



Producción

Contenido

- Mejorar el rendimiento
- Configurar la aplicación
- Subir a la nube: PaaS heroku

Mejora del rendimiento

- Utilizar la **compresión** de gzip
 - La compresión de gzip puede disminuir significativamente el tamaño del cuerpo de respuesta y, por lo tanto, aumentar la velocidad de una aplicación web. Utilizar el middleware de compresión para la compresión de gzip en la aplicación Express
 - Si se usa un proxy inverso (Nginx) no es necesario el middleware
- **No** utilizar **funciones síncronas**
 - Ralentizan las llamadas. Utilizar --trace-sync **en desarrollo** para detectar las funciones síncronas
- Utilizar un **middleware** para el servicio de **archivos estáticos** (serve-static) en lugar de res.sendFile()
- Realizar un **registro correcto**
 - console.log y console.err son funciones síncronas. Cambiar por un **módulo de depuración** (debug)
- Manejar todas las **excepciones** correctamente. Utilizar:
 - try-catch
 - promesas

Configurar la aplicación

- Configurar variables de entorno
- Establecer NODE_ENV en “production”
- Asegurarse de que la aplicación se reinicia automáticamente
- Ejecutar la aplicación en un clúster
- Almacenar en la caché los resultados de la solicitud
- Utilizar un equilibrador de carga
- Utilizar un proxy inverso

Variables de entorno

- Antes del despliegue a producción, hay que preparar el código de la aplicación.
- Proteger la información privada dentro de variables de entorno:
 - Claves de API,
 - Passwords Admin
 - Rutas de db
 - Puerto HTTP
 - Indicar la base de datos de Desarrollo, Pruebas, Integración o Producción
 - URLs de recursos del servidor
 - CDNs para testing vs. production...

Variables de entorno: **process.env**

- Variable global inyectada por Node en tiempo de ejecución
- Se accede a través del objeto process, la propiedad env y la variable que queremos invocar. Ej.:

```
process.env.PORT
```

- Cuando no están definidos los valores de las variables se pueden definir como:

```
const port = process.env.PORT || 3000;
```

```
process.env.PORT = process.env.PORT || 3000;
```

- Definición de las variables en tiempo de ejecución:

```
$ PORT = 3000 node server
```

Variables de entorno

- En fichero config.js:

```
//config.js
module.export = {
  port: process.env.PORT || 3001,
  db: process.env.MONGODB_URI || "mongodb:http://localhost:27017/test",
  SECRET_TOKEN: "mi clave de token"
}
```

- En fichero *.env*
 - Incluir en fichero .gitignore

```
NODE_ENV=development
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=myspassword
DB_NAME=desarrolloactivo_db
```

```
npm install dotenv
```

```
require('dotenv').config();
```

NODE_ENV = 'production'

- Almacena en la caché las plantillas de vistas
- Almacena en la caché los archivos CSS generados en las extensiones CSS
- Genera menos mensajes de error detallados

Reinicio Automático

- Utilizar un gestor de procesos: contenedor de aplicaciones que facilita el despliegue, proporciona una alta disponibilidad y permite gestionar la aplicación en el tiempo de ejecución.
- Además de reiniciar la aplicación cuando se bloquea, un gestor de procesos permite:
 - Obtener información útil sobre el rendimiento en tiempo de ejecución y el consumo de recursos.
 - Modificar dinámicamente los valores para mejorar el rendimiento.
 - Controlar la agrupación en clúster (StrongLoop PM y pm2).
- Los gestores de procesos más conocidos para Node son los siguientes:
 - [StrongLoop Process Manager](#)
 - [PM2](#)
 - [Forever](#)

Gestor de procesos: forever.js

- Herramienta de interfaz de línea de mandatos simple que permite garantizar la ejecución continua (forever/siempre) de un determinado script
- Ideal para ejecutar los despliegues más pequeños de scripts y aplicaciones Node.js

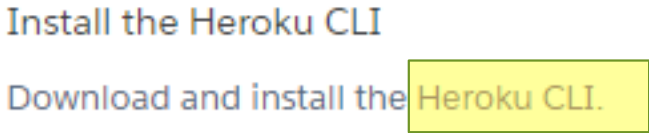

```
npm install forever -g
```

- Para iniciar un script:

```
$ forever start script.js
```

- Más información: <https://github.com/foreverjs/forever>.

Heroku: Instalación

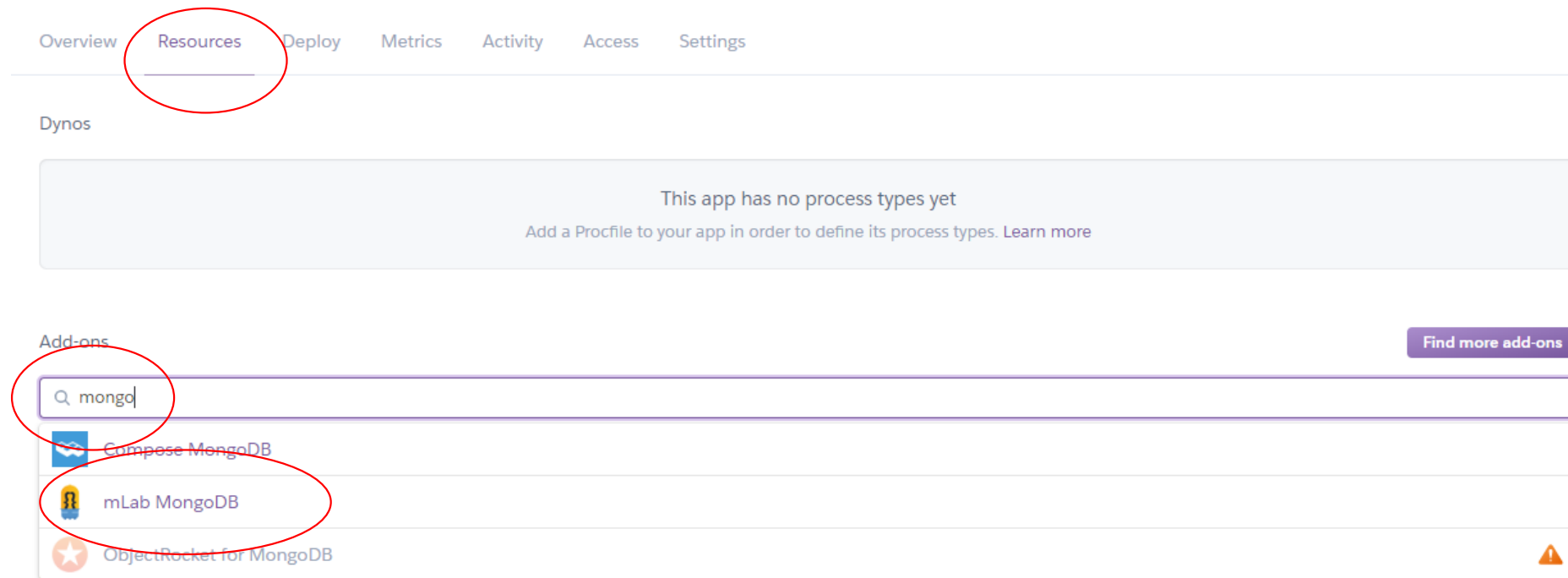
- Heroku es un servicio en la nube tipo PaaS (plataforma como servicio) donde se pueden **alojar y desplegar aplicaciones**
- Pasos para subir una aplicación a Heroku:
 1. Crear una cuenta en <https://signup.heroku.com/>
 2. Crear una nueva aplicación
 3. Clicar en  para instalar el cliente Heroku CLI:

 4. Comprobar la versión instalada en consola: `heroku -v`

Heroku: Subir aplicación

- Pasos para subir una aplicación a Heroku:
 5. Logarse desde el terminal: `$ heroku login`
 6. Crear un repositorio .git local (si no estaba creado) `$ git init`
 7. Crear un repositorio remoto git desde la ruta de la aplicación apuntando a la aplicación creada en heroku: `$ heroku git:remote -a api-rest-rgl`
 8. Subimos la aplicación al repositorio:
 1. Añadir código al repositorio `$ add .`
 2. Hacer commit de todo `$ git commit -m "versión ok"`
 3. Subir el código a heroku `$ git push heroku master`

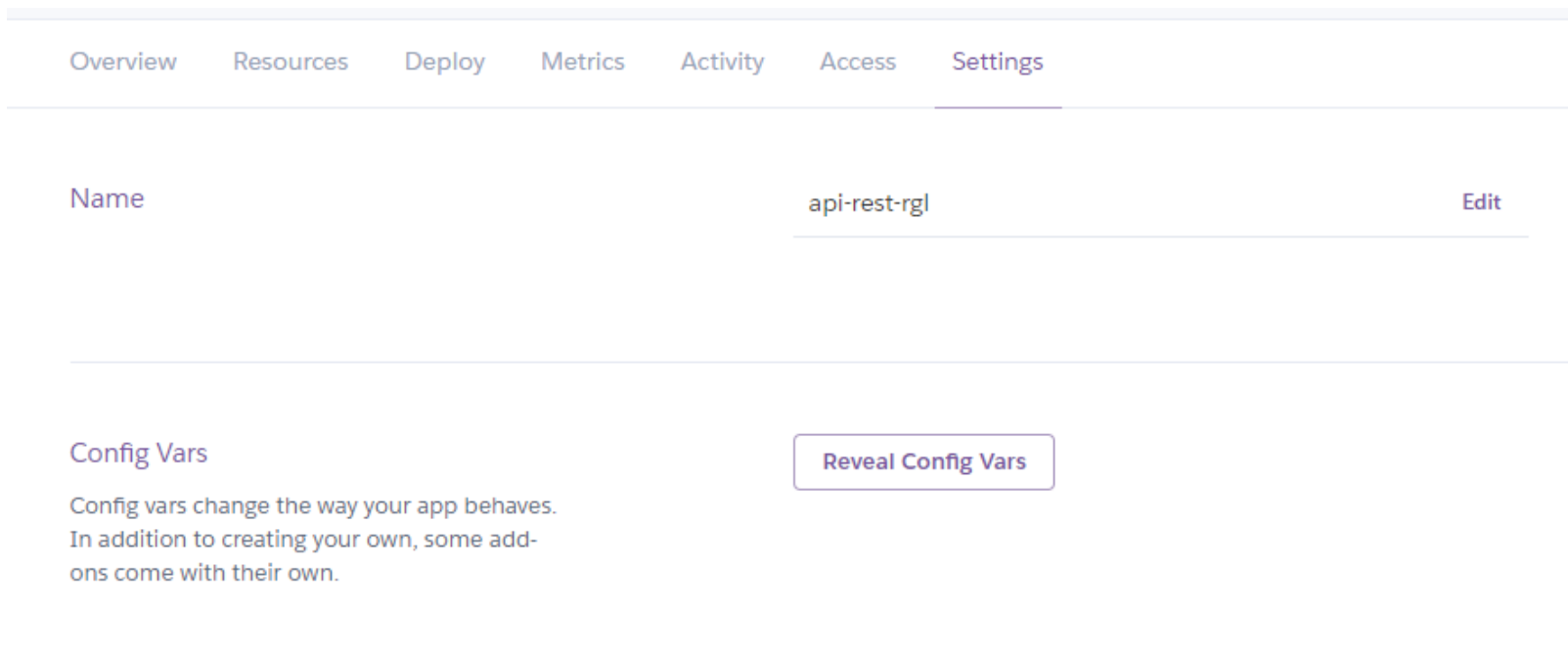
Heroku: Configurar aplicación

9. Añadir MongoDB:



Heroku: Subir aplicación

10. Configurar variables de entorno en Heroku:



The screenshot shows the Heroku Settings page for an application. At the top, there is a navigation bar with tabs: Overview, Resources, Deploy, Metrics, Activity, Access, and Settings. The Settings tab is selected. Below the navigation bar, there is a table with one row showing the environment variable 'api-rest-rgl' with an 'Edit' link. Below the table, there is a section titled 'Config Vars' with a 'Reveal Config Vars' button. The text below the button explains that config vars change the way the app behaves and that some add-ons come with their own config vars.

Overview	Resources	Deploy	Metrics	Activity	Access	Settings
----------	-----------	--------	---------	----------	--------	----------

Name	api-rest-rgl	Edit
------	--------------	------

Config Vars

Reveal Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.