



# Capítol 3: Deployment

Aina Palacios

# Aina Palacios

- Enginyera de Telecomunicacions especialitzada en Audiovisuals
- Màster en Tecnologies Avançades especialitzada en deep learning en Multimèdia!
- Experiència en programació web i machine learning.
- Mentora a IT Academy de **Vuejs**



<https://www.linkedin.com/in/ainapc/>

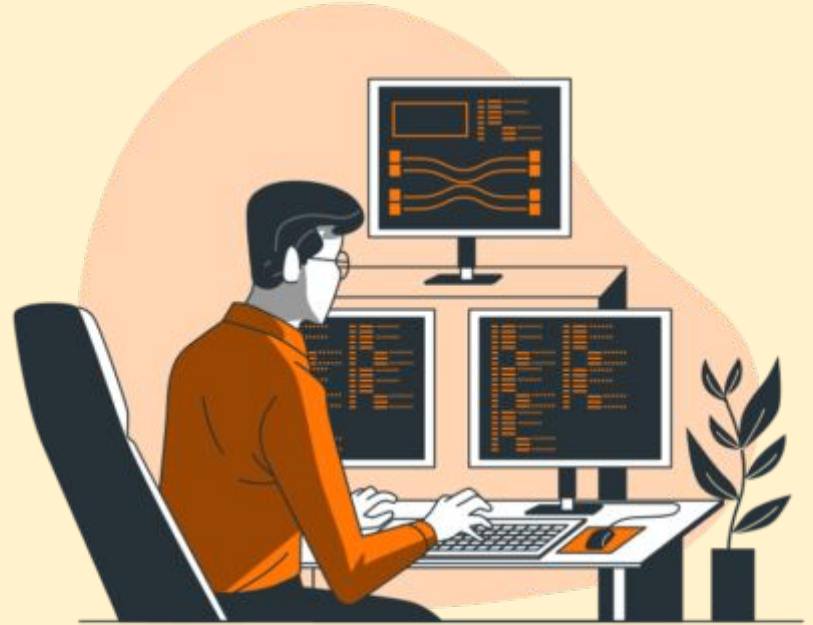


ainaPali#2617

# Deployment

En el context d'**administració de xarxes**, entenem per deployment el procés de preparar una estructura o sistema per estar llest per producció i habilitar el seu ús.

Dependrà del servei que li vulguem donar al nostre producte, on fer el deployment!



# Virtual Environment

Un virtual environment en Python és bàsicament un entorn d'interpretació de Python on hi ha totes les llibreries i script que romandran separades d'altres entorns o del teu sistema Python principal.

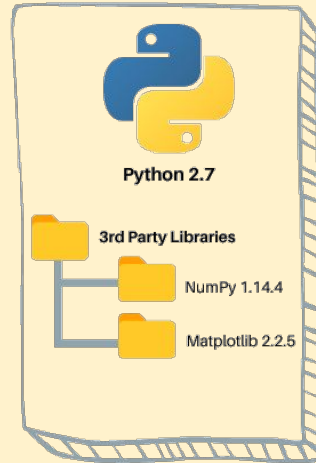
L'objectiu principal és crear **entorns individuals** pels nostres projectes! Així s'evita els errors per les diferents versions utilitzades.



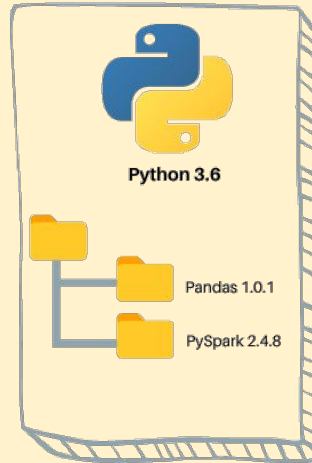
# Avantatges

- ❖ Pots utilitzar diferents Pythons version
- ❖ Queda aïllat de les teves altres llibreries
- ❖ Pots compartir fàcilment l'entorn per l'ús dels altres developers
- ❖ Pots instal·lar altres entorns fàcilment

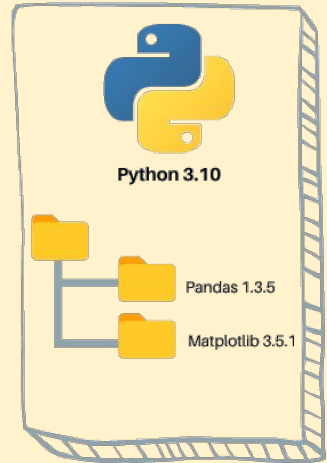
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



# Passos a seguir!

1. Instalar virtualenv
  - 1.1. **pip install virtualenv**
2. Crear virtualenv
  - 2.1. **python<version> -m venv <virtual-environment-name>**
3. Activa!
  - 3.1. **source env/bin/activate**
4. Comprova que funciona
  - 4.1. **pip list**
5. Instalar requirements
  - 5.1. **pip install -r requirements.txt**
6. Crea el teu requirements.txt!
  - 6.1. **pip freeze > requirements.txt**
7. Desactiva
  - 7.1. **deactivate**

```
mkdir projectA
cd projectA
python3.8 -m venv env
```

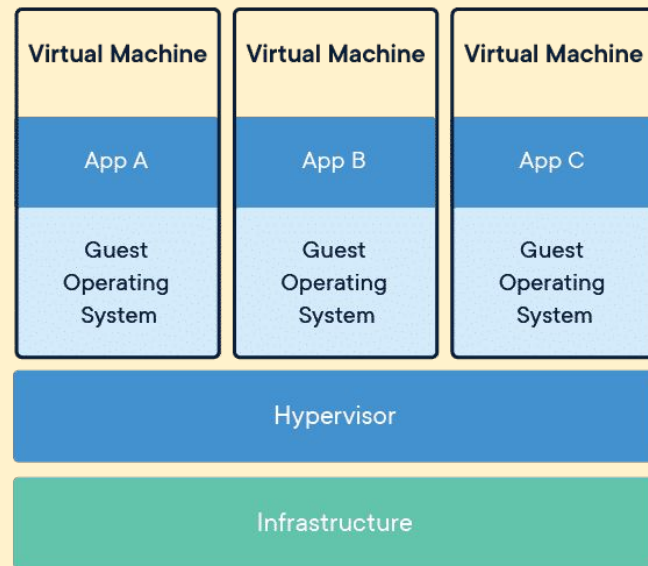
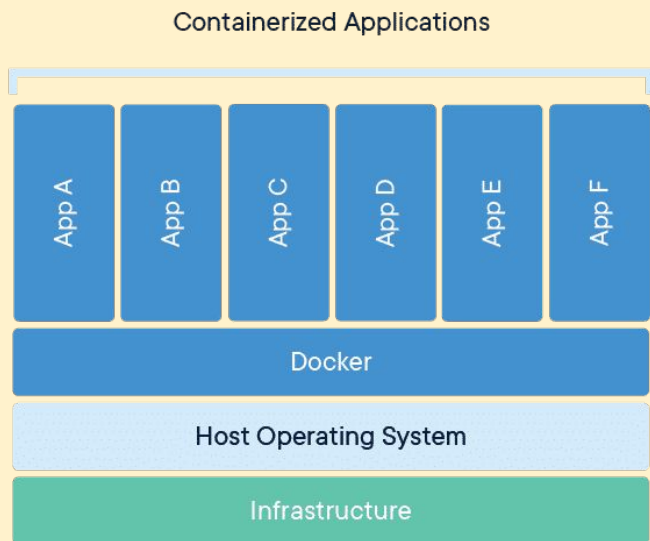
# Servidors

- Un servidor és un entorn que ens permet tenir el nostre entorn i programa en viu! És recomanable adjudicar manteniment del servidor a tercers, ja que aquest pot ser complicat.
- En el nostre cas utilitzarem un servidor gratuït per fer el deployment de la nostra API!



# Docker

- Un servidor és un espai de deployment, però imagina que tens molts programes amb diferents entorns, necessites separar aquests blocks. Per fer-ho, existeix el que se'n diu Docker!





# PostgreSQL

- Volem una base de dades a la nostra API per guardar informació! En aquest cas, farem ús de PostgreSQL, un sistema de bases de dades compatible amb moltes eines del marcat i de fàcil ús!



# Part 1: Tutorial de creació d'una API

Build your first REST  
API with Flask and  
PostgreSQL

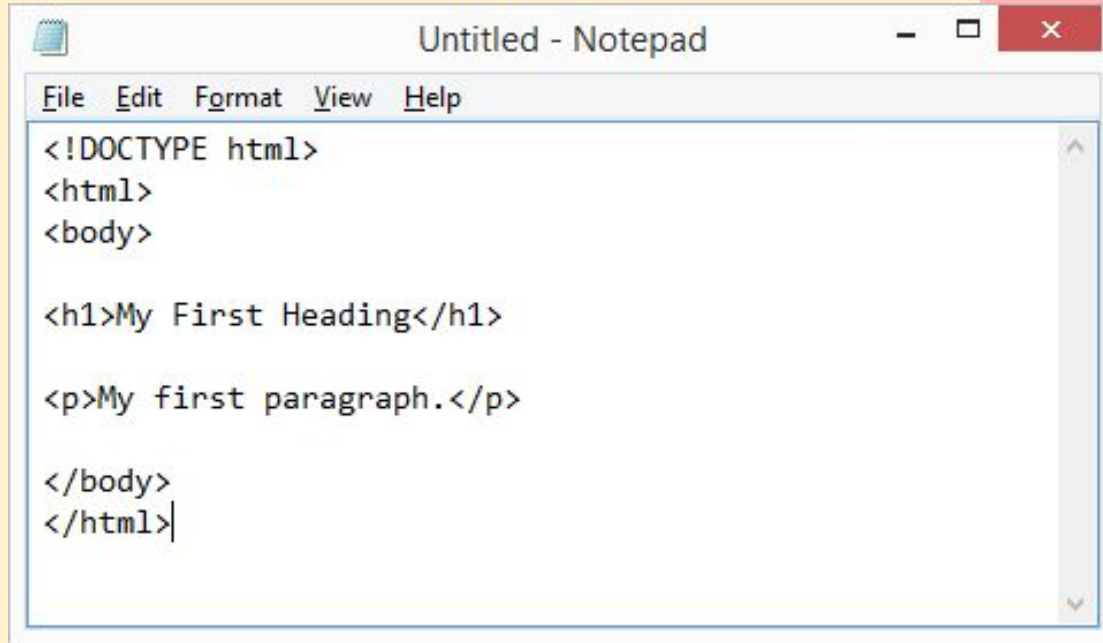
# Exercici 1

Ara prova de crear un JSON que retorni la predicció del teu model!

Pista: Obre un jupyter notebook i mira quines variables necessites  
Pista 2: Retorna un error en cas de no tenir l'objecte



# **Part 2:** **HTML**



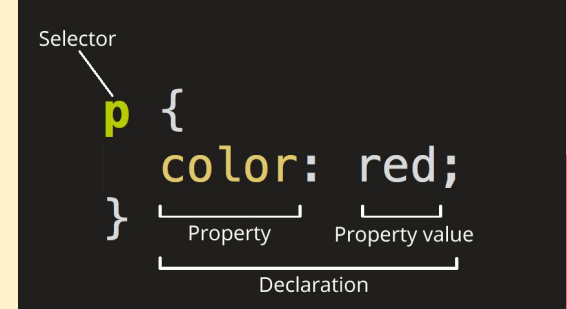
```
<!DOCTYPE html>
<html>
<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>
</html>
```

# HTML, CSS i Bootstrap!



# Exercici 2

Modifica l'HTML, crea un Títol i posa-li color!

Pista: Utilitza els tags i les classes!

# Tutorial Flask + HTML

Aprenem a crear un Formulari!

[Tutorial](#)

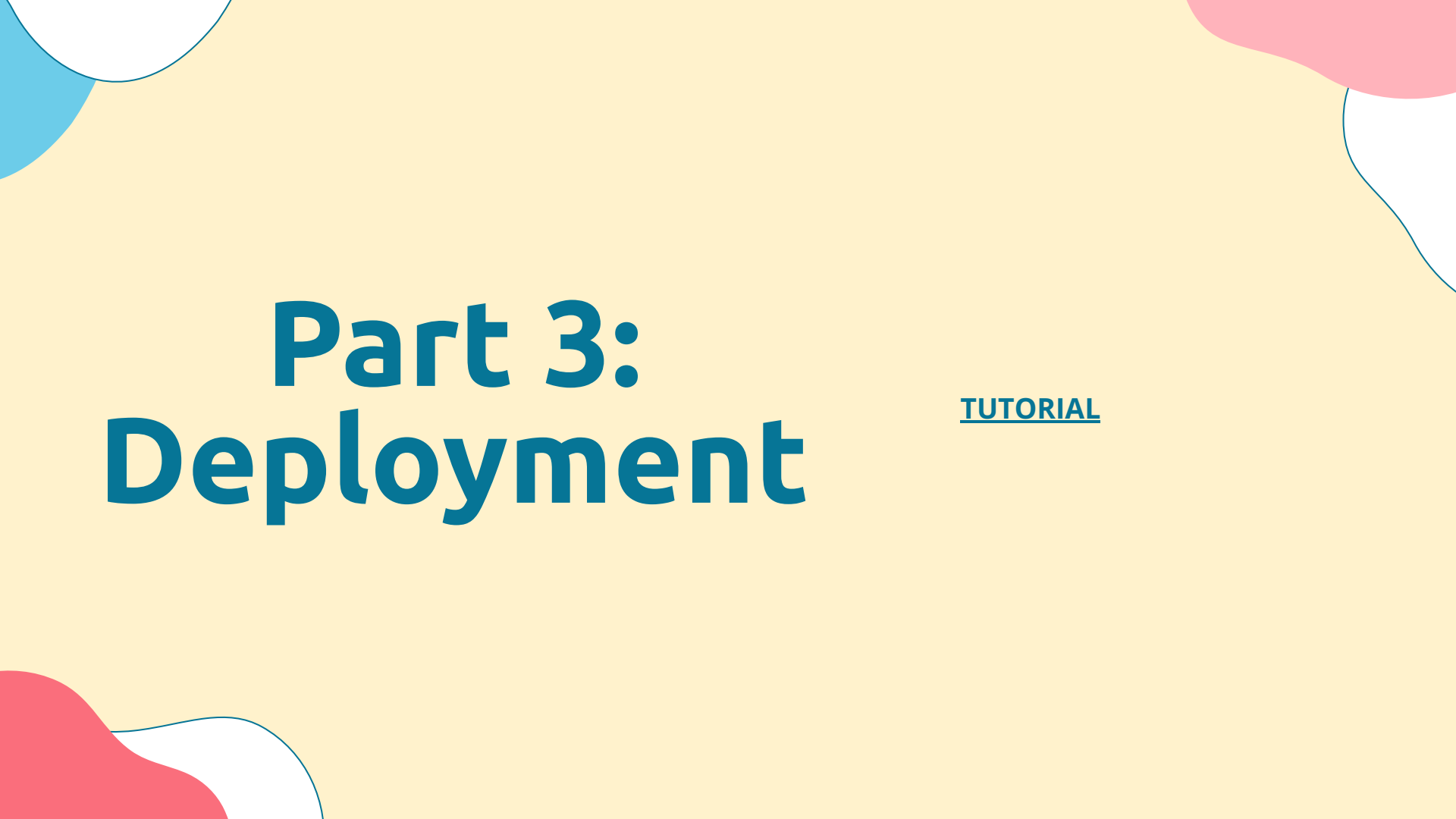
[Mega Tutorial](#)



# Flask

web development,  
one drop at a time

[Un altre tutorial Flask](#)  
[HTML dinàmic](#)



# Part 3: Deployment

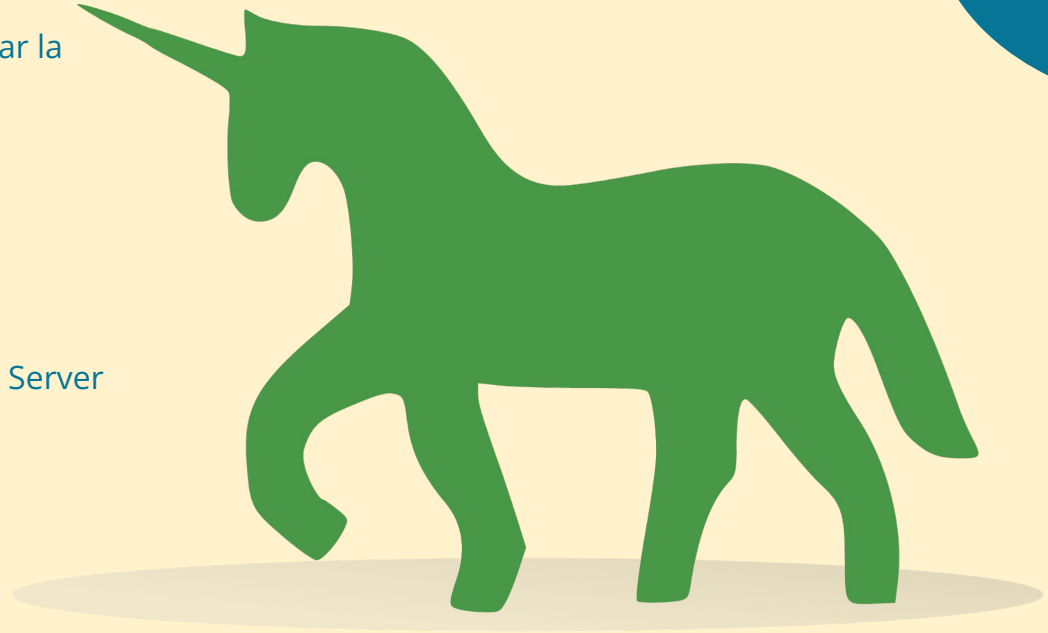
TUTORIAL



# IMPORTANT!

- No posar cap Key al GITHUB! -> Pots posar la Key directament al deployment
- Posar el Python version com a key!

Perquè utilitzar **Gunicorn** -> És un Python Web Server  
Getaway Interface HTTP server!





**Capítol 3  
acabat!**