Clusters & Worker Threads

Rene Corrales

JavaScript Developer

¿Qué es un proceso?

¿Qué es un thread?

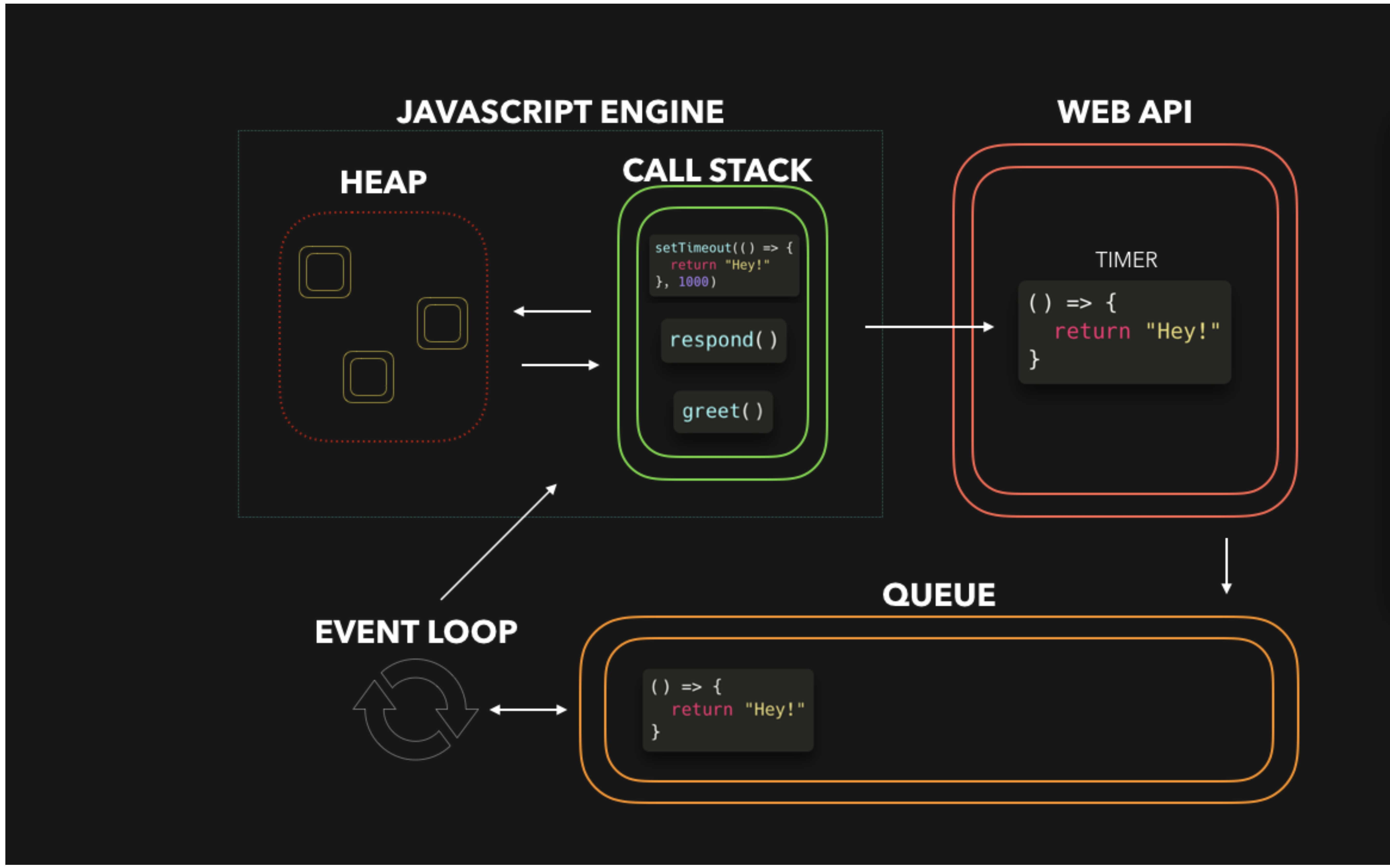
¿Es JavaScript single thread?

¿Es JavaScript single thread?

Si.

¿Cómo funciona?

Event Loop



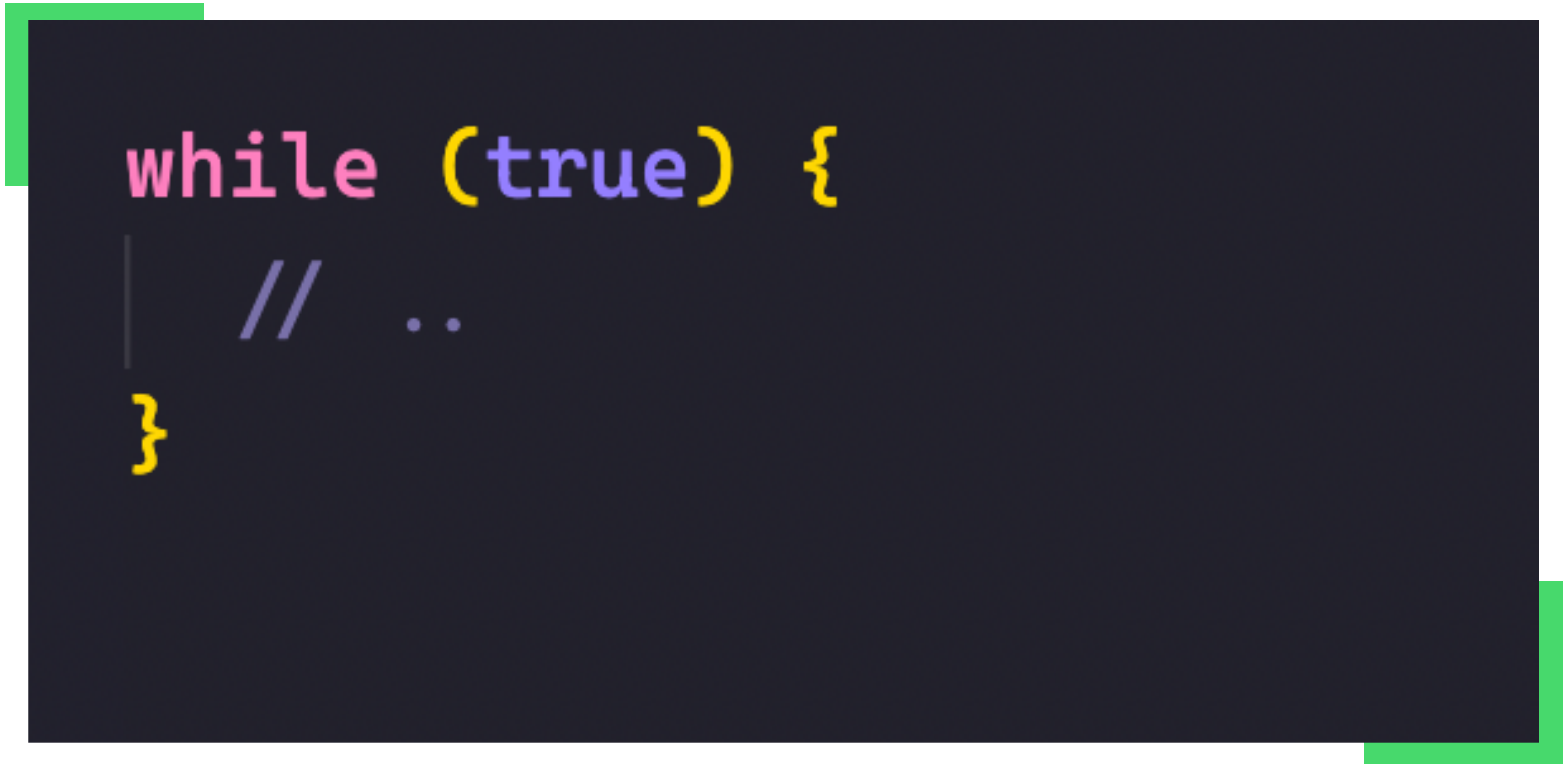
```
function greet() {
  return "Hello!"
}

function respond() {
  return setTimeout(() => {
    return "Hey!"
  }, 1000))
}

greet()
respond()
```

¿Es JavaScript single thread?

Entonces, **JavaScript** se ejecuta en un solo thread pero las **Web APIs** No.

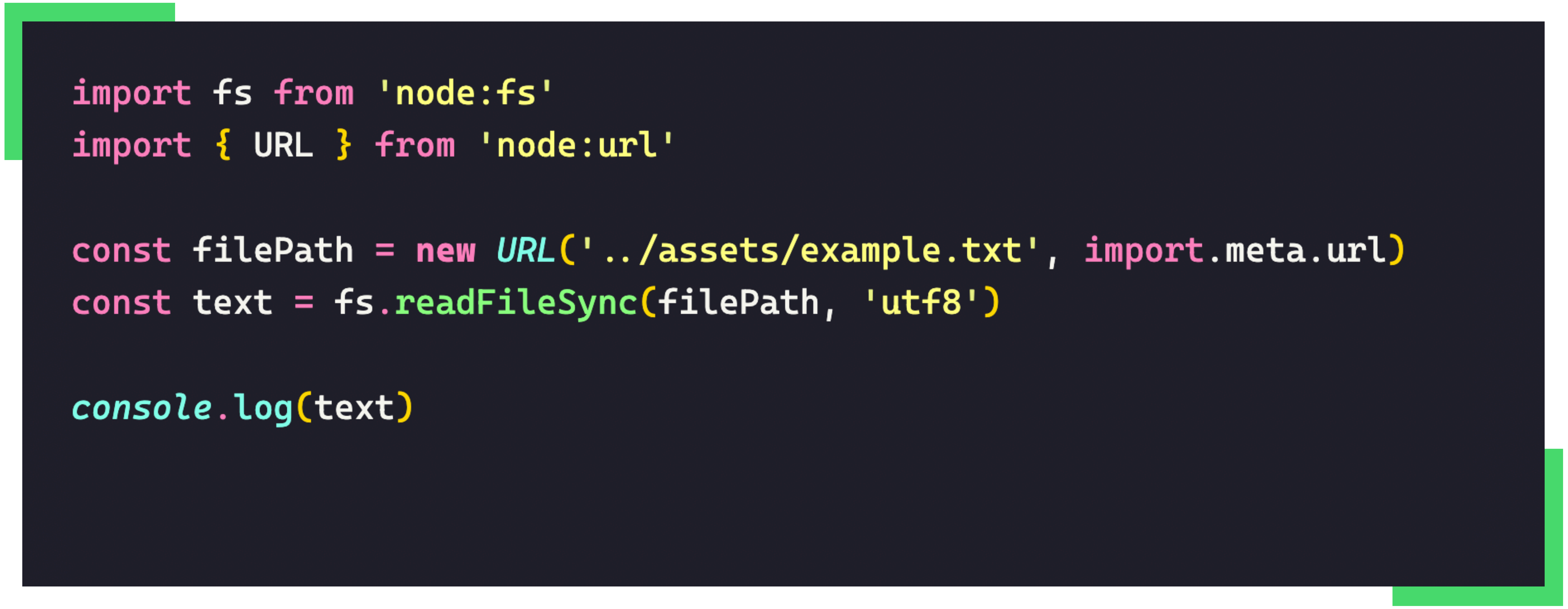
¿Qué cosas **no** debemos hacer?

¿Qué cosas no debemos hacer?



```
while (true) {
  // ..
}
```

¿Qué cosas no debemos hacer?



```
import fs from 'node:fs'
import { URL } from 'node:url'

const filePath = new URL('../assets/example.txt', import.meta.url)
const text = fs.readFileSync(filePath, 'utf8')

console.log(text)
```

¿Qué cosas no debemos hacer?

```javascript
import { pbkdf2Sync, randomBytes } from 'node:crypto'

const pass = randomBytes(20).toString()
const salt = Buffer.allocUnsafe(20)
const keylen = 400
const iterations = 5e6

const passHash = pbkdf2Sync(pass, salt, iterations, keylen, 'sha256')

console.log('Simple Hash', passHash.toString('hex'))
```

¿Cómo no bloquear el **Event Loop**?

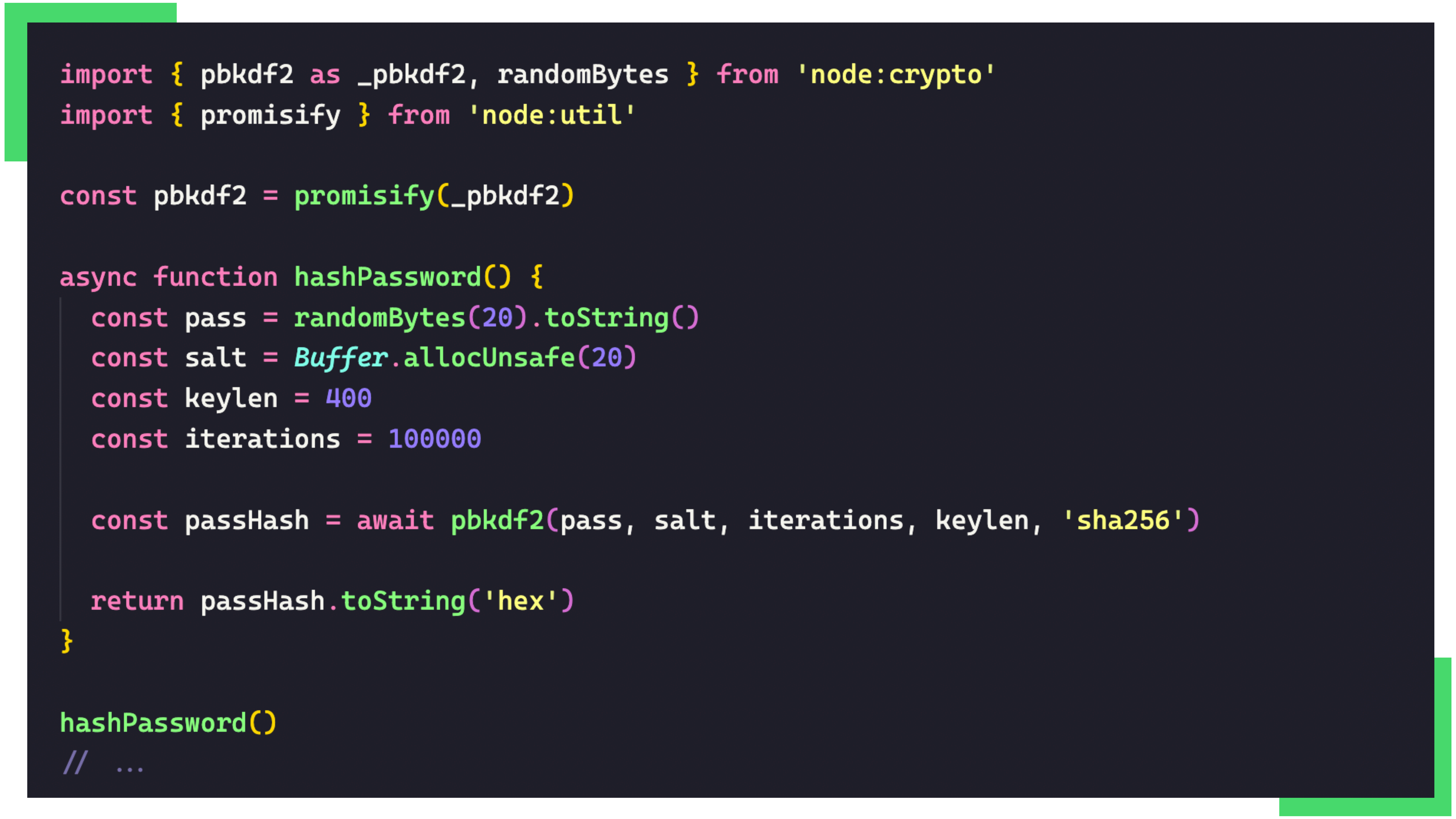
¿Cómo no bloquear el Event Loop?

Usar las api asíncronas de NodeJS

¿Cómo no bloquear el Event Loop?

```
import fs from 'node:fs/promises'

async function printFileData() {
  const filePath = new URL('../assets/example.txt', import.meta.url)
  const text = await fs.readFile(filePath, 'utf8')
  console.log(text)
}

printFileData()
// ...
```

¿Qué cosas no debemos hacer?

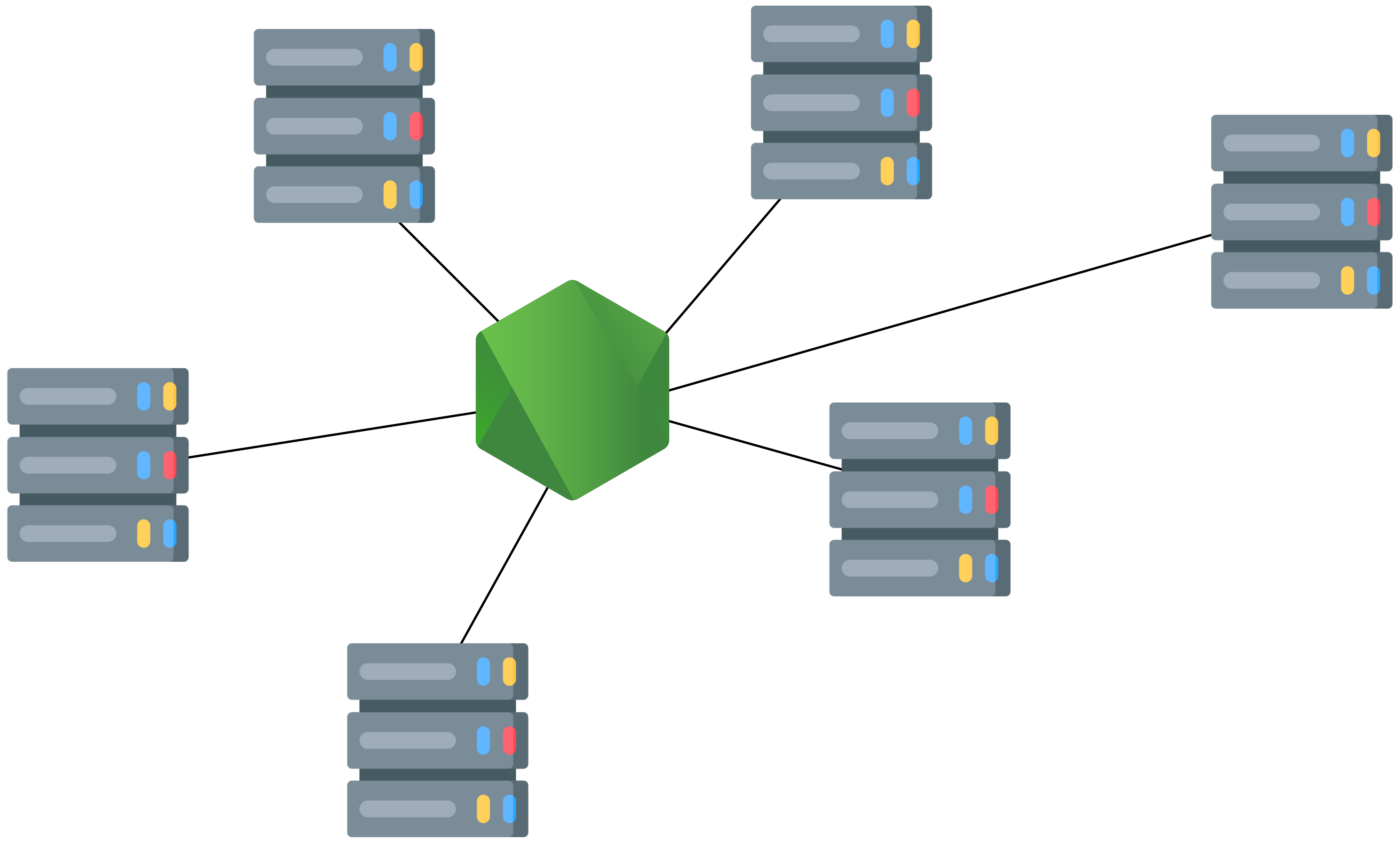


```js
import { pbkdf2 as _pbkdf2, randomBytes } from 'node:crypto'
import { promisify } from 'node:util'

const pbkdf2 = promisify(_pbkdf2)

async function hashPassword() {
  const pass = randomBytes(20).toString()
  const salt = Buffer.allocUnsafe(20)
  const keylen = 400
  const iterations = 100000

  const passHash = await pbkdf2(pass, salt, iterations, keylen, 'sha256')

  return passHash.toString('hex')
}

hashPassword()
// ...
```

Clusters

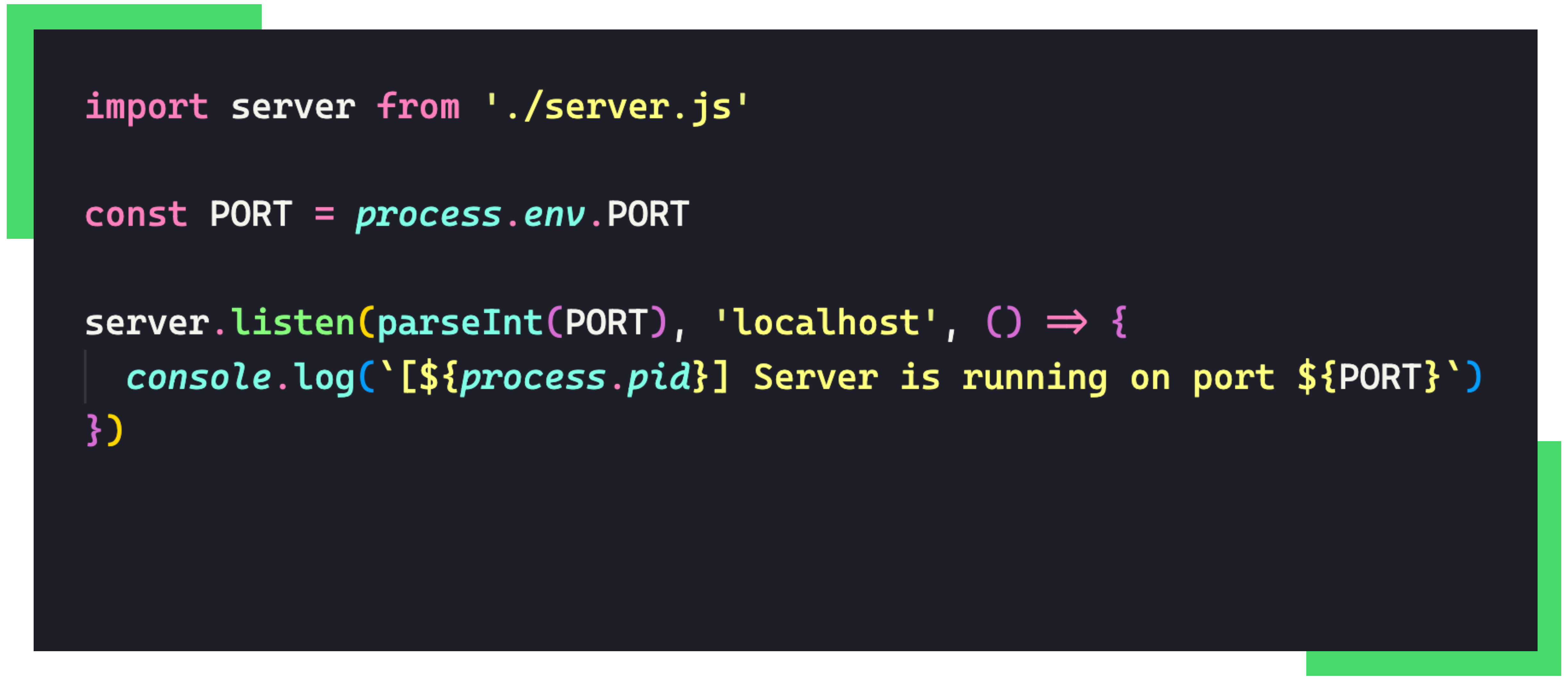
¿Que es Clusters?

¿Que es Clusters?

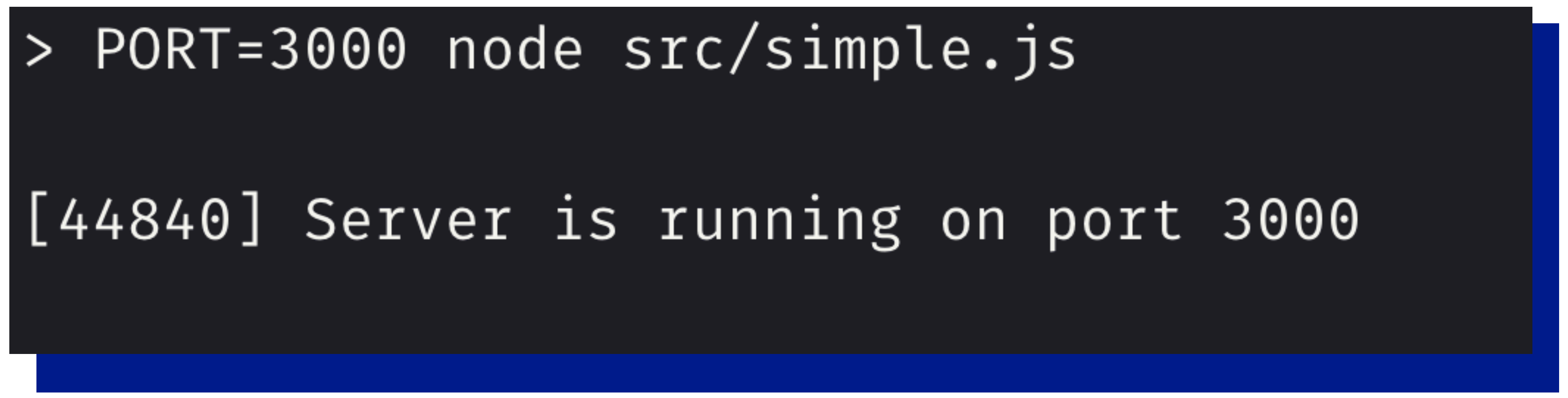


Modulo cluster en NodeJS

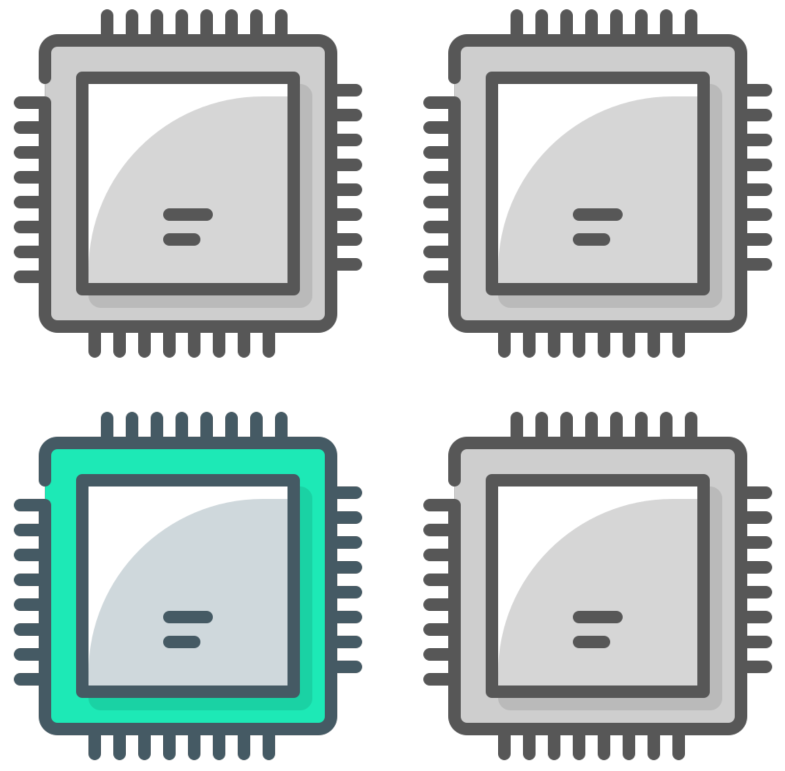
Modulo cluster en NodeJS



```js
import server from './server.js'

const PORT = process.env.PORT

server.listen(parseInt(PORT), 'localhost', () => {
  console.log(`[${process.pid}] Server is running on port ${PORT}`)
})
```



```
> PORT=3000 node src/simple.js

[44840] Server is running on port 3000
```

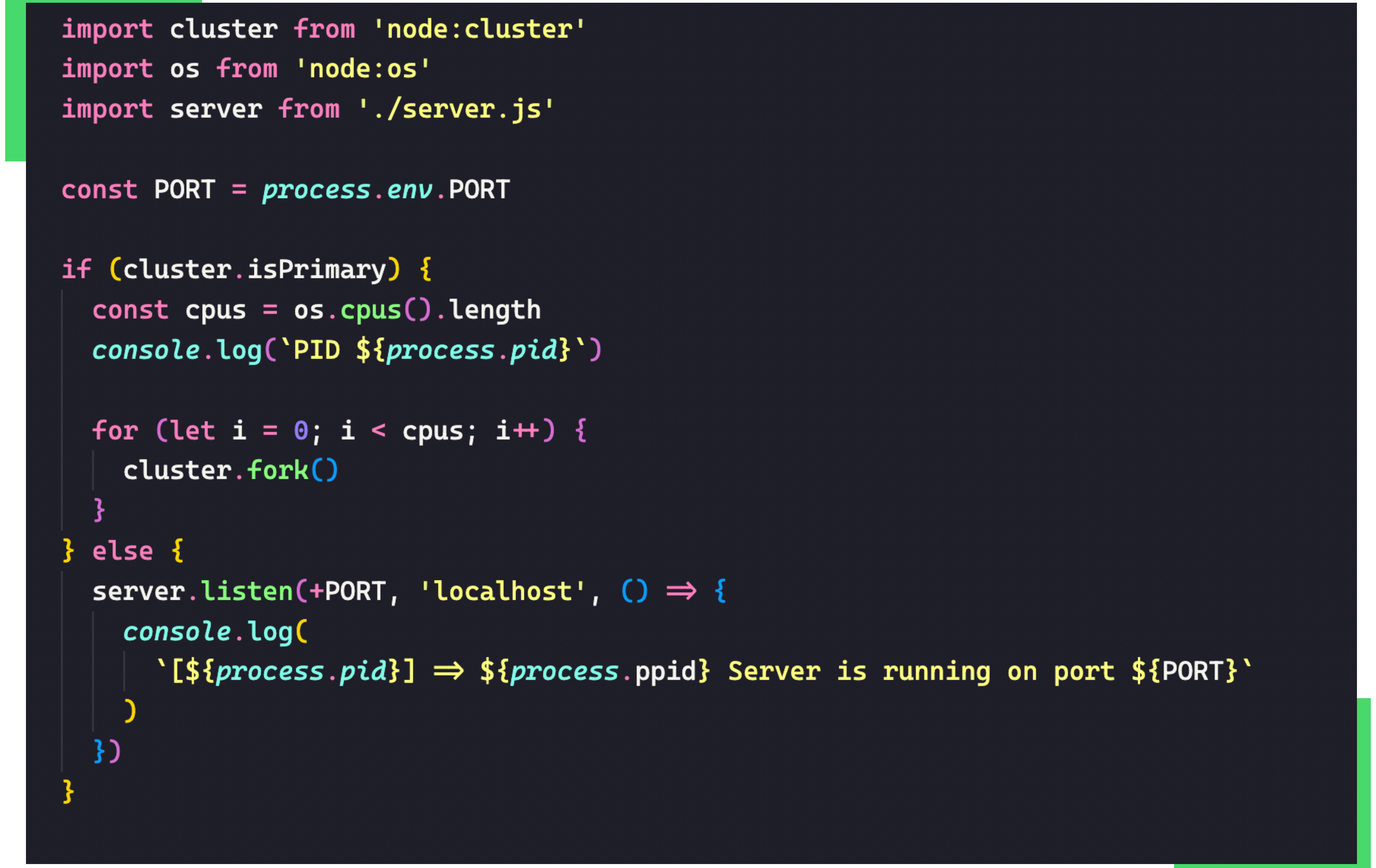


4 CPU

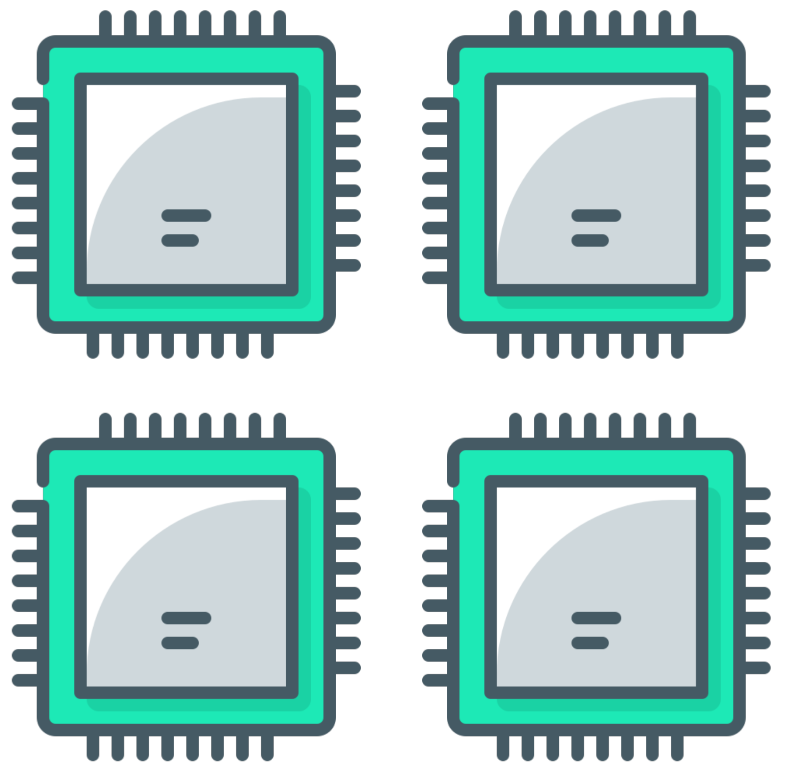
Modulo cluster en NodeJS

```
import cluster from 'node:cluster'
```

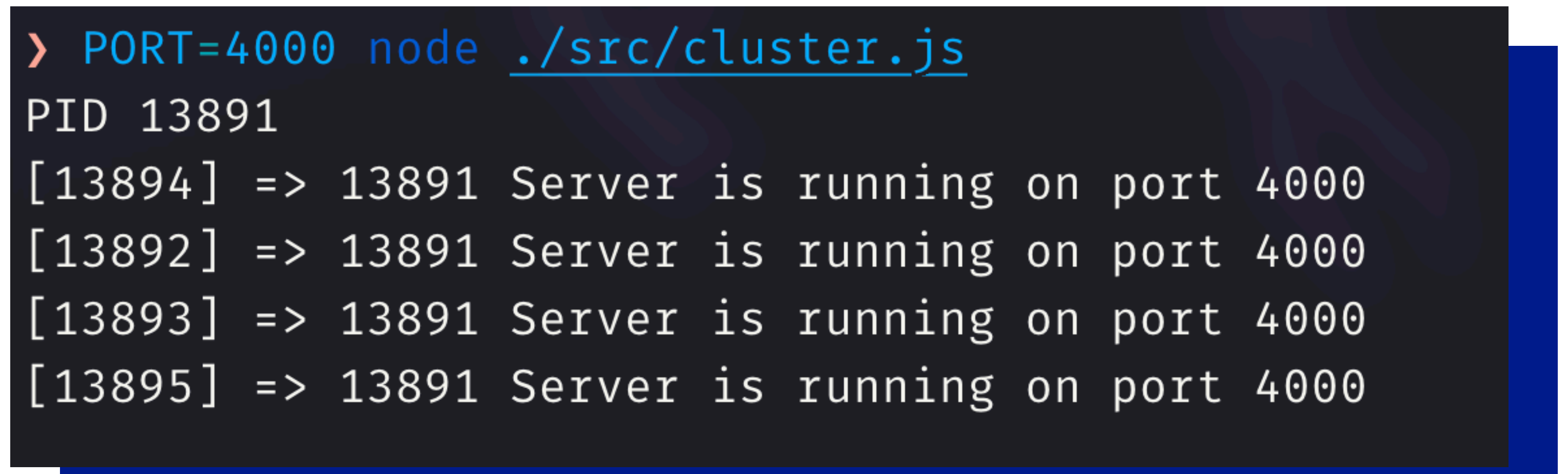
Modulo cluster en NodeJS



```js
import cluster from 'node:cluster'
import os from 'node:os'
import server from './server.js'

const PORT = process.env.PORT

if (cluster.isPrimary) {
  const cpus = os.cpus().length
  console.log(`PID ${process.pid}`)

  for (let i = 0; i < cpus; i++) {
    cluster.fork()
  }
} else {
  server.listen(+PORT, 'localhost', () => {
    console.log(
      `[${process.pid}] => ${process.ppid} Server is running on port ${PORT}`
    )
  })
}
```

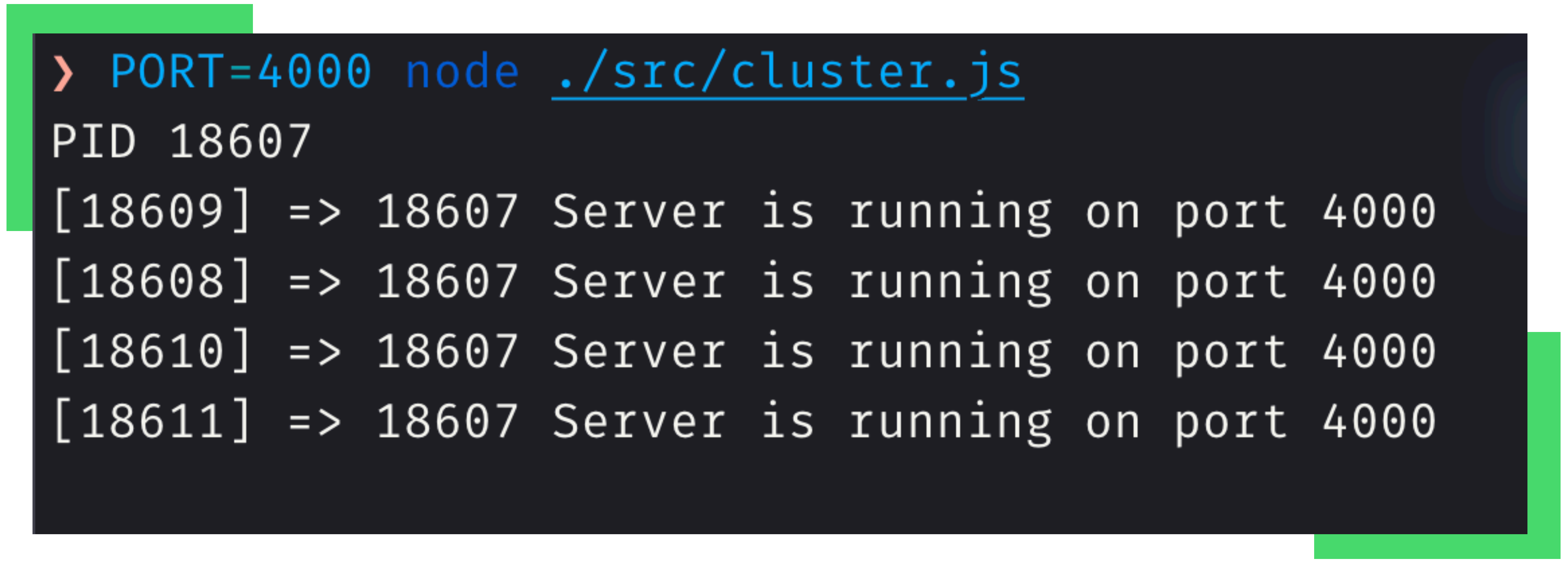


4 CPU



```
❯ PORT=4000 node ./src/cluster.js
PID 13891
[13894] => 13891 Server is running on port 4000
[13892] => 13891 Server is running on port 4000
[13893] => 13891 Server is running on port 4000
[13895] => 13891 Server is running on port 4000
```

Modulo cluster en NodeJS



```
❯ PORT=4000 node ./src/cluster.js
PID 18607
[18609] => 18607 Server is running on port 4000
[18608] => 18607 Server is running on port 4000
[18610] => 18607 Server is running on port 4000
[18611] => 18607 Server is running on port 4000
```

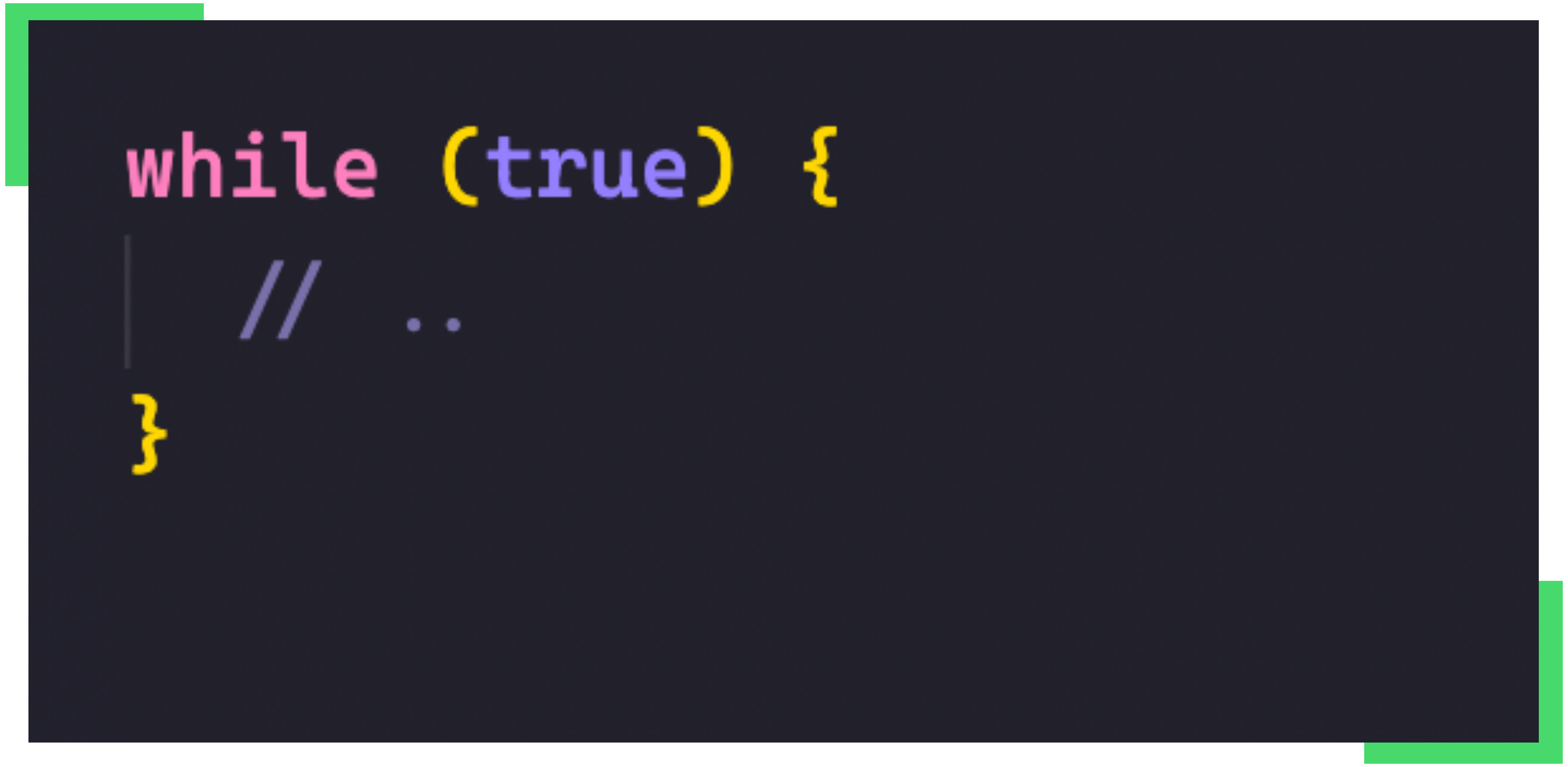
Modulo cluster en NodeJS

Veamos algo de Código

Pero aun así podríamos bloquear el
Even Loop

¿Cómo?

🤔



```
while (true) {
  // ..
}
```

Workers Threads

Modulo worker threads

Worker threads

```
import { Worker } from 'node:worker_threads'
```

Worker threads

```
import { Worker, MessageChannel } from 'node:worker_threads'
```

Worker threads

Veamos algo de Código

Gracias