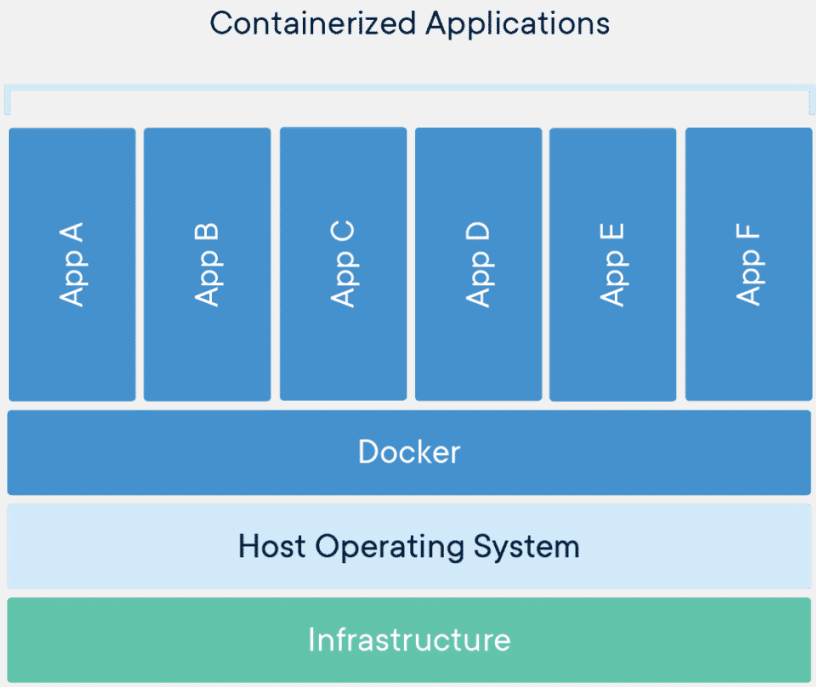
Table of Contents

Tools

Container

> A container is the standard unit of software that packages code and all dependencies so that the application works within another computing environment without depending on it. The most famous container is Docker who can create diferents apps as a show into the graph. More information



NoSQL Database

> A nosql database refers to the great variety of technologies developed to reduce cascading development, storage of large volumes of information, unstructured and polymorphic data, and many others. More Information

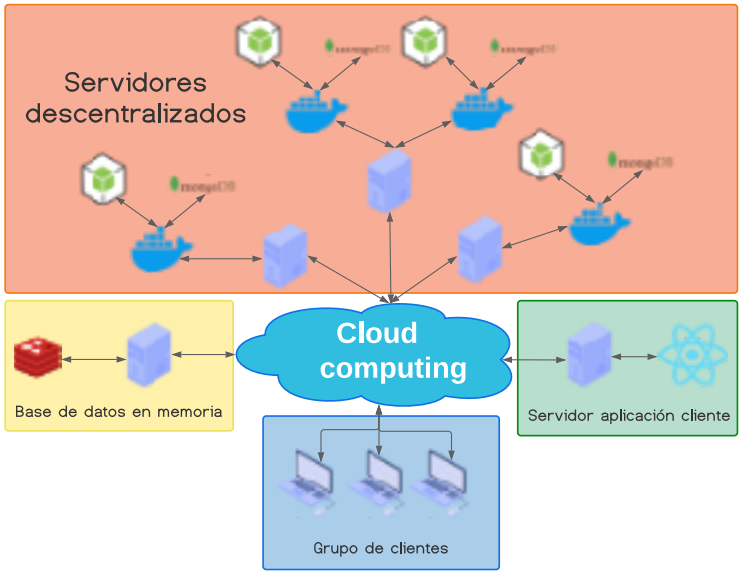
Express

> It is a framework used for web development, written in javascript and hosted within the NodeJS environment, it allows you to configure the environment and perform common tasks within web development and publishing. More Information

Inmemory Database

> In-memory databases are specially designed to rely primarily on memory rather than on disk or SSD data storage. Being ideal for applications where microsecond response times are required or have large peaks of information traffic. More Information

Architecture



Redis

> It is an in-memory data structure store used as a database, cache, message broker, and transmission engine. Including a publish and subscribe function. More Information

PubSub

> To manage decentralized systems it is very helpful to have subscriptions, unsubscriptions and publication. Decoupling of publishers and subscribers enables greater scalability with a more dynamic network topology. More Information

Mongo

> One of this is MongoDB, it has high availability, native horizontal scalability and has an image container into dockerHub. More Information

Installation

Create dockerfile

Dockerfile has an image name and version first structure.

```
FROM node:16
```

We declare the route where we are going to work inside the container

```
WORKDIR /usr/src/app
```

We copy the necessary files where the third-party libraries are to install the application

```
COPY package*.json ./
```

```
COPY yarn*.lock ./
```

We execute the installation command this depends on whether it is developed if it is yarn or npm

```
RUN yarn install
```

```
RUN npm install
```

We copy all the folders that we need to execute correctly if they are all we use colons

```
COPY . .
```

We place the port where the application is running

```
EXPOSE 3000
```

Finally, the command that will execute the application

```
CMD [ "node", "./built/index.js" ]
```

The file containing all the configuration

Once the dockerfile is ready, we execute the command inside the file path

```
docker build -t username/image:num-version .
```

Push imagen to dockerhub

After having the desired application ready, tested and with the latest updates, we write the following command which will upload the image to docker hub so that it can be used publicly

```
docker push username/image:num-version
```

Create docker-compose

Once the image is uploaded to the docker hub, we can download it to any computer using pull or using docker compose for the latter we have:

The name of the service that groups the containers

```
eca_chain_server
```

The name that we will give to the container

```
container_name: eca-chain
```

The name of the image uploaded to dockerHub with its respective version

```
image: dipaz/eca-chain-server:v1
```

Environment variables used in the container image

> environment:
>
> - MONGO_URI_1: mongodb://root:example@IpPcHost:27017/admin

The port where you want to run the desktop application and the port that runs inside the container

> ports:
>
> - "3000:3000"

The file containing all the configuration

Once the docker-compose.yml is ready, we execute the command inside the file path

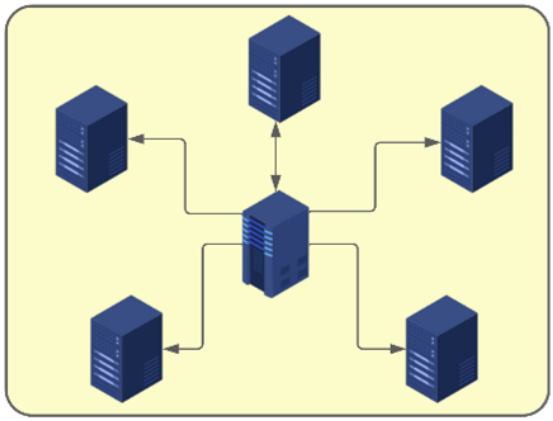
```
docker-compose up -D
```

(MVC) arquitecture

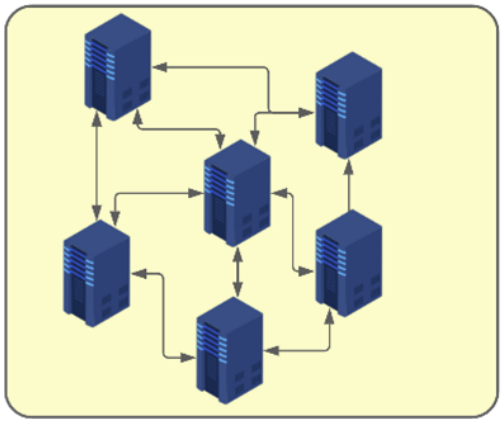
> - **Model** contains a representation of the data handled by the system, its business logic, and its persistence mechanisms.
> - **View** user interface, which composes the information that is sent to the client and the mechanisms of interaction with it.
> - **Controller** acts as an intermediary between the Model and the View, managing the flow of information between them and the transformations to adapt the data to the needs of each one. More Information

Decentralization

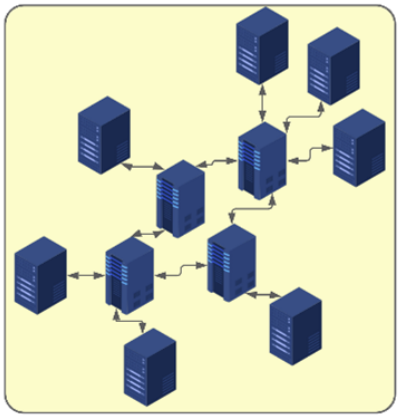
> Within the digital world, to solve latency and hardware overload, edge, fog and cloud computing are used. With edge devices for link nodes that aggregate and package data using cloud resources, obtaining the optimal form for information systems (Song et al., 2018), a novel design for decentralized storage systems is proposed, preserving privacy based on signatures and unique addresses. (Kopp y otros, 2017)



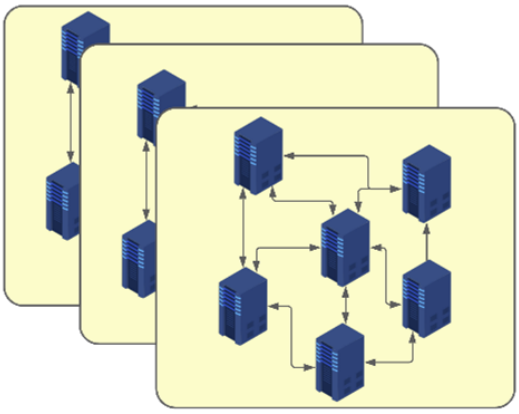
Centralizado



Completamente Descentralizado
(Distribuido)



Localmente Centralizado
(Descentralizado



Capas