Institut Puig Castellar
Santa Coloma de Gramenet

# Automatització amb Ansible de serveis amb dockers

**Projecte de desenvolupament**

CFGS Administració de Sistemes Informàtics i Xarxes

**Dylan Zafra Robledo**
**ASIX2B**

**Resum del projecte (màxim 250 paraules):**

**Temàtica del projecte**
Aquest projecte es basa en la gestió automatitzada de serveis en contenidors mitjançant Ansible i Docker. L'objectiu és crear diverses maquetes amb diferents serveis, provocar fallades intencionades i restaurar-los automàticament amb Ansible.

**Objectiu del projecte**
L'objectiu principal és desenvolupar un entorn d'infraestructura com a codi (IaC) que permeti desplegar, gestionar i recuperar serveis en contenidors de manera eficient i automatitzada. A més, es busca integrar aquesta infraestructura en un entorn de núvol per assegurar escalabilitat i alta disponibilitat.

**Metodologia seguida per a aconseguir l'objectiu**
Primer, es definiran els serveis i es desplegaran utilitzant Docker en diferents contenidors. A continuació, es configurarà Ansible per gestionar aquests serveis, definir rols i playbooks per automatitzar el seu desplegament i recuperació. Es provaran escenaris de fallada controlada i es validarà que Ansible sigui capaç de restaurar els serveis automàticament. Finalment, es traslladarà aquesta infraestructura a un proveïdor de núvol per garantir la seva disponibilitat.

**Resum de les conclusions**
Aquest projecte demostrarà com Ansible i Docker poden treballar conjuntament per gestionar serveis de manera eficient. La implementació en el núvol permetrà una major resiliència, automatització i optimització dels recursos.

**Paraules clau (entre 4 i 8):**

Ansible, automatització, serveis, dockers, fallada

**Abstract (in English, 250 words or less):**

**Project Theme**
This project focuses on the automated management of containerized services using Ansible and Docker. The goal is to create multiple mock environments with different services, intentionally cause failures, and restore them automatically using Ansible.

**Project Objective**
The main objective is to develop an Infrastructure as Code (IaC) environment that allows for efficient deployment, management, and recovery of containerized services in an automated way. Additionally, the project aims to integrate this infrastructure into a cloud environment to ensure scalability and high availability.

**Methodology to Achieve the Objective**
First, the services will be defined and deployed using Docker in different

containers. Then, Ansible will be configured to manage these services, defining roles and playbooks to automate deployment and recovery. Controlled failure scenarios will be tested to validate Ansible's ability to restore services automatically. Finally, this infrastructure will be migrated to a cloud provider to ensure its availability.

**Summary of Conclusions**
This project will demonstrate how Ansible and Docker can work together to efficiently manage services. The implementation in the cloud will enable greater resilience, automation, and resource optimization.

**Keywords (entre 4 i 8):**

Ansible, automatization, services, dockers, errors

# Índex

## Llista de figures

# 1 Introducció

Descripció general del projecte

Aquest projecte té com a finalitat desplegar una infraestructura automatitzada per gestionar serveis contenitzats mitjançant Ansible i Docker. L'objectiu és crear un entorn on diversos serveis s'executin dins de contenidors i sotmetre'ls a fallades intencionades per avaluar la capacitat d'Ansible de restaurar-los automàticament. A més, tot aquest sistema es desplegarà en el núvol per garantir escalabilitat i alta disponibilitat.

Objectius principals

Aquest projecte busca:

- Automatitzar la gestió i el desplegament de serveis contenitzats.
- Simular fallades per validar la resiliència del sistema.
- Implementar mecanismes de recuperació automàtica amb Ansible.
- Migrar la infraestructura al núvol per reforçar-ne la robustesa.

Metodologia

Per assolir aquests objectius, es seguirà el procés següent:

- **Definició dels serveis** → Es determinaran quins serveis són necessaris i es desplegaran en contenidors Docker.
- **Configuració d'Ansible** → Es desenvoluparan rols i playbooks per gestionar-los eficientment.
- **Simulació de fallades** → Es provocaran errors controlats per analitzar el temps i l'eficàcia de la seva recuperació.
- **Desplegament al núvol** → Es garantirà que la infraestructura sigui òptima per operar en un proveïdor de núvol.

## 1.1 Context

L'automatització i la contenització s'han convertit en elements clau dins del sector de la informàtica i l'administració de sistemes. Cada cop més empreses recorren a Docker per desplegar aplicacions de manera eficient i escalable, mentre que eines com Ansible faciliten la gestió i automatització de la configuració d'infraestructures de forma àgil i fiable.

Tot i això, molts professionals que volen especialitzar-se en aquest àmbit es troben amb una dificultat: adquirir experiència pràctica en la implementació d'aquestes tecnologies en entorns reals. D'aquí neix aquest projecte, que busca proporcionar un escenari pràctic i controlat per treballar amb Ansible i Docker, reproduint situacions habituals en el món laboral, com el desplegament de serveis, la gestió d'errors i la recuperació automàtica.

## 1.2 Justificació

Aquest projecte té com a objectiu aprendre i aplicar Ansible i Docker per a la gestió automatitzada d'infraestructures TI. Es desenvoluparan diverses maquetes amb serveis contenitzats, els quals es trencaran intencionadament per testar mecanismes automàtics de recuperació amb Ansible. Això permetrà entendre millor la resolució de problemes i l'administració eficient de sistemes.

A més, els serveis es desplegaran en el núvol, afegint un component d'escalabilitat i seguretat. Aquest enfocament pràctic busca simular escenaris reals del món laboral, millorant l'adaptabilitat i l'experiència tècnica.

## 1.3 Objectius

Un cop finalitzat el projecte, es pretén adquirir competències pràctiques en l'ús d'Ansible i Docker per a la gestió i automatització de sistemes. Es vol assolir una capacitat per desplegar, gestionar i recuperar serveis contenitzats en entorns de producció, incloent-hi la seva configuració en el núvol. Es simulen escenaris on hi hagi caigudes dels serveis i resoldre els problemes.

### 1.3.1 Objectiu general

De forma general, es pretén aconseguir un domini sòlid de les eines Ansible i Docker per a l'automatització de tasques, la gestió eficient de contenidors i la creació d'entorns de treball en el núvol. L'objectiu és adquirir habilitats pràctiques per desplegar, monitoritzar i mantenir serveis de manera automàtica, millorant així l'eficiència i l'escalabilitat dels sistemes.

### 1.3.2 Objectius específics

**Aprendre a configurar i utilitzar Docker**: Entendre la creació, gestió i execució de contenidors, així com la seva configuració i orquestració.

**Adquirir coneixements sobre Ansible**: Aprendre a automatitzar tasques d'administració de sistemes i gestionar configuracions de manera eficient amb Ansible.

**Desplegar serveis utilitzant Docker i Ansible**: Crear entorns de treball automatitzats amb Docker i gestionar-los de manera automàtica amb Ansible.

**Simular fallades i reacció automàtica**: Desenvolupar i provar processos d'automatització per a la recuperació de serveis en cas de fallada, millorant la resiliència dels entorns.

**Gestionar infraestructures al núvol**: Almacenar, configurar i gestionar serveis i contenidors a la nube per tal de millorar la disponibilitat i escalabilitat dels entorns.

**Monitoritzar els serveis i contenidors**: Aprendre a monitoritzar serveis i contenidors, identificant fallades i optimitzant el rendiment dels sistemes.

**Garantir la seguretat dels entorns creats**: Implementar pràctiques de seguretat en la creació i gestió de contenidors i serveis per tal d'assegurar la protecció de les dades i la

infraestructura.

**Obtenir una experiència pràctica real**: Desenvolupar projectes pràctics on es pugui aplicar el coneixement adquirit de Docker i Ansible en un entorn controlat.

**Consolidar el coneixement per a l'àmbit laboral**: Preparar-se per afrontar reptes laborals en l'automatització de sistemes i gestió d'infraestructures amb eines modernes.

## 1.4 Estratègia i planificació del projecte

### Estratègies per dur a terme el treball

**Desenvolupar un producte nou**
Aquesta estratègia suposa la creació d'un sistema totalment nou emprant Ansible i Docker des de zero. Ofereix un alt grau de personalització, però també implica una inversió significativa de temps i recursos, ja que caldria construir un entorn complet, integrar diversos serveis i assegurar-ne el funcionament conjunt.

**Adaptar un producte existent**
En comptes de començar des de zero, aquesta opció aposta per reutilitzar una infraestructura ja existent, adaptant-la a les necessitats específiques del projecte. Aquesta via permet aprofitar configuracions prèvies i entorns ja funcionals, optimitzant-los per incorporar els serveis requerits.

**Integrar eines i serveis per a la seva gestió**
Aquesta alternativa es basa en la incorporació de Docker i Ansible dins d'un entorn de treball que ja està operatiu. L'objectiu és aconseguir una gestió automatitzada dels serveis, cosa que afavoreix l'automatització de processos i la millora del rendiment sense haver de construir la infraestructura des del principi.

### Estratègia triada

S'ha optat per l'estratègia d'**integrar eines i serveis per a la seva gestió**, ja que es considera la més adequada per assolir els objectius del projecte per diversos motius:

**Estalvi de temps i recursos**
En lloc de desenvolupar un sistema des de zero, la integració d'eines com Docker i Ansible en un entorn existent permet accelerar el desplegament i posar el focus en la configuració i automatització dels serveis, que és el nucli del projecte.

**Viabilitat tècnica**
L'ús d'eines àmpliament adoptades com Docker i Ansible, amb una extensa documentació i comunitats actives, facilita tant la implementació com l'aprenentatge. A més, l'automatització de la infraestructura ja és una pràctica consolidada, cosa que en simplifica la integració.

**Sostenibilitat**
Integrar noves eines dins un sistema funcional permet desenvolupar una solució escalable i fàcil de mantenir, amb capacitat per adaptar-se a futures necessitats i millores.

**Objectius laborals**
Aquesta estratègia aporta una experiència pràctica amb tecnologies molt presents al món professional. A més, dona com a resultat un producte final sòlid i preparat per ser utilitzat en contextos reals.

# 1.5 Metodologia de treball

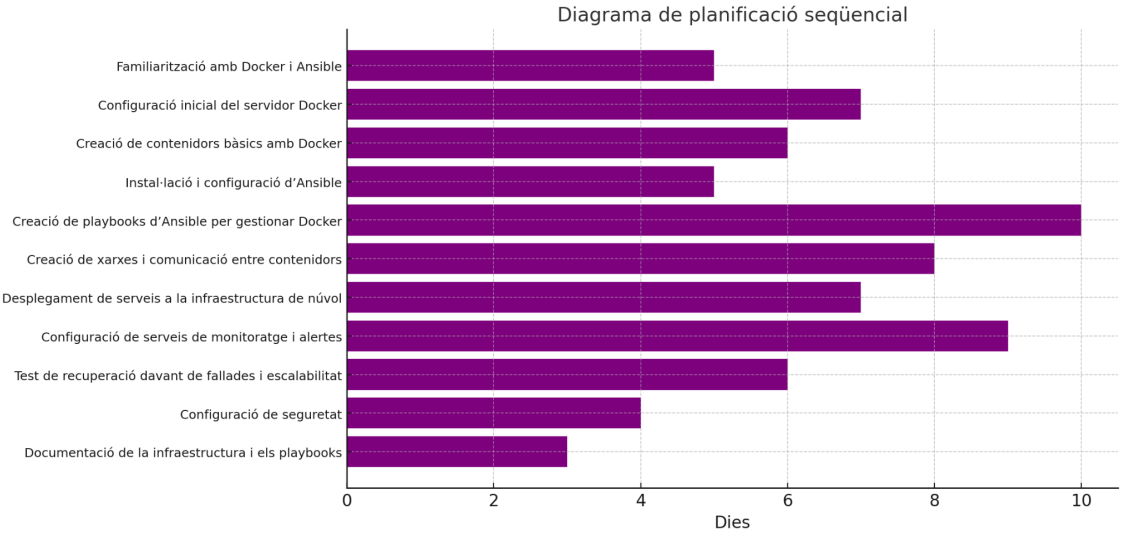
### Metodologia de treball

Per al desenvolupament d'aquest projecte, es farà servir una metodologia **agile**, concretament **Scrum**, ja que ofereix una gestió flexible i iterativa del treball. Aquest enfocament permet adaptar-se a canvis i millores contínues, fet especialment útil en projectes d'infraestructura i automatització com aquest. Mitjançant la divisió en **sprints** curts, es garanteix un seguiment constant del progrés i una millora progressiva de les solucions implementades.

### Eines de seguiment

Per tal de controlar i fer el seguiment del projecte, s'utilitzaran diverses eines:

- **Diagrama de Gantt**: Per a la planificació visual de les tasques i la definició de les seves dependències.
- **Trello**: Plataforma de gestió de tasques que permet monitoritzar l'estat de cada sprint.
- **GitHub**: Utilitzat per al control de versions del codi i la col·laboració en els **playbooks** d'Ansible i les configuracions de Docker.
- **Monitoratge amb Prometheus i Grafana**: Fonamental per analitzar i avaluar el rendiment de la infraestructura desplegada.




# 1.6 Estudi econòmic i pressupostari

### Instal·lació i configuració d'Ansible

**Tasques principals:**

- Instal·lar Ansible tant en entorns locals com en servidors remots.

- Definir i ajustar els fitxers d'inventari per a una gestió eficient dels nodes.
- Desenvolupar playbooks orientats a automatitzar processos com la instal·lació de Docker i altres serveis relacionats.

### Requisits i materials:

- Accés a màquines virtuals o físiques per implementar la configuració.
- Connexió a internet per descarregar paquets i dependències necessàries.

**Cost estimat:**
L'eina Ansible no té cost, però el temps destinat a la seva configuració pot variar segons el grau d'experiència de l'equip tècnic.

## Instal·lació i configuració de Docker

### Tasques principals:

- Instal·lar Docker en els sistemes físics o virtuals per executar aplicacions en contenidors.
- Configurar les xarxes Docker per assegurar una comunicació fluida entre contenidors.
- Realitzar proves pràctiques de creació i execució de contenidors.

### Requisits i materials:

- Equips amb capacitat suficient per fer funcionar Docker sense problemes.
- Un sistema operatiu compatible, com Ubuntu o Debian.

### Cost estimat:

Docker és una solució gratuïta, però els recursos de maquinari (CPU, memòria i disc) representen un cost a considerar.

## Desenvolupament de playbooks per a la gestió automatitzada de serveis

### Tasques principals:

- Crear playbooks amb Ansible per controlar serveis dins dels contenidors (com ara DNS o DHCP).
- Automatitzar l'execució i la recuperació dels serveis en cas d'errors.
- Establir sistemes de supervisió i alertes per garantir-ne el funcionament continu.

### Requisits i materials:

- Ús d'eines com Nagios, Prometheus o Grafana per al seguiment dels contenidors.

- Connexió a internet per accedir a actualitzacions i recursos externs.

### Cost estimat:
Es requereix una inversió en hores de desenvolupament per configurar i validar els serveis automatitzats.

## Configuració de la infraestructura de xarxa

### Tasques principals:

- Dissenyar xarxes internes amb Docker i establir la comunicació entre contenidors.
- Assignar IPs fixes als contenidors.
- Implementar serveis essencials com DNS, DHCP i altres.

### Requisits i materials:

- Routers o switches virtuals per gestionar la xarxa interna de manera eficient.

### Cost estimat:
Pot implicar despeses en maquinari físic o llicències per a màquines virtuals, segons l'entorn utilitzat.

## Proves de fallades i recuperació

### Tasques principals:

- Simular caigudes de contenidors i verificar que els playbooks d'Ansible els restaurin automàticament.
- Configurar sistemes de còpia de seguretat per contenidors i dades associades.

### Requisits i materials:

- Espai d'emmagatzematge suficient per allotjar les còpies de seguretat, localment o en xarxa.

### Cost estimat:
És possible que es requereixi espai d'emmagatzematge addicional si no es disposa de recursos prèviament assignats.

## Almacenament al núvol

### Tasques principals:

- Integrar serveis al núvol per a l'emmagatzematge i gestió dels contenidors (com AWS, Google Cloud o Azure).

- Configurar la sincronització automàtica i la pujada de configuracions als serveis en línia.

### Requisits i materials:

- Cal disposar d'una subscripció activa a algun servei de núvol.

### Cost estimat:

Tot i que molts proveïdors ofereixen una capa gratuïta, els costos finals dependran de l'ús real de recursos com l'espai, la CPU o el trànsit.

## Costos de desenvolupament i manteniment

### Costos inicials:

- Subscripció a plataformes de núvol o adquisició de maquinari, si escau.
- Consum energètic associat al funcionament dels servidors o màquines virtuals.
- Temps de dedicació de l'equip per a la configuració completa d'Ansible, Docker i serveis associats.

### Costos de manteniment:

- Monitoratge constant de serveis i contenidors desplegats.
- Actualitzacions periòdiques d'eines i sistemes (Ansible, Docker, etc.).
- Gestió i manteniment de plataformes de supervisió com Grafana o Prometheus.
- Despeses recurrents de la infraestructura en el núvol, si s'utilitza.

### Costos laborals:

- L'equip de desenvolupament suposarà un cost segons les hores dedicades a la implementació i proves.
- També pot ser necessari comptar amb professionals en xarxes o administració de sistemes.

### Altres costos addicionals:

- Llicències de programari, en cas d'utilitzar eines comercials per a la supervisió o la gestió.

# 2 Descripció del projecte

## 2.1.1 Anàlisi de requisits [projecte d'implementació]

### Requisits de funcionalitat

#### Desplegament automàtic de contenidors

Cal que Ansible permeti desplegar, gestionar i configurar contenidors Docker de manera automàtica en diferents entorns, ja siguin màquines físiques, virtuals o núvol.

Exemple d'ús: Un usuari hauria de poder executar un playbook d'Ansible que instal·li automàticament un conjunt de serveis com bases de dades dins de contenidors.

#### Automatització en la gestió de serveis

Els serveis allotjats dins els contenidors (servidors web) han de poder ser configurats automàticament mitjançant Ansible.

Exemple d'ús: Si un contenidor deixa de funcionar, el sistema hauria de reiniciar-lo de forma automàtica sense requerir accions manuals.

#### Monitoratge i alertes automàtiques

És imprescindible integrar solucions de monitoratge (com Prometheus o Grafana) que permetin controlar l'estat tant dels serveis com dels contenidors, generant alertes immediates en cas de fallades.

Exemple d'ús: Si un contenidor cau o no respon, el sistema ha de notificar l'incident i activar els mecanismes de recuperació.

#### Recuperació automàtica de fallades

Els contenidors i serveis han de disposar de mecanismes per reiniciar-se i restaurar la seva configuració original en cas d'errors.

Exemple d'ús: Davant una caiguda, el contenidor s'ha de reiniciar i tornar al seu estat inicial sense intervenció externa.

#### Execució en entorns de núvol

Els contenidors han de poder operar i ser gestionats en plataformes cloud com AWS, GCP o Azure.

Exemple d'ús: L'usuari ha de poder pujar una imatge Docker a una instància al núvol i allotjar-la remotament.

### Requisits de robustesa

#### Alta disponibilitat (HA)

El sistema ha de garantir el funcionament continuat dels serveis, amb capacitat de failover per traslladar-los automàticament en cas de fallada del maquinari o del programari.

Exemple d'ús: Si un servidor falla, el sistema hauria de migrar els serveis a una màquina operativa sense perdre disponibilitat.

#### Escalabilitat

La infraestructura ha de permetre afegir nous contenidors o màquines de manera senzilla per adaptar-se a increments de càrrega.

Exemple d'ús: Si el volum de feina augmenta, el sistema ha de desplegar més instàncies de servei de forma automàtica per mantenir el rendiment.

### Ús eficient dels recursos

Els contenidors s'han de configurar per aprofitar els recursos disponibles de manera òptima, evitant sobrecàrregues del sistema.

Exemple d'ús: Els contenidors han d'adaptar-se a la càrrega de treball, optimitzant l'ús de CPU, RAM i disc.

## Requisits de seguretat

### Autenticació i autorització

Tots els serveis han d'estar protegits per mecanismes que garanteixin l'accés exclusiu a usuaris autoritzats.

Exemple d'ús: Un administrador ha d'identificar-se correctament abans de fer canvis en la configuració dels serveis o contenidors.

### Xifratge de dades

És necessari xifrar tant les dades emmagatzemades com les comunicacions entre serveis i contenidors per evitar vulnerabilitats.

Exemple d'ús: Quan un contenidor transmet informació confidencial, aquesta ha de ser xifrada abans d'enviar-se.

### Auditoria i registre de seguretat

El sistema ha de mantenir un registre detallat de totes les accions i esdeveniments rellevants per garantir la traçabilitat.

Exemple d'ús: Els administradors han de poder revisar els logs per saber qui ha accedit als serveis o ha realitzat modificacions.

## Requisits de facilitat d'ús

### Playbooks comprensibles

Els playbooks d'Ansible han de ser clars, ben documentats i fàcils d'executar, fins i tot per usuaris amb poca experiència.

Exemple d'ús: Amb unes instruccions bàsiques, qualsevol usuari ha de poder desplegar serveis sense dificultat.

### Documentació detallada

Cal proporcionar guies completes sobre la configuració, ús i resolució d'incidències del sistema.

Exemple d'ús: Un usuari pot seguir una guia pas a pas per afegir nous serveis o solucionar errors habituals.

## Requisits previs mínims del sistema

### Maquinari i màquines virtuals

Cada contenidor ha de disposar com a mínim de 2 CPU, 4 GB de RAM i 20 GB de disc. A més, la connexió de xarxa ha de ser estable.

### Emmagatzematge

És necessari disposar d'espai suficient per imatges Docker, registres i còpies de seguretat.

### Xarxa i serveis associats

Cal una connexió a Internet per descarregar imatges i actualitzacions. També es requereixen xarxes virtuals per a la comunicació interna i serveis com DHCP i DNS per a la gestió d'IP.

### Sistema operatiu

Són compatibles distribucions Linux com Ubuntu 20.04 o CentOS. Docker i Ansible han d'estar instal·lats i funcionals.

### Aplicacions necessàries

- Docker: instal·lat i llest per executar contenidors.
- Ansible: configurat per automatitzar processos.
- Monitoratge: eines com Prometheus o Grafana per controlar l'activitat dels serveis.

**Seguretat de xarxa**

El sistema ha de disposar d'un firewall adequadament configurat i mantenir totes les aplicacions i sistemes actualitzats per prevenir vulnerabilitats conegudes.

# 2.2 Tecnologies

## 2.2.1 Comparativa de les tecnologies valorades

| Tecnologia | Descripció | Avantatges | Inconvenients |
|---|---|---|---|
| **Ansible** | Eina d'automatització de configuració i desplegament d'infraestructures basada en YAML i SSH. | - No necessita agents en els servidors.<br>- Fàcil d'aprendre i utilitzar.<br>- Integració amb múltiples sistemes. | - Execució seqüencial pot ser lenta.<br>- No apte per a tasques molt complexes comparat amb Puppet o Chef. |
| **Docker** | Plataforma de virtualització basada en contenidors lleugers i portables. | - Lleuger i eficient.<br>- Facilita el desplegament i escalabilitat.<br>- Portabilitat entre sistemes. | - Pot requerir coneixements avançats per optimitzar imatges.<br>- Manca d'una gestió nativa d'estat persistent. |
| **Puppet** | Eina de gestió de configuració basada en declaracions. | - Ideal per a grans infraestructures.<br>- Escalabilitat i control granular. | - Corba d'aprenentatge pronunciada.<br>- Necessita agents als servidors. |
| Kubernetes | Orquestrador de contenidors per gestionar aplicacions distribuïdes. | - Altament escalable.<br>- Auto-recuperació i desplegament automatitzat. | - Configuració complexa.<br>- Requereix molta infraestructura i experiència prèvia. |
| Terraform | Eina d'infraestructura com a codi (IaC) per gestionar recursos en el núvol. | - Compatible amb múltiples proveïdors de núvol.<br>- Permet definir infraestructures de forma declarativa. | - Pot ser complicat per a usuaris nous.<br>- No és ideal per a tasques puntuals de gestió de sistemes. |

## 2.2.2 Tecnologies escollides

**Tecnologies escollides per al projecte**

*Després d'analitzar diverses alternatives, s'ha optat per utilitzar **Ansible** i **Docker** com a tecnologies principals per al desenvolupament del projecte. La selecció s'ha fet tenint en compte la seva facilitat d'ús, escalabilitat i aplicació en entorns empresarials actuals.*

### 1. Ansible – Automatització d'infraestructures

### *Motivació per la seva elecció:*

- ***Simplicitat**: Utilitza fitxers YAML senzills (Playbooks) que faciliten la configuració i el desplegament automatitzat.*
- ***Sense agents**: No necessita instal·lar cap client en els servidors gestionats, ja que es basa en connexions SSH.*
- ***Escalabilitat**: És apte tant per a petites com per a grans infraestructures, amb capacitat per gestionar centenars de servidors de manera centralitzada.*
- ***Integració**: Compatible amb Docker, Kubernetes i proveïdors de núvol com AWS, Azure i Google Cloud.*

### 2. Docker – Contenidors per a la gestió d'aplicacions

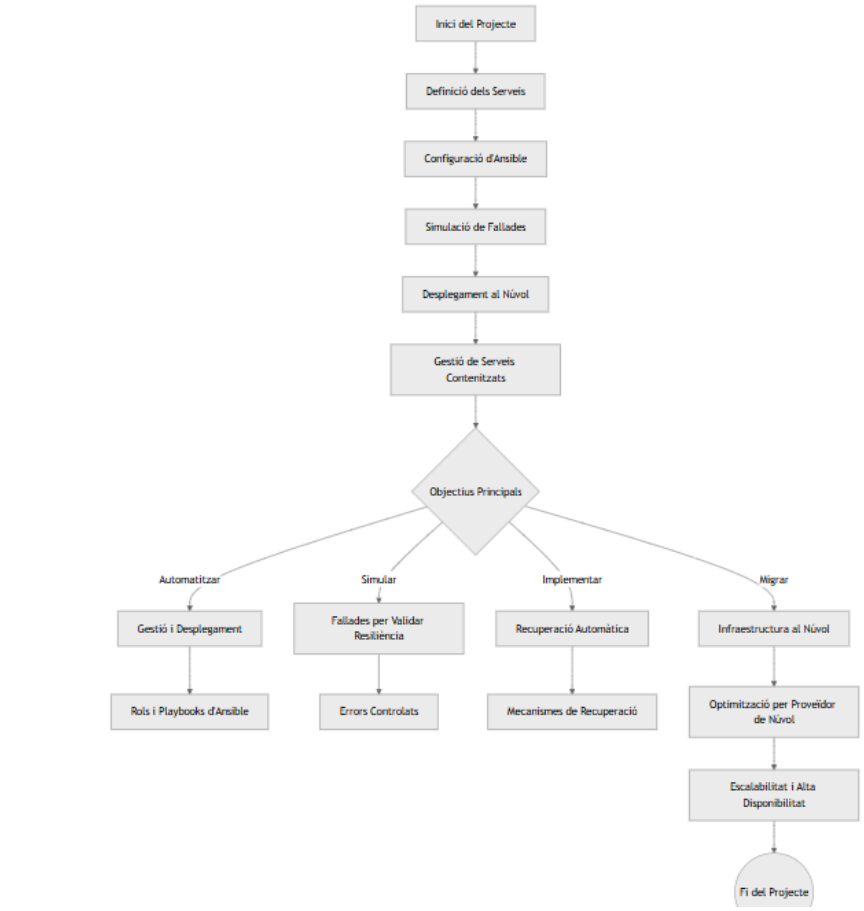
### *Motivació per la seva elecció:*

- ***Lleugeresa**: Els contenidors consumeixen menys recursos que les màquines virtuals, permetent executar múltiples serveis sense impactar significativament el rendiment.*
- ***Portabilitat**: Els contenidors es poden moure fàcilment entre entorns de desenvolupament, proves i producció, assegurant coherència.*
- ***Facilitat de desplegament**: Permet empaquetar aplicacions amb totes les seves dependències, evitant problemes de compatibilitat.*
- ***Orquestració**: Compatible amb Kubernetes per gestionar arquitectures de microserveis i escalar aplicacions de manera eficient.*

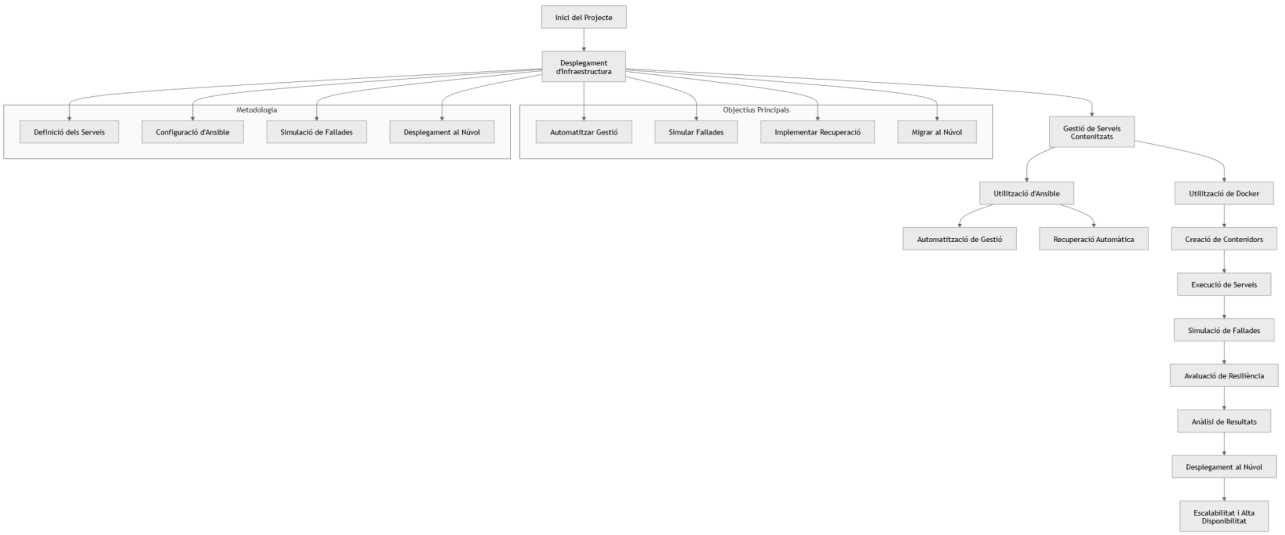
### Per què no s'han escollit altres tecnologies?

- ***Puppet i Chef** requereixen agents en els servidors i tenen una corba d'aprenentatge més pronunciada.*
- ***Kubernetes** és una eina poderosa per l'orquestració de contenidors, però per aquest projecte inicial s'ha optat per un enfocament més senzill amb Docker.*

# 2.3 Estructura del projecte




## 2.4 Descripció dels components



## 2.5 Definició de les funcionalitats [projecte d'implementació]

1. Desplegament automatitzat de serveis

- **Funcionalitat:** Permet desplegar serveis en contenidors Docker mitjançant Playbooks d'Ansible.
- Procés:
    1. Es defineixen els serveis a desplegar en un fitxer YAML d'Ansible.
    2. S'executa el Playbook, que crea i configura els contenidors de manera automàtica.
    3. Els serveis queden operatius sense intervenció manual.
- **Estat d'implementació: Totalment implementat.**

2. Monitorització dels serveis i detecció de fallades

- **Funcionalitat:** Verifica l'estat dels serveis desplegats i detecta si algun deixa de funcionar.
- Procés:
    1. Es configuren sondes de monitorització que comproven periòdicament la disponibilitat dels serveis.
    2. Si un servei cau, es genera una alerta i es desencadena un procés de recuperació.
- **Estat d'implementació: Parcialment (monitorització bàsica, però no s'ha integrat amb eines avançades com Prometheus o Grafana).**

3. Recuperació automàtica de serveis fallits

- **Funcionalitat:** Quan es detecta una fallada en un servei, Ansible el reinicia automàticament.
- Procés:
    1. La monitorització detecta que un servei ha deixat de respondre.
    2. Ansible executa una tasca per reiniciar el contenidor afectat.
    3. Si el servei continua fallant, es pot reconfigurar o redeployar automàticament.
- **Estat d'implementació: Parcialment**

4. Gestió centralitzada de configuracions

- **Funcionalitat:** Manté els fitxers de configuració dels serveis en un repositori versionat i aplicable a diferents entorns.
- Procés:

1. Es defineixen les configuracions en Playbooks d'Ansible.
2. Quan es desplega un servei, es carrega la seva configuració de manera automàtica.
3. Es poden fer modificacions centralitzades i desplegar-les a tots els nodes.

- **Estat d'implementació: Parcialment**

5. Escalabilitat dels serveis

- **Funcionalitat:** Possibilitat d'augmentar el nombre d'instàncies d'un servei de manera dinàmica.
- Procés:
  1. Es defineix un nombre mínim i màxim d'instàncies en la configuració d'Ansible.
  2. Si la càrrega augmenta, es poden desplegar més contenidors automàticament.
  3. Quan la càrrega baixa, els contenidors innecessaris es poden eliminar per estalviar recursos.
- **Estat d'implementació: Reservat per a una ampliació futura (no s'ha integrat autoscaling per falta de temps).**

# 3 Components rellevants i funcionalitats

# 4 Funcionament del projecte

Aquest apartat explica com s'han desplegat i com funcionen els diferents serveis que formen part de la infraestructura, tots gestionats i automatitzats mitjançant Ansible i executats dins de contenidors Docker. Cada servei té una funció concreta dins del sistema i ha estat integrat de manera coordinada per garantir-ne l'operativitat, la monitorització i la recuperació automàtica.

## 4.1.1 Docker

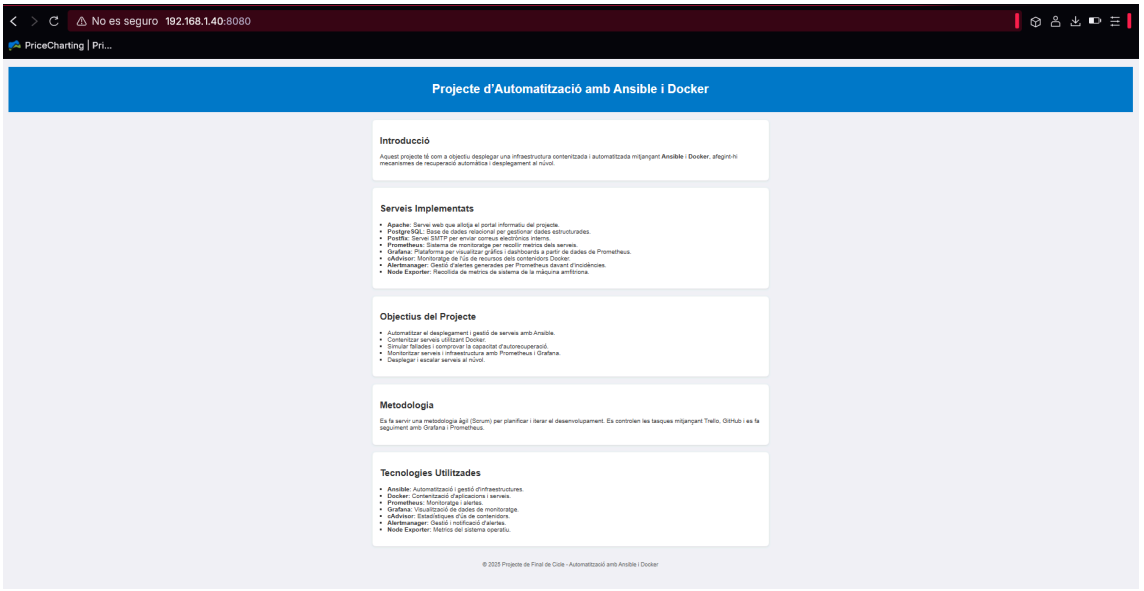
Docker és la base del projecte, ja que permet contenitzar cada servei de manera aïllada i fàcilment replicable. S'ha utilitzat per crear imatges personalitzades i definir contenidors amb configuracions específiques. Gràcies a aquesta eina, es poden controlar millor les dependències i el comportament dels serveis, i gestionar-los de forma eficient amb comandes i playbooks.

Cada servei s'ha desplegat amb la configuració adequada de ports, volums i variables d'entorn, assegurant la persistència de les dades i la seva accessibilitat des de l'exterior. A més, s'han definit polítiques de reinici automàtic (restart_policy: always) per garantir la seva disponibilitat contínua.

## 4.1.2 Apache

Apache s'ha desplegat com a servidor web principal, amb l'objectiu de servir un índex HTML informatiu del projecte i oferir enllaços d'accés als serveis desplegats. Escolta pel port 8080 i mostra una pàgina personalitzada que centralitza l'accés a tot l'entorn de pràctiques.

També actua com a punt d'entrada per consultar l'estat del sistema i accedir a panells de monitoratge, FTP o duplicació de dades.




## 4.1.3 PostgreSQL

PostgreSQL és el sistema de bases de dades relacional triat pel projecte. S'ha desplegat dins d'un contenidor amb persistència mitjançant volums Docker. La base de dades funciona com a

servei d'emmagatzematge d'exemple i s'hi han introduït dades de prova mitjançant scripts SQL executats amb un playbook.

Aquest servei ha estat útil per comprovar tant la persistència com el monitoratge dels recursos que consumeix una base de dades activa.

## 4.1.4 Postfix

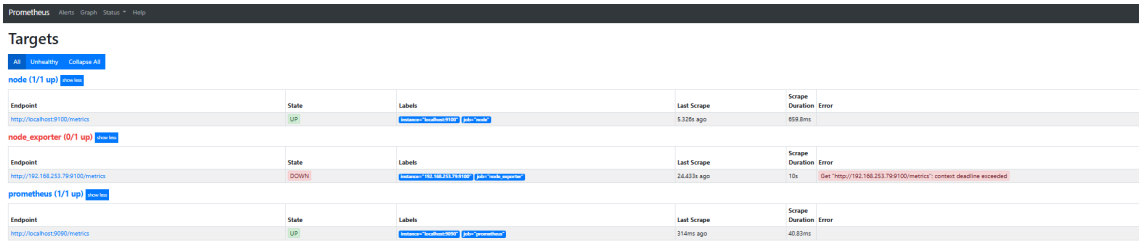
S'ha desplegat un servidor Postfix per gestionar l'enviament de correu electrònic. Inicialment es va intentar configurar amb un relay SMTP extern (com Gmail o Outlook), però, per limitacions d'autenticació i configuració, es va optar per un relay intern funcional.

El servei s'ha configurat amb fitxers com /etc/mailname, main.cf i sasl_passwd, i s'ha integrat amb el sistema per fer proves d'enviament local de correus entre contenidors. També ha permès validar alertes internes del sistema.

## 4.1.5 Prometheus

Prometheus s'ha desplegat com a eina de monitoratge per recollir mètriques dels serveis. La configuració es fa mitjançant el fitxer prometheus.yml, on s'especifiquen els serveis a monitorar (com cAdvisor o el mateix Prometheus).

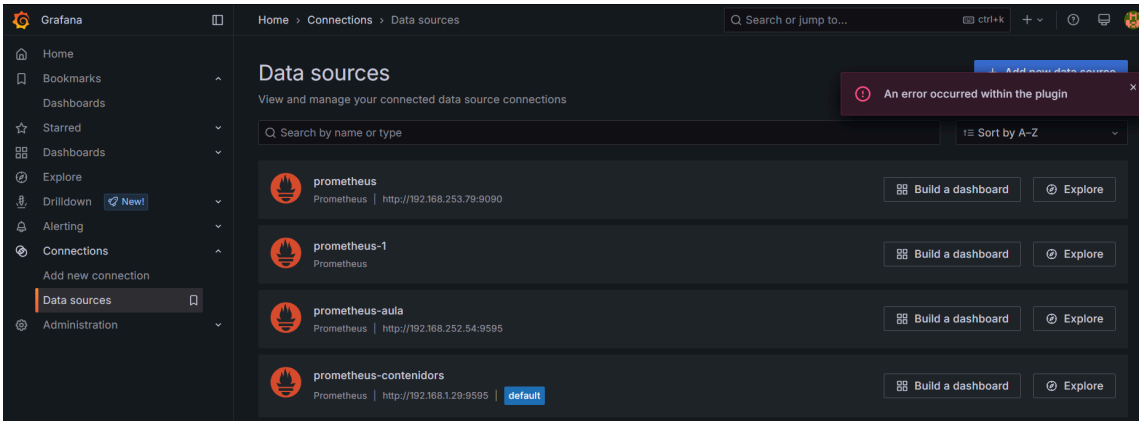
Recull dades sobre l'ús de CPU, memòria i estat dels contenidors, i permet visualitzar-les o generar alertes a través de Alertmanager.




## 4.1.6 Grafana

Grafana complementa Prometheus i permet crear dashboards interactius per visualitzar l'estat dels serveis. S'ha desplegat en un contenidor i configurat perquè es connecti a Prometheus com a font de dades.

Els dashboards personalitzats mostren mètriques com l'ús de recursos i l'estat dels serveis, oferint un seguiment visual i dinàmic del sistema.



### 4.1.7 cAdvisor

cAdvisor (Container Advisor) és l'eina encarregada de monitorar l'ús de recursos dels contenidors Docker. Ofereix informació detallada sobre CPU, memòria, I/O i ús de xarxa per contenidor.

S'ha integrat amb Prometheus per recollir mètriques i mostrar-les posteriorment a Grafana. És una peça clau per detectar anomalies o contenidors que han deixat de funcionar correctament.

### 4.1.8 FileBrowser

S'ha desplegat FileBrowser per facilitar la visualització i gestió gràfica dels fitxers del sistema. L'objectiu principal ha estat permetre la navegació per directoris com /srv/ftp, compartir arxius entre contenidors i accedir a còpies de seguretat.

Aquest servei ha estat útil per validar visualment tasques com la generació de backups i l'estat

### 4.1.9 FTP (vsftpd)

El servei FTP s'ha desplegat utilitzant la imatge de vsftpd per actuar com a repositori de còpies de seguretat i com a canal d'intercanvi de fitxers entre serveis.

S'ha configurat perquè tingui accés al directori /home/usuario/ansible-k8s, permetent que serveis com Duplicati hi escriguin fitxers de còpia o restauració.

### 4.1.10 BorgBackup

BorgBackup s'ha integrat com a eina principal per a la generació de còpies de seguretat incrementals, eficients i xifrades. El seu desplegament s'ha realitzat mitjançant Ansible, sense contenidors, per permetre una execució directa sobre el sistema host.

S'ha configurat perquè creï còpies del directori del projecte i d'altres rutes crítiques, emmagatzemant-les dins del directori /home/usuario/projecte-asix/admin/repo. El sistema utilitza un repositori xifrat amb clau privada (repokey) i genera backups amb noms únics basats en data i hora, facilitant així l'historial de versions.

Aquesta solució permet fer còpies eficients, gràcies a la deduplicació, i assegurar-ne la integritat. També serveix com a prova de persistència, ja que permet restaurar totalment els fitxers del sistema o carpetes específiques en cas de fallada.

### 4.1.11 rclone

rclone complementa BorgBackup actuant com a eina de transferència remota de les còpies generades. Ha estat configurat per establir connexió amb el servidor FTP del projecte (vsftpd), aprofitant un perfil definit a ~/.config/rclone/rclone.conf amb les credencials de l'usuari admin.

Després de crear i comprimir cada còpia de seguretat en format .tar.gz, rclone l'envia automàticament al directori remot backup/, ubicat dins del servidor FTP. Aquesta transferència es realitza de manera automàtica mitjançant un script que pot ser programat via cron o llançat manualment.

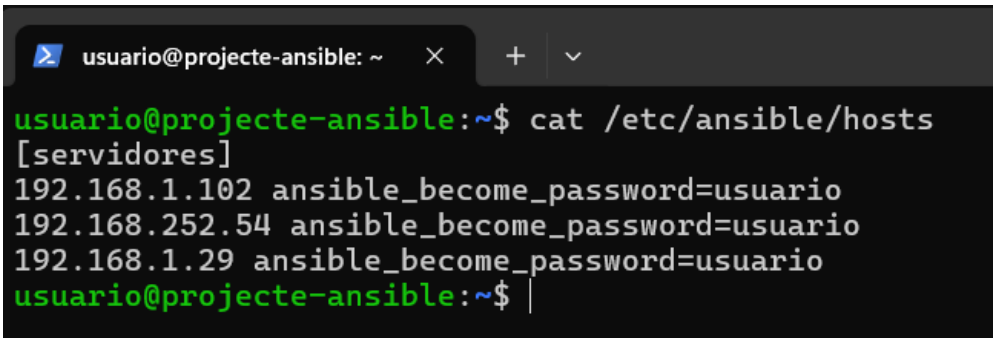
Amb aquest enfocament, les còpies de seguretat no només queden disponibles localment, sinó que també es conserven en un repositori remot, millorant la resiliència del sistema i permetent la recuperació fins i tot en cas de pèrdua local.

# 5 Arxiu d'inventari d'Ansible

L'arxiu **`/etc/ansible/hosts`** defineix la llista d'equips gestionats per Ansible i s'utilitza com a inventari estàtic per identificar les màquines on s'executaran les tasques automatitzades. Aquest fitxer segueix una estructura basada en grups, permetent organitzar els servidors segons rols o característiques comunes.

En aquest cas, s'ha creat un grup anomenat **`[servidores]`**, que inclou tres adreces IP corresponents a màquines que formen part de la infraestructura. A cadascuna se li ha associat el paràmetre **`ansible_become_password`**, que indica la contrasenya necessària per escalar privilegis mitjançant **`sudo`** durant l'execució de les tasques.

Aquest arxiu permet que Ansible identifiqui correctament els nodes remots i accedeixi a ells amb els permisos adequats per aplicar les configuracions definides als playbooks.




# 6 Playbooks

Els playbooks són una peça fonamental d'Ansible, ja que permeten definir de manera declarativa totes les tasques que cal executar en els nodes o servidors remots. En aquest projecte, s'han emprat per automatitzar la instal·lació, desplegament, configuració i monitoratge de serveis contenitzats amb Docker.

Estan escrits en format YAML, fàcilment llegible i mantenible, i es componen de diverses tasques (tasks) agrupades segons l'objectiu funcional. A continuació, es detallen els diferents tipus de playbooks desenvolupats, així com la seva estructura i finalitat dins del sistema.

## 6.1 Apache

Aquest playbook desplega un **contenidor Docker que inclou un servidor web Apache i accés SSH**, útil tant per a la visualització web com per a proves de connexió remota dins del projecte.

**Objectiu**

Automatitzar la creació d'un contenidor amb dos serveis integrats:

- **Apache** com a servidor web, exposat al port 8080.

- **SSH** per accedir remotament al contenidor amb autenticació per contrasenya, al port 2222.

**Estructura i explicació del playbook**

```
- name: Crear contenedor Docker amb Apache i SSH
  hosts: servidores
  become: true
  tasks:
```

S'executa sobre el grup servidores amb permisos d'administrador (become: true).

**1. Crear la carpeta del Dockerfile**

```
  - name: Crear carpeta del Dockerfile
    file:
     path: /opt/apache-ssh
     state: directory
```

Crea el directori /opt/apache-ssh on es guardarà el Dockerfile que defineix la imatge personalitzada.

**2. Copiar el Dockerfile**

```
  - name: Copiar Dockerfile
    copy:
    dest: /opt/apache-ssh/Dockerfile
```

Aquí es crea el Dockerfile amb el següent contingut:

```
content: |
FROM debian:bookworm
ENV DEBIAN_FRONTEND=noninteractive
RUN apt update && apt install -y apache2 openssh-server && \
   mkdir /var/run/sshd && \
   echo "root:root" | chpasswd && \
   sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config && \
   sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config && \
   echo 'ServerName localhost' >> /etc/apache2/apache2.conf
EXPOSE 80 22
CMD service apache2 start && /usr/sbin/sshd -D
```

- Base Debian Bookworm.
- Instal·lació de apache2 i openssh-server.
- Habilitació del login root per SSH amb contrasenya (root:root).
- Exposició dels ports 80 i 22 per a HTTP i SSH respectivament.
- Comandament final (CMD) per iniciar Apache i el servei SSH simultàniament.

**3. Construir la imatge Docker**

```
    - name: Construir imatge Docker
    community.docker.docker_image:
      name: apache-ssh
      source: build
      build:
        path: /opt/apache-ssh
```

Aquest pas construeix la imatge personalitzada `apache-ssh` a partir del Dockerfile creat.

**4. Llançar el contenidor**

```
- name: Assegurar que el contenidor està corrent
  community.docker.docker_container:
    name: apache_final
    image: apache-ssh
    state: started
    restart_policy: always
    published_ports:
     - "8080:80"
     - "2222:22"
```

Crea i inicia un contenidor anomenat apache_final:

- Publica el port **80** al **8080** (accés web).
- Publica el port **22** al **2222** (accés SSH).
- Estableix política de reinici automàtic (restart_policy: always).

```
usuario@projecte-ansible: ~/
  GNU nano 7.2                                          crear-contenedor-apache.yml
- name: Crear contenedor Docker amb Apache i SSH
  hosts: servidores
  become: true
  tasks:

    - name: Crear carpeta del Dockerfile
      file:
        path: /opt/apache-ssh
        state: directory

    - name: Copiar Dockerfile
      copy:
        dest: /opt/apache-ssh/Dockerfile
        content: |
          FROM debian:bookworm
          ENV DEBIAN_FRONTEND=noninteractive
          RUN apt update && apt install -y apache2 openssh-server && \
              mkdir /var/run/sshd && \
              echo "root:root" | chpasswd && \
              sed -i 's/#PermitRootLogin prohibit-password/PermitRootLogin yes/' /etc/ssh/sshd_config && \
              sed -i 's/#PasswordAuthentication yes/PasswordAuthentication yes/' /etc/ssh/sshd_config && \
              echo 'ServerName localhost' >> /etc/apache2/apache2.conf
          EXPOSE 80 22
          CMD service apache2 start && /usr/sbin/sshd -D

    - name: Construir imatge Docker
      community.docker.docker_image:
        name: apache-ssh
        source: build   # <-- Esto es lo que te faltaba
        build:
          path: /opt/apache-ssh

    - name: Assegurar que el contenidor està corrent
      community.docker.docker_container:
        name: apache_final
        image: apache-ssh
        state: started
```

```
restart_policy: always
published_ports:
  - "8080:80"
  - "2222:22"
```

# 6.2 Injectar index

**1. Crear el directori dins del contenidor**

- name: Crear directori dins el contenidor si no existeix
  command: docker exec apache_final mkdir -p /usr/local/apache2/htdocs

- Utilitza docker exec per **crear un directori dins del contenidor** apache_final.
- Mkdir -p assegura que es creï la ruta si no existeix, sense error si ja hi és.
- Tot i que Apache sol servir el contingut des de /var/www/html, aquest pas pot estar pensat com a precaució o per compatibilitat amb una altra imatge.

**2. Copiar el fitxer index.html al contenidor**

- name: Copiar index.html al contenidor apache_final
  command: docker cp /tmp/index.html apache_final:/var/www/html/index.html

- **Copia un fitxer local index.html**, ubicat a /tmp/index.html, **dins del contenidor** al directori /var/www/html/.
- Aquest directori és el que Apache utilitza habitualment com a **directori web per defecte**.

3. Reiniciar Apache

- name: Reiniciar Apache dins del contenidor
  command: docker exec apache_final service apache2 restart

- Reinicia el servei Apache **dins del contenidor** per assegurar que **el nou fitxer `index.html` sigui servit correctament**.
- Això és necessari sobretot si es fan canvis de configuració o contingut actiu.

```yaml
- name: Injectar index.html dins del contenidor Apache
  hosts: servidores
  become: yes
  tasks:
    - name: Crear directori dins el contenidor si no existeix
      command: docker exec apache_final mkdir -p /usr/local/apache2/htdocs

    - name: Copiar index.html al contenidor apache_final
      command: docker cp /tmp/index.html apache_final:/var/www/html/index.html

    - name: Reiniciar Apache dins del contenidor
      command: docker exec apache_final service apache2 restart
```

# 6.3 PostgreSQL

Aquest playbook s'encarrega del desplegament d'un contenidor Docker amb el sistema de gestió de bases de dades **PostgreSQL**, configurat automàticament amb una base de dades, una taula i algunes dades d'exemple. Aquesta automatització és útil per simular entorns reals on una base de dades ha d'estar llesta per funcionar immediatament després del desplegament.

- name: Crear contenidor PostgreSQL i afegir dades
  hosts: servidores

```
become: true
vars:
  container_name: postgres_test
  postgres_password: mysecretpassword
  db_name: empresa
  table_name: empleats
```

S'estableixen les variables principals per reutilitzar-les durant el playbook: nom del contenidor, contrasenya de l'usuari postgres, nom de la base de dades i de la taula.

**1. Instal·lació de Docker**

```
- name: Assegurar que Docker està instal·lat
  apt:
    name: docker.io
    state: present
    update_cache: yes
```

Es garanteix que Docker estigui instal·lat abans d'intentar crear qualsevol contenidor.

**2. Creació i llançament del contenidor PostgreSQL**

```
- name: Iniciar contenidor PostgreSQL
  community.docker.docker_container:
    name: "{{ container_name }}"
    image: postgres:15
    state: started
    restart_policy: always
    env:
      POSTGRES_PASSWORD: "{{ postgres_password }}"
    ports:
      - "5432:5432"
```

- Es desplega un contenidor amb la imatge oficial postgres:15.
- Es defineix la contrasenya de l'usuari per defecte (POSTGRES_PASSWORD) i es publica el port 5432.

**3. Esperar que PostgreSQL estigui disponible**

```
- name: Esperar que PostgreSQL estigui llest
  wait_for:
    host: 127.0.0.1
    port: 5432
    timeout: 30
```

Es fa una espera activa perquè el servei PostgreSQL dins del contenidor estigui completament operatiu abans de passar a executar ordres SQL.

**4. Creació d'un script SQL temporal**

```
- name: Crear fitxer SQL temporal
  copy:
    dest: /tmp/init.sql
    content: |
      CREATE DATABASE {{ db_name }};
      \c {{ db_name }}
      CREATE TABLE IF NOT EXISTS {{ table_name }} (
        id SERIAL PRIMARY KEY,
        nom TEXT,
        edat INT
      );
```

```
INSERT INTO {{ table_name }} (nom, edat) VALUES ('Joan', floor(random()*30 + 20)::int);
INSERT INTO {{ table_name }} (nom, edat) VALUES ('Maria', floor(random()*30 + 20)::int);
```

- Es crea un fitxer SQL a /tmp/init.sql amb ordres per crear una base de dades, una taula i inserir dades simulades.

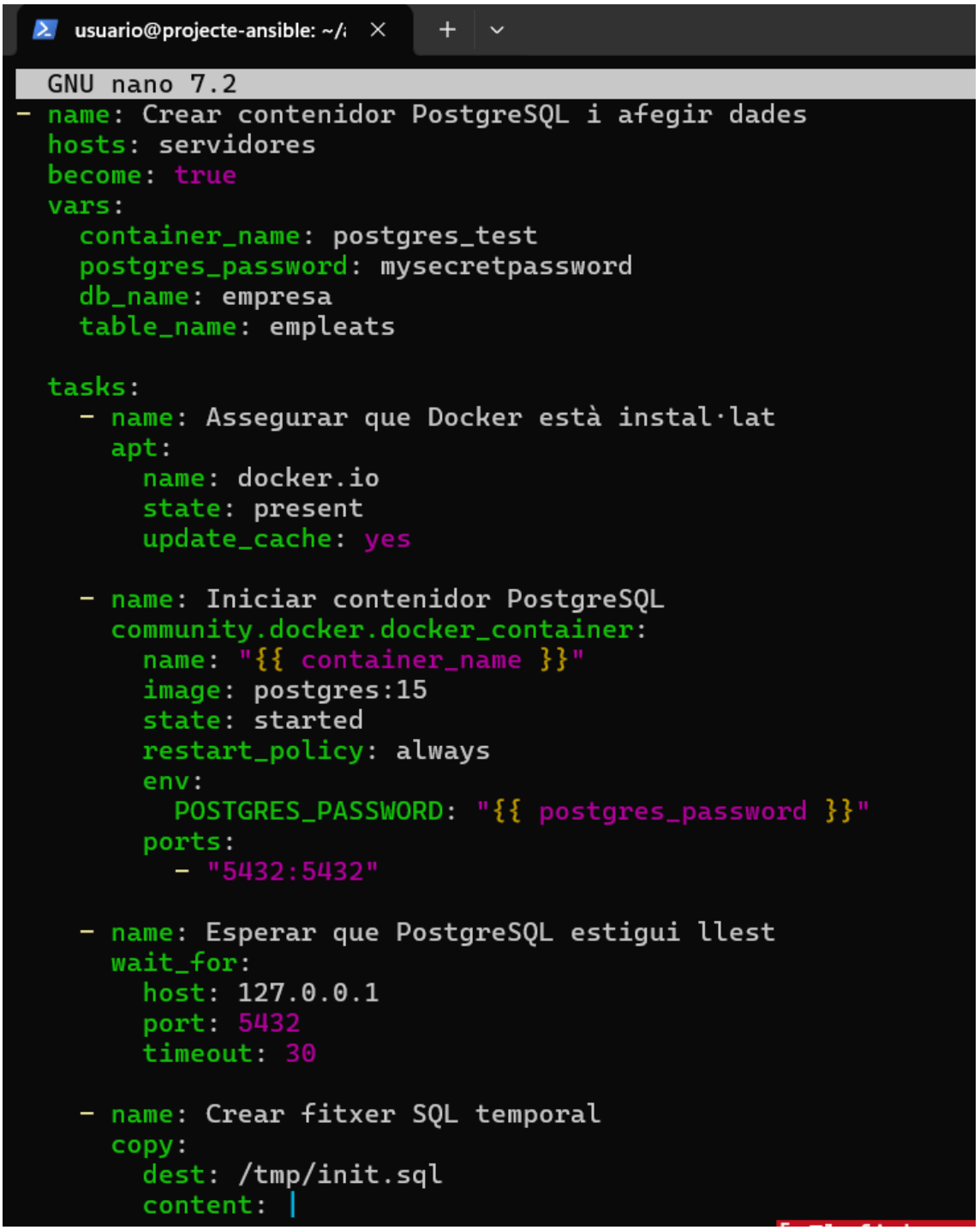
- Es fa servir PostgreSQL per avaluar expressions com random() i floor() per generar edats aleatòries.

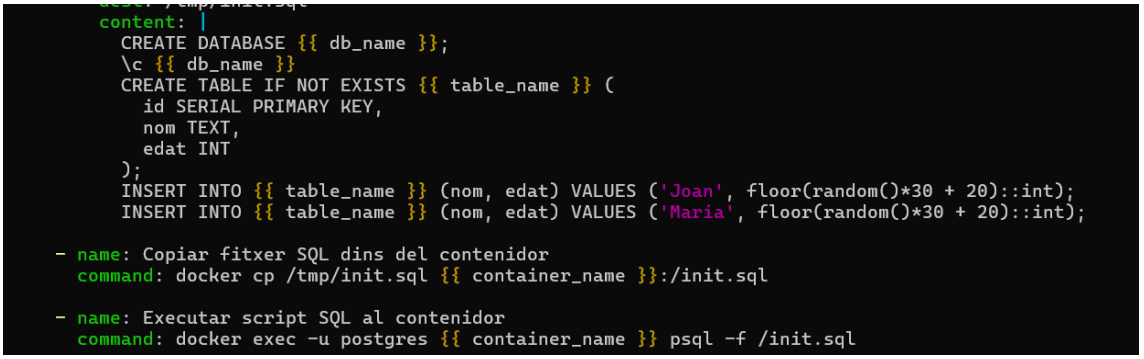
**5. Còpia i execució de l'script SQL**

```
- name: Copiar fitxer SQL dins del contenidor
  command: docker cp /tmp/init.sql {{ container_name }}:/init.sql
- name: Executar script SQL al contenidor
  command: docker exec -u postgres {{ container_name }} psql -f /init.sql
```

- L'script es copia dins del contenidor.

- Posteriorment, es executa el script SQL com a usuari postgres, utilitzant psql.


usuario@projecte-ansible: ~/
GNU nano 7.2
- name: Crear contenidor PostgreSQL i afegir dades
hosts: servidores
become: true
vars:
container_name: postgres_test
postgres_password: mysecretpassword
db_name: empresa
table_name: empleats
tasks:
- name: Assegurar que Docker està instal·lat
apt:
name: docker.io
state: present
update_cache: yes
- name: Iniciar contenidor PostgreSQL
community.docker.docker_container:
name: "{{ container_name }}"
image: postgres:15
state: started
restart_policy: always
env:
POSTGRES_PASSWORD: "{{ postgres_password }}"
ports:
- "5432:5432"
- name: Esperar que PostgreSQL estigui llest
wait_for:
host: 127.0.0.1
port: 5432
timeout: 30
- name: Crear fitxer SQL temporal
copy:
dest: /tmp/init.sql
content: |


content: |
CREATE DATABASE {{ db_name }};
\c {{ db_name }}
CREATE TABLE IF NOT EXISTS {{ table_name }} (
id SERIAL PRIMARY KEY,
nom TEXT,
edat INT
);
INSERT INTO {{ table_name }} (nom, edat) VALUES ('Joan', floor(random()*30 + 20)::int);
INSERT INTO {{ table_name }} (nom, edat) VALUES ('Maria', floor(random()*30 + 20)::int);
- name: Copiar fitxer SQL dins del contenidor
command: docker cp /tmp/init.sql {{ container_name }}:/init.sql
- name: Executar script SQL al contenidor
command: docker exec -u postgres {{ container_name }} psql -f /init.sql

# 6.4 FTP i Filebrowser

Aquest playbook permet desplegar un servei de **FTP segur** mitjançant el contenidor fauria/vsftpd, juntament amb l'aplicació web **Filebrowser**, que permet gestionar fitxers gràficament a través del navegador. Ambdós serveis comparteixen el directori /home/usuario/ansible-k8s, oferint així una ruta centralitzada per gestionar arxius tant via FTP com via web.

**1. Creació del directori compartit**

```
- name: Assegurar que el directori existeix
  file:
    path: /home/usuario/ansible-k8s
    state: directory
    mode: '0755'
```

Aquesta tasca garanteix que el directori base /home/usuario/ansible-k8s existeixi i sigui accessible pel contenidor FTP i Filebrowser. Aquest directori actuarà com a **espai compartit** per als dos serveis.

**2. Llançament del contenidor FTP**

```
- name: Llançar contenidor FTP
  community.docker.docker_container:
    name: ftp-server
    image: fauria/vsftpd
    state: started
    restart_policy: always
    published_ports:
      - "21:21"
      - "21000-21010:21000-21010"
    env:
      FTP_USER: "admin"
      FTP_PASS: "admin"
      PASV_MIN_PORT: "21000"
      PASV_MAX_PORT: "21010"
      PASV_ADDRESS: "{{ ansible_host }}"
    volumes:
      - /home/usuario/ansible-k8s:/home/vsftpd
```

- Es desplega un contenidor amb el servidor **vsftpd**.
- Es defineixen usuaris amb credencials (admin/admin).
- Es configuren ports passius de 21000 a 21010 per millorar la compatibilitat amb tallafocs.
- El directori local /home/usuario/ansible-k8s es mapeja al directori d'FTP /home/vsftpd, permetent pujar i baixar fitxers des del contenidor.

**3. Llançament del contenidor Filebrowser**

```
- name: Llançar contenidor Filebrowser
  community.docker.docker_container:
    name: filebrowser
    image: filebrowser/filebrowser
    state: started
    restart_policy: always
```

```
published_ports:
  - "8081:80"
volumes:
  - /home/usuario/ansible-k8s:/srv
command: ["--port", "80", "--noauth"]
```

- Filebrowser proporciona una interfície web per gestionar arxius, ideal per a usuaris que no utilitzen FTP tradicional.
- Amb l'opció --noauth, no es requereix login per accedir a la pàgina (configurable).
- El contenidor exposa el port **8081** i mostra el contingut de /srv, que correspon al mateix directori compartit.

```yaml
- name: Desplegar servidor FTP i Filebrowser amb accés a /home/usuario/ansible-k8s
  hosts: servidores
  become: true

  tasks:
    - name: Assegurar que el directori principal existeix
      file:
        path: /home/usuario/ansible-k8s
        state: directory
        mode: '0755'

    - name: Crear subdirectori per a l'usuari FTP admin
      file:
        path: /home/usuario/ansible-k8s/admin
        state: directory
        owner: 1000
        group: 1000
        mode: '0755'

    - name: Llançar contenidor FTP
      community.docker.docker_container:
        name: ftp-server
        image: fauria/vsftpd
        state: started
        restart_policy: always
        published_ports:
          - "21:21"
          - "21000-21010:21000-21010"
        env:
          FTP_USER: "admin"
          FTP_PASS: "admin"
          PASV_MIN_PORT: "21000"
          PASV_MAX_PORT: "21010"
          PASV_ADDRESS: "{{ ansible_host }}"
        volumes:
          - /home/usuario/ansible-k8s:/home/vsftpd
```

```yaml
- name: Llançar contenidor Filebrowser
  community.docker.docker_container:
    name: filebrowser
    image: filebrowser/filebrowser
    state: started
    restart_policy: always
    published_ports:
      - "8081:80"
    volumes:
      - /home/usuario/ansible-k8s:/srv
    command: ["--port", "80", "--noauth"]
```

## 6.5 Postfix

Aquest playbook desplega un servei de correu electrònic bàsic utilitzant **Postfix** dins d'un contenidor Docker. S'ha configurat com un **relay intern**, és a dir, sense cap servidor SMTP extern (com Gmail o Outlook). Aquesta configuració és útil per a entorns locals o pràctiques on no es necessita enviar correus fora de la xarxa.

**1. Creació del directori de configuració**

```yaml
- name: Crear carpeta /opt/postfix-intern/etc
  file:
    path: /opt/postfix-intern/etc
    state: directory
    mode: '0755'
```

Aquesta tasca crea la ruta on es col·locaran els fitxers de configuració de Postfix dins del projecte. És essencial per mantenir la configuració estructurada i modularitzada.

**2. Fitxer mailname**

```yaml
- name: Crear fitxer /etc/mailname
  copy:
    dest: /opt/postfix-intern/etc/mailname
    content: "elpuig.example.com\n"
    mode: '0644'
```

El fitxer mailname indica el nom de domini del sistema per als correus sortints, com ara root@elpuig.example.com.

**3. Fitxer main.cf per configuració del relay intern**

```yaml
- name: Crear fitxer main.cf per relay intern
  copy:
    dest: /opt/postfix-intern/etc/main.cf
    content: |
      myhostname = elpuig.example.com
```

```
    mydestination = $myhostname, localhost.localdomain, localhost
    relayhost =
    inet_interfaces = all
    smtpd_banner = $myhostname ESMTP
    biff = no
    append_dot_mydomain = no
    readme_directory = no
    smtp_tls_security_level = may
    smtpd_tls_security_level = may
  mode: '0644'
```

Aquest fitxer defineix els paràmetres bàsics de funcionament de Postfix com a servidor MTA. El paràmetre relayhost buit indica que els correus es gestionaran **internament**, sense cap servei extern.

**4. Creació del Dockerfile per construir la imatge**

```
- name: Crear Dockerfile per Postfix bàsic
  copy:
    dest: /opt/postfix-intern/Dockerfile
    content: |
      FROM debian:bookworm
      ENV DEBIAN_FRONTEND=noninteractive
      COPY etc/main.cf /etc/postfix/main.cf
      COPY etc/mailname /etc/mailname
      RUN apt update && apt install -y postfix bsd-mailx
      EXPOSE 25
      CMD ["postfix", "start-fg"]
```

Aquest Dockerfile instal·la **Postfix** i les eines per enviar correus (bsd-mailx). El servei queda exposat pel port 25 dins del contenidor.

**5. Construcció de la imatge**

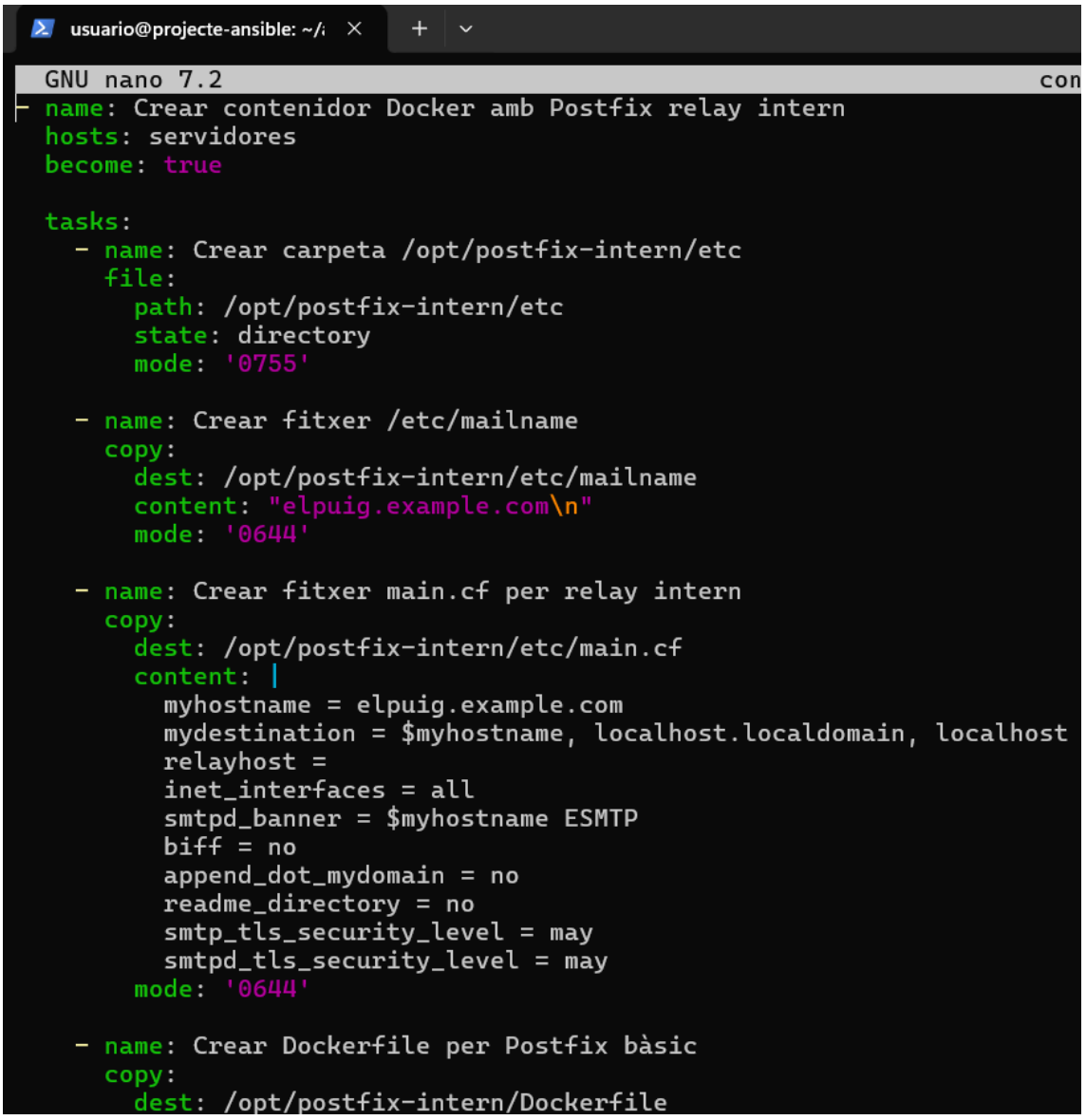
```
- name: Construir imatge Docker de Postfix bàsic
  community.docker.docker_image:
    name: postfix-intern-image
    source: build
    build:
      path: /opt/postfix-intern
```

Es genera una imatge Docker anomenada postfix-intern-image a partir del directori /opt/postfix-intern, que conté la configuració i el Dockerfile.

**6. Execució del contenidor Postfix**

```
- name: Llançar contenidor Postfix intern
  community.docker.docker_container:
    name: postfix-intern-container
    image: postfix-intern-image
    state: started
    restart_policy: always
    published_ports:
      - "2525:25"
    dns_servers:
      - "8.8.8.8"
      - "8.8.4.4"
```

Finalment, el contenidor queda actiu i accessible a través del port 2525 (mapejat al port 25 intern). També es configuren **DNS externs** per garantir la resolució de dominis dins del contenidor.



```
      content: |
        FROM debian:bookworm
        ENV DEBIAN_FRONTEND=noninteractive
        COPY etc/main.cf /etc/postfix/main.cf
        COPY etc/mailname /etc/mailname
        RUN apt update && apt install -y postfix bsd-mailx
        EXPOSE 25
        CMD ["postfix", "start-fg"]

    - name: Construir imatge Docker de Postfix bàsic
      community.docker.docker_image:
        name: postfix-intern-image
        source: build
        build:
          path: /opt/postfix-intern

    - name: Llançar contenidor Postfix intern
      community.docker.docker_container:
        name: postfix-intern-container
        image: postfix-intern-image
        state: started
        restart_policy: always
        published_ports:
          - "2525:25"
        dns_servers:
          - "8.8.8.8"
          - "8.8.4.4"
```

# 6.6 Prometheus

Prometheus és una eina de monitoratge **basada en mètriques** àmpliament utilitzada en entorns DevOps i de sistemes. En aquest projecte, s'ha desplegat com a contenidor Docker mitjançant Ansible per recollir mètriques dels serveis i contenidors, especialment gràcies a la integració amb cAdvisor i Alertmanager.

Aquest playbook automatitza tot el procés de configuració i desplegament de Prometheus i defineix alertes bàsiques per detectar si algun contenidor deixa de funcionar.

**1. Creació del directori de configuració**

```
- name: Crear directori de configuració Prometheus
  file:
    path: /opt/prometheus
    state: directory
    mode: '0755'
```

Es crea el directori local on es guardaran els fitxers de configuració que es muntaran dins del contenidor.

**2. Fitxer prometheus.yml amb Alertmanager i cAdvisor**

```
- name: Crear fitxer prometheus.yml amb Alertmanager i regles
  copy:
    dest: /opt/prometheus/prometheus.yml
    content: |
      global:
```

```
    scrape_interval: 15s

  alerting:
    alertmanagers:
      - static_configs:
          - targets:
              - '192.168.1.29:9093'

  rule_files:
    - "alert_rules.yml"

  scrape_configs:
    - job_name: 'prometheus'
      static_configs:
        - targets: ['192.168.1.29:9595']

    - job_name: 'cadvisor'
      static_configs:
        - targets: ['192.168.1.29:8585']
```

- L'interval global de consulta de mètriques.
- La connexió amb Alertmanager a través del port 9093.
- La inclusió de fitxers de regles (alert_rules.yml).
- Les fonts de dades que cal monitoritzar: Prometheus i cAdvisor.

**3. Fitxer de regles d'alertes alert_rules.yml**

```
- name: Crear fitxer de regles d'alertes alert_rules.yml
  copy:
    dest: /opt/prometheus/alert_rules.yml
    content: |
      groups:
      - name: docker_alerts
        rules:
        - alert: ContainerDown
          expr: up == 0
          for: 1m
          labels:
            severity: critical
          annotations:
            summary: "Un contenidor ha caigut"
            description: "El contenidor {{ $labels.instance }} no respon."
```

Aquesta regla activa una alerta anomenada ContainerDown si algun contenidor monitoritzat deixa d'estar disponible (up == 0) durant més d'un minut.

**4. Execució del contenidor Prometheus**

```
- name: Llançar contenidor Prometheus
  community.docker.docker_container:
    name: prometheus
    image: prom/prometheus:latest
    state: started
    restart_policy: always
```

```
    published_ports:
      - "9595:9090"
    volumes:
      - /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
      - /opt/prometheus/alert_rules.yml:/etc/prometheus/alert_rules.yml
```

Aquest contenidor s'executa amb el port 9595 exposat i amb els fitxers de configuració muntats des del sistema amfitrió. Gràcies a això, qualsevol canvi en les regles o configuració es pot fer fàcilment sense reconstruir la imatge.

# 6.7 cAdvisor

**cAdvisor** (Container Advisor) és una eina desenvolupada per Google que permet recollir informació en temps real sobre el rendiment dels contenidors Docker. Aquesta eina proporciona mètriques detallades sobre l'ús de CPU, memòria, disc i xarxa de cada contenidor, i s'integra fàcilment amb **Prometheus** per visualitzar-les i fer-ne seguiment.

```
- name: Llançar contenidor cAdvisor
  community.docker.docker_container:
    name: cadvisor
    image: gcr.io/cadvisor/cadvisor:v0.47.1
    state: started
    restart_policy: always
    published_ports:
      - "8585:8080"
    volumes:
      - "/:/rootfs:ro"
      - "/var/run:/var/run:rw"
      - "/sys:/sys:ro"
      - "/var/lib/docker/:/var/lib/docker:ro"
      - "/sys/fs/cgroup:/sys/fs/cgroup:ro"
```

Aquest bloc és el que llança realment el contenidor de **cAdvisor**. Les opcions més destacades són:

- **Imatge**: gcr.io/cadvisor/cadvisor:v0.47.1 — s'utilitza una versió estable recent.
- **Port 8585**: Exposa el port 8080 del contenidor al port 8585 de l'amfitrió per accedir via web o via Prometheus.
- **Volums muntats**:
    - **/:/rootfs:ro:** Accés al sistema de fitxers arrel per llegir metadades.
    - **/var/run:/var/run:**rw: Accés als sockets i PID del sistema.
    - **/sys:/sys:**ro, /sys/fs/cgroup:/sys/fs/cgroup:ro: Necessaris per accedir a informació del sistema operatiu i cgroups.
    - **/var/lib/docker/:/var/lib/docker:ro:** Permet veure els contenidors Docker en execució.

Aquesta configuració permet que cAdvisor pugui monitoritzar **tots els contenidors** de l'amfitrió i exposar les seves mètriques perquè Prometheus les consulti.

usuario@projecte-ansible: ~/

```
 GNU nano 7.2                                                    moni
- name: Llançar Prometheus i cAdvisor
  hosts: servidores
  become: true

  tasks:
    - name: Crear directori de configuració Prometheus
      file:
        path: /opt/prometheus
        state: directory
        mode: '0755'

    - name: Crear fitxer prometheus.yml amb Alertmanager i regles
      copy:
        dest: /opt/prometheus/prometheus.yml
        content: |
          global:
            scrape_interval: 15s

          alerting:
            alertmanagers:
              - static_configs:
                  - targets:
                      - '192.168.1.29:9093'  # IP i port de l'Alertmanager

          rule_files:
            - "alert_rules.yml"

          scrape_configs:
            - job_name: 'prometheus'
              static_configs:
                - targets: ['192.168.1.29:9595']

            - job_name: 'cadvisor'
              static_configs:
                - targets: ['192.168.1.29:8585']
```

```
    - name: Crear fitxer de regles d'alertes alert_rules.yml
      copy:
        dest: /opt/prometheus/alert_rules.yml
        content: |
          groups:
          - name: docker_alerts
            rules:
            - alert: ContainerDown
              expr: up == 0
              for: 1m
              labels:
                severity: critical
              annotations:
                summary: "Un contenidor ha caigut"
                description: "El contenidor {{'{{'}} $labels.instance {{'}}'}} no respon."

    - name: Llançar contenidor Prometheus
      community.docker.docker_container:
        name: prometheus
        image: prom/prometheus:latest
        state: started
        restart_policy: always
        published_ports:
          - "9595:9090"
        volumes:
          - /opt/prometheus/prometheus.yml:/etc/prometheus/prometheus.yml
          - /opt/prometheus/alert_rules.yml:/etc/prometheus/alert_rules.yml
```

```
- name: Llançar contenidor cAdvisor
  community.docker.docker_container:
    name: cadvisor
    image: gcr.io/cadvisor/cadvisor:v0.47.1
    state: started
    restart_policy: always
    published_ports:
      - "8585:8080"
    volumes:
      - "/:/rootfs:ro"
      - "/var/run:/var/run:rw"
      - "/sys:/sys:ro"
      - "/var/lib/docker/:/var/lib/docker:ro"
      - "/sys/fs/cgroup:/sys/fs/cgroup:ro"
```

## 6.8 Grafana

**Grafana** és una eina de visualització de dades molt potent que permet crear dashboards interactius per monitoritzar sistemes, serveis i infraestructures. Al projecte, es fa servir Grafana per representar gràficament les mètriques recollides per **Prometheus**, com l'estat dels contenidors Docker, l'ús de recursos i les alertes configurades.

```
- name: Llançar contenidor Grafana
  community.docker.docker_container:
    name: grafana
    image: grafana/grafana
    state: started
    restart_policy: always
    published_ports:
      - "3000:3000"
    volumes:
      - grafana-storage:/var/lib/grafana
```

- **Imatge**: `grafana/grafana` — imatge oficial.
- **Port 3000**: Permet accedir al panell de Grafana via navegador.
- **Volum `grafana-storage`**: Guarda la configuració, dashboards i dades persistents de l'usuari.

# 6.9 Borg backup + rclone

```
- name: Instal·lar Borg i rclone i configurar backup a FTP
  hosts: servidores
  become: true

  vars:
    backup_dir: /home/usuario/projecte-asix/admin
    borg_repo: /home/usuario/projecte-asix/admin/repo
    borg_passphrase: "admin"
    rclone_remote_name: "ftpserver"
    rclone_host: "localhost"
    rclone_user: "admin"
    rclone_pass_obscured: "admin"
```

Aquesta primera part defineix el playbook. S'executarà sobre el grup de màquines anomenat servidores i utilitzarà permisos d'administrador. Tot seguit es declaren variables que s'utilitzaran en les tasques següents, com ara el directori de còpia, el repositori de BorgBackup, la contrasenya per protegir-lo, i les dades de connexió per a rclone, que permetran enviar els backups a un servidor FTP local.

```
  tasks:
    - name: Instalar BorgBackup i rclone
      apt:
        name:
          - borgbackup
          - rclone
        state: present
        update_cache: yes
```

Aquesta tasca s'encarrega d'instal·lar BorgBackup i rclone utilitzant el gestor de paquets `apt`. S'assegura també que la llista de paquets estigui actualitzada abans de fer la instal·lació, per garantir que es descarreguen les versions més recents disponibles als repositoris.

```
    - name: Crear carpeta de backup
      file:
        path: "{{ backup_dir }}"
        state: directory
        owner: usuario
        mode: '0755'
```

Aquí es crea el directori on s'emmagatzemaran les còpies de seguretat, amb permisos perquè l'usuari normal (usuario) el pugui utilitzar. Aquesta carpeta serà compartida després amb el servei FTP per permetre'n la descàrrega o consulta des d'altres equips o serveis.

```
    - name: Crear carpeta per config rclone
      file:
        path: /home/usuario/.config/rclone
        state: directory
        owner: usuario
        mode: '0700'
```

Aquesta tasca crea la ruta on rclone emmagatzemarà la seva configuració. Es troba dins del directori .config de l'usuari. Els permisos són restrictius per assegurar que només aquest usuari pugui llegir o escriure aquest fitxer, ja que conté credencials sensibles.

```
- name: Crear fitxer de configuració rclone.conf
  copy:
    dest: /home/usuario/.config/rclone/rclone.conf
    owner: usuario
    mode: '0600'
    content: |
      [{{ rclone_remote_name }}]
      type = ftp
      host = {{ rclone_host }}
      user = {{ rclone_user }}
      pass = {{ rclone_pass_obscured }}
```

Amb aquesta tasca es crea el fitxer rclone.conf, que defineix el perfil de connexió cap al servidor FTP. Inclou el tipus de connexió (FTP), el host (que és localhost si el servidor està en la mateixa màquina), l'usuari i la contrasenya ja ofuscada. Aquest fitxer és fonamental perquè rclone pugui pujar arxius automàticament cap al servidor definit.

```
- name: Crear script de backup
  copy:
    dest: /home/usuario/projecte-asix/borgcs/backup-ftp.sh
    owner: usuario
    mode: '0755'
    content: |
      #!/bin/bash

      export BORG_REPO={{ borg_repo }}
      export BORG_PASSPHRASE='{{ borg_passphrase }}'

      mkdir -p "$BORG_REPO"

      # Inicialitza si no existeix
      if [ ! -d "$BORG_REPO/data" ]; then
          borg init --encryption=repokey "$BORG_REPO"
      fi

      # Crear backup amb borg
              borg create --stats "$BORG_REPO"::"backup-$(date +%Y-%m-%d_%H-%M-%S)" /etc
/home/usuario/projecte-asix

      # Crear nom de fitxer únic
      FECHA=$(date +%Y-%m-%d_%H-%M-%S)
      ARCHIVO="{{ backup_dir }}/backup-${FECHA}.tar.gz"

      # Comprimir el repo
      tar -czf "$ARCHIVO" "$BORG_REPO"

      # Enviar al servidor FTP
      rclone copy "$ARCHIVO" {{ rclone_remote_name }}:backup
```

Finalment, aquesta tasca genera l'script que automatitza tot el procés de còpia. L'script exporta les variables necessàries per utilitzar BorgBackup, crea el repositori si no existeix, realitza un backup amb data i hora per tenir una còpia única, el comprimeix en un fitxer .tar.gz i el puja al servidor FTP configurat amb rclone. L'script es desa al directori del projecte i se li assignen permisos perquè pugui ser executat fàcilment.

# 6.10 Aixecar contenidors

```
- name: Crear y lanzar relanzar.sh en segundo plano
  hosts: servidores
  become: true
  vars:
    base_path: /home/usuario/projecte-asix
    script_path: /home/usuario/projecte-asix/relanzar.sh
```

El seu objectiu és crear i executar un script que supervisa si els contenidors Docker estan en funcionament. Es llança sobre el grup de màquines anomenat servidores i s'executa amb permisos d'administrador per poder accedir a tot el sistema. Les dues variables declarades defineixen la ruta base del projecte i la ruta on es crearà l'script relanzar.sh.

```
  tasks:
    - name: Crear script relanzar.sh
      copy:
        dest: "{{ script_path }}"
        mode: '0755'
        content: |
          #!/bin/bash
          BASE_PATH="{{ base_path }}"
          echo "$(date): relanzar.sh ejecutado" >> "$BASE_PATH/debug.log"

          check_container() {
            CONTAINER_NAME=$1
            PLAYBOOK_PATH=$2
            STATUS=$(docker inspect -f '{{"{{"}}.State.Running{{"}}"}}' "$CONTAINER_NAME" 2>/dev/null)
            if [ $? -ne 0 ]; then
                                    echo "$(date): ❌ Contenedor '$CONTAINER_NAME' no existe." >>
"$BASE_PATH/monitor.log"
              return
            fi
            if [ "$STATUS" != "true" ]; then
                        echo "$(date): ⚠️ $CONTAINER_NAME está caído. Reiniciando con Ansible..." >>
"$BASE_PATH/monitor.log"
                                    ansible-playbook -i "$BASE_PATH/inventory.ini" "$PLAYBOOK_PATH" >>
"$BASE_PATH/monitor.log" 2>&1
            else
                                echo "$(date): ✅ $CONTAINER_NAME está funcionando correctamente." >>
"$BASE_PATH/monitor.log"
            fi
          }

          check_container "grafana" "$BASE_PATH/monitorizatge3000/monitorizar.yml.yml"
          check_container "cadvisor" "$BASE_PATH/monitorizatge3000/monitorizar.yml.yml"
          check_container "prometheus" "$BASE_PATH/monitorizatge3000/monitorizar.yml"
          check_container "postfix-intern-container" "$BASE_PATH/postfix/contenidor-postfix.yml"
          check_container "filebrowser" "$BASE_PATH/monitorizatge3000/contenidor-filebrowser.yml"
          check_container "ftp-server" "$BASE_PATH/ftp8081/contenidor-ftp.yml"
          check_container "apache_final" "$BASE_PATH/apache8080/crear-contenedor-apache.yml"
          check_container "postgres" "$BASE_PATH/bbdd/contenidor-postgres.yml"
```

Aquesta tasca crea físicament el fitxer relanzar.sh dins del projecte. El seu contingut és un script en bash que comença registrant la seva execució en un fitxer debug.log. Després defineix una funció anomenada check_container, que comprova si un contenidor concret està actiu.

Aquesta comprovació es fa mitjançant docker inspect. Si el contenidor no existeix o no està en execució, s'escriu un missatge al fitxer monitor.log i s'executa el playbook corresponent amb Ansible per tornar a posar-lo en marxa. A continuació, l'script crida aquesta funció per a cadascun dels contenidors definits en el sistema, passant-li el nom del contenidor i la ruta del seu playbook específic. D'aquesta manera, es pot recuperar de forma automàtica qualsevol servei que hagi fallat.

```
- name: Lanzar relanzar.sh en segundo plano con nohup (si no está corriendo)
  shell: |
    if ! pgrep -f "relanzar.sh" > /dev/null; then
      nohup {{ script_path }} >> {{ base_path }}/monitor.log 2>&1 &
    fi
  args:
    executable: /bin/bash
```

Aquesta darrera tasca comprova si el script relanzar.sh ja s'està executant. Si no ho està, llança el procés en segon pla utilitzant nohup, assegurant que es mantingui actiu encara que l'usuari tanqui la sessió. La sortida del procés es redirigeix al fitxer monitor.log, on es poden revisar els missatges d'estat de cada contenidor. D'aquesta manera, el sistema de supervisió s'activa de forma automàtica i contínua, vigilant l'estat dels serveis del projecte sense necessitat de cap acció manual posterior.

# 7 Github

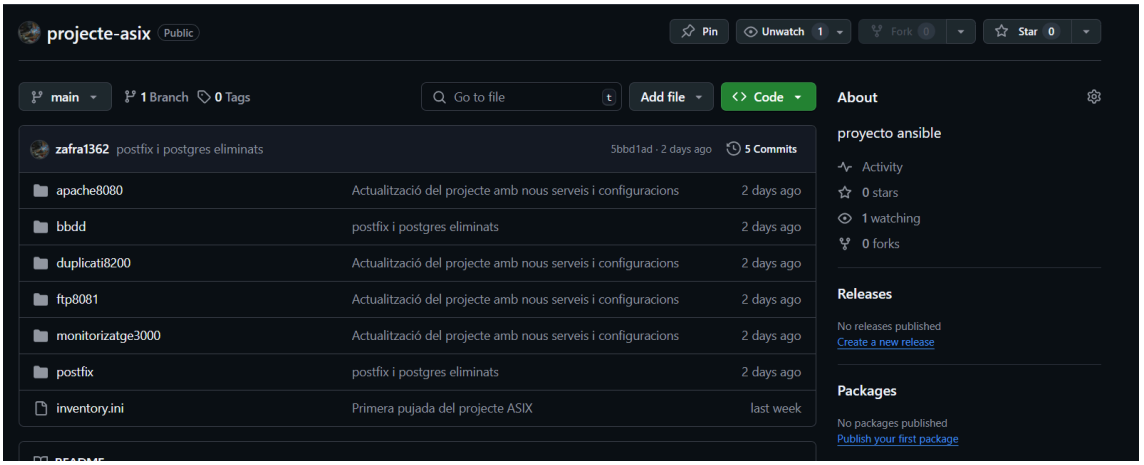
Durant el desenvolupament del projecte, s'ha utilitzat **GitHub** com a eina principal per al control de versions i per mantenir centralitzada tota la configuració, codi i documentació. Aquesta eina ha estat fonamental per a garantir la traçabilitat, recuperació i col·laboració del projecte.

**Objectius d'ús de GitHub**

- **Control de versions**: Per tenir un registre clar de l'evolució dels playbooks i scripts.
- **Recuperació en cas de pèrdua**: GitHub ha permès recuperar tot el projecte després d'una eliminació accidental.
- **Organització per serveis**: Cada servei desplegat (Apache, PostgreSQL, Postfix, FTP, Prometheus, etc.) té el seu propi directori dins del repositori.
- **Documentació centralitzada**: La memòria, índex web i configuracions de Prometheus i Grafana també s'han versionat al repositori.

**Aprenentatge obtingut**

GitHub no només ha estat una eina de còpia de seguretat, sinó que ha ajudat a treballar de manera més organitzada, detectar errors i provar configuracions sense por a perdre informació. A més, ha sigut essencial per a una bona documentació i presentació del projecte final.




# 8 Automatització i idempotència

Els playbooks es dissenyen per ser idempotents, és a dir, poden executar-se tantes vegades com calgui sense modificar l'estat final del sistema ni generar conflictes.

Exemples:

- Si el contenidor ja existeix, no es torna a crear.
- Si el fitxer ja hi és i no ha patit cap canvi, no es torna a copiar.

Aquesta característica és clau per garantir la coherència del sistema i prevenir errors en entorns de producció.

## 8.1 Beneficis d'usar playbooks

- **Repetibilitat**: Els serveis es poden desplegar igual en qualsevol entorn.
- **Escalabilitat**: Es poden aplicar a diversos servidors a la vegada.
- **Mantenibilitat**: Qualsevol canvi s'aplica només modificant el playbook.
- **Documentació viva**: Els *playbooks* actuen com a documentació tècnica del sistema.

## 8.2 Automatització de còpies de seguretat

Per assegurar que les còpies de seguretat es facin de manera constant i sense haver d'intervenir manualment, s'ha posat en marxa un sistema automatitzat mitjançant scripts i la planificació de tasques amb cron.

L'script principal, anomenat *backup-ftp.sh*, s'executa localment i fa tres passos essencials: primer, crea la còpia amb BorgBackup; després, la comprimeix amb un nom únic basat en la data i hora; i finalment, la puja al servidor FTP utilitzant rclone. Aquest procediment garanteix

que cada còpia es desa de manera separada i permet mantenir un registre ordenat dels backups.

Perquè s'executi automàticament, s'ha afegit una entrada al cron del sistema, que llança l'script cada dia a una hora concreta. Així s'assegura que sempre hi hagi còpies actualitzades, fet que simplifica la recuperació en cas d'error o pèrdua de dades. Aquesta programació es pot ajustar fàcilment segons les necessitats del sistema o de l'usuari.

En conjunt, és una solució de còpia de seguretat completa, eficient i totalment automatitzada, que encaixa perfectament amb la infraestructura ja desplegada.

## 8.3 Automatització dels serveis

Aquest playbook ha estat creat amb l'objectiu d'automatitzar tant la supervisió com la recuperació dels contenidors Docker que formen part de la infraestructura del projecte. La seva funció clau és garantir que, si algun contenidor s'atura, es posi en marxa automàticament el playbook corresponent per restablir-lo, tot això sense cap intervenció manual.

Per aconseguir-ho, es genera un script anomenat *relanzar.sh*, encarregat de revisar periòdicament l'estat de contenidors específics (com grafana, cadvisor, prometheus, postfix, filebrowser, entre d'altres). Aquest script utilitza la comanda docker inspect per comprovar si els contenidors estan actius i, si detecta que algun ha deixat de funcionar, executa el seu playbook associat amb Ansible per reiniciar-lo.

Un cop creat, l'script es posa en marxa en segon pla mitjançant la comanda nohup, la qual permet que continuï actiu encara que l'usuari tanqui la sessió. A més, s'inclou una comprovació per evitar que hi hagi més d'una instància del mateix script funcionant alhora, cosa que prevé duplicacions innecessàries.

Cada execució queda registrada en un fitxer de log anomenat *monitor.log*, que facilita el seguiment dels contenidors: tant els que funcionen correctament com els que han estat reiniciats. L'script s'ha concebut perquè operi indefinidament en segon pla, assegurant una supervisió constant. Tot i això, si es vol ajustar la freqüència de comprovació (per exemple, cada minut), es pot fer servir cron o afegir un bucle amb sleep dins del propi script.

Aquest mecanisme garanteix l'alta disponibilitat dels serveis desplegats, assegurant que qualsevol fallada o parada accidental es resolgui de forma automàtica a través del seu playbook original, sense que calgui vigilància contínua per part de l'usuari.

# 9 Conclusions

## 9.1 Conclusions generals del projecte

El desenvolupament d'aquest projecte ha estat una oportunitat molt valuosa per posar en pràctica tant els coneixements teòrics com tècnics adquirits al llarg del cicle formatiu. He pogut dissenyar, desplegar i fer el seguiment d'una infraestructura complexa utilitzant eines actuals com Ansible i Docker, dins d'un entorn controlat però amb situacions i reptes molt similars als que es troben en l'àmbit professional.

Des del punt de vista acadèmic, aquest projecte m'ha ajudat a aprofundir en:

- La creació i gestió de serveis contenitzats amb Docker.
- L'automatització de la infraestructura mitjançant Ansible.

- El monitoratge i anàlisi de rendiment amb Prometheus i Grafana.
- La gestió d'incidències, recuperació d'errors i manteniment continu dels serveis.

Pel que fa a l'àmbit professional, m'ha servit per:

- Desenvolupar una manera de treballar més estructurada, seguint metodologies àgils.
- Enfrontar-me a problemes reals i adaptar-me als imprevistos durant el desplegament.
- Consolidar coneixements pràctics sobre tecnologies molt utilitzades avui dia en el món laboral.
- Guanyar seguretat a l'hora d'afrontar reptes relacionats amb la infraestructura, la seguretat i l'escalabilitat.

## 9.2 Consecució dels objectius

**Automatitzar el desplegament i la gestió de serveis amb Ansible**
**Assolit.**
S'han desenvolupat diversos playbooks per desplegar serveis com Apache, PostgreSQL, Postfix, Prometheus, Grafana, entre d'altres. Gràcies a això, es disposa d'una gestió totalment automatitzada i reproduïble.

**Utilitzar Docker per contenitzar serveis en maquetes**
**Assolit.**
Tots els serveis s'han executat dins de contenidors Docker, aprofitant la seva flexibilitat i portabilitat per fer proves i recrear entorns realistes.

**Simular fallades i comprovar la capacitat d'autorecuperació**
**Assolit.**
S'han forçat fallades de manera manual als contenidors per validar que Ansible pogués reiniciar-los o substituir-los, mitjançant restart_policy: always i tasques correctores executades remotament.

**Desplegar i escalar els serveis al núvol**
**Assolit.**
El projecte s'ha desplegat sobre màquines virtuals en un entorn que simula un núvol privat, amb capacitat per escalar serveis i fer-ne el seguiment remotament.

**Monitoritzar els serveis amb Prometheus i Grafana**
**Assolit.**
S'han integrat Prometheus i Grafana per recollir mètriques dels contenidors (amb cAdvisor) i configurar alertes amb Alertmanager. També s'han creat panells per visualitzar l'estat dels serveis.

**Fer còpies de seguretat automàtiques amb Duplicati**
**No assolit completament**.
Tot i que s'ha desplegat correctament el contenidor de Duplicati i la seva configuració és funcional, s'han trobat problemes d'accés al directori FTP. La connexió amb el servei vsftpd fallava per permisos denegats. Aquesta funcionalitat es considera pendent de revisar en futures iteracions.

## 9.3 Valoració de la metodologia i planificació

Tot i comptar amb una metodologia clara i ben estructurada, no sempre s'ha pogut seguir amb la constància prevista, principalment per qüestions laborals i personals. El fet de treballar als matins ha estat un condicionant rellevant, ja que m'ha dificultat trobar franges horàries llargues i productives per dedicar al projecte. Això ha comportat haver de treballar en estones soltes o durant caps de setmana, cosa que ha impactat tant en el ritme de desenvolupament com en la capacitat de resposta davant d'imprevistos tècnics.

S'han hagut d'introduir ajustos a la planificació inicial, sobretot pel que fa a les fases de prova i resolució d'errors. Alguns serveis, com Duplicati amb connexió FTP, han requerit molt més temps del previst per problemes relacionats amb permisos, configuracions complexes i limitacions pròpies del programari.

Pel que fa a la coordinació i seguiment del projecte:

L'ús de Git i GitHub per al control de versions ha estat força efectiu, tot i que a l'inici es van produir dificultats amb la pujada del projecte, a causa de conflictes i l'estructura de carpetes, que es van haver de resoldre manualment.

En relació amb el pressupost, no s'han superat els límits establerts, ja que totes les eines utilitzades han estat de codi obert i gratuïtes.

## 9.4 Problemes sorgits i solucions

Durant el desenvolupament del projecte han anat apareixent diversos entrebancs tècnics i logístics que han afectat tant el ritme com l'orientació de la feina. Malgrat tot, aquests problemes han estat una font d'aprenentatge molt valuosa i m'han ajudat a desenvolupar habilitats pràctiques per resoldre incidències reals.

Un dels moments més crítics va ser quan vaig **perdre** accidentalment el projecte local. Per una manipulació errònia dels fitxers, es va esborrar una part del directori principal. Per sort, havia estat fent servir **GitHub** com a sistema de control de versions, cosa que em va salvar: vaig poder recuperar-ho tot —playbooks, configuracions i documents inclosos. Aquesta situació em va fer entendre de debò com n'és d'essencial treballar amb còpies de seguretat al núvol i tenir un bon **control de versions**.

També vaig topar amb **problemes** en la gestió dels ports dels contenidors **Docker**. Diversos serveis volien utilitzar el mateix port per defecte (com el 8080), cosa que generava conflictes i impedia l'execució normal. Per solucionar-ho, vaig revisar tots els ports exposats i els vaig reorganitzar, assignant-ne d'exclusius a cada servei. Aquest procés em va ensenyar a ser més previsor amb l'assignació de recursos en entorns contenitzats.

Amb **Postfix**, vaig voler configurar un relay **extern** fent servir servidors **SMTP** com **Gmail** o **Outlook**. Tot i provar diverses configuracions amb **SASL** i **TLS**, no vaig aconseguir establir una connexió estable, probablement per les polítiques de seguretat estrictes dels proveïdors. Al final, vaig optar per muntar un relay intern funcional per enviar correus entre contenidors, que sí que va funcionar sense problemes.

I pel que fa a les **còpies de seguretat**, vaig decidir provar **Duplicati** integrat amb un servei **FTP** propi. Tot i aconseguir posar en marxa el servidor i connectar-lo amb Duplicati, el sistema no acabava de funcionar bé: mostrava errors de permisos i autenticació. Ara per ara l'he deixat operatiu de manera parcial i, de cara a futures iteracions, tinc previst explorar alternatives més robustes com BorgBackup o Restic.

## 9.5 Visió de futur

Tot i que el projecte ha aconseguit complir la major part dels objectius inicials, encara queden algunes línies de treball obertes que podrien desenvolupar-se en una fase posterior per millorar i ampliar l'entorn desplegat.

Una de les primeres tasques pendents seria reforçar el sistema de còpies de seguretat. Tot i haver desplegat **Duplicati** amb un servidor **FTP** com a destinació, no es va arribar a tenir un sistema del tot estable i fiable. En el futur, valdria la pena explorar solucions més robustes com **Restic, BorgBackup o Bacula**, i fins i tot plantejar la integració de còpies remotes amb xifratge automàtic i control d'accés per rols.

També ha quedat per completar la implementació d'un sistema de notificacions d'alertes, utilitzant Alertmanager i Prometheus, per enviar avisos per correu o per missatgeria (**Telegram**, **Slack**...) en cas de caiguda d'algun contenidor o rendiment anòmal. **Alertmanager** ja està configurat, però la integració amb canals externs no es va poder completar per manca de temps.

Una altra millora interessant seria afegir un sistema centralitzat de gestió d'usuaris, com **Keycloak**, que permetés una autenticació comuna en aplicacions com **Filebrowser** o **Grafana**. Aquesta capa aportaria més seguretat i un millor control dels accessos.

Pel que fa al desplegament, actualment tot funciona sobre contenidors **Docker** en una màquina virtual local. Un pas natural seria migrar-ho a un entorn **Kubernetes**, cosa que facilitaria l'escalabilitat, la recuperació automàtica de pods i una gestió de serveis molt més avançada.

# 10 Glossari

- **Ansible**: Eina d'automatització que permet configurar sistemes, desplegar aplicacions i coordinar tasques d'administració en diversos servidors mitjançant fitxers YAML (playbooks).

- **Docker**: Plataforma de contenidors que permet empaquetar aplicacions amb les seves dependències en un entorn lleuger, portable i aïllat, facilitant-ne l'execució en diferents sistemes

- **Contenidor**: Entorn virtual lleuger que comparteix el nucli del sistema amfitrió però funciona de manera aïllada respecte altres aplicacions.

- **Playbook**: Fitxer YAML utilitzat per Ansible que defineix un conjunt de tasques automatitzades a executar en un o més nodes.

- **Prometheus**: Eina de monitoratge que recull mètriques de sistemes i serveis, i permet generar alertes segons condicions definides.

- **Grafana**: Plataforma de visualització de dades que permet crear dashboards interactius a partir de fonts com Prometheus.

- **PostgreSQL**: Sistema de gestió de bases de dades relacional, de codi obert, conegut per la seva robustesa i flexibilitat.

- **Postfix**: Agent de transferència de correu (MTA) per enviar correus electrònics des de sistemes Linux.

- **FTP (File Transfer Protocol)**: Protocol per transferir fitxers entre sistemes a través d'una xarxa.

- **Duplicati**: Eina de còpies de seguretat amb suport per xifratge i destinacions com FTP o serveis al núvol.

- **cAdvisor**: Eina de Google per monitorar l'ús de recursos (CPU, memòria, I/O, etc.) dels contenidors Docker.

- **VSFTPD (Very Secure FTP Daemon)**: Servidor FTP segur i estable, habitual en entorns de producció.

- **Filebrowser**: Aplicació web que facilita la gestió gràfica i intuïtiva de fitxers dins d'un sistema.

# 11 Bibliografia

Ansible Documentation. Official documentation. Disponible a: https://docs.ansible.com/ (Consulta: 14 de març de 2025).

An Introduction to Configuration Management with Ansible. DigitalOcean Conceptual Guide. Disponible a: https://www.digitalocean.com/community/conceptual-articles/an-introduction-to-configuration-management-with-ansible (Consulta: 16 de març de 2025).

Docker Documentation. Official documentation. Disponible a: https://docs.docker.com/ (Consulta: 18 de març de 2025).

A Docker Tutorial for Beginners. Docker Curriculum. Disponible a: https://docker-curriculum.com/ (Consulta: 19 de març de 2025).

Prometheus Documentation. Official documentation. Disponible a: https://prometheus.io/docs/ (Consulta: 23 de març de 2025).

What is Prometheus Monitoring? A Beginner's Guide. Better Stack Guide. Disponible a: https://betterstack.com/community/guides/monitoring/prometheus/ (Consulta: 25 de març de 2025).

Grafana Documentation. Official documentation. Disponible a: https://grafana.com/docs/grafana/latest/ (Consulta: 28 de març de 2025).

Grafana Tutorial: A Detailed Guide to Your First Dashboard. SentinelOne Blog. Disponible a: https://www.sentinelone.com/blog/grafana-tutorial-detailed-guide-dashboard/ (Consulta: 30 de març de 2025).

Postfix Documentation. Official documentation. Disponible a: https://www.postfix.org/documentation.html (Consulta: 2 d'abril de 2025).

How To Install and Configure Postfix on Ubuntu 22.04. DigitalOcean Tutorial. Disponible a: https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-postfix-on-ubuntu-22-04 (Consulta: 4 d'abril de 2025).

PostgreSQL Documentation. Official documentation. Disponible a: https://www.postgresql.org/docs/current/ (Consulta: 6 d'abril de 2025).

How To Install and Use PostgreSQL on Ubuntu 22.04. DigitalOcean Tutorial. Disponible a: https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-22-04 (Consulta: 7 d'abril de 2025).

File Browser Documentation. Official documentation. Disponible a: https://filebrowser.org/ (Consulta: 10 d'abril de 2025).

How to Access Server Files in a Web Browser with filebrowser. Tony Teaches Tech Tutorial. Disponible a: https://tonyteaches.tech/filebrowser-tutorial/ (Consulta: 11 d'abril de 2025).

Duplicati Documentation. Official documentation. Disponible a: https://docs.duplicati.com/ (Consulta: 16 d'abril de 2025).

Install and Use Duplicati to Back Up Files on Ubuntu 20.04. Snapshooter Guide. Disponible a: https://snapshooter.com/learn/linux/duplicati (Consulta: 17 d'abril de 2025).

cAdvisor (Container Advisor). GitHub Official Repo. Disponible a: https://github.com/google/cadvisor (Consulta: 20 d'abril de 2025).

cAdvisor Tutorial. Kubecost Guide. Disponible a: https://www.kubecost.com/kubernetes-devops-tools/cadvisor/ (Consulta: 21 d'abril de 2025).

vsftpd (Very Secure FTP Daemon). Official documentation. Disponible a: https://security.appspot.com/vsftpd.html (Consulta: 24 d'abril de 2025).

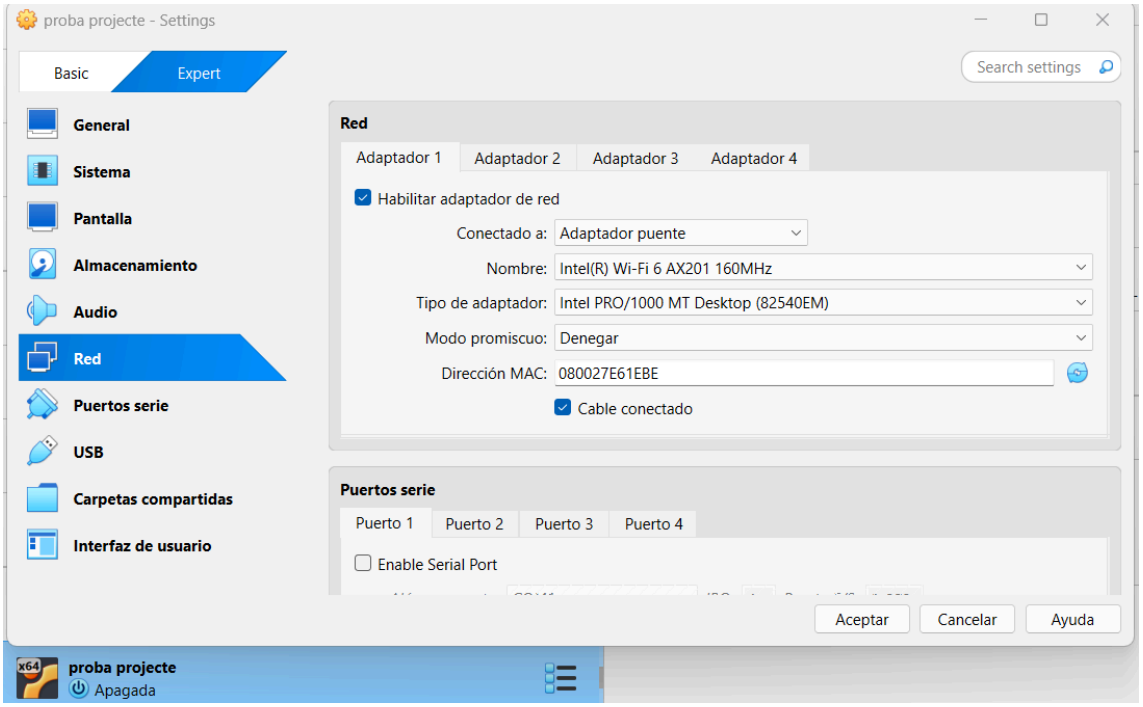
How To Set Up vsftpd for a User's Directory on Ubuntu 20.04. DigitalOcean Tutorial. Disponible a: https://www.digitalocean.com/community/tutorials/how-to-set-up-vsftpd-for-a-user-s-directory-on-ubuntu-20-04 (Consulta: 27 d'abril de 2025).

# 12 Annexos

## Manual d'usuari

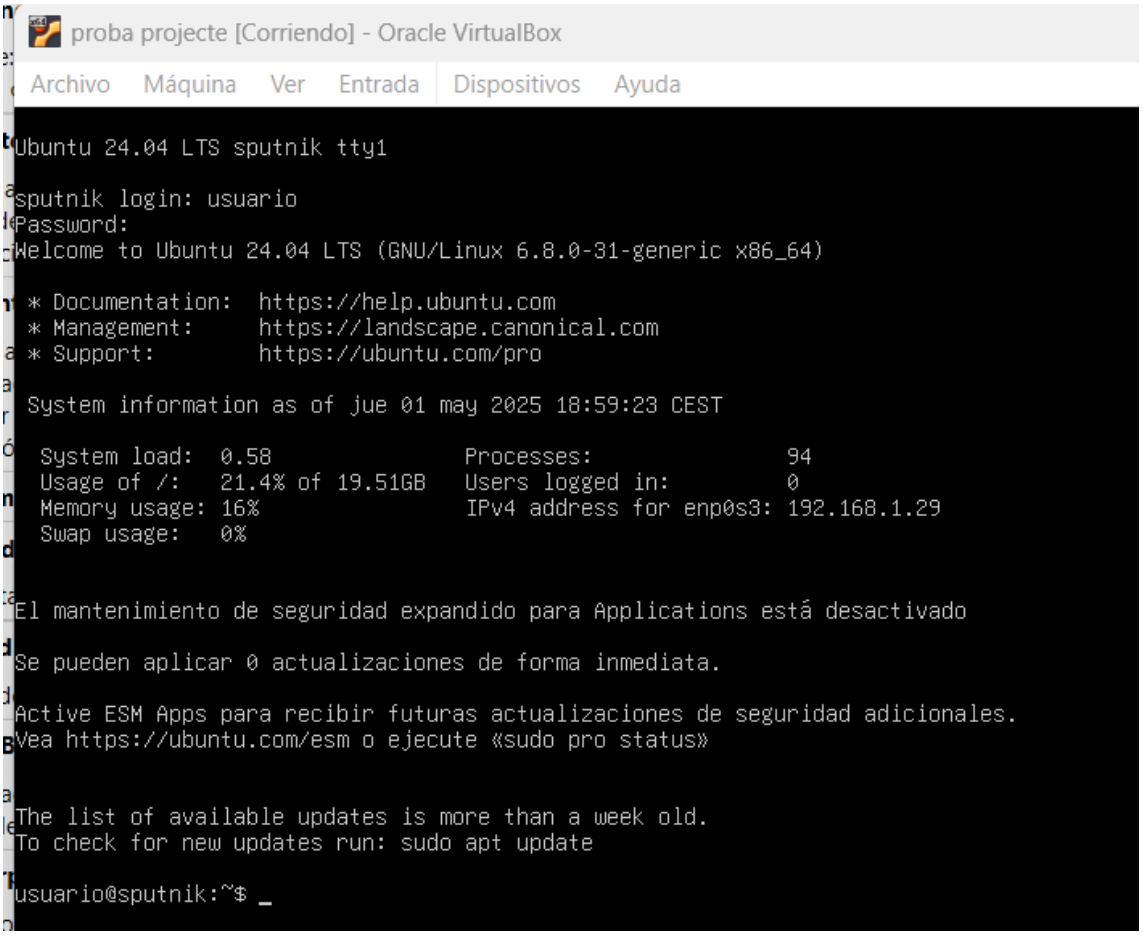
### Crear una màquina virtual

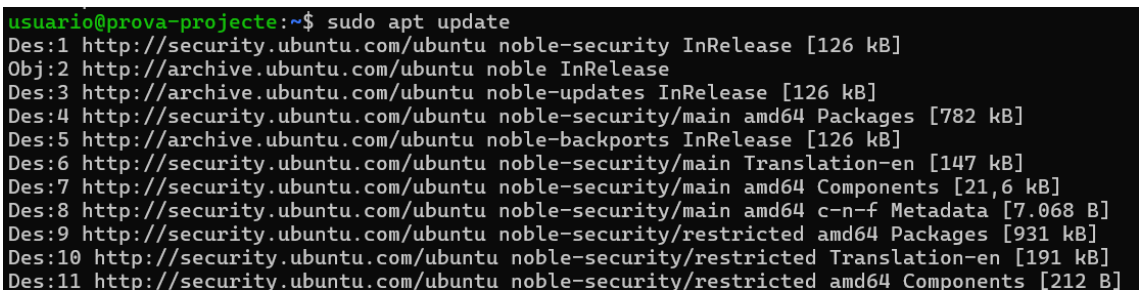
Crearem una màquina virtual server o desktop, el que es prefereixi i la posarem en adaptador pont per tenir connexió.



## Iniciarla i actualitzar

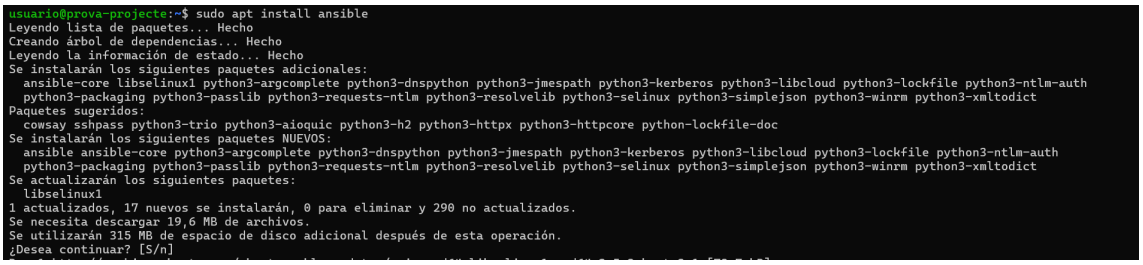
Iniciem la màquina virtual i fem **sudo apt update** per agafar totes les actualitzacions.



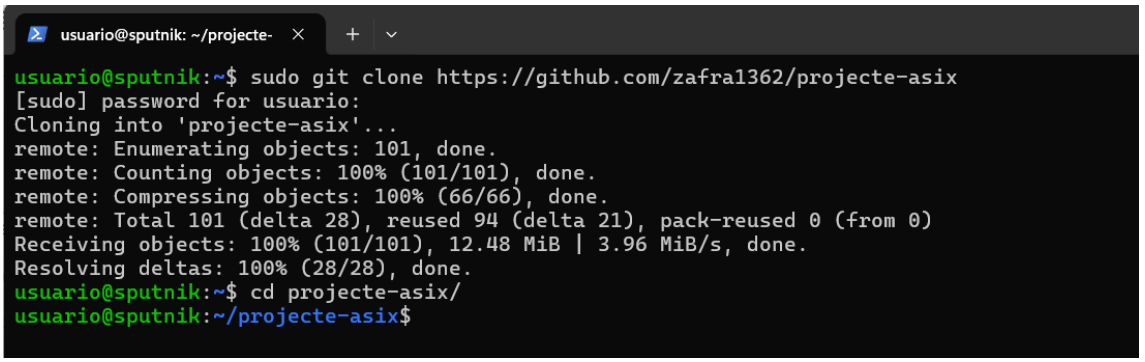




## Instalar ansible i docker

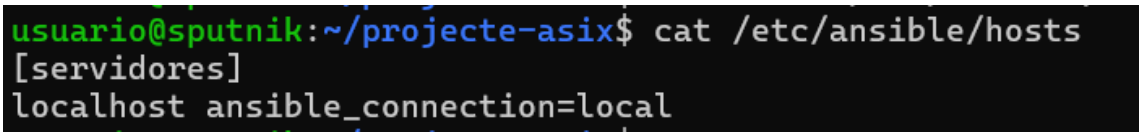
Instal·lem Ansible amb un simple sudo apt install ansible



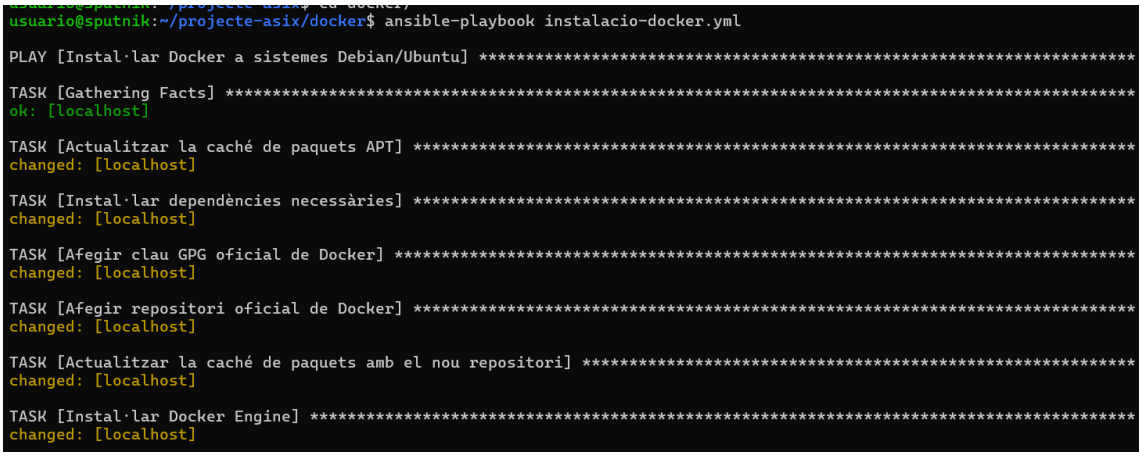
Clonem el meu repositori



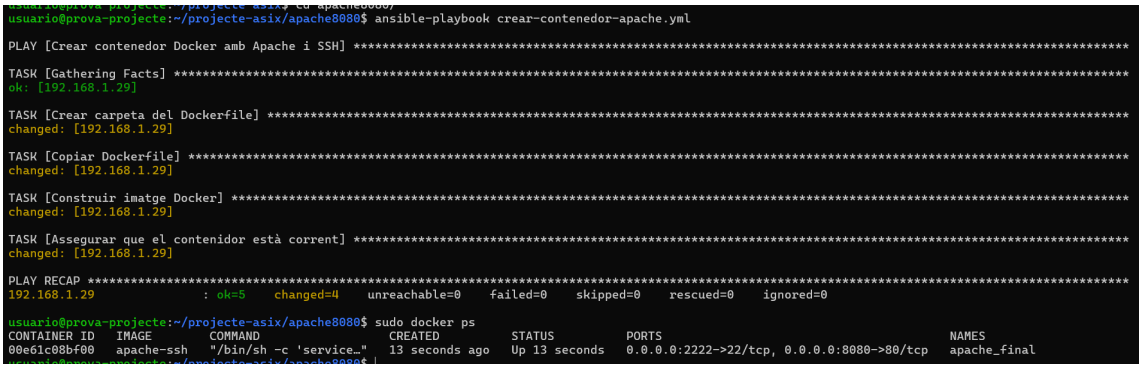

Utilitzarem el contingut del **inventory.ini** per posar-ho a **/etc/ansible/hosts** això indica que al grup **[servidores]** es troba la nostre máquina.




Per començar amb la instal·lació llençarem el playbook per instal·lar **docker**



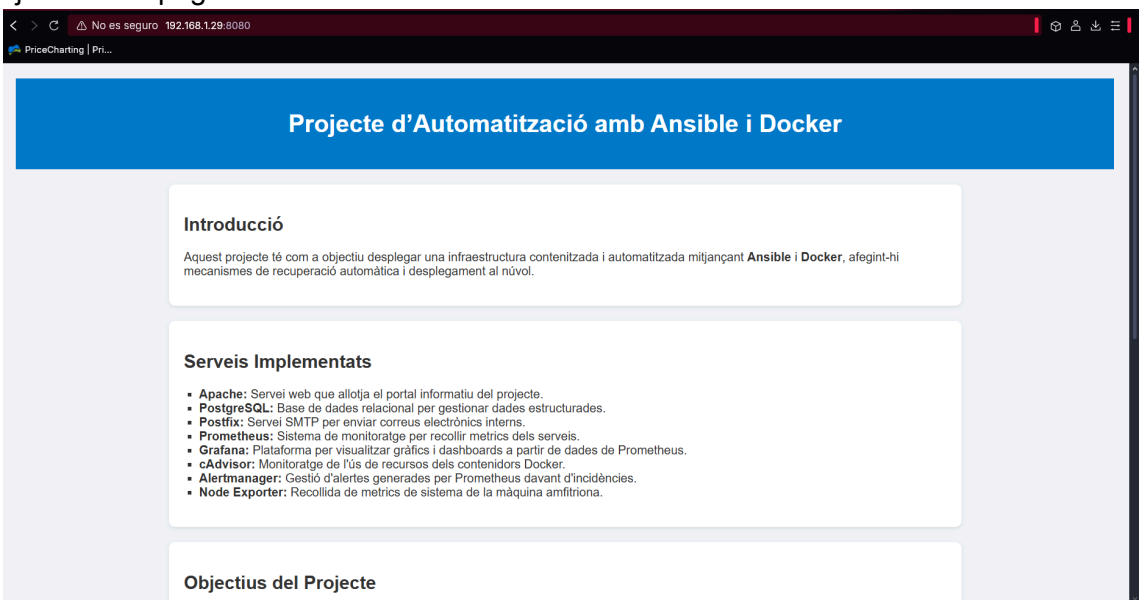
Una vegada creat llançem el playbook d'apache i al finalitzar ja el tenim



Injectem l'index

```
usuario@prova-projecte:~/projecte-asix/apache8080$ ansible-playbook inyectar-web-contenedor.yml

PLAY [Injectar index.html dins del contenidor Apache] ***************************************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************************************************************************
ok: [192.168.1.29]

TASK [Crear directori dins el contenidor si no existeix] ************************************************************************************************************************
changed: [192.168.1.29]

TASK [Copiar index.html al contenidor apache_final] *****************************************************************************************************************************
changed: [192.168.1.29]

TASK [Reiniciar Apache dins del contenidor] *************************************************************************************************************************************
changed: [192.168.1.29]

PLAY RECAP **********************************************************************************************************************************************************************
192.168.1.29               : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

I ja tenim la página




Es llença seguidament el playbook de postgres

```
usuario@prova-projecte:~/projecte-asix/bbdd$ ansible-playbook contenidor-postgres.yml

PLAY [Crear contenidor PostgreSQL i afegir dades] *******************************************************************************************************************************

TASK [Gathering Facts] **********************************************************************************************************************************************************
ok: [192.168.1.29]

TASK [Iniciar contenidor PostgreSQL] ********************************************************************************************************************************************
changed: [192.168.1.29]

TASK [Esperar que PostgreSQL estigui llest] *************************************************************************************************************************************
ok: [192.168.1.29]

TASK [Crear fitxer SQL temporal] ************************************************************************************************************************************************
changed: [192.168.1.29]

TASK [Copiar fitxer SQL dins del contenidor] ************************************************************************************************************************************
changed: [192.168.1.29]

TASK [Executar script SQL al contenidor] ****************************************************************************************************************************************
changed: [192.168.1.29]

PLAY RECAP **********************************************************************************************************************************************************************
192.168.1.29               : ok=6    changed=4    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

usuario@prova-projecte:~/projecte-asix/bbdd$ sudo docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED              STATUS              PORTS                                                  NAMES
c0b9952b913c   postgres:15    "docker-entrypoint.s…"   About a minute ago   Up About a minute   0.0.0.0:5432->5432/tcp                                 postgres_test
00e61c08bf00   apache-ssh     "/bin/sh -c 'service…"   36 minutes ago       Up 36 minutes       0.0.0.0:2222->22/tcp, 0.0.0.0:8080->80/tcp             apache_final
```

Es verifica que s'han creat les taules

```
postgres=# \l
empresa|postgres|UTF8|en_US.utf8|en_US.utf8||libc|
postgres|postgres|UTF8|en_US.utf8|en_US.utf8||libc|
template0|postgres|UTF8|en_US.utf8|en_US.utf8||libc|=c/postgres
postgres=CTc/postgres
template1|postgres|UTF8|en_US.utf8|en_US.utf8||libc|=c/postgres
postgres=CTc/postgres
```

Seguidament s'instal.la el servei de correu Postfix

```
usuario@prova-projecte:~/projecte-asix/postfix$ ansible-playbook contenidor-postfix.yml

PLAY [Crear contenidor Docker amb Postfix relay intern] ************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
ok: [192.168.1.29]

TASK [Crear carpeta /opt/postfix-intern/etc] ***********************************************************************************************************************
changed: [192.168.1.29]

TASK [Crear fitxer /etc/mailname] **********************************************************************************************************************************
changed: [192.168.1.29]

TASK [Crear fitxer main.cf per relay intern] ***********************************************************************************************************************
changed: [192.168.1.29]

TASK [Crear Dockerfile per Postfix bàsic] **************************************************************************************************************************
changed: [192.168.1.29]

TASK [Construir imatge Docker de Postfix bàsic] ********************************************************************************************************************
changed: [192.168.1.29]

TASK [Llançar contenidor Postfix intern] ***************************************************************************************************************************
changed: [192.168.1.29]

PLAY RECAP *********************************************************************************************************************************************************
192.168.1.29               : ok=7    changed=6    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

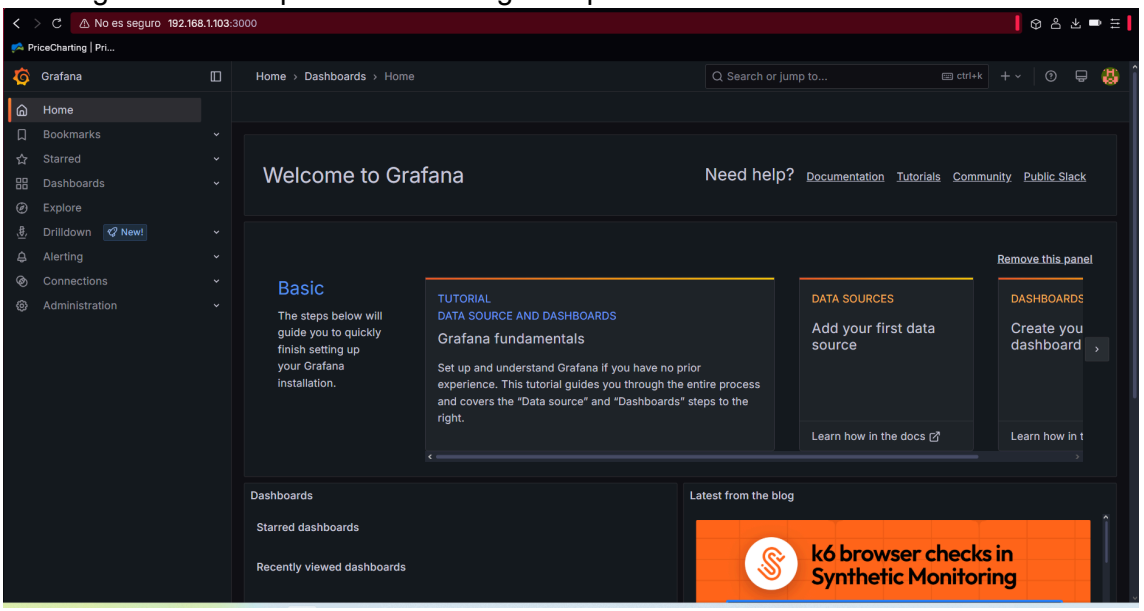
Seguidament enviem un correu i verifiquem que hi a arribat

```
root@a45096ca6bff:/# mailq
Mail queue is empty
root@a45096ca6bff:/# echo "Hola, este es un mensaje de prueba" | mail -s "Prueba Postfix" root@elpuig.example.com
root@a45096ca6bff:/# mailq
Mail queue is empty
root@a45096ca6bff:/# exit
exit
```

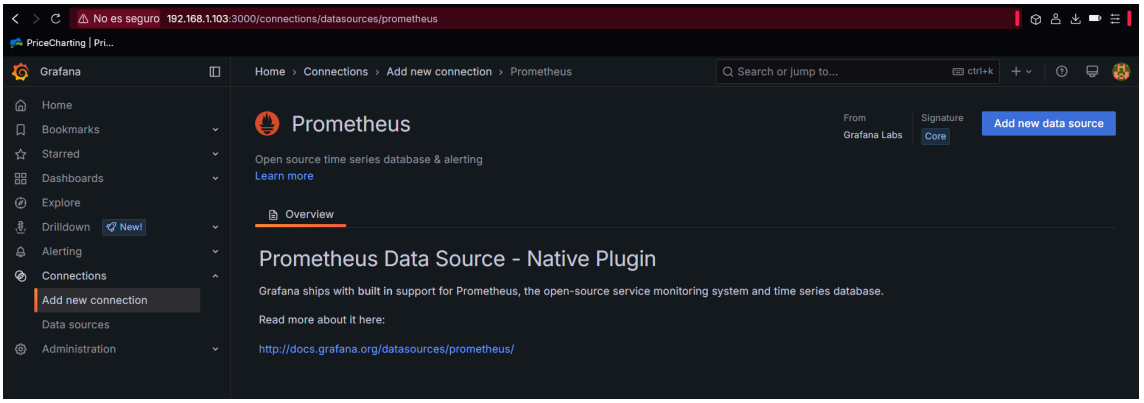
Ara pasarem a instalar el sistema de monitoritzatge

```
usuario@prova-projecte: ~   Windows PowerShell

usuario@prova-projecte:~/projecte-asix/monitorizatge3000$ ansible-playbook monitoritzar.yml

PLAY [Llançar Prometheus, cAdvisor i Grafana] **********************************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************************************************
The authenticity of host '192.168.1.103 (192.168.1.103)' can't be established.
ED25519 key fingerprint is SHA256:LaXOwkx15AQMwvSNuypfrroJq1yKr6EagzI1Xg4nrLk.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:1: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? fatal: [192.168.1.29]: UNREACHABLE! => {"changed": false, "msg": "Failed to connect to
the host via ssh: ssh: connect to host 192.168.1.29 port 22: No route to host", "unreachable": true}
yes
ok: [192.168.1.103]

TASK [Crear directori de configuració Prometheus] ******************************************************************************************************************
ok: [192.168.1.103]

TASK [Crear fitxer prometheus.yml amb Alertmanager i regles] *******************************************************************************************************
changed: [192.168.1.103]

TASK [Crear fitxer de regles d'alertes alert_rules.yml] ************************************************************************************************************
ok: [192.168.1.103]

TASK [Llançar contenidor Prometheus] *******************************************************************************************************************************
ok: [192.168.1.103]

TASK [Eliminar contenidor cAdvisor si existeix] ********************************************************************************************************************
changed: [192.168.1.103]

TASK [Llançar contenidor cAdvisor] *********************************************************************************************************************************
changed: [192.168.1.103]

TASK [Crear volum Docker per Grafana si no existeix] ***************************************************************************************************************
ok: [192.168.1.103]

TASK [Llançar contenidor Grafana] **********************************************************************************************************************************
ok: [192.168.1.103]

PLAY RECAP *********************************************************************************************************************************************************
192.168.1.103              : ok=9    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.1.29               : ok=0    changed=0    unreachable=1    failed=0    skipped=0    rescued=0    ignored=0
```

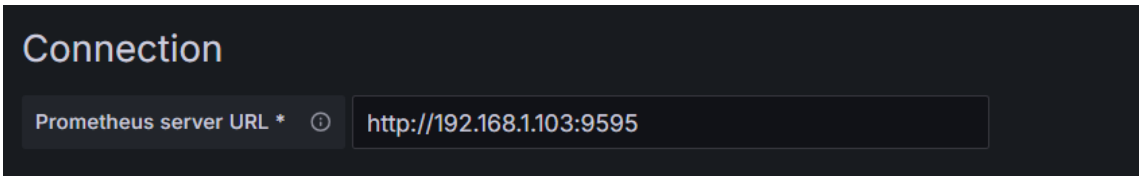
Una vegada s'instl·la posarem al navegador ip:3000



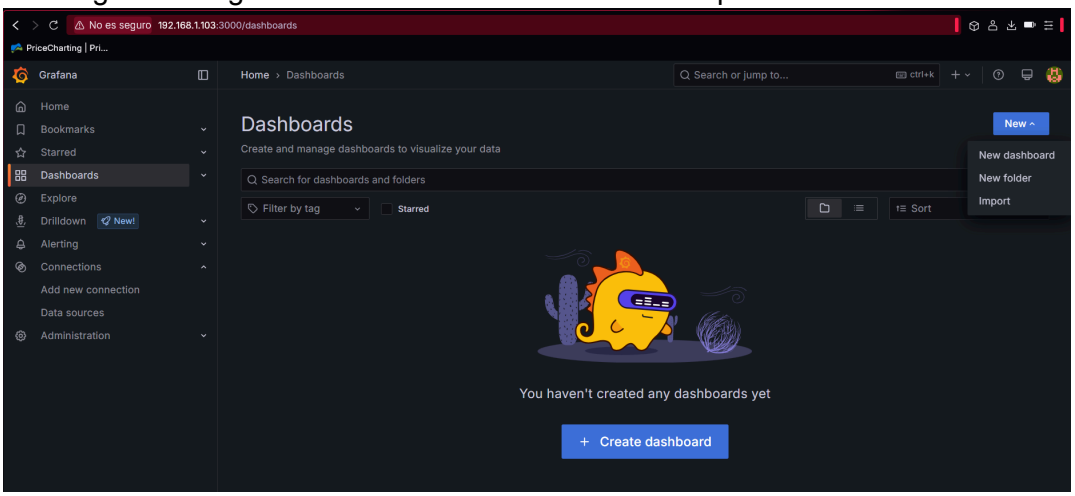

Seguidament afegirem prometheus connections->add new connection e instal·lem Prometheus



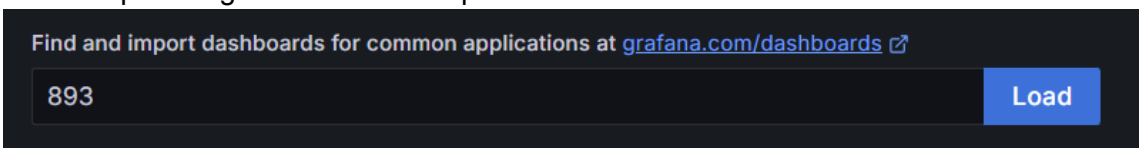

Seguidament afegim la nova instancia de Prometheus, nomes afegim la direccio del Prometheus



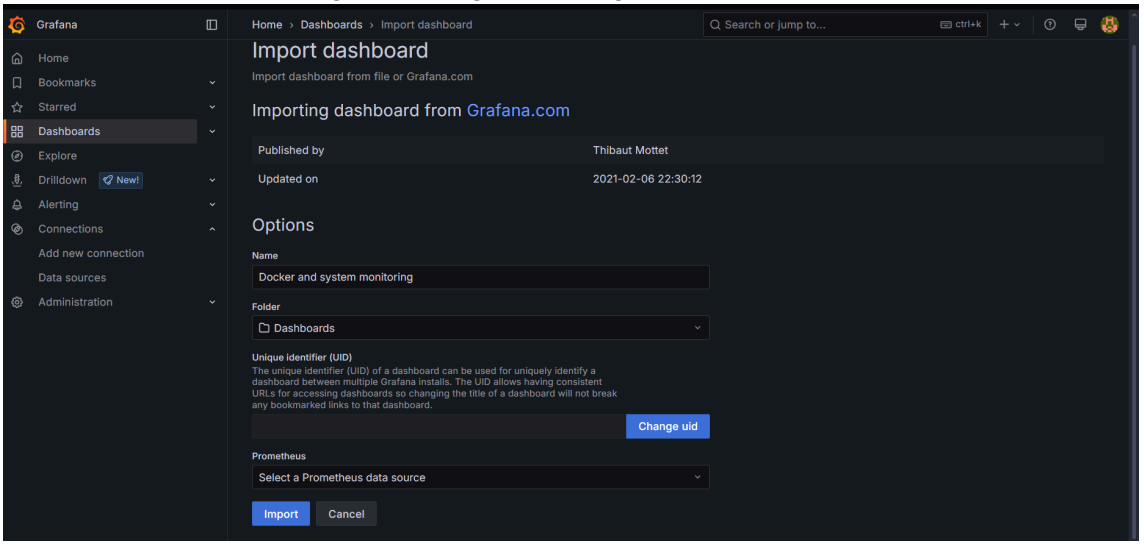
Una vegada configurat anirem a dashboards->new->import



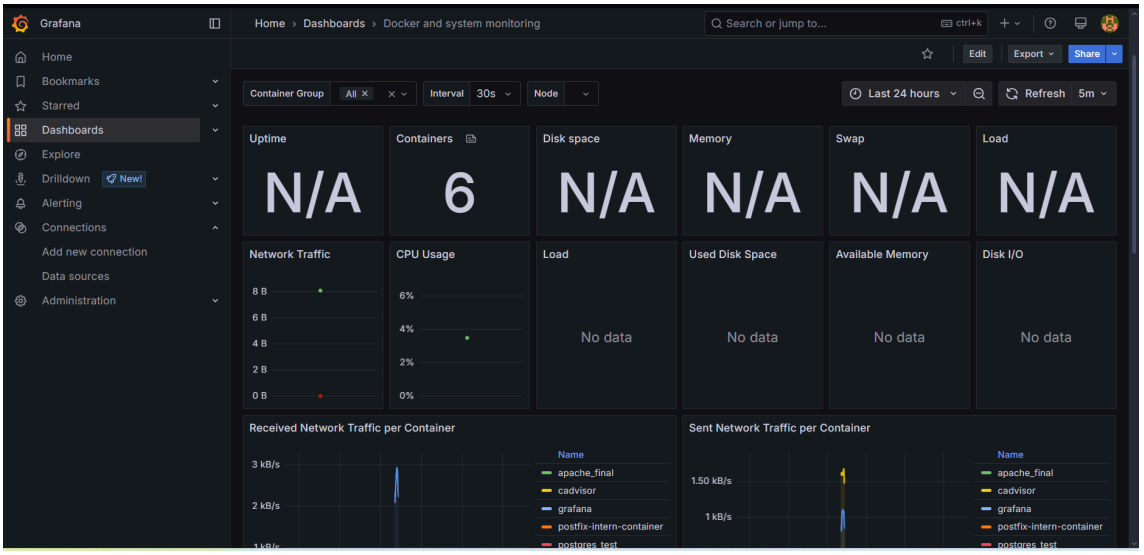

Dins d'import afegirem el codi 893 que es l'id del Prometheus




Li donem a Load i una vegada carregat el configurem al nostre mode



Gràcies a Prometheus i cAdvisor podem visualitzar-ho tot a Grafana, incloent nous contenidors




Quan s'hagi intal·lat tots els serveis llençarem el playbook per aixecar els contenidors

```
usuario@sputnik:~/projecte-asix/levantar-contenedores$ ansible-playbook levantar.yml

PLAY [Crear y lanzar relanzar.sh en segundo plano] ******************************************************************

TASK [Gathering Facts] **********************************************************************************************
ok: [localhost]

TASK [Crear script relanzar.sh] *************************************************************************************
changed: [localhost]

TASK [Lanzar relanzar.sh en segundo plano con nohup (si no está corriendo)] *****************************************
changed: [localhost]

PLAY RECAP **********************************************************************************************************
localhost                  : ok=3    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

usuario@sputnik:~/projecte-asix/levantar-contenedores$
```

Aquest deixarà un log de execució i el que mostra a la carpeta del projecte pers saber quan s'executa (si es vol programar es farà amb cron)

```
usuario@sputnik: ~/projecte-
usuario@sputnik:~/projecte-asix$ sudo ./relanzar.sh
usuario@sputnik:~/projecte-asix$ cat
admin/              borgcs/             ftp8081/            levantar-contenedores/ postfix/
apache8080/         debug.log           .git/               monitorizatge3000/     relanzar.sh
bbdd/               docker/             inventory.ini       monitor.log
usuario@sputnik:~/projecte-asix$ cat monitor.log
mar 20 may 2025 19:53:00 CEST: ❌ Contenedor 'grafana' no existe.
mar 20 may 2025 19:53:00 CEST: ❌ Contenedor 'cadvisor' no existe.
mar 20 may 2025 19:53:00 CEST: ❌ Contenedor 'prometheus' no existe.
mar 20 may 2025 19:53:01 CEST: ❌ Contenedor 'postfix-intern-container' no existe.
mar 20 may 2025 19:53:01 CEST: ❌ Contenedor 'filebrowser' no existe.
mar 20 may 2025 19:53:01 CEST: ❌ Contenedor 'ftp-server' no existe.
mar 20 may 2025 19:53:01 CEST: ❌ Contenedor 'apache_final' no existe.
mar 20 may 2025 19:53:01 CEST: ❌ Contenedor 'postgres' no existe.
mar 20 may 2025 19:53:32 CEST: ❌ Contenedor 'grafana' no existe.
mar 20 may 2025 19:53:32 CEST: ❌ Contenedor 'cadvisor' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'prometheus' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'postfix-intern-container' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'filebrowser' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'ftp-server' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'apache_final' no existe.
mar 20 may 2025 19:53:33 CEST: ❌ Contenedor 'postgres' no existe.
mar 20 may 2025 20:01:28 CEST: ❌ Contenedor 'grafana' no existe.
mar 20 may 2025 20:01:28 CEST: ❌ Contenedor 'cadvisor' no existe.
mar 20 may 2025 20:01:28 CEST: ❌ Contenedor 'prometheus' no existe.
mar 20 may 2025 20:01:28 CEST: ✅ postfix-intern-container está funcionando correctamente.
mar 20 may 2025 20:01:28 CEST: ✅ filebrowser está funcionando correctamente.
mar 20 may 2025 20:01:28 CEST: ✅ ftp-server está funcionando correctamente.
mar 20 may 2025 20:01:28 CEST: ✅ apache_final está funcionando correctamente.
mar 20 may 2025 20:01:28 CEST: ✅ postgres está funcionando correctamente.
mar 20 may 2025 20:05:02 CEST: ✅ grafana está funcionando correctamente.
mar 20 may 2025 20:05:02 CEST: ✅ cadvisor está funcionando correctamente.
mar 20 may 2025 20:05:02 CEST: ✅ prometheus está funcionando correctamente.
mar 20 may 2025 20:05:02 CEST: ✅ postfix-intern-container está funcionando correctamente.
mar 20 may 2025 20:05:02 CEST: ✅ filebrowser está funcionando correctamente.
mar 20 may 2025 20:05:03 CEST: ✅ ftp-server está funcionando correctamente.
mar 20 may 2025 20:05:03 CEST: ✅ apache_final está funcionando correctamente.
mar 20 may 2025 20:05:03 CEST: ✅ postgres está funcionando correctamente.
usuario@sputnik:~/projecte-asix$
```

```
usuario@sputnik:~/projecte-asix$ cat debug.log
mar 20 may 2025 19:53:00 CEST: relanzar.sh ejecutado
mar 20 may 2025 19:53:32 CEST: relanzar.sh ejecutado
mar 20 may 2025 20:01:27 CEST: relanzar.sh ejecutado
mar 20 may 2025 20:05:01 CEST: relanzar.sh ejecutado
usuario@sputnik:~/projecte-asix$
```

Si un contenidor es cau s'aixecará amb l'escript i t'ho dirá el log

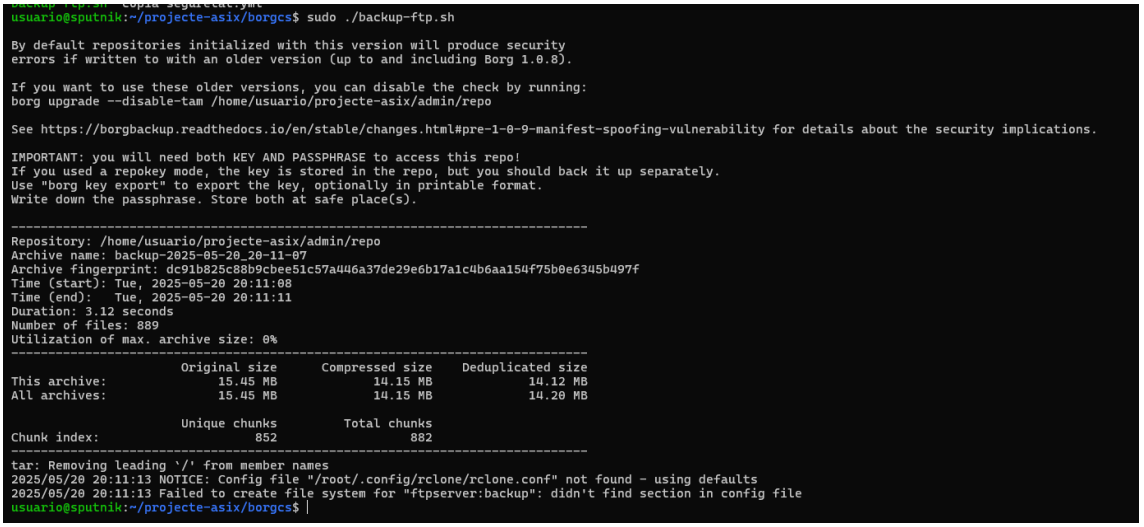
```
usuario@sputnik:~/projecte-asix$ sudo docker stop apache_final
apache_final
```

```
mar 20 may 2025 20:07:44 CEST: ⚠ apache_final está caído. Reiniciando con Ansible...

PLAY [Crear contenedor Docker amb Apache i SSH] ******************************

TASK [Gathering Facts] *******************************************************
ok: [localhost]

TASK [Crear carpeta del Dockerfile] ******************************************
ok: [localhost]

TASK [Copiar Dockerfile] *****************************************************
ok: [localhost]

TASK [Construir imatge Docker] ***********************************************
ok: [localhost]

TASK [Assegurar que el contenidor està corrent] ******************************
changed: [localhost]

PLAY RECAP *******************************************************************
localhost                  : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

mar 20 may 2025 20:07:56 CEST: ✅ postgres está funcionando correctamente.
```

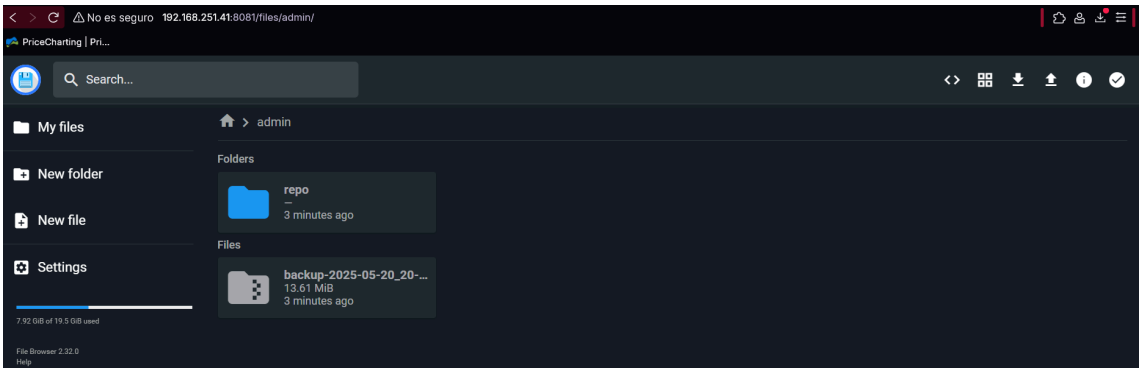
Finalment configurarem les cópies automatiques, llençarem el playbook per configurar borg amb l'ftp i rclone

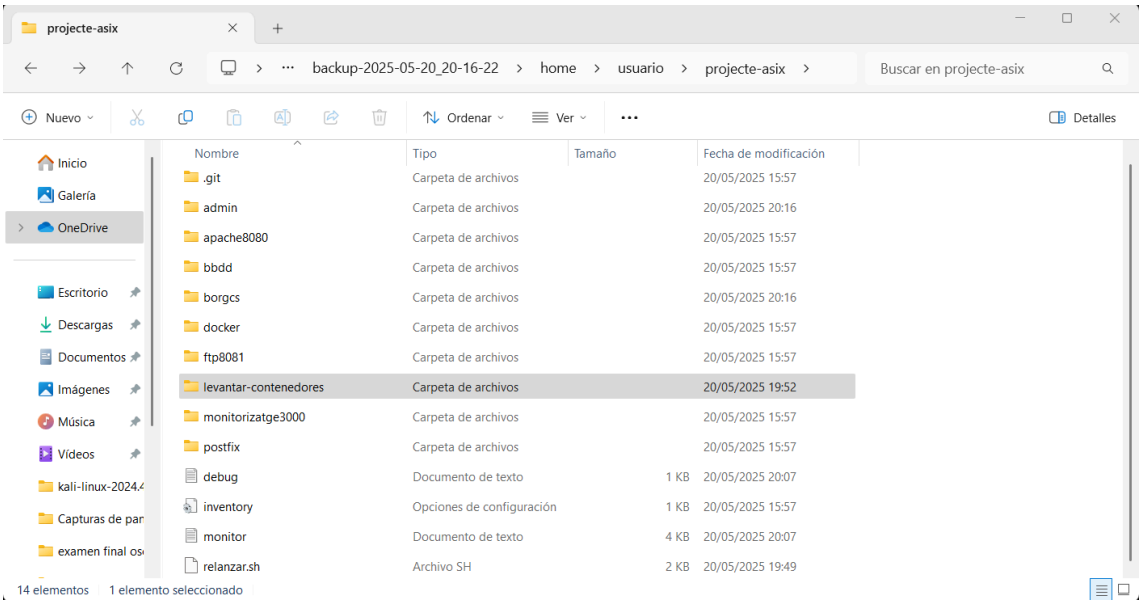
```
usuario@sputnik:~/projecte-asix/borgcs$ ansible-playbook copia-seguretat.yml

PLAY [Instal·lar Borg i rclone i configurar backup a FTP] *********************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************************************************
ok: [localhost]

TASK [Instalar BorgBackup i rclone] ******************************************************************************************************************************
changed: [localhost]

TASK [Crear carpeta de backup] ***********************************************************************************************************************************
changed: [localhost]

TASK [Crear carpeta per config rclone] ***************************************************************************************************************************
changed: [localhost]

TASK [Crear fitxer de configuració rclone.conf] ******************************************************************************************************************
changed: [localhost]

TASK [Crear script de backup] ************************************************************************************************************************************
changed: [localhost]

PLAY RECAP *******************************************************************************************************************************************************
localhost                  : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

usuario@sputnik:~/projecte-asix/borgcs$ ls
backup-ftp.sh  copia-seguretat.yml
```

Seguidament deixarà un script, el llançem i fará la còpia




Finalment ànem a l'FTP i comprovem a la ruta indicada el .tar http://latevaip:8081




Es comprova la còpia i es fa sencera de la carpeta.