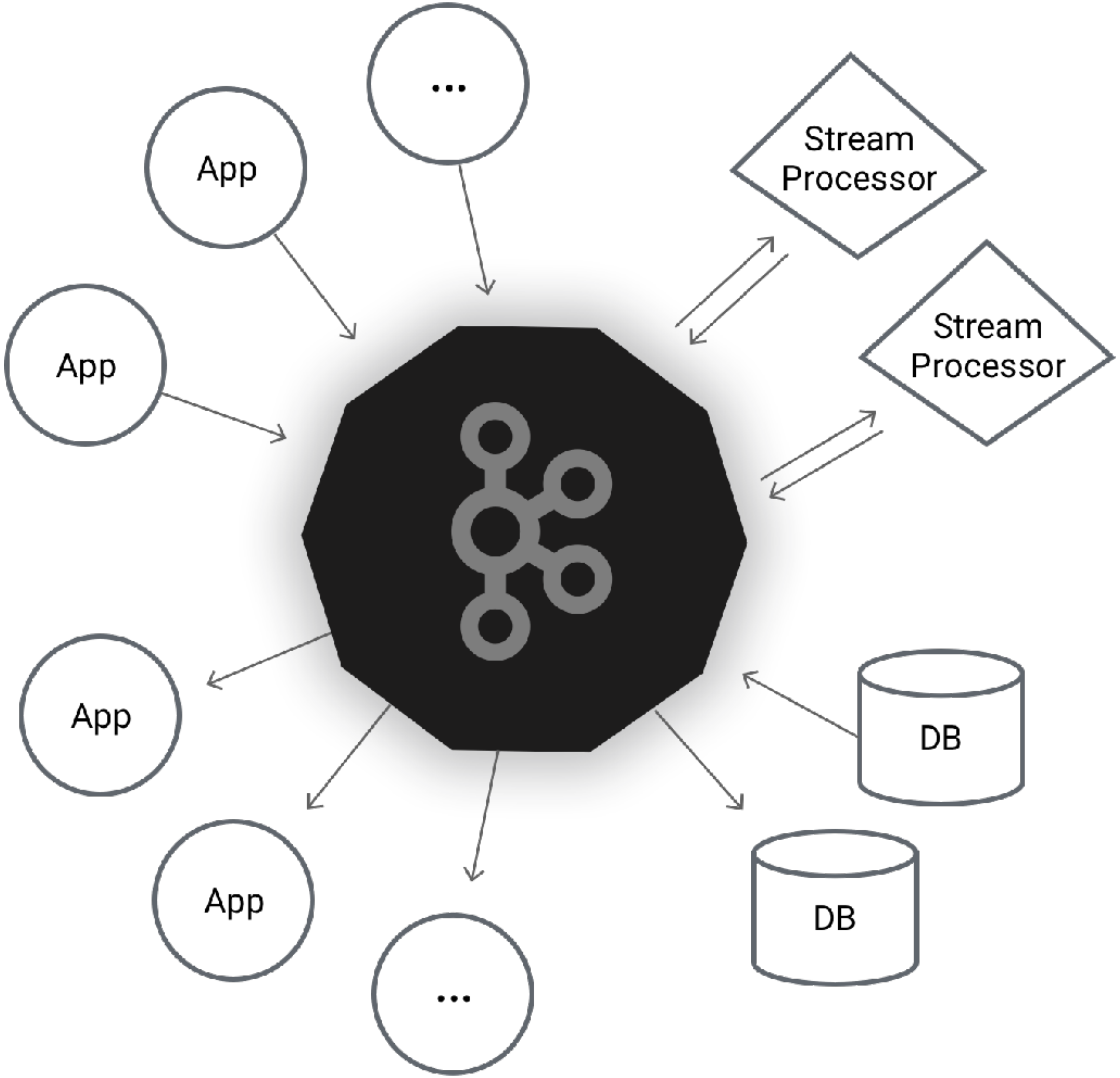

Kubernetes Master
Developer / Operator
API Server
Controller Manager
Scheduler
etcd
Users
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
...
Kubelet
cAdvisor
Kube-Proxy
Pod
Pod
...
Pod
Plugin Network (eg Flannel, Weavenet, etc)
Kubernetes Node
Kubernetes Node

Service communication

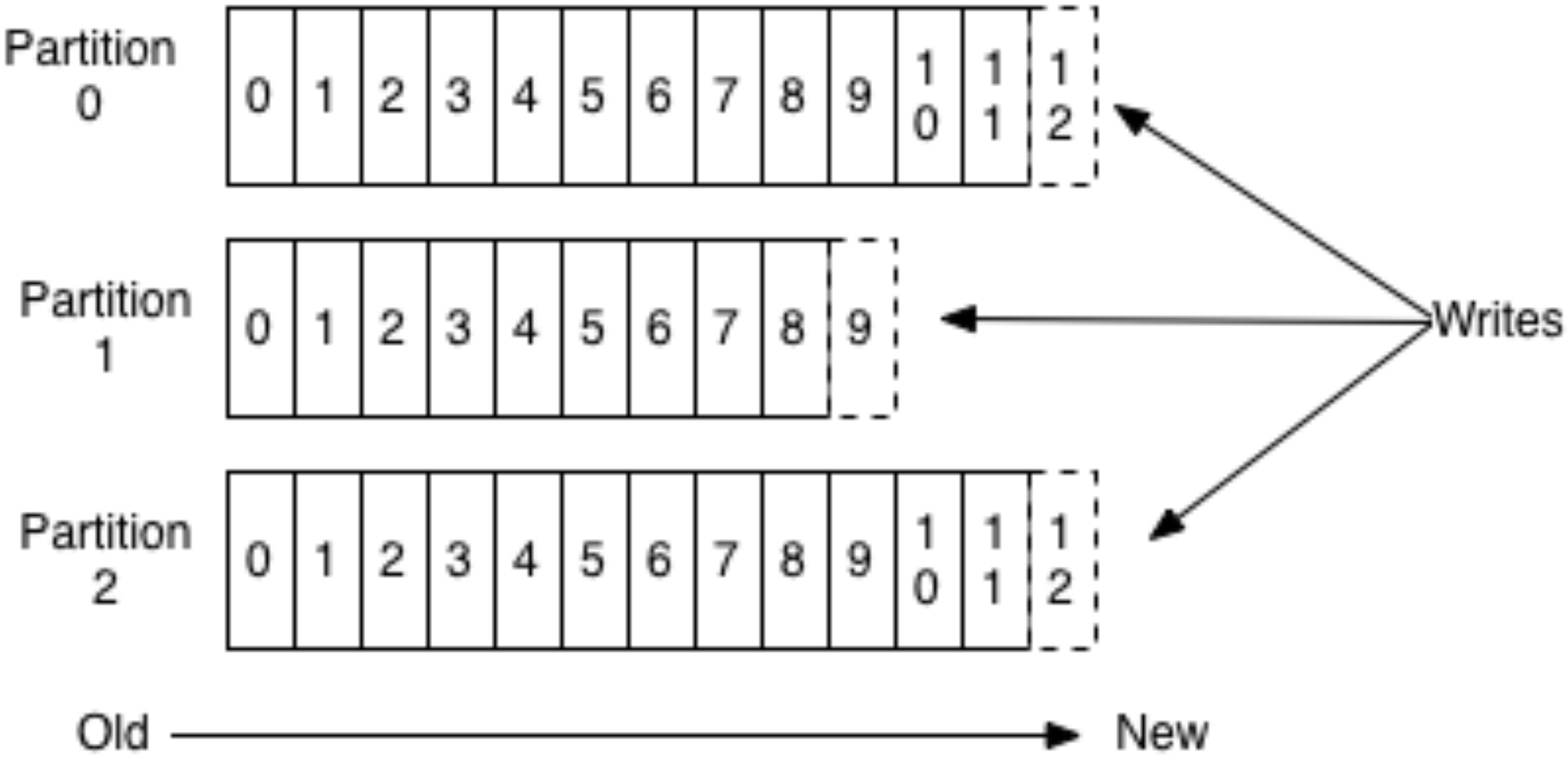
Service communication

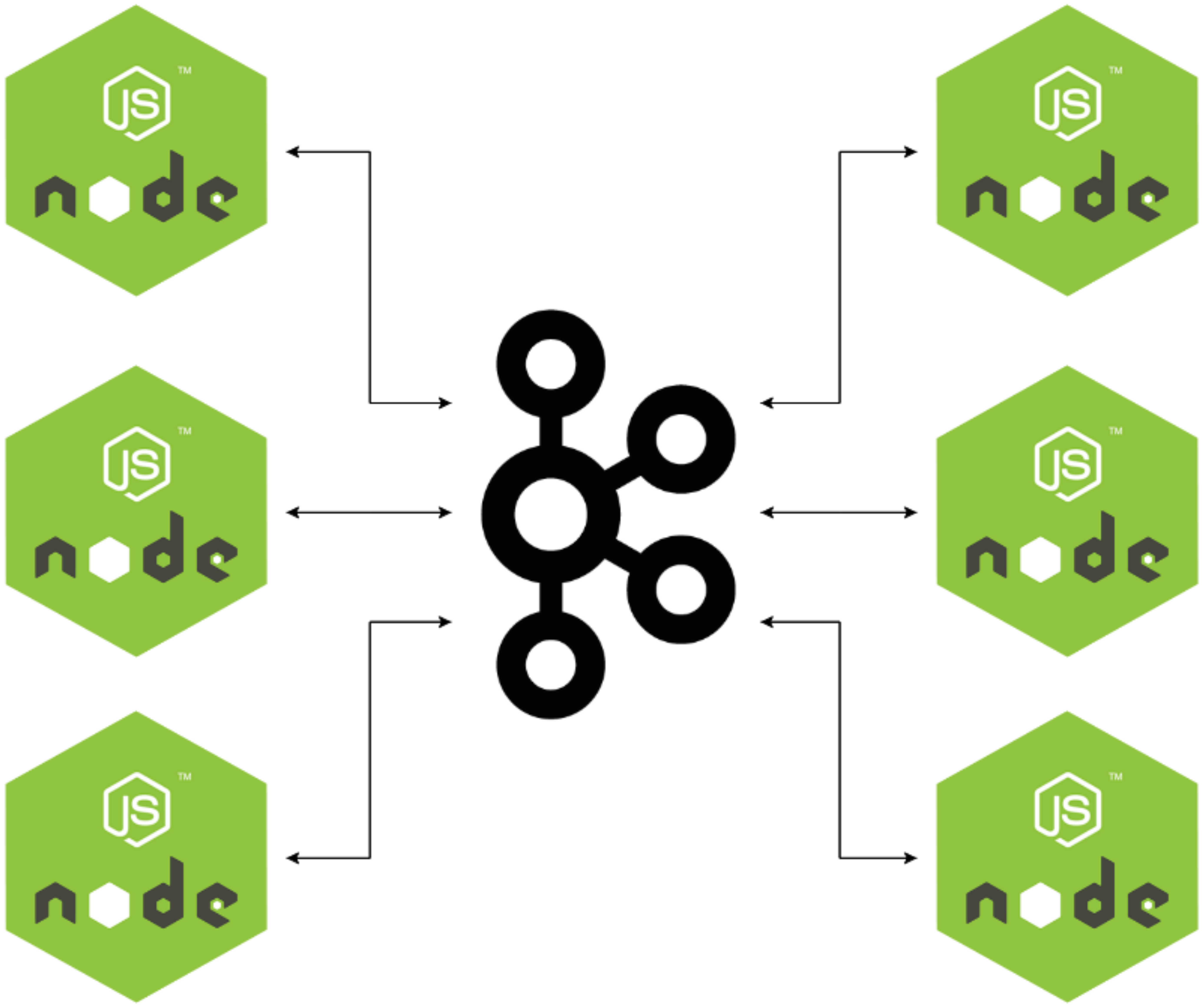
- **Synchronous** - HTTP/REST API
- **Asynchronous** - Messaging

APACHE
kafka®

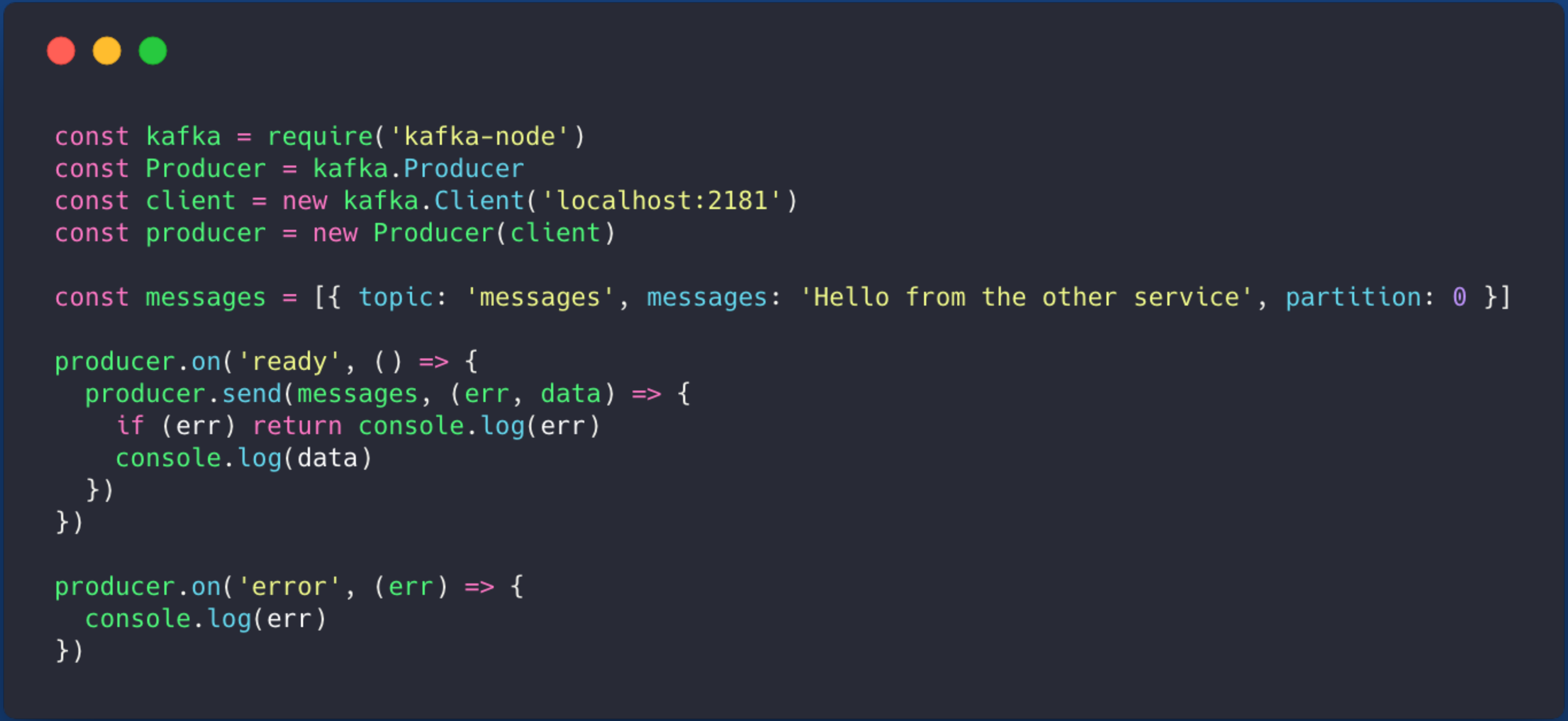

...
App
App
Stream Processor
Stream Processor
App
App
...
DB
DB

Anatomy of a Topic



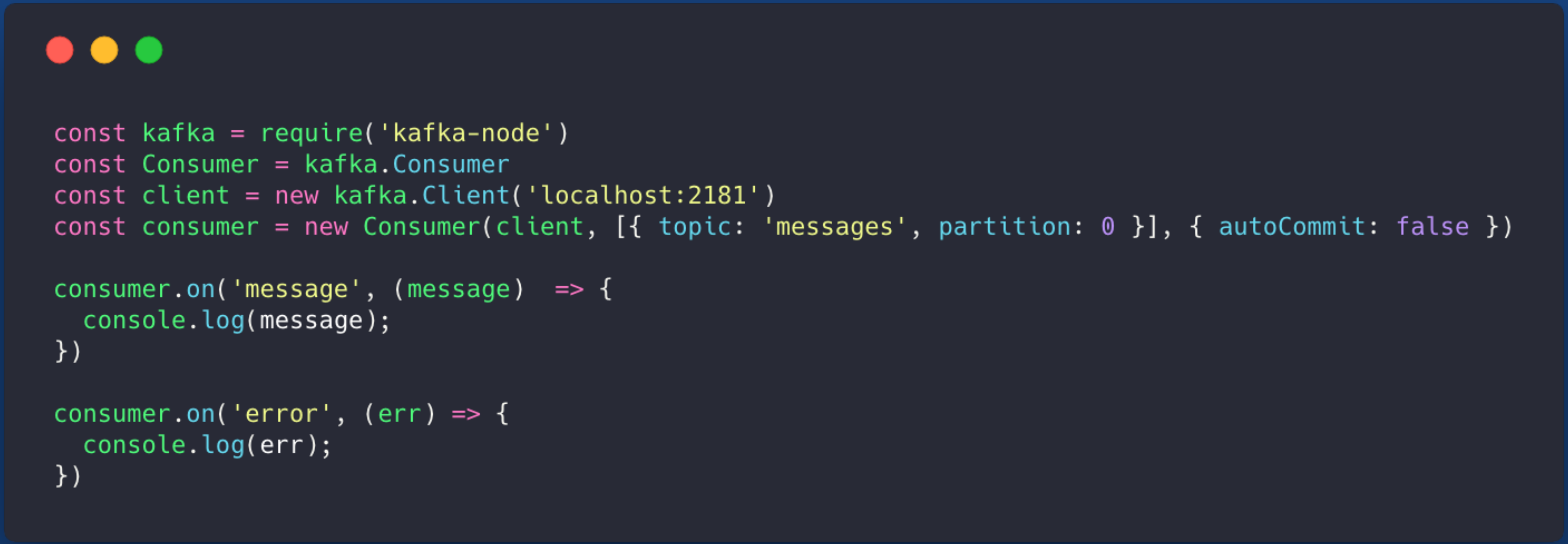

JS
node
JS
node
JS
node
JS
node
JS
node
JS
node

Producer



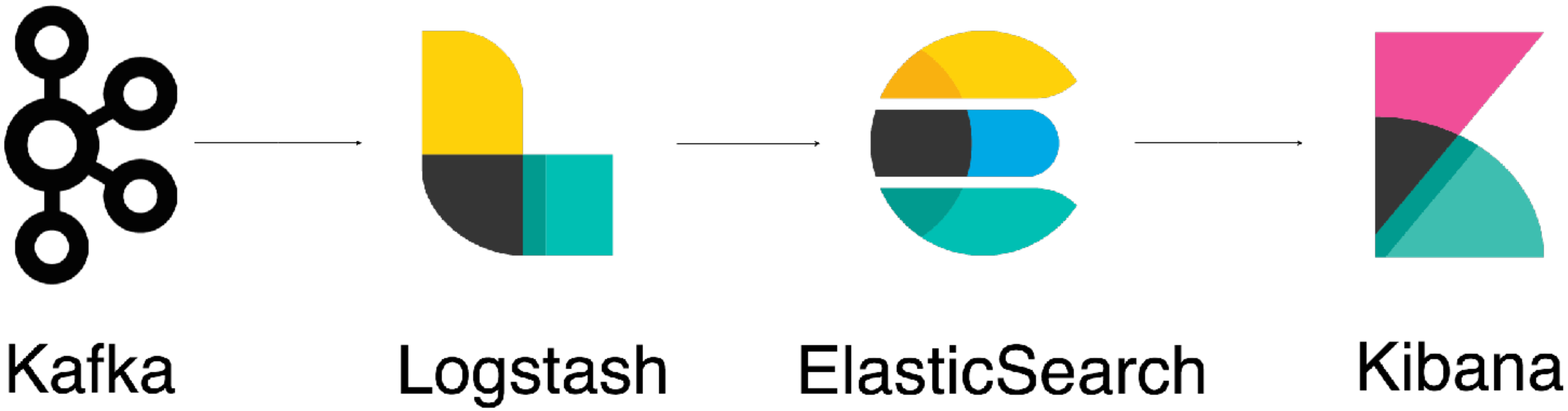
```
const kafka = require('kafka-node')
const Producer = kafka.Producer
const client = new kafka.Client('localhost:2181')
const producer = new Producer(client)

const messages = [{ topic: 'messages', messages: 'Hello from the other service', partition: 0 }]

producer.on('ready', () => {
  producer.send(messages, (err, data) => {
    if (err) return console.log(err)
    console.log(data)
  })
})

producer.on('error', (err) => {
  console.log(err)
})
```

Consumer

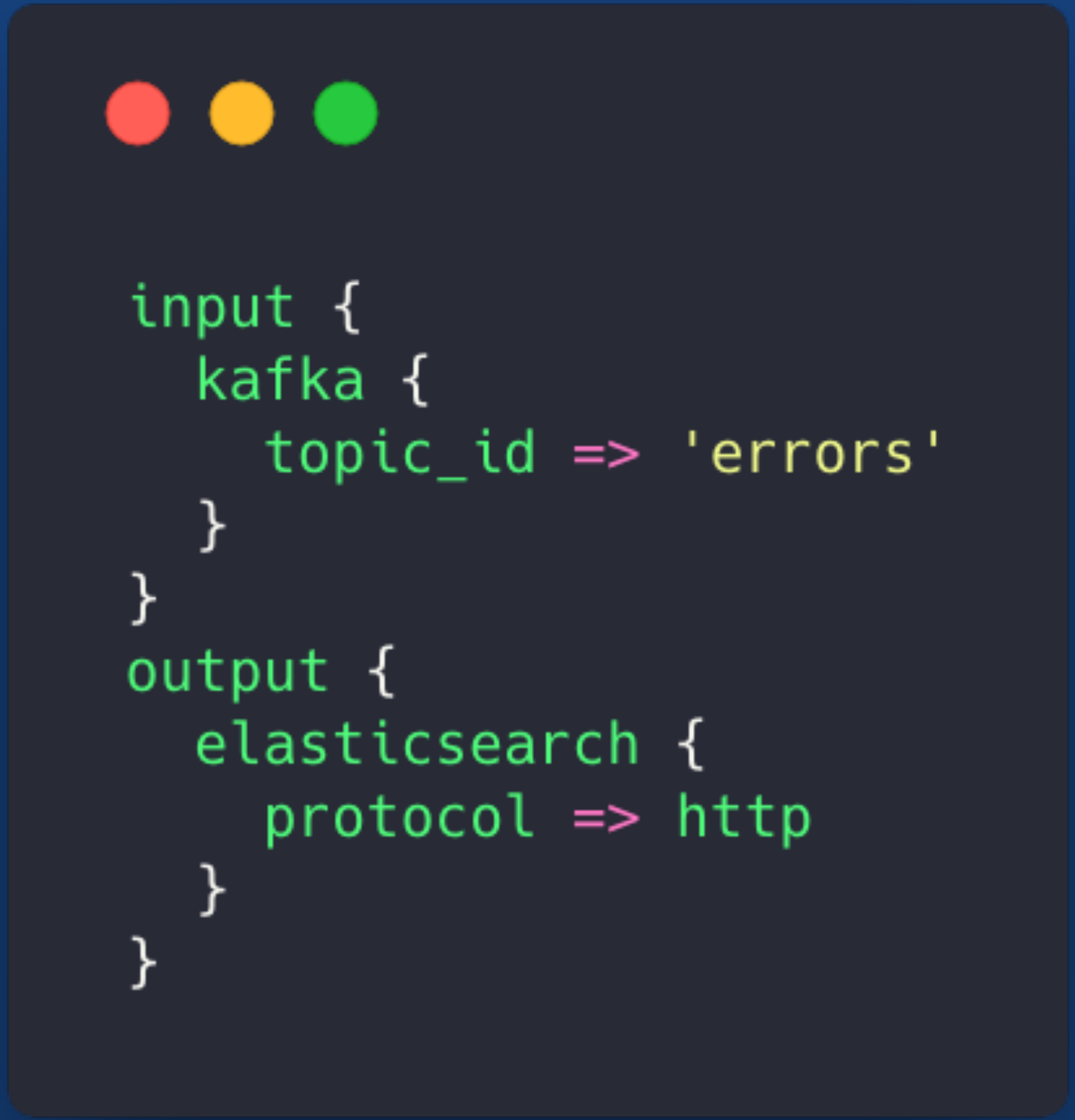


```
const kafka = require('kafka-node')
const Consumer = kafka.Consumer
const client = new kafka.Client('localhost:2181')
const consumer = new Consumer(client, [{ topic: 'messages', partition: 0 }], { autoCommit: false })

consumer.on('message', (message)  => {
  console.log(message);
})

consumer.on('error', (err) => {
  console.log(err);
})
```

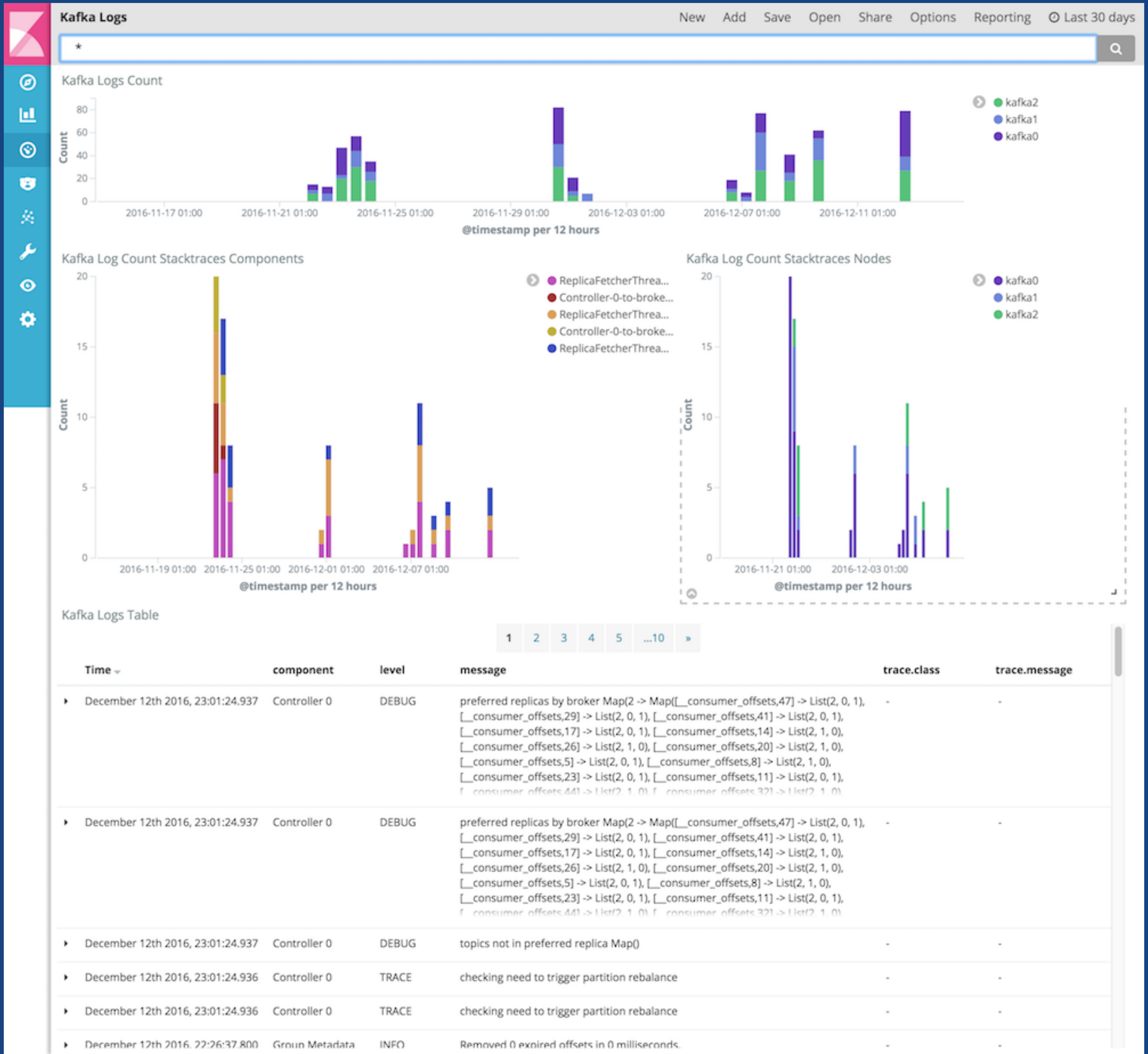
elastic



Kafka → Logstash → ElasticSearch → Kibana



```
input {
  kafka {
    topic_id => 'errors'
  }
}
output {
  elasticsearch {
    protocol => http
  }
}
```



Time	component	level	message	trace.class	trace.message
December 12th 2016, 23:01:24.937	Controller 0	DEBUG	preferred replicas by broker Map(2 -> Map([__consumer_offsets,47] -> List(2, 0, 1), [__consumer_offsets,29] -> List(2, 0, 1), [__consumer_offsets,41] -> List(2, 0, 1), [__consumer_offsets,17] -> List(2, 0, 1), [__consumer_offsets,14] -> List(2, 1, 0), [__consumer_offsets,26] -> List(2, 1, 0), [__consumer_offsets,20] -> List(2, 1, 0), [__consumer_offsets,5] -> List(2, 0, 1), [__consumer_offsets,8] -> List(2, 1, 0), [__consumer_offsets,23] -> List(2, 0, 1), [__consumer_offsets,11] -> List(2, 0, 1), [__consumer_offsets,44] -> List(2, 1, 0), [__consumer_offsets,32] -> List(2, 1, 0),	-	-
December 12th 2016, 23:01:24.937	Controller 0	DEBUG	preferred replicas by broker Map(2 -> Map([__consumer_offsets,47] -> List(2, 0, 1), [__consumer_offsets,29] -> List(2, 0, 1), [__consumer_offsets,41] -> List(2, 0, 1), [__consumer_offsets,17] -> List(2, 0, 1), [__consumer_offsets,14] -> List(2, 1, 0), [__consumer_offsets,26] -> List(2, 1, 0), [__consumer_offsets,20] -> List(2, 1, 0), [__consumer_offsets,5] -> List(2, 0, 1), [__consumer_offsets,8] -> List(2, 1, 0), [__consumer_offsets,23] -> List(2, 0, 1), [__consumer_offsets,11] -> List(2, 0, 1), [__consumer_offsets,44] -> List(2, 1, 0), [__consumer_offsets,32] -> List(2, 1, 0),	-	-
December 12th 2016, 23:01:24.937	Controller 0	DEBUG	topics not in preferred replica Map()	-	-
December 12th 2016, 23:01:24.936	Controller 0	TRACE	checking need to trigger partition rebalance	-	-
December 12th 2016, 23:01:24.936	Controller 0	TRACE	checking need to trigger partition rebalance	-	-
December 12th 2016, 22:26:37.800	Group Metadata	INFO	Removed 0 expired offsets in 0 milliseconds.	-	-

Conclusion

- Use NodeJS!
- Use Async communication
- Use containers
- Use Kubernetes for container orchestration
- Log everything to ElasticSearch

Thank you 🎉

Twitter
@ivanjov96

Blog
https://ivanjov.com