SISTEMAS OPERATIVOS 1

CLASE 3

Leonel Aguilar

aux.leoaguilar@gmail.com

Sebastián Sánchez

sebastiantuchez0@gmail.com

Módulos y Docker

Nodejs, DockerFile

AGENDA DE LA CLASE

Módulos Kernel
¿Qué es un módulo? ¿Para qué se usa?

01

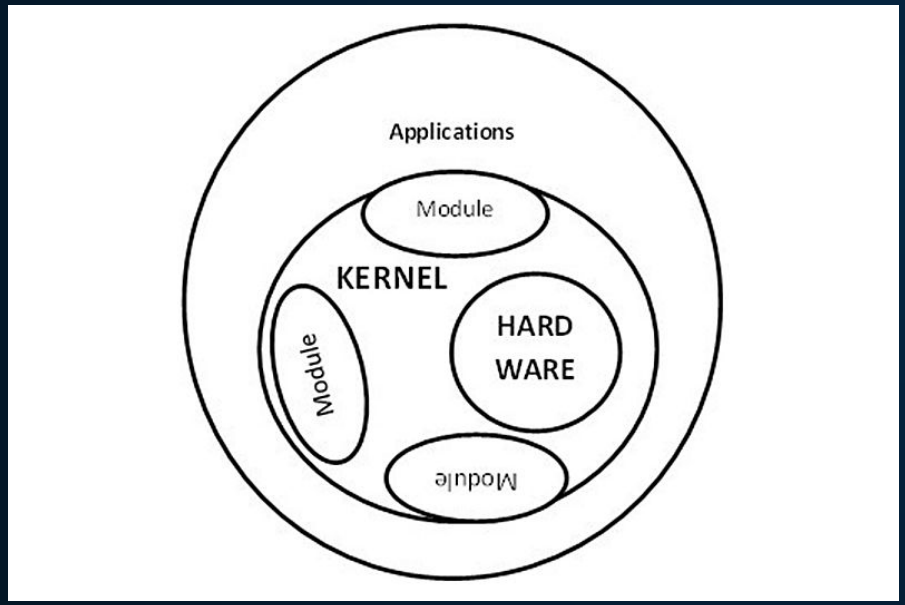
NodeJS, Docker
Express, Mongo Atlas, Docker Build, Docker Run

02

Ejemplo Práctico
Docker, Mongo Atlas, Node JS, Módulos

03

MÓDULOS KERNEL



- Código cargado a demanda
- Extender la funcionalidad del sistema
- El kernel tiene diseño modular

MÓDULOS KERNEL

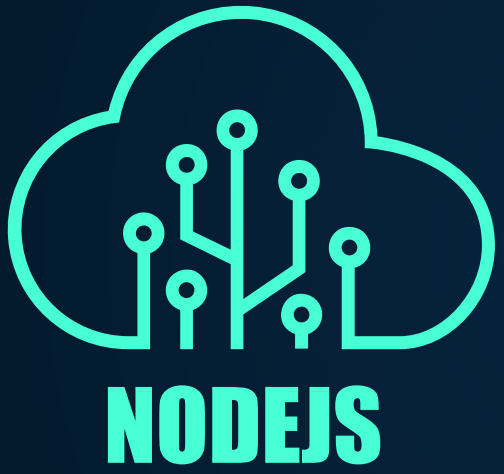
```c
// Librerías a cargar
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kernel.h>

MODULE_LICENSE("MODULO");
MODULE_AUTHOR("AUTOR");
MODULE_DESCRIPTION("DESCRIPCION");
MODULE_VERSION("VERSION");

// Definicion de evento principal
static int __init event_init (void) {

    // Codigo dentro del evento INIT

    // Retornar 0 si todo está bien
    // Retornar [num] como código de error
    return 0;
}

static void __exit event_exit(void) {

    // Código dentro del evento EXIT
}

// esta llamada carga la función que se ejecutará en el init
module_init(event_init);

// esta llamada carga la función que se ejecutará en el exit
module_exit(event_exit);
```

```
# Ver módulos almacenados
$ /lib/modules/[nombre_kernel]

# Ver la versión del kernel
$ uname -r

# Revisar módulos instalados, esto listará todos los módulos instalados en el
kernel
$ lsmod

# Revisamos o "buscamos" unicamente un módulo
$ lsmod | grep "[nombre_modulo]"
```

```
# Instalar el módulo con el comando insmod
# https://linux.die.net/man/8/insmod#:~:text=insmod%20is%20a%20trivial%20program,
is%20taken%20from%20standard%20input
# No se espera ninguna salida cuando este comando es exitoso.

$ sudo insmod [nombre_modulo].ko
```

```
# Revisar los logs de los modulos
$ sudo dmesg

# Ahora, revisamos el documento generado, cada vez que lo revisemos se reescribirá
$ cat /proc/[nombre_modulo]
```

```
# Eliminar el modulo
# No se espera ninguna salida cuando este comando es exitoso.

$ sudo rmmod "[nombre_modulo]"
```



¿QUÉ ES? ¿PARA QUÉ?

- Entorno de tiempo de ejecución
- Batteries-included
- Modelo de entrada y salida sin bloqueo controlado por eventos.

¿QUÉ USAREMOS?

EXPRESS

Express cómo framework para realizar nuestra API y app. web.

MONGOOSE

Utilizaremos este módulo de nodejs para conectarnos a una base de datos en Mongo ATLAS.

DOCKER

Nuestra API utilizará un contenedor de nodejs.

DOCKERFILE

- Crear imágenes
- Archivo de texto que contiene instrucciones

```
# Sintaxis básica de docker build

$ docker build [opciones] RUTA | URL | -

# En nuestro caso:
# Ir al directorio donde esta el Dockerfile
$ cd [PATH_AL_DOCKERFILE]

# Crear imagen utilizando build, notar el -t
# -t define un tag con el que nos referiremos a la imagen
$ docker build . -t node:14
```

DOCKER RUN

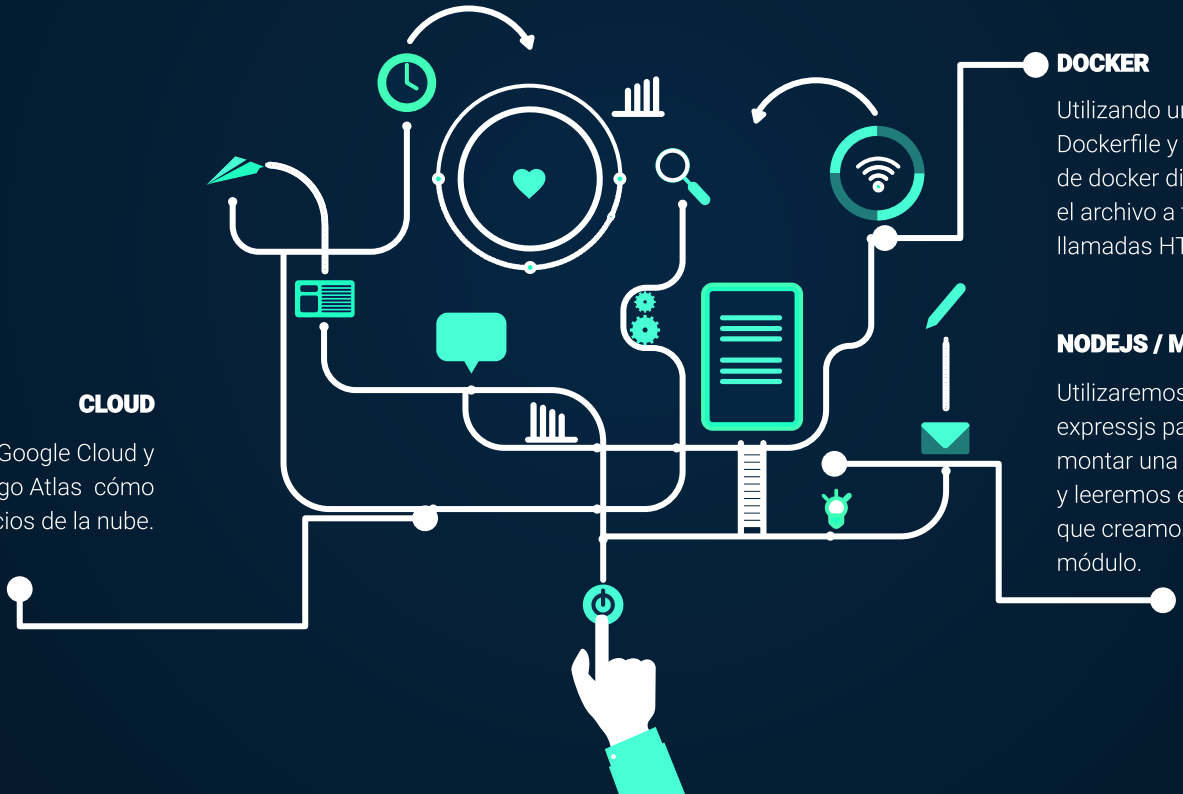
- Crear contenedor en base a una imagen
- Correr el contenedor

```
# Sintaxis básica de docker run
$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

# En nuestro caso:
# -dit dice a docker que haga detach (correr en segundo plano), y que el contenedor podra ser interactivo con una consola de tipo tty.
# --name le asigna un nombre al container
# -v copia al volumen del contenedor /elements/procs el contenido de la carpeta /procs en el sistema operativo
# -p dice que puerto exponer al mundo
# por ultimo ponemos el tag que le definimos a la imagen
$ docker run -dit --name node-api -v /proc/:/elements/procs/ -p 80:80 node:14
```

¿QUÉ HAREMOS?

Mostrar la hora del servidor en una página web, utilizando cloud, containers y módulos.

PROJECT OVERVIEW



DOCKER

Utilizando un archivo Dockerfile y un container de docker distribuiremos el archivo a través de llamadas HTTP.

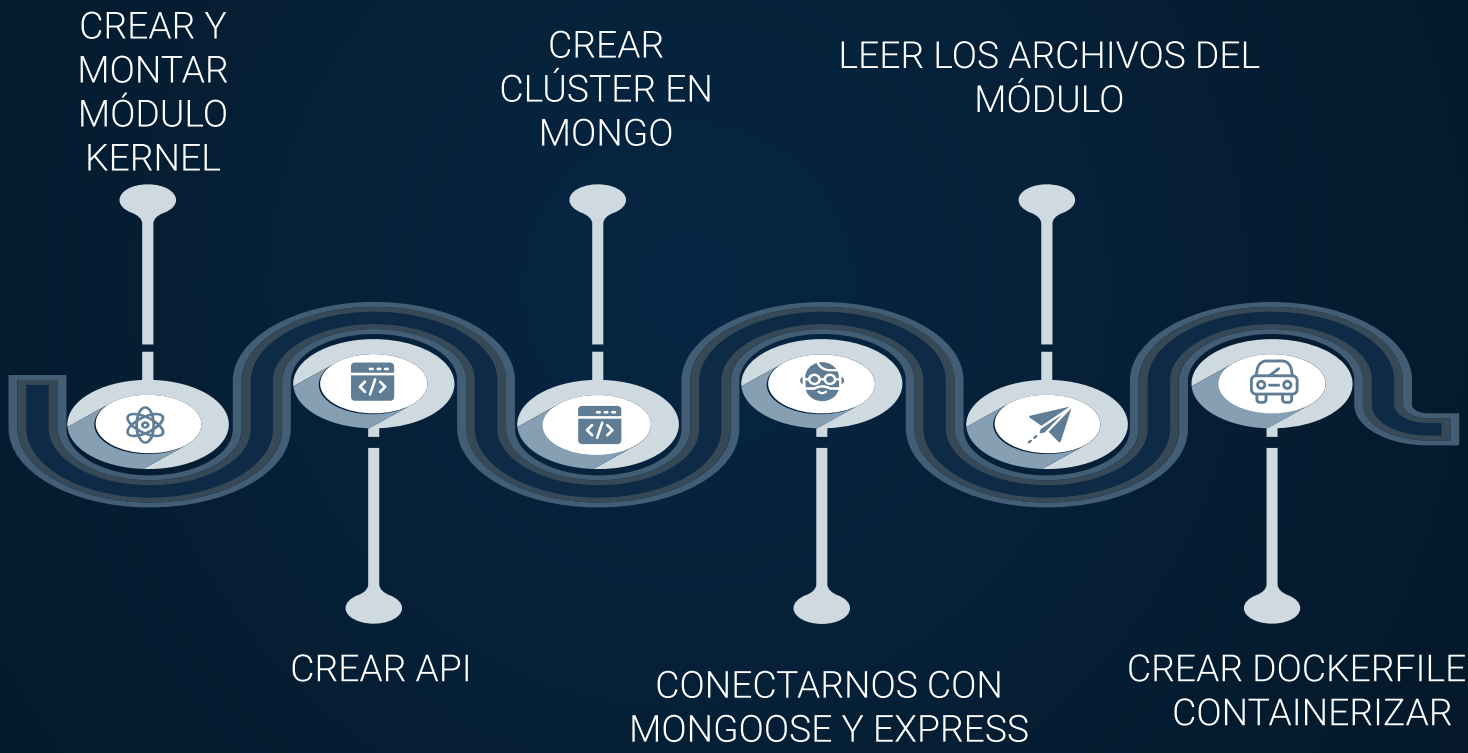
NODEJS / MODULO

Utilizaremos expressjs para montar una API rest, y leeremos el archivo que creamos con el módulo.

CLOUD

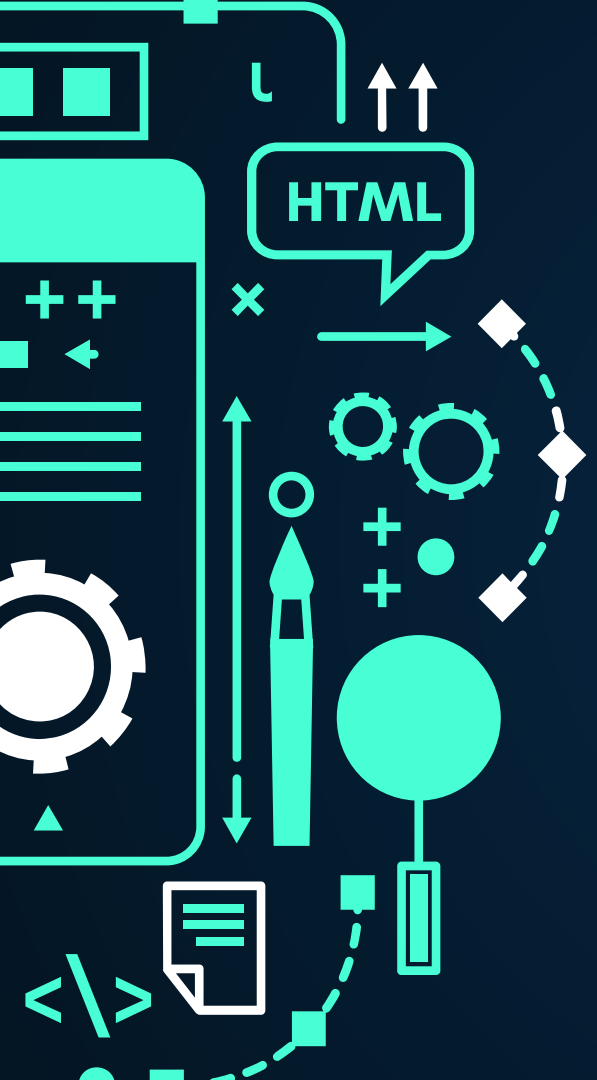
Utilizar Google Cloud y Mongo Atlas cómo servicios de la nube.

DETAILED STEPS



PARTE PRÁCTICA

https://github.com/leoaguilar97/so1-course/tree/clase-3/Tutoriales



¡GRACIAS!

¿PREGUNTAS?
¿COMENTARIOS?