
This is the **published version** of the bachelor thesis:

Muñoz Otero, Marco. *Automatització del desplegament d'infraestructures IT per a petites empreses*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334402>

under the terms of the  license.

Disseny i Avaluació d'un Framework d'Automatització d'Infraestructures IT

Marco Muñoz

30 de juny de 2026

Resum– El desplegament tradicional d'infraestructures IT (*Information Technology*) es caracteritza per processos manuals, lents i sovint irreproducibles que poden generar inconsistències en els sistemes. Aquest projecte presenta el disseny i la implementació d'un framework d'automatització pel desplegament d'infraestructures virtualitzades en entorns locals (*On-Premise*). Mitjançant l'ús de Proxmox i eines d'Infraestructura com a Codi (IaC) com Terraform i Ansible, es proposa una solució que permet desplegar serveis de forma desatesa. L'enfocament se centra a resoldre el problema de la rigidesa del maquinari físic mitjançant l'automatització, combinant màquines virtuals i contenidors per oferir una alternativa professional, escalable i econòmica que garanteix la sobirania de les dades sense els costos de gestió del núvol públic.

Paraules clau– Automatització, cloud privat, framework, infraestructura com a codi (IaC), on-premise, PME, virtualització.

Abstract– Traditional IT infrastructure deployment is characterized by manual, slow, and often non-reproducible processes that can lead to system inconsistencies. This project presents the design and implementation of an automation framework for deploying virtualized infrastructures in local (On-Premise) environments. Through the use of Proxmox and Infrastructure as Code (IaC) tools such as Terraform and Ansible, a solution is proposed that enables the unattended deployment of services. The approach focuses on addressing the rigidity of physical hardware through automation, combining virtual machines and containers to provide a professional, scalable, and cost-effective alternative that ensures data sovereignty without the management costs of the public cloud.

Keywords– Automation, Framework, Infrastructure as a Code (IaC), On-Premise, Private Cloud, SME, Virtualization.



1 INTRODUCCIÓ - CONTEXT DEL TREBALL

EL panorama actual de la digitalització a Espanya mostra una dualitat clara. Mentre que les grans empreses tenen una adopció del *cloud* com a infraestructura del 78,24%, només el 10,55% de les empreses d'entre 0 i 9 treballadors les fan servir [1]. Aquesta diferència tan gran confirma que el model de núvol públic no acaba d'encaixar en les necessitats de les microempreses i els negocis locals. Tot i l'adopció generalitzada del cloud, sobretot en empreses grans, informes destaquen dificultats amb la gestió de costos i la càrrega operativa associada[2].

Per a negocis de proximitat i Petites i Mitjanes Empreses (PIMES), solucions basades en proveïdors com Amazon Web Services (AWS) o Microsoft Azure resulten sovint sobredimensionades, complexes de gestionar i amb costos operatius creixents i impredecibles que no garanteixen la proximitat tècnica requerida [3]. El cost acumulat de les subscripcions i la manca de proximitat tècnica fan que el model *on-premise* (local) mantingui un fort atractiu. Com assenyalava Fisher [4], en molts escenaris el cost de tenir equipament propi ben gestionat és inferior al de la infraestructura com a servei (IaaS) a llarg termini. A més, els negocis locals valoren el control de les dades i la figura del tècnic de confiança, aspectes que queden diluïts en els grans proveïdors de núvol públic [2].

Malgrat els avantatges del model local, la realitat de la gestió de la infraestructura *on-premise* tradicional presenta alguns problemes a tenir en compte; la dependència absoluta de processos manuals, lents i altament propensos a errors humans. El desplegament d'un servidor físic de manera

• E-mail de contacte: marcomunozotero@gmail.com

• Menció: Enginyeria de Computadors

• Treball tutoritzat per: Alvaro Wong Gonzalez (Departament d'Arquitectura de Computadors i Sistemes Operatius)

• Curs 2025/26

tradicional requereix que una persona, normalment tècnica, configuri manualment el sistema operatiu, les xarxes i cada servei de manera aïllada. Aquest enfocament provoca una manca de reproductibilitat, inconsistències en els sistemes i dificultats per escalar o recuperar el sistema davant d'una fallada del maquinari. La motivació principal d'aquest treball neix de l'experiència professional de l'autor en el sector IT, on s'ha detectat que molts petits negocis requereixen exactament el mateix conjunt de serveis professionals: servidors de fitxers, bases de dades, gestió d'usuaris, etc.

Per resoldre aquest problema, es proposa un model basat en un node de control centralitzat que gestiona la lògica del desplegament mitjançant una interfície web. Aquest sistema permet escollir els serveis com ara els de fitxers, bases de dades o gestió d'usuaris segons les necessitats de cada client a través d'un formulari. Un cop feta la selecció, un *script* de Python processa les dades i invoca de manera automatitzada l'API de l'hipervisor Proxmox VE [12] utilitzant Terraform [13] per a l'aprovisionament del maquinari. Immediatament després, el sistema llança *playbooks* d'Ansible [17] via SSH per automatitzar la instal·lació i configuració dels serveis que corren sobre les màquines i contenidors. Aquest desplegament basat en IaC [7] permet a un proveïdor de IT configurar el maquinari de destí sense intervenció manual i remota, un enfocament que no només permet reduir els costos operatius, sinó que garanteix que els recursos, sovint limitats en aquest tipus d'equips, s'utilitzin de la forma més eficient possible.

Aquest enfocament presenta avantatges respecte a les alternatives actuals de mercat. A diferència d'entorns com OpenStack [19] o plataformes com Nutanix [18], les quals exigeixen infraestructures hiperconvergentes costoses i una administració complexa, aquesta solució està dissenyada per optimitzar l'ús de recursos sobre un únic servidor físic de baix cost. A més, en actuar com un pont d'abstracció, automatitza el coneixement tècnic darrere d'una interfície visual, transformant un procés manual en un desplegament industrial, ràpid i lliure d'errors.

Pel que fa als resultats obtinguts i les contribucions del treball, s'ha aconseguit consolidar un prototip de *framework* funcional que automatitza amb èxit el cicle de vida i el desplegament d'una infraestructura sobre un hipervisor *bare-metal* de forma centralitzada.

A continuació, es llista els objectius del projecte. Seguidament, s'analitza l'estat de l'art de les tecnologies de virtualització i orquestració actuals. A continuació, es descriu la metodologia i el disseny del sistema, pas previ al detall de la implementació tècnica i el desenvolupament del codi. El treball finalitza amb una fase de validació i proves del sistema, per acabar amb les conclusions extretes i les possibles continuïtats futures.

2 OBJECTIUS

L'objectiu d'aquest projecte és dissenyar, implementar i avaluar un *framework* d'automatització basat en programari de codi obert que optimitzi el desplegament d'infraestructures IT en entorns locals, amb la finalitat de reduir els temps de provisió, eliminar els errors manuals i oferir a les PIMES una alternativa professional, econòmica i escalable que garanteixi el control de les seves dades.

Per assolir aquest propòsit, es defineixen els següents objectius específics:

- **Disseny d'una arquitectura modular de serveis:** Definir els serveis essencials d'una empresa, com poden ser gestió d'usuaris, emmagatzematge compartit, bases de dades que es puguin combinar segons les necessitats de cada negoci.
- **Dissenyar un mecanisme d'interacció per parametritzar desplegaments** Crear una aplicació web senzilla que actuï com a node de control, que permeti parametritzar el desplegament i generar automàticament els fitxers de configuració necessaris per a la infraestructura.
- **Automatització del desplegament i la configuració de serveis IT** Implementar un flux de treball basat en estàndards de seguretat que permeti l'aprovisionament i la configuració de serveis de manera totalment automàtica, minimitzant les errades operatives.
- **Avaluar el rendiment del sistema en hardware de recursos limitats** Validar l'eficiència del sistema en un *servidor* amb 16GB de RAM, prioritzant l'ús de LXC davant de VMs per garantir un funcionament estable de l'entorn.
- **Validació tècnica del sistema:** Dur a terme proves de desplegament d'extrem a extrem, verificant els temps d'execució, la connectivitat entre serveis i la seguretat dels perímetres de xarxa creats.

3 ESTAT DE L'ART

A continuació, s'analitzen els models de desplegament actuals, les tecnologies de virtualització que permeten optimitzar el maquinari i el conjunt d'eines que fan possible desplegar serveis de forma reproducible mitjançant codi.

3.1 Models d'infraestructures actuals

La transformació digital ha impulsat l'adopció de diferents models d'infraestructura per donar suport als serveis empresarials. Entre ells destaquen el núvol públic, el model local (*on-premise*) i els entorns híbrids [4]. El núvol públic ha experimentat un creixement significatiu gràcies a la seva capacitat per reduir la inversió inicial en programari i maquinari i proporcionar una escalabilitat pràcticament immediata. A més, permet externalitzar gran part de la gestió de la infraestructura cap a proveïdors especialitzats, oferint nivells de disponibilitat difícils d'assolir en instal·lacions locals [2].

No obstant això, alguns estudis destaquen que aquesta flexibilitat pot implicar un increment progressiu dels costos operatius a llarg termini, especialment en organitzacions amb necessitats relativament estables de computació [4]. Així mateix, la dependència de proveïdors externs planteja reptes relacionats amb la sobirania de les dades, el compliment normatiu i el control de la infraestructura [2]. Aquestes limitacions han mantingut l'interès pels models *on-premise*, on l'organització conserva el control físic dels sistemes i de la informació emmagatzemada [5].

Com a alternativa intermèdia sorgeix el model híbrid, que combina recursos locals amb serveis al núvol per equilibrar

control i flexibilitat. Malgrat que aquesta aproximació permet aprofitar els avantatges d'ambdós models, també introdueix una complexitat addicional en la gestió, integració i seguretat dels sistemes distribuïts [6].

3.2 Automatització i IaC

En paral·lel a l'evolució dels models d'infraestructura, l'automatització ha esdevingut un dels pilars de l'administració de sistemes. Anteriorment, el desplegament de servidors i serveis es realitzava mitjançant procediments manuals que depenien del coneixement de l'administrador i resultaven difícils de reproduir. Per resoldre aquestes limitacions va sorgir l'IaC, que permet definir i gestionar infraestructures mitjançant arxius de configuració reproduïbles [7].

L'adopció de l'IaC aporta avantatges significatius, com l'estandardització de configuracions, la reducció d'errors humans, la traçabilitat de canvis i la capacitat de reconstruir entorns complets de forma automatitzada. No obstant això, diversos treballs adverteixen que una automatització incorrectament dissenyada pot propagar errors, per la qual cosa resulta imprescindible aplicar principis de seguretat des de les primeres fases del disseny [8]. Malgrat aquests riscos, l'IaC s'ha consolidat com una pràctica essencial per a la gestió eficient d'infraestructures modernes [20].

3.3 Virtualització vs. Contenedors

Un altre aspecte fonamental en entorns amb recursos limitats és la tecnologia de virtualització emprada. Per una banda, les màquines virtuals proporcionen un aïllament complet mitjançant la virtualització del maquinari i l'execució de sistemes operatius independents, fet que les converteix en una solució per a entorns on la seguretat i l'aïllament són requisits prioritaris. Per altra banda, els contenedors comparteixen el nucli del sistema amfitrió [23] i presenten una sobrecàrrega considerablement menor en termes de memòria i processament [9].

Els contenedors ofereixen millors temps d'arrencada i un consum més eficient de recursos, mentre que les màquines virtuals continuen sent l'opció preferida quan es requereix un aïllament estricte entre serveis [10]. Aquesta diferència resulta especialment rellevant en infraestructures de baix cost, on l'optimització de la memòria RAM i de la capacitat de còput condiciona directament el nombre de serveis que es poden executar simultàniament.

3.4 Eines de l'ecosistema

3.4.1 Proxmox

L'elecció d'una plataforma de virtualització adequada és un factor clau per a l'èxit del desplegament d'una infraestructura tecnològica. VMware es va consolidar com l'estàndard en entorns empresarials. No obstant això, els canvis recents en el seu model de llicenciament després de l'adquisició per part de Broadcom han provocat increments significatius en els costos [11]. Com a conseqüència, plataformes de codi obert com Proxmox Virtual Environment han adquirit una rellevància creixent [12].

Proxmox destaca per combinar en una mateixa plataforma la gestió de màquines virtuals basades en KVM i contenedors LXC, proporcionant una interfície unificada d'ad-

ministració i una API que permet la seva integració amb eines d'automatització. Aquesta capacitat resulta especialment interessant en entorns amb recursos limitats, on la combinació d'ambdues tecnologies permet optimitzar l'ús del maquinari disponible.

3.4.2 Terraform

L'automatització de l'aprovisionament d'infraestructura requereix, a més, eines capaces d'interactuar amb la plataforma de virtualització mitjançant llenguatge de descripció de *hardware*. En aquest àmbit, Terraform s'ha consolidat com una de les solucions més esteses per a l'orquestració de recursos mitjançant enfocaments declaratius [13]. La seva independència respecte al proveïdor d'infraestructura permet descriure l'estat desitjat del sistema i delegar en l'eina la creació i modificació dels recursos necessaris mitjançant proveïdors específics [21].

Tanmateix, l'automatització de la infraestructura no està lliure de riscos. Estudis adverteixen que les configuracions de l'IaC poden introduir vulnerabilitats de seguretat i errors sistemàtics si no es gestionen correctament [14]. Malgrat això, diversos estudis destaquen que l'automatització declarativa redueix dràsticament els temps de provisió i millora l'escalabilitat operativa gràcies al principi d'idempotència, que garanteix resultats consistents fins i tot després de múltiples execucions del mateix codi [15].

3.4.3 Cloud-init

Un cop desplegats els recursos virtuals, és necessari automatitzar la configuració inicial dels sistemes desplegats. Per dur a terme això, Cloud-init proporciona un mecanisme àmpliament utilitzat que permet personalitzar instàncies des del moment de la seva creació mitjançant imatges de núvol optimitzades [22], configurant paràmetres com la xarxa, el nom de l'equip (*hostname*) o les claus SSH d'accés [16]. Aquesta capacitat facilita la creació de plantilles genèriques reutilitzables i constitueix una peça fonamental en els processos moderns de desplegament automatitzat.

3.4.4 Ansible

Complementàriament, la gestió de la configuració del programari requereix eines per instal·lar, actualitzar i mantenir els serveis de forma consistent. Entre les solucions més conegudes es troben Puppet, Chef i SaltStack [17]. No obstant això, aquestes plataformes solen basar-se en arquitectures que requereixen agents residents en cada node administrat. El software d'Ansible adopta un enfocament sense agents (*agentless*) basat en connexions SSH i fitxers de configuració declaratius (*playbooks*) escrits en llenguatge de programació YAML, reduint el consum de recursos i simplificant l'administració dels sistemes [17].

La combinació de Terraform, Cloud-init i Ansible representa actualment una de les arquitectures més utilitzades per automatitzar infraestructures modernes, realitzant les fases de provisionament, inicialització i configuració final dels serveis.

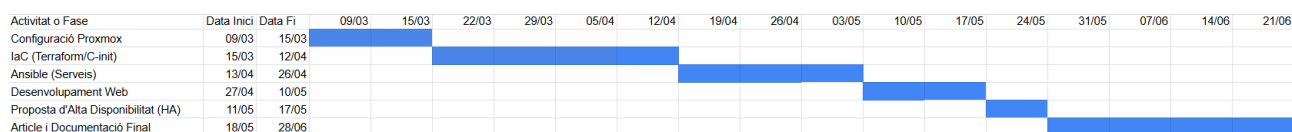


Fig. 1: Diagrama de Gantt

3.5 Proposta

Malgrat l'existència de plataformes comercials per a la gestió d'infraestructures, moltes d'elles es recolzen en solucions propietàries d'elevat cost, com Nutanix [18], mentre que alternatives de codi obert com OpenStack presenten una complexitat operativa considerable i uns requisits de maquinari difícils d'assumir per a petites organitzacions [19]. Aquesta situació genera una oportunitat per desenvolupar solucions que apropin les capacitats d'automatització de la infraestructura empresarial a entorns locals de baix cost.

En conseqüència, l'anàlisi elaborada indica que les tecnologies utilitzades per automatitzar infraestructures virtualitzades en aquest projecte es troben consolidades. Tanmateix, continua havent la necessitat d'integrar aquestes eines en solucions accessibles per a petites i mitjanes empreses que permetin desplegar serveis de forma modular, reproducible sobre maquinari local. Aquesta necessitat constitueix la principal motivació del marc de treball (*framework*) proposat en el present treball i accessible de forma oberta [24].

3.6 Escalabilitat i flexibilitat On-Premise

L'estudi [6] destaca que un dels punts crítics en l'escalabilitat d'infraestructures *on-premise* és la gestió de la complexitat adaptativa. En entorns que no són nativament configurables per API, com els centres de dades tradicionals de les PIMES, l'escalabilitat sol derivar en sistemes rígids i difícils de mantenir. Mitjançant l'ús de models que integren l'IaC, és possible evolucionar els sistemes físics cap a entorns dinàmics de manera ràpida i reproducible. En aquest treball, la utilització d'aquesta metodologia sobre Proxmox permet que el sistema pugui actualitzar-se amb la modificació de variables de configuració o addició de nous nodes es pugui realitzar de manera sistemàtica, reduint els errors de disseny i millorant la resposta davant futures expansions del negoci.

4 PLANIFICACIÓ

A la figura 1 es pot observar la planificació del projecte. El treball s'ha dividit en quatre fases principals, on en cadascuna i tindrà lloc subfases d'investigació, disseny, desenvolupament i tests.

S'ha dedicat un esforç intensiu a la configuració de Proxmox VE i al desenvolupament del codi de Terraform. Aquesta fase ha estat extensa, ja que establir una base sòlida d'IaC i garantir la comunicació amb l'API de l'hipervisor

és important. Qualsevol inconsistència en aquesta fase bloquejaria el desplegament dels serveis superiors. En aquest període, també s'ha integrat *Cloud-init* per automatitzar la identitat i l'aprovisionament de xarxa dels nodes.

Aquesta fase enllaça directament amb l'ús d'Ansible, on es preveu una dedicació important per al desenvolupament de *playbooks*. L'estratègia se centra en la divisió del sistema en components més petits i independents, facilitant així no només la instal·lació del programari, sinó també la gestió de la persistència de dades en volums separats d'una manera més controlada.

El desenvolupament de la interfície web s'executa en la fase final, actuant com un integrador de totes les eines prèviament validades i assegurant que el flux d'extrem a extrem sigui funcional. Aquest període conclou amb la realització de proves exhaustives i l'escriptura de l'article tècnic final.

S'ha assignat un marge de temps dins de cada fase per a la resolució de conflictes, com per exemple amb l'API de Proxmox o la persistència de dades, aspectes que podrien presentar una corba d'aprenentatge elevada.

5 DISSENY I ARQUITECTURA DEL SISTEMA

El desenvolupament del *framework* s'ha realitzat de manera incremental i modular. En primer lloc, es va dissenyar l'arquitectura general de la solució i es va configurar l'entorn de virtualització basat en Proxmox. Posteriorment, es van desenvolupar els mòduls d'aprovisionament mitjançant Terraform per a automatitzar la creació de màquines virtuals i contenidors. Una vegada validada aquesta capa d'infraestructura, es va integrar *Cloud-init* per a automatitzar la configuració inicial dels sistemes desplegats. En una fase posterior es van desenvolupar els *playbooks* d'Ansible responsables de la instal·lació i configuració automàtica dels serveis. Finalment, es va implementar una interfície web que actua com a punt central d'orquestració i es van realitzar proves de validació per a verificar el correcte funcionament del sistema. Aquesta metodologia permet desacoblar les diferents capes del *framework*, facilitant el seu manteniment, ampliació i reutilització en futurs desplegaments[20].

5.1 Perspectiva de l'ús del sistema

El sistema s'ha dissenyat des de dues perspectives complementàries. Des del punt de vista del **tècnic IT**, el desplegament complet s'orquestra mitjançant codi, reduint la intervenció manual a la configuració inicial dels paràmetres del client. Això permet reproduir infraestructures equivalents

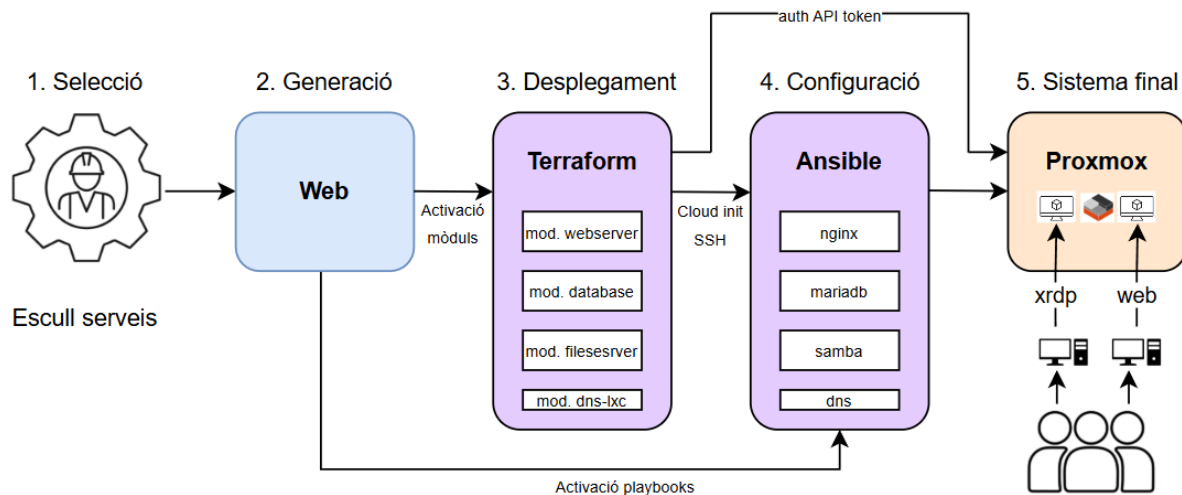


Fig. 2: Model proposat.

de forma consistent i minimitzant el risc d'errors derivats de les intervencions manuals.

Des del punt de vista del **client final**, l'accés als serveis es realitza de forma transparent. Depenent de les necessitats de cada negoci, es contemplen dues modalitats d'accés.

5.2 Flux d'automatització i descripció del model arquitectònic

El procés d'orquestració i provisió s'ha dissenyat sota els principis de desacoblament de responsabilitats i modularitat estricta. Com es mostra detalladament a la figura 2, el sistema s'estructura com un *pipeline* seqüencial dividit en cinc blocs funcionals interconnectats: Selecció, Generació, Desplegament de Maquinari, Configuració de Programari i Sistema Final.

A continuació, s'analitza el comportament i el flux de dades de cada component descrit en el diagrama de flux del sistema:

5.2.1 Mòdul 1: Selecció i interacció de l'operador

Aquest primer bloc té la responsabilitat principal de permetre al tècnic informàtic especificar de manera personalitzada i simplificada el catàleg de serveis requerits per la PIME client en funció de les seves necessitats de negoci. Les dades d'entrada d'aquest mòdul estan constituïdes directament per les respostes del tècnic al formulari interactiu de la interfície web, en el qual l'operador selecciona quins mòduls desitja activar (com ara servidors de fitxers, bases de dades, servidors web, controladors de domini, gestors DNS o servidors intermediaris *proxy*). A més, per a cadascun dels serveis escollits, es requereix definir de manera parametritzada el nivell de càrrega previst per al servei mitjançant un selector de perfils dimensionats (baixa, mitjana o alta intensitat de treball). Durant el processament, el mòdul recull aquesta matriu de selecció funcional d'entorn. Les dades de sortida del mòdul són els paquets i descriptors lògics de serveis triats pel tècnic, establint la seva relació amb el següent mòdul en transmetre l'estructura de la petició cap a la capa d'orquestració dinàmica del servidor web.

5.2.2 Mòdul 2: Interfície i Selecció Dinàmica de Mòduls

Un cop el tècnic ha seleccionat els serveis que requereix el client a la pàgina web, l'aplicació executa una lògica d'orquestració selectiva. La interfície web actua com un filtre per evitar que l'operador hagi de manipular directament els fitxers de configuració o triar els components manualment a través de la consola de comandaments.

L'script de Python rep les opcions del formulari i, de manera dinàmica, activa o descarta els mòduls de Terraform i els *playbooks* d'Ansible que ja han estat prèviament desenvolupats, com els components estàndard de base de dades, servidor web o VPN. D'aquesta manera, es tradueixen els clics de la pantalla en una ordre d'execució focalitzada que crida únicament a les peces de programari i maquinari requerides per a la solució d'aquell client concret, garantint un desplegament modular, net i lliure d'errors sintàctics.

5.2.3 Mòdul 3: Desplegament massiu d'infraestructura (Terraform)

Aquest mòdul aprovisiona i desplega la infraestructura de maquinari dins de l'entorn de virtualització de Proxmox VE. Rep els mòduls a aplicar des de la interfície de control i estableix una connexió remota automatitzada contra l'API REST de Proxmox utilitzant un token d'autenticació xifrat. També injecta la clau pública SSH a les metadades de configuració inicial de Cloud-init. Com a dades de sortida, el mòdul lliura les instàncies virtuals, és a dir, les màquines virtuals i contenidors, ja creades i enceses. Això defineix la seva relació seqüencial amb el següent bloc, en deixar els mecanismes i la via d'accés completament oberts perquè l'orquestrador de configuració prengui el control del node de forma segura i autònoma.

5.2.4 Mòdul 4: Inicialització i Configuració de Programari (Ansible)

La responsabilitat d'aquest quart mòdul consisteix a actuar com l'orquestrador de la configuració interna del sistema, transformant els nodes gestionats¹ en serveis de producció operatius. Utilitzant com a dades d'entrada els *playbooks*

seleccionats pel tècnic a la interfície web, el seu processament es basa a establir una connexió via SSH als nodes per a llançar l'execució de les tasques de configuració de programari. Atès que les adreces IP estàtiques de les màquines virtuals i contenidors de Proxmox ja estan definides prèviament a l'interior d'aquests propis *playbooks*, l'orquestrador pot iniciar la comunicació de manera immediata i desatesa per a instal·lar i aixecar serveis com ara bases de dades, servidors web o VPN. Les dades de sortida d'aquest procés són els serveis funcionals, la qual cosa estableix la seva relació en transferir aquest estat lògic consolidat cap al darrer mòdul per a la seva utilització final.

5.2.5 Mòdul 5: Sistema Final Virtualitzat (Proxmox VE)

Aquest darrer mòdul assumeix la responsabilitat d'actuar com l'entorn de producció definitiu, representant el resultat final de tot el procés d'automatització: una infraestructura virtualitzada completament operativa desplegada sobre un node de Proxmox VE. El bloc rep com a dades d'entrada l'estat lògic consolidat de les d'aplicacions del mòdul anterior.

Pel que fa al seu processament, una vegada tancades les fases seqüencials de provisió i parametrització, els serveis s'estructuren en màquines virtuals o contenidors LXC segons el seu nivell de criticitat, consum de recursos i rendiment desitjat. Aquest disseny flexibilitza l'entorn segons la naturalesa de cada organització. Per exemple, serveis lleugers com ara servidors DNS o VPN es poden desplegar sobre contenidors LXC per a maximitzar l'aprofitament i la densitat dels recursos disponibles. Per altra banda, aquells serveis que requereixen un major nivell d'aïllament o funcionalitats i extensions específiques del nucli del sistema operatiu, com és el cas dels controladors de domini, servidors web, servidor de fitxers o base de dades, s'executen sobre màquines virtuals dedicades.

D'aquesta manera, una petita empresa podria disposar d'una infraestructura optimitzada formada per un servidor web corporatiu, una base de dades MariaDB i un sistema d'emmagatzematge compartit, mentre que una altra organització amb necessitats diferents podria complementar fàcilment aquests mateixos serveis bàsics amb una VPN WireGuard per a oferir accés remot segur o amb un controlador de domini per a dur a terme la gestió centralitzada d'usuaris i permisos.

La combinació de màquines virtuals i contenidors en aquest entorn permet mantenir un equilibri entre eficiència, modularitat i seguretat, optimitzant al màxim l'ús del maquinari físic disponible sense haver de renunciar a la flexibilitat necessària per a futures expansions del negoci. Com a dades de sortida, el mòdul lliura la infraestructura IT corporativa consolidada i llista per a l'explotació operativa del client. Gràcies a haver-se generat mitjançant un flux automatitzat basat en el paradigma de l'IaC, qualsevol modificació, ampliació o reconstrucció parcial de la infraestructura davant d'una incidència es pot realitzar de forma totalment controlada des del node de control inicial, reduint la intervenció manual i millorant la mantenibilitat global del sistema.

6 RESULTATS

6.1 Entorn experimental

Per dur a terme l'avaluació del sistema, s'ha configurat un escenari de proves dividit en dues capes de computació diferenciades: un node de control i un node de destinació local. Aquest entorn experimental s'ha utilitzat per validar el funcionament integral del *framework*, avaluant la capacitat de desplegar serveis de forma automatitzada sobre maquinari de recursos limitats. Les proves realitzades se centren a analitzar els temps de desplegament, el consum de recursos i la correcta integració dels serveis generats.

Per una banda, el node de control (computador de l'autor) allotja l'entorn de desenvolupament basat en el subsistema Linux per a Windows (WSL) on hi ha instal·lat Ansible, l'interpret de Python per al servidor web de gestió, així com el binari de Terraform. D'altra banda, el node de destinació consisteix en un servidor local físic dedicat on s'executa l'hipervisor Proxmox VE, sobre el qual es despleguen els diferents mòduls d'infraestructura virtualitzada.

Aquest servidor d'assaig compta amb les següents especificacions:

- **Processador:** Intel Core i5-10400 de 6 nuclis i 12 fils amb una freqüència base de 2,90 GHz.
- **Memòria RAM:** 16 GB DDR4 a 2667 MT/s, actuant com el principal factor de limitació en la densitat de virtualització de l'entorn.
- **Emmagatzematge:** Disc d'estat sòlid SATA de 240 GB per a l'allotjament del sistema de fitxers de l'hipervisor i els volums de dades.
- **Placa base i gràfics:** MSI H510M PRO-E amb una unitat de processament gràfic integrada Intel UHD Graphics 630.

6.2 Configuració d'accés i connectivitat del laboratori

Per al funcionament correcte de l'orquestració dins del laboratori experimental, s'han hagut d'aplicar un conjunt de decisions de disseny focalitzades en el comportament de la xarxa i l'accés segur. Aquestes decisions permeten validar el funcionament segur del *framework* dins d'un entorn controlat, preservant la reproductibilitat dels experiments realitzats.

Pel que fa a la gestió d'accessos, s'utilitza de manera temporal l'usuari `root` per establir les connexions SSH inicials des del node de control WSL cap als diferents contenidors i màquines virtuals. Aquesta configuració simplificada està justificada pel fet que tot el sistema s'executa íntegrament dins d'un entorn de xarxa local tancat i controlat d'assaig. Cal fer constar de forma explícita que, per a un entorn de producció real, aquesta metodologia s'haurà de modificar obligatòriament per complir amb les polítiques de seguretat estàndard, substituint l'accés directe de l'administrador per un usuari limitat configurat dins del grup de seguretat `sudoers`.

Aquestes decisions de disseny permeten simplificar l'experimentació mantenint unes condicions de seguretat ade-

quades per a un entorn de laboratori, sense comprometre futures adaptacions a escenaris de producció.

6.3 Validació funcional: Desplegament pas a pas

Per demostrar la viabilitat de l'arquitectura dissenyada, es presenta un cas d'ús concret corresponent al desplegament d'un entorn web. Aquest escenari comprèn un servidor web basat en Nginx, un gestor de bases de dades relacionals MariaDB i una VPN amb Wireguard. El flux d'execució validat segueix els següents passos seqüencials:

- 1. El tècnic selecciona els mòduls desitjats, en aquest cas el servidor Web, la base de dades i la VPN a través del formulari de la interfície d'orquestració (veure Fig. 3).
- 2. L'script de Python processa la petició, gestiona l'assignació de recursos segons la càrrega definida i inicia el procés d'automatització.
- 3. Terraform aprovisiona el contenidors i les màquines virtuals mitjançant l'API de Proxmox (veure Fig. 4).
- 4. Cloud-init realitza la configuració inicial de tots els recursos, establint les adreces IP estàtiques i injectant les claus SSH necessàries per habilitar l'accés remot segur.
- 5. Ansible executa les tasques de configuració. En el cas del mòdul VPN, el *playbook* genera dinàmicament les claus criptogràfiques de WireGuard, configura el túnel tant al servidor com al client, i utilitza el mòdul *fetch* per descarregar automàticament el fitxer de configuració del client (*client.conf*) cap al directori de treball del tècnic.
- 6. Un cop finalitzat el desplegament, el tècnic importa el fitxer *client.conf* a la seva aplicació WireGuard per activar el túnel segur. D'aquesta manera, s'estableix un accés xifrat a la xarxa local provisionada, garantint que els serveis restin aïllats d'exposició pública.
- 7. Finalment, es valida el desplegament verificant la connectivitat interna i el funcionament dels serveis (vegeu Fig. 5).

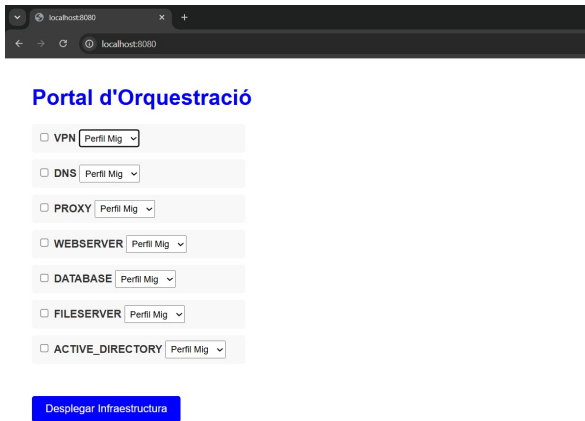


Fig. 3: Interfície web de control amb els perfils de càrrega per a l'orquestració.

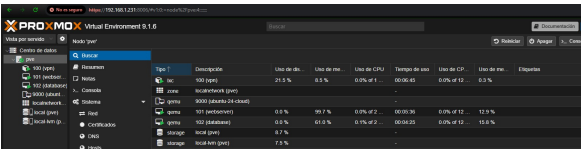


Fig. 4: Màquines virtuals i contenidors desplegats de forma automàtica a Proxmox VE per Terraform.



Fig. 5: Pàgina web mostrant la connexió establerta i les dades de MariaDB.

6.4 Serveis desplegats i connectivitat

Mitjançant el flux descrit anteriorment, s'ha validat amb èxit la creació del catàleg de serveis corporatius essencials. La TAULA 1 resumeix els mòduls funcionals que el *framework* és capaç de desplegar actualment de forma autònoma.

TAULA 1: CATÀLEG DE SERVEIS AUTOMATITZATS PEL *framework*.

Servei	Descripció i Tecnologia
VPN	Túnel encriptat amb WireGuard per a accés segur.
DNS	Servidor BIND9 per a la resolució de noms locals.
Web	Servidor HTTP/HTTPS basat en Nginx.
Base de dades	Sistema relacional MariaDB securitzat.
Fileserver	Emmagatzematge compartit amb protocol Samba.
Domini	Màquina virtual per a gestió d'identitats (AD).

6.5 Rendiment del desplegament

S'ha completat i validat amb èxit el desenvolupament de l'orquestrador integral, assolint un nivell d'automatització complet. Les proves de rendiment demostren que el *framework* és capaç de desplegar l'ecosistema sencer de la TAULA 1 de manera modular i sense intervenció manual en un temps operatiu proper als 10 minuts, una reducció dràstica respecte als mètodes de configuració tradicionals *on-premise*.

6.6 Consum de recursos i anàlisi de memòria

A més de la validació funcional, es va realitzar una anàlisi empírica del comportament dels recursos de maquinari durant l'orquestració massiva dels nodes per comprovar-ne la viabilitat en servidors de recursos limitats. Un dels fets més destacables detectats a la interfície nativa de Proxmox VE va ser la saturació gràfica del consum de memòria RAM a les instàncies del servidor Web, el *Fileserver* i el *Domain Controller*, les quals mostraven un indicador d'utilització proper al 100%.

Per analitzar aquest comportament, es va realitzar una auditoria interna entrant per SSH a la instància de la web (webserver, configurada amb 2 GB de RAM) i executant la comanda `free -m`. Les mètriques obtingudes es detallen a la TAULA 2 i van revelar l'estructura de memòria real del sistema:

TAULA 2: ESTAT DE LA MEMÒRIA RAM AL NODE *web-server* EN MEGABYTES (MB).

Total	Used	Free	Shared	Buff/Cache	Available
1967	390	262	3	1510	157

Aquestes dades numèriques de la TAULA 2 avaluades aporten una justificació tècnica crítica sobre com interactuen els sistemes operatius virtualitzats en entorns de recursos limitats:

- **Càrrega real de l'espai d'usuari (*Used RAM*):** El programari i els dimonis en execució de producció (Nginx, serveis base) només consumeixen de forma estricta *390MB* de la memòria del sistema (19%), confirmant que el dimensionament assignat és totalment viable per al maquinari del laboratori.
- **Retenció massiva per a memòria cau (*Buff/Cache*):** El Kernel d'Ubuntu ocupa un volum de *1510MB* destinat exclusivament a la memòria cau de pàgines. Durant la fase d'arrencada inicial del node i la posterior instal·lació automatitzada de binaris mitjançant els *playbooks* d'Ansible, el sistema operatiu llegeix massivament fitxers del disc de Proxmox. Per optimitzar l'accés d'E/S, el nucli bolca aquests binaris a la RAM en forma de memòria cau [23].

Aquesta validació demostra que, tot i els indicadors d'alerta visuals a Proxmox, el disseny d'infraestructura lleugera manté els nodes estables, operant dins dels llindars físics requerits i garantint l'èxit de l'automatització.

7 CONCLUSIONS

La finalització d'aquest projecte corrobora l'assoliment dels objectius plantejats en la fase inicial, demostrant l'efectivitat i la viabilitat de concebre un entorn d'orquestració automatitzat i local per al sector de la petita i mitjana empresa. A través d'una combinació coordinada d'eines sota un únic panell de control unificat en Python, s'ha consolidat un mètode capaç d'aixecar serveis corporatius complexos en pocs minuts, garantint la independència tecnològica i la gestió de dades respecte als operadors de núvol públic. En concret, les fites assolides confirmen la maduresa del disseny proposat, havent estructurat amb èxit un catàleg de serveis modulars, programat una interfície web intuïtiva per a la parametrització de desplegaments, i automatitzat tant el provisionament de maquinari com l'estat intern del programari amb Terraform i Ansible sobre un servidor de laboratori amb recursos altament limitats.

Pel que fa a les contribucions més significatives, aquest treball aporta un marc pràctic per a la provisió de recursos sota demanda d'extrem a extrem, acompanyat d'un model de seguretat fonamentat en tokens d'API amb privilegis concrets i l'enllaç de claus criptogràfiques SSH de forma

desatada des de WSL. Tot i l'èxit del prototip, cal reconèixer algunes limitacions pròpies d'aquesta fase de desenvolupament; el mòdul de la VPN es troba restringit a escenaris de prova locals, ja que l'exposició de trànsit cap a Internet s'ha ajornat conscientment per raons de prudència operativa, evitant aixecar vectors d'atac abans de disposar d'un estudi sistemàtic de vulnerabilitats. De manera global, aquesta recerca evidencia que és plenament viable estendre les pràctiques d'enginyeria de programari més avançades i la Infraestructura com a Codi (IaC) a entorns de baix pressupost, posant a l'abast de petites organitzacions una solució altament reproducible, flexible i econòmicament sostenible per a la governança dels seus sistemes informàtics locals.

Com a línies de continuïtat i futures extensions, es desenvoluparia un disseny de producció encara més complet. Es podria integrar un assistent de dimensionament operatiu a la interfície de control, basat en el volum d'usuaris concurrents previst per a la PIME. A partir d'aquestes dades de negoci, l'orquestrador en Python podria calcular al·gorímicament i assignar de manera proporcional els nuclis de processament i la memòria RAM requerits, injectant aquests valors directament als fitxers de configuració just abans d'iniciar la provisió. Aquesta extensió evitaria qualsevol malbaratament de recursos i optimitzaria al màxim la vida útil del servidor de virtualització local.

ÚS D'INTEL·LIGÈNCIA ARTIFICIAL GENERATIVA

En l'elaboració d'aquest Treball de Final de Grau, l'ús d'eines d'intel·ligència artificial generativa s'ha limitat exclusivament a tasques de suport tecnològic i editorial, sense substituir en cap cas l'anàlisi ni la presa de decisions de l'autor:

- **Resolució d'errors:** Suport puntual per analitzar *logs* de la terminal i diagnosticar problemes durant la configuració i desplegament de la infraestructura.
- **Format en LaTeX:** Ajuda en l'edició a Overleaf, especialment per resoldre dubtes de sintaxi, generar taules complexes i inserir imatges.
- **Redacció i consultes:** Resolució de dubtes conceptuals puntuals i revisió d'estil per garantir la cohesió i el to acadèmic del document en català.

AGRAÏMENTS

Gràcies a totes les persones que m'han ajudat a formar-me com a enginyer i com a persona. Agrair al meu tutor del treball, l'Álvaro Wong, l'esforç dedicat a guiar-me per dur a terme aquest projecte. Als professors i professores que m'han ensenyat durant el grau, així com als companys i amics amb qui he compartit aquesta etapa. Finalment, agrair a la meua família i parella pel suport que em donen.

REFERÈNCIES

- [1] Eurostat. *Cloud computing services by size class of enterprises*. European Union, 2026.

- [2] HBR *Accelerating Forward: The State of Cloud-Driven Transformation*. Harvard Business Review, 2021
- [3] M. Twum-Darko, y J.L. Sibanyoni. (2014) *Role of cloud computing as a technology innovation in small and medium enterprises in South Africa*. ResearchGate.
- [4] C. Fisher *Cloud versus On-Premise Computing*. Scientific Research Publishing (SCIRP), 2018.
- [5] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu *Edge Computing: Vision and challenges*. WebOfScience, 2016.
- [6] Eric S. Enos and Daniel R. Herber *A Reference Architecture for On-Premises Software-Defined Infrastructure Using Model-Based Systems Engineering* IEEEExplore, 2025
- [7] Kief Morris *Infrastructure as Code*. O'Reilly, 2016
- [8] Hashicorp *Terraform Cloud Security and Compliance White Paper*. Disponible a: <https://www.datocms-assets.com/2885/1597254052-tfc4bsecuritybriefv2.pdf>
- [9] Qi Zhang, Ling Liu, Calton Pu, Qiwei Dou, Liren Wu and Wei Zhou *A Comparative Study of Containers and Virtual Machines in Big Data Environment*. Arxiv, 2018
- [10] *Virtual machine vs container: an application performance review*. Disponible a Academia.edu
- [11] Gyana Swain *Los clientes de VMware en Europa se enfrentan a incrementos de precios de hasta el 1.500 por ciento tras la adquisición por parte de Broadcom*. Computer World, 2015
- [12] *Proxmox VE*. Disponible a pve.proxmox.com/wiki
- [13] Hashicorp *Terraform*. Disponible a: <https://developer.hashicorp.com/terraform>
- [14] Rahman *Security Vulnerabilities in Infrastructure as Code: What, How Many, and Who*. ResearchGate, 2023.
- [15] Guerriero, Michele and Garriga, Martin and Tamburri, Damian A. and Palomba, Fabio. *Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry*. WebOfScience, 2019.
- [16] Proxmox, *Cloud-Init Support*. Disponible a: pve.proxmox.com, 2025
- [17] Redhat *Automatización empresarial: Ansible, Terraform, Puppet, Chef y Salt*. Disponible a: <https://www.redhat.com/es/topics/automation/understanding-ansible-vs-terraform-puppet-chef-and-salt>, 2026
- [18] *Nutanix*. Disponible a: <https://www.nutanix.com/>
- [19] *OpenStack*. Disponible a: <https://www.openstack.org/>
- [20] Jim Holdsworth i Annie Badman (IBM) *¿Qué es la infraestructura como código (IaC)?* Disponible a: <https://www.ibm.com/es-es/think/topics/infrastructure-as-code>
- [21] Hashicorp *Proxmox Provider BPG* Disponible a: <https://registry.terraform.io/providers/bpg/proxmox/latest/docs>
- [22] *RamNode Cloud Image vs ISO Boot Source* Disponible a: <https://ramnode.com/support/documentation/cloud-vps/cloud-image-vs-iso>
- [23] Robert Love. *Linux Kernel Development*. 3a edició, Addison-Wesley Professional, 2010.
- [24] Muñoz, M. Framework d'Automatització d'Infraestructures IT. Repositori GitHub. Disponible a: <https://github.com/marcomunoz10/TFG>

APÈNDIX

A Captures de pantalla addicionals

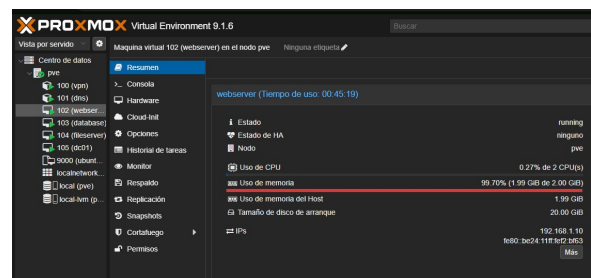


Fig. 6: Màquina del webserver consumint el 100% de la RAM.

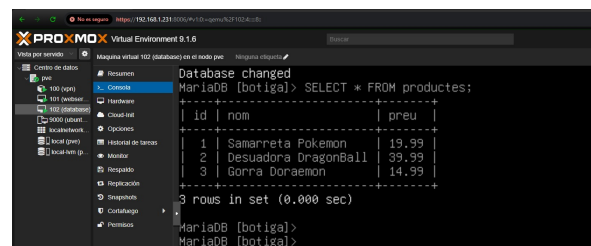


Fig. 7: Comprovació interna de les dades guardades a la instància de MariaDB.

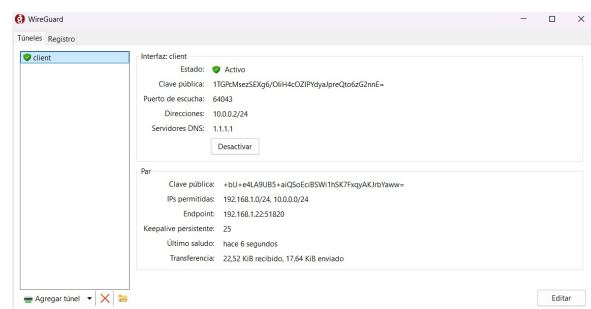


Fig. 8: Validació de la connectivitat del túnel VPN WireGuard des del node de control.

B Glossari de termes

Per facilitar la comprensió d'aquest article, es defineixen els següents conceptes clau utilitzats en l'àmbit de la virtualització de sistemes:

Node gestionat: Màquina remota, com per exemple un servidor, una màquina virtual o un contenidor que Ansible configura o administra.

Host (Sistema Amfitrió): Representa el maquinari físic real sobre el qual s'instal·la l'hipervisor Proxmox VE. És l'encarregat de gestionar i subministrar de forma directa els recursos reals de CPU, memòria RAM, emmagatzematge i interfícies de xarxa.

Guest (Sistema Convidat): Es refereix a qualsevol instància aïllada lògicament que s'executa dins de l'hipervisor, ja sigui en forma de MV o d'LXC. El *guest* opera amb el seu propi espai d'usuari i creu que disposa d'un maquinari dedicat, tot i estar compartint el ferro de l'amfitrió.

KVM (Kernel-based Virtual Machine): Tecnologia de virtualització de codi obert integrada en el Kernel de Linux que utilitza Proxmox per generar màquines virtuals pures. Permet que l'hipervisor actue com a amfitrió dividint els recursos físics de forma estricta.