MEMÒRIA

“Infraestructura com a codi, el seu cicle de desenvolupament i les millors pràctiques de DevOps”

Assignatura: TFG - Administració de xarxes i sistemes operatius

Curs: 2023-24 (1r Semestre) - Aula 1

Nom de l’Alumne/a: JAIRO VARO MUNERA

Índex

1. Introducció del TFG

Motivat per la meva experiència en l'entorn acadèmic i laboral, he decidit triar una temàtica que defineix la forma de treballar en el dia a dia de molts professionals i que, a més m'interessa especialment.

Des de l'inici de la meva carrera, he centrat els meus esforços a desenvolupar les meves habilitats en el camp de la automatització, la infraestructura com a codi, la integració i el lliurament continu i les bones pràctiques de desenvolupament.

1.1. Descripció i estructura de la memòria

Aquest treball final de grau té com a objectiu explorar el marc de treball i la filosofia actual de DevOps en un món en constant desenvolupament i evolució. La seva finalitat és descriure i promoure les millors pràctiques per al desenvolupament, manteniment i entrega de codi.

Iniciarem el recorregut amb una exploració del concepte de DevOps, analitzant l'evolució d'aquest moviment i destacant els beneficis de la seva aplicació en la gestió d'infraestructures.

Seguidament, ens centrarem en el concepte d'Infraestructura com a Codi, amb un enfocament especial en l'ús de Terraform com eina principal. A més, donat que el cicle de vida i el desenvolupament del codi ocupen una posició central en aquest treball, explorarem les eines de gestió col·laborativa de codi i el seu important paper en el control de versions.

No es pot passar per alt la importància de la computació al núvol, i, per tant, introduirem un proveïdor: Amazon Web Services, on es destacaran algunes de les seves eines i serveis clau per a la gestió de la infraestructura.

Per concloure, ens enfocarem en les eines relacionades amb l'automatització i la creació de processos automàtics per al desplegament de codi mitjançant la integració i entrega contínua: CI/CD. La part més emocionant d'aquest treball radica en els exemples pràctics que s'ofereixen a la part final, on es realitzaran laboratoris per il·lustrar de manera tangible els conceptes que s'han explorat al llarg de la memòria.

1.2. Justificació i motius de la temàtica

La selecció d'aquesta temàtica per al Treball Final de Grau es fonamenta en diverses raons significatives:

- **Marc Actual:** L'àmbit tecnològic està en un estat de canvi constant i ràpid. És essencial adaptar-se a aquesta dinàmica, i la filosofia de DevOps i les seves pràctiques són fonamentals per a una gestió eficient d'infraestructures i una entrega eficaç de codi.

- **Interès Personal:** La meva experiència amb la tecnologia, la gestió de sistemes i la seva infraestructura m'ha conduït a especialitzar-me en àrees com la infraestructura com a codi, l'automatització i les pràctiques de desenvolupament. Aquesta temàtica em permet fusionar els meus interessos professionals amb els meus objectius acadèmics.

- **Investigació:** Aquest treball hem dona l'oportunitat d'explorar a fons conceptes clau com la Infraestructura com a Codi, l'ús de Terraform, la computació en el núvol i algunes de les eines mes potents de CI/CD, els quals considero que són fonamentals en el meu futur professional.

- **Contribució al Coneixement:** Mitjançant la realització d'aquest TFG, puc fer una contribució a l'àrea d'administració de sistemes al proporcionar exemples pràctics i avaluacions de les millors pràctiques.

Aquests són els arguments que m'han guiat a l'hora d'escollir aquesta temàtica per al TFG, i estic segur que aquest projecte serà altament enriquidor i gratificant des d'una perspectiva acadèmica i personal.

1.3. Objectius del treball

El principal objectiu d'aquest Treball de Fi de Grau és aprofundir en el camp de la infraestructura com a codi i les pràctiques de DevOps, proporcionant una comprensió exhaustiva i aplicada d'aquests conceptes clau per a la gestió d'infraestructures tecnològiques.

1.3.1. Objectius Principals

- Comprendre a fons la infraestructura com a codi com a mètode per a la gestió i aprovisionament d'infraestructures, així com la automatització de processos.

- Aprofundir en la Filosofia DevOps.

1.3.2. Objectius Parcials i Tasques Associades

- Analitzar i documentar les millors pràctiques de DevOps i la seva integració en el desenvolupament i la gestió de la infraestructura.

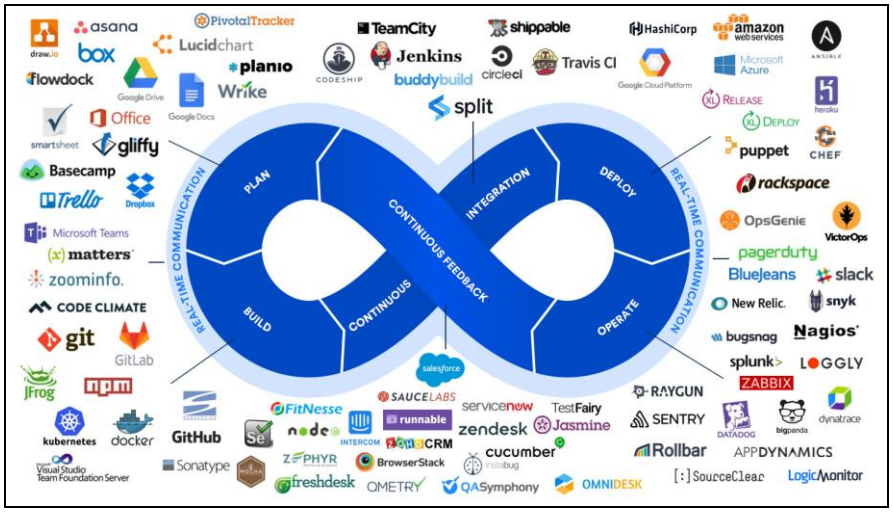
- Estudiar la metodologia DevOps en el cicle de desenvolupament de codi des del punt de vista de la infraestructura.

- Investigar i comprendre els principis de Semantic Versioning i Conventional Commits com a part integral del desenvolupament de codi i la gestió de canvis en la infraestructura quan treballem amb un controlador de versions.

- Implementar la Infraestructura com a Codi mitjançant Terraform, desenvolupant codi i exemples de mòduls per als laboratoris.

- Dominar l'ús de Terraform com a eina principal per a la descripció i el desplegament de la infraestructura com a codi.
- Utilitzar Amazon Web Services com a proveïdor de núvol.
- Desplegar infraestructures en AWS i crear laboratoris amb exemples reals.
- Implementar CI/CD amb GitHub Actions i/o Jenkins per a l'automatització del cicle de vida del codi i les actualitzacions de la infraestructura.

Figura 1. Capturant la diversitat d'eines clau, la DevOps cloud community destaca la automatització, integració i gestió de la infraestructura com a codi. Explorarem les millors pràctiques en el desenvolupament i gestió de la infraestructura, iniciant el desxiframent del complex món de DevOps.



Font Alibaba: Cloud Community DevOps.

1.4. Requeriments per a la realització del TFG

Per al desenvolupament d'aquest TFG, s'utilitzaran els següents requisits:

1.4.1. Maquinari: Entorn local de desenvolupament

- Sistemes operatius: A disposició he tingut aquests sistemes operatius:
    - MacOS High Sierra.
    - Windows 11 + WSL Ubuntu.
- Per a les tasques relacionades s'utilitzaran dos maquines amb suficient espai d'emmagatzematge lliure per a dur a terme el desenvolupament i proves de forma eficaç.

- MacBook Air 13, amb i5 i 8 GB de RAM.
- Lenovo ThinkPad 15, amb AMD Ryzen 5 i 16 GB de RAM.

1.4.2. Programari: Eines i plataformes

- Comptes de GitHub i AWS: Utilitzant una llicència gratuïta o de desenvolupament, per a l'accés i l'ús de les corresponents plataformes.
- Software de gestió de codi com Git i GitHub per a control de versions i col·laboració.
- Terraform > 5.7 per a la creació i gestió de la infraestructura com a codi.
- Jenkins > 2.0 com a eina de CI/CD per a automatitzar les fases de desenvolupament i distribució.
- *Visual Studio Code* per a l'edició de codi i desenvolupament.

1.4.3. Documentació

- Descrit en l'apartat "11. Referències"

1.4.4. Legal i Seguretat

- Assegurar que totes les pràctiques i les solucions implementades compleixen les millors practiques de seguretat.
- Abordar les consideracions de seguretat relacionades amb la infraestructura i el desenvolupament de codi, incloent-hi pràctiques de seguretat i protecció de dades.

1.5. Planificació del TFG

Per tal de poder complir amb les dates de entrega i portar a terme la feina necessària per a la realització d'aquest TFG s'ha realitzat una planificació elaborada amb Microsoft Project i basada en Gantt.

Aquest fitxer es troba disponible dins de l'apartat "10. Annexos":

Nom tasca	Comença	Fi
Inici del projecte	**dl 25/09/23**	**dc 04/10/23**
Kick off	dl 25/09/23	dg 01/10/23
Enviar email amb els objectius de treball dins del TFG al Tutor	dg 01/10/23	dg 01/10/23
Reunió PAC1 amb el tutor i validació proposta	dc 04/10/23	dc 04/10/23
Fase I - Entrega PAC 1 - 10%	**dc 04/10/23**	**dg 15/10/23**
Proposta de pla de treball	dc 04/10/23	dg 15/10/23
Descripció del treball	dc 04/10/23	dg 15/10/23

Objectiu del treball	dc 04/10/23	dg 15/10/23
Requeriments	dc 04/10/23	dg 15/10/23
Planificació	dc 04/10/23	dg 15/10/23
Esbós de sumari	dc 04/10/23	dg 15/10/23
Fase II - Entrega PAC 2 - 40 %	**dl 16/10/23**	**dg 19/11/23**
Obtenció de llicencies i comptes per a eines del TFG: AWS, GitHub, etc	dl 16/10/23	dg 19/11/23
Creació de recursos i preparació del entorn de desenvolupament	dl 16/10/23	dg 19/11/23
Inici de memòria, formats i documents associats	dl 16/10/23	dg 19/11/23
Memòria: Apartat 1. Introducció del TFG	dl 16/10/23	dg 19/11/23
Memòria: Apartat 2. DevOps: Concepte i Marc Actual	dl 16/10/23	dg 19/11/23
Investigar, llegir articles i història del concepte	dl 16/10/23	dg 19/11/23
Llegir i investigar documentació de Terraform	dl 16/10/23	dg 19/11/23
Llegir i investigar documentació de AWS	dl 16/10/23	dg 19/11/23
Memòria: Apartat 3. Computació al Núvol	dl 16/10/23	dg 19/11/23
Estudi i comparativa de proveïdors al núvol	dl 16/10/23	dg 19/11/23
Memòria: Apartat 4. Infraestructura com a Codi	dl 16/10/23	dg 19/11/23
Investigar dels mòduls de Terraform a desenvolupar	dl 16/10/23	dg 19/11/23
Definir els mòduls de Terraform a desenvolupar	dl 16/10/23	dg 19/11/23
Generar guia d'instal·lació de Terraform	dl 16/10/23	dg 19/11/23
Desenvolupament: Crear i desenvolupar mòduls de Terraform	dl 16/10/23	dg 19/11/23
Preparació i documentació dels laboratoris proposats	dl 16/10/23	dg 19/11/23
Fase III - Entrega PAC 3 - 40%	**dl 20/11/23**	**dg 24/12/23**
Memòria: Apartat 5. Cicle de vida i Desenvolupament de Codi	dl 20/11/23	dg 24/12/23
Llegir i investigar documentació de Jenkins i GitHub Actions	dl 20/11/23	dg 24/12/23
Definició + inici desenvolupament del Laboratori 1	dl 20/11/23	dg 24/12/23
Definició + inici desenvolupament del Laboratori 2	dl 20/11/23	dg 24/12/23
Definició + inici desenvolupament del Laboratori 3	dl 20/11/23	dg 24/12/23
Estudi de viabilitat de ampliació dels laboratoris proposats	dl 20/11/23	dg 24/12/23
Memòria: Apartat 6. Eines de CI / CD	dl 20/11/23	dg 24/12/23
Validació dels recursos desenvolupats associats als laboratoris	dl 20/11/23	dg 24/12/23
Memòria: Apartat 7. Laboratoris d'exemple	dl 20/11/23	dg 24/12/23
Desenvolupament: Codi per el laboratori 1	dl 20/11/23	dg 24/12/23
Desenvolupament: Codi per el laboratori 2	dl 20/11/23	dg 24/12/23
Desenvolupament: Codi per el laboratori 3	dl 20/11/23	dg 24/12/23
Memòria: Format final de referencies i bibliografia	dl 20/11/23	dg 24/12/23
Revisió documents annexos i documentació dels laboratoris	dl 20/11/23	dg 24/12/23
Fase Final: Lliurament final de Memòria i presentació - 10%	**dl 25/12/23**	**dg 14/01/24**
Revisió final memòria i documents associats	**dl 25/12/23**	**dg 14/01/24**
Realitzar la presentació de la memòria	**dl 25/12/23**	**dg 14/01/24**
Gravació de la presentació final	**dl 25/12/23**	**dg 14/01/24**
Defensa TFG davant del tribunal	**dl 15/01/24**	**dt 30/01/24**

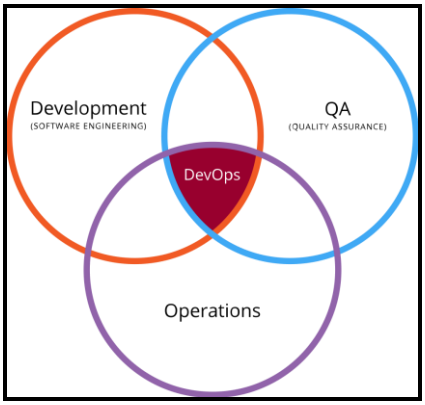
2. DevOps: Concepte i Marc Actual

Aquesta secció proporciona una descripció detallada del concepte i l'entorn actual del moviment DevOps, que són fonamentals per comprendre l'abordatge d'aquest treball.

Principalment s'explorarà l'essència del concepte de com un enfocament interdisciplinari i col·laboratiu que uneix les àrees de desenvolupament de software i operacions. Es discutiran els principis bàsics que defineixen aquest concepte, com ara la integració contínua, l'entrega contínua, i l'automatització de processos.

Respecte al marc actual, s'analitzarà l'entorn tecnològic, que està caracteritzat per un canvi constant i ràpid. Es discutirà com les pràctiques i els seus principis són essencials per afrontar aquesta dinàmica canviant i com contribueixen a una gestió eficient de les infraestructures i una entrega de codi eficaç.

Figura 2. En aquesta visualització, els cercles de Desenvolupament, Control de Qualitat i Operacions convergeixen, simbolitzant la intersecció crucial de DevOps. Aquesta imatge captura la síntesi de pràctiques i processos entre aquests àmbits.



Font Wikipedia: Intersecció de DevOps en desenvolupament i operacions.

2.1. Historia, definició i significat de DevOps

El terme DevOps va ser utilitzat per primera vegada durant la conferència Agile del 2008 a Toronto, en una xerrada centrada en la "Infraestructura Àgil", des de llavors, el terme ha experimentat un creixent reconeixement i popularització.

Al llarg del temps, diverses definicions han emergit per capturar la seva veritable essència, tot i que la mes popular es referir-se al terme com un conjunt de pràctiques dissenyades per reduir el temps entre l'aprovació d'un canvi en una aplicació i la seva implementació en producció.

Aquesta definició destaca la importància d'aspectes com la celeritat, la integritat i l'automatització com a components fonamentals de DevOps. El seu propòsit fonamental radica en la unió harmoniosa dels processos de desenvolupament de software i operacions de IT, amb l'objectiu de refinament del cicle de vida del desenvolupament de software.

El moviment té com a objectiu principal accelerar el lliurament de codi cap a l'entorn de producció, aquest objectiu es busca aconseguir mitjançant la implementació de pràctiques com la integració contínua, l'entrega contínua i l'automatització, les quals es despleguen a tots els nivells operatius.

2.2. L'evolució del moviment DevOps

El moviment DevOps ha evolucionat des d'un conjunt de pràctiques fins a convertir-se en un dels perfils més demandats a la indústria TIC. Tot i que es va originar de forma inicial com una cultura i un enfocament per fomentar la col·laboració en el cicle de vida de desenvolupament de programari, avui dia s'ha convertit en un terme àmpliament utilitzat per descriure un conjunt divers de tasques i responsabilitats.

Per a molts professionals, no és tracta simplement d'un rol o un conjunt d'eines, sinó que és un concepte que engloba l'adopció d'eines, enfocaments àgils per al desenvolupament de programari, la creació de processos eficients i l'automatització.

La transició a DevOps i l'adopció de les seves pràctiques sovint implica superar barreres culturals i operatives a les organitzacions. A mesura que el terme ha evolucionat, el perfil s'ha convertit en un dels més sol·licitats a la indústria TIC. Tot i que DevOps pot implicar una àmplia gamma de tasques i habilitats, és essencial comprendre l'amplitud i diversitat d'aquest rol i reconèixer la necessitat de comptar amb un conjunt variat de coneixements.

Tot i que s'ha convertit en un terme de moda i un títol de feina comú, és fonamental recordar que segueix sent un concepte profund i multifacètic que fomenta la col·laboració entre equips de desenvolupament i operacions. En resum, el perfil de DevOps s'ha transformat de un conjunt de pràctiques a un dels perfils més demandats a la indústria tecnològica.

Podem dir que l'evolució del moviment DevOps es pot descriure en cinc etapes (Figura 3):

- **Normalització del stack tecnològic:** En aquesta etapa, els equips adopten mètodes àgils i estableixen un control de versions adequat. S'eliminen tecnologies redundants i s'estandarditza la pila tecnològica. També es col·loquen les configuracions de les aplicacions al control de versions.

- **Estandardització i reducció de la variabilitat:** Els equips treballen per reduir la variabilitat, consolidant la tecnologia en una única família de sistemes operatius, minimitzant la complexitat del procés i explorant oportunitats de col·laboració. Aquí es important que les configuracions del sistema es posin al control de versions i les aplicacions es redissenyin segons les necessitats del negoci.

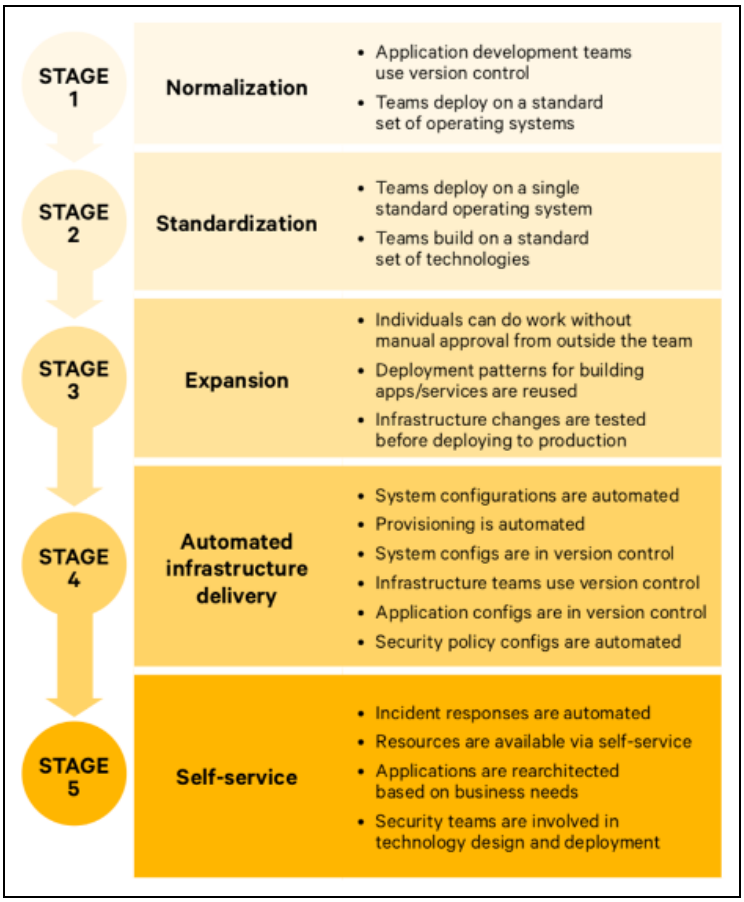
- **Expansió de les pràctiques de DevOps:** Amb les bases establertes, s'aborden les discrepàncies post-canvis. S'assegura que el resultat del desenvolupament concordi amb l'entrega, eliminant la burocràcia per permetre canvis immediats.

- **Automatització de l'entrega d'infraestructura:** L'automatització de l'entrega d'infraestructura redueix la discrepància entre la producció del desenvolupament i els terminis de lliurament de les operacions. Això permet lliuraments més ràpids i configuracions per a futurs autoserveis.

- **Proveir capacitats d'autoservei:** Aquí s'amplien i es posen a disposició més recursos d'autoservei. Els desenvolupadors poden implementar entorns de proves pel seu compte.

Pel general, aquestes etapes marquen el camí cap a una implementació d'èxit de DevOps, i cadascuna d'elles contribueix a optimitzar la col·laboració i l'entrega més eficient de programari.

Figura 3. És detalla l'evolució de DevOps en cinc fases clau. Cada etapa contribueix a la convergència de pràctiques, eliminant discrepàncies i permetent l'entrega àgil d'infraestructura.



Font Weave: The DevOps evolution model.

2.3. Millors pràctiques i beneficis de DevOps en la gestió d'infraestructures

DevOps ha aconseguit transformar la gestió d'infraestructures mitjançant l'ús de l'Automatització. Aquest concepte implica la gestió de la infraestructura mitjançant codi, aquest és un aspecte crucial en l'adopció plena de les pràctiques de DevOps.

Quan la infraestructura es gestiona correctament, garanteix que els recursos estiguin configurats adequadament, siguin segurs, es facin còpies de seguretat regularment i es supervisin periòdicament.

Una de les pràctiques més importants és la Infraestructura com a Codi, que permet aprovisionar l'entorn IT amb codi en comptes de configuracions manuals.

Això agilitza les operacions que, de manera manual podrien trigar hores o fins i tot dies a ser configurades, i ofereix una infraestructura àgil i preparada per a les proves.

Els beneficis d'utilitzar DevOps en la gestió de la infraestructura inclouen:

- **Predictibilitat:** La taxa d'error en nous desplegament és significativament menor ja que els productes es testegen en les etapes inicials del procés de desenvolupament.

- **Millora de la Qualitat:** Es permet als equips crear aplicacions de millor qualitat tenint en compte els aspectes reals de la infraestructura.

- **Eliminació d'Errors i Temps de Recuperació Reduït:** Les pràctiques que es fomenten són efectives per eliminar l'impacte de revessos, punts de bloqueig o falles en la implementació que puguin afectar l'eficiència. La detecció ràpida d'errors facilita el treball d'equips d'operacions i desenvolupament.

- **Reducció del Temps de Comercialització:** S'agilitza el lliurament del programari, cosa que accelera el procés de comercialització i beneficia al negoci.

3. Computació al Núvol

Aquesta secció es centra en un concepte tecnològic que ha revolucionat la infraestructura empresarial: la Computació al Núvol. Aquesta tecnologia ha esdevingut una peça clau en empreses i organitzacions de diversos sectors.

Es desglossaran els fonaments d'aquest concepte, es discutiran les raons que han portat a la seva àmplia adopció i s'exploraran els diferents models que ofereix. A més, s'abordaran aspectes crítics com la seguretat i la privacitat en l'entorn de la Computació al Núvol.

Aprofundint més, considerant que Amazon Web Services és el proveïdor seleccionat per a aquest treball, es realitzarà una breu comparació amb altres proveïdors rellevants. També es donarà una visió general del conjunt d'eines i serveis fonamentals proporcionats per AWS en relació amb la gestió de la infraestructura.

Figura 4. Ofereix una visió sintètica del panorama divers de la computació al núvol, basada en el contingut d'aquesta. Aquesta imatge proporciona una comprensió visual dels possibles elements en la computació al núvol.

Font Helpdeskgeek: *10 Types of Cloud Computing You Should Know About.*

3.1. Introducció del concepte

La computació en núvol és la disponibilitat sota demanda de recursos de sistemes informàtics, especialment emmagatzematge de dades i potència de computació, sense gestió activa directa per part de l'usuari.

Aquesta tecnologia sovint disposta de funcions distribuïdes en múltiples ubicacions, cadascuna de les quals és un centre de dades. La computació en el núvol es basa en compartir recursos per aconseguir coherència i típicament utilitza un model de pagament per ús, que pot ajudar a reduir despeses de capital però també pot conduir a despeses operatives inesperades per als usuaris.

Principalment, el concepte inclou cinc característiques essencials, segons la definició de l'Institut Nacional d'Estàndards i Tecnologia dels Estats Units:

- **Servei a la carta:** Els usuaris poden aprovisionar de forma autònoma, capacitats informàtiques segons les seves necessitats, sense requerir interacció humana directa.

- **Accés a través de la xarxa:** Aquestes capacitats són accessibles a través de la xarxa i poden ser utilitzades en diferents dispositius, des de telèfons mòbils fins a estacions de treball.

- **Agrupament de recursos:** Els proveïdors de núvol allotgen eficientment recursos entre diversos usuaris, ajustant-los dinàmicament segons la demanda dels consumidors.

- **Elasticitat en l'escalat:** El núvol permet l'escalabilitat ràpida dels recursos en resposta a canvis en la demanda, sovint de manera automàtica.

- **Seguretat i confiança:** Els sistemes de núvol supervisen i optimitzen l'ús dels recursos mitjançant un mesurament de la informació sobre els serveis, garantint la transparència tant per als proveïdors com per als usuaris.

La història de la computació al núvol comença en la dècada de 1960, quan les empreses tenien l'opció de llogar temps en grans equips informàtics en lloc de comprar-ne assumint els costos que suposava.

Aquest concepte es coneixia com a "serveis de compartició de temps". Amb l'arribada dels primers ordinadors personals, la propietat d'un equip informàtic es va fer molt més assequible i popular, relegant els serveis de compartició de temps a un segon pla.

Tot i això, la idea de llogar l'accés a la potència informàtica va ressorgir en diverses ocasions. A finals de la dècada de 1990 i principis dels anys 2000, es van explorar conceptes com els proveïdors de serveis d'aplicacions, la computació d'utilitat i la computació en graella.

Finalment, es va consolidar amb el naixement de la computació al núvol, que va agafar impuls amb l'aparició del programari com a servei i els grans proveïdors de núvol a escala global, com Amazon Web Services.

3.2. Per que la computació al Núvol és tan utilitzada actualment?

En l'actualitat, la computació al núvol ha esdevingut un element essencial, moltes empreses estan adoptant aquesta tecnologia amb rapidesa. Però, per què ha guanyat tanta popularitat?

Històricament, la infraestructura informàtica era costosa i complexa, limitant l'accés a les empreses més grans. La computació al núvol va canviar aquesta dinàmica. Amb els serveis de núvol, les empreses poden accedir a recursos informàtics sense haver de

realitzar inversions significatives en hardware i manteniment. Això ha obert les portes a petites i mitjanes empreses i ha democratitzat la tecnologia.

L'escalabilitat és un altre factor crític que ha contribuït a la popularitat de la computació al núvol. Les empreses poden augmentar o reduir els seus recursos segons les seves necessitats, la qual cosa resulta més eficient i econòmic. La capacitat d'adaptar-se ràpidament a les fluctuacions en la demanda del mercat és fonamental, i la computació al núvol ho fa possible.

També ofereix flexibilitat en termes de treball remot i col·laboració gracies a l'accés a les dades i aplicacions des de qualsevol ubicació amb connexió a Internet, els empleats poden treballar de manera més flexible. Això s'ha convertit en un avantatge significatiu en el món empresarial actual, on la mobilitat i la col·laboració són crucials.

Per acabar, l'augment de la seguretat i la fiabilitat en els serveis de núvol ha fet que les empreses confiïn en aquesta tecnologia. Les grans empreses de serveis de núvol inverteixen en mesures de seguretat i certificacions per garantir la protecció de les dades dels seus clients. Això ha estat fonamental per guanyar la confiança de les empreses en la computació al núvol.

3.3. Models de desplegament i de serveis de la computació al Núvol

Principalment revisarem de forma global com les organitzacions accedeixen als recursos de la núvol i com utilitzen aquesta tecnologia per millorar les seves operacions i serveis.

3.3.1. Models de desplegament de la computació al núvol

Els models de desplegament en la computació al Núvol ofereixen una visió essencial sobre com les organitzacions gestionen i accedeixen als recursos de la núvol. Aquests models, tal com estandarditza l'Institut Nacional d'Estàndards i Tecnologia, es divideixen en quatre categories amb base en la ubicació, la gestió i l'accessibilitat dels usuaris:

- **Pública:** La infraestructura del servei és oferta per una empresa proveïdora i és accessible al públic en general. Per exemple, Amazon Web Services ofereix una ampla gamma de serveis de núvol públic a empreses i desenvolupadors de tot el món. Aquests serveis estan disponibles per a qualsevol que vulgui utilitzar-los.

- **Privada:** La infraestructura del servei és propietat d'una organització i només és accessible per als usuaris de forma privada i sota la seva autorització. Un exemple seria quan una gran empresa decideix crear el seu propi núvol privat per emmagatzemar dades sensibles i aplicacions internes, assegurant que només els seus empleats autoritzats tinguin accés a aquestes dades.

- **Híbrida:** Implica l'ús d'una combinació d'infraestructures, tant públiques com privades. Un exemple seria quan una organització utilitza un núvol públic per a les seves necessitats de càrregues de treball variables i, al mateix temps, manté una infraestructura privada per a dades més sensibles i controlades internament.

- **Comunitària:** La infraestructura del servei està dissenyada per satisfer les necessitats d'un grup de diverses organitzacions amb interessos comuns. Un exemple podria ser un grup d'organitzacions del sector de la salut que col·labora per crear un núvol comunitari que compleixi amb uns requisits de privadesa i seguretat específics.

3.3.2. Models de servei de la computació al núvol

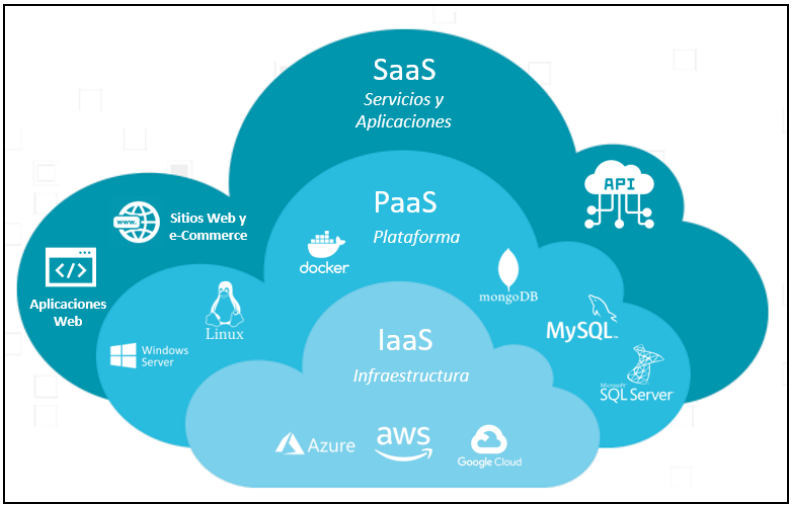
Pel que fa als models de serveis, la computació al Núvol ofereix principalment tres models fonamentals, cadascun d'ells essencial per a les funcions que ofereixen:

- **Programari com a servei (SaaS):** Aquest model permet als usuaris accedir i utilitzar aplicacions i programari a través d'Internet. Un exemple seria l'ús de Microsoft 365, on els usuaris poden accedir a aplicacions com Word, Excel i Outlook mitjançant una connexió a Internet, sense necessitat d'instal·lar aquests progbranches als seus dispositius.

- **Plataforma com un servei (PaaS):** En aquest model, la plataforma ofereix als desenvolupadors eines i entorns per crear, provar i implementar aplicacions. Google Cloud Platform és un exemple de PaaS que proporciona eines i recursos per al desenvolupament d'aplicacions i serveis, sense que els desenvolupadors hagin de gestionar la infraestructura subjacents.

- **Infraestructura com un servei (IaaS):** Aquest model proporciona recursos informàtics virtuals, com màquines virtuals i emmagatzematge, mitjançant Internet. Amazon Web Services ofereix una gama d'opcions d'IaaS, permetent als usuaris crear i gestionar recursos informàtics segons les seves necessitats específiques, com màquines virtuals per a servidors web o emmagatzematge escalable per a dades empresarials. Això ofereix als usuaris un major control i flexibilitat per gestionar la seva infraestructura de núvol.

Figura 5. És una visió resumida dels models de servei en la computació al núvol, tal com es presenta a la secció 3.3.2. Aquesta representació proporciona una comprensió clara dels tres models fonamentals: SaaS, PaaS i IaaS.



Font Qualitapps: *IaaS, PaaS i SaaS: Els models de serveis a la núvol.*

3.4. Seguretat i Privacitat en la Computació al Núvol

A continuació s'exploraran els desafiaments de seguretat i les solucions associades a la protecció de dades i a la garantia de la confidencialitat en un entorn de núvol.

La seguretat i la privacitat són dos aspectes crítics en l'entorn de la computació al núvol. A mesura que les empreses i les organitzacions emmagatzemen i gestionen cada vegada més dades i aplicacions en serveis de núvol, es fa necessari abordar les preocupacions relacionades amb la seguretat i la protecció de la privadesa.

3.4.1 Seguretat en la Computació al Núvol

La seguretat en la computació al núvol es refereix a les pràctiques i les mesures adoptades per garantir que les dades i les aplicacions emmagatzemades en el núvol estiguin protegides contra amenaces i atacs cibernètics. Algunes de les principals preocupacions de seguretat en la computació al núvol inclouen:

- **Accés no autoritzat:** L'accés a dades i recursos sensibles per part de persones no autoritzades és una amenaça important. Les empreses han de garantir que només els usuaris autoritzats puguin accedir a les seves dades en el núvol.

- **Integritat de les dades:** Assegurar-se que les dades emmagatzemades no siguin alterades de manera no autoritzada és fonamental. Qualsevol canvi no autoritzat pot tenir conseqüències fatals per a la integritat de les dades.

- **Disponibilitat:** La disponibilitat de les dades és essencial. Les empreses han de preveure fallades del sistema i assegurar-se que les dades siguin accessibles en tot moment.

- **Conformitat legal:** Les empreses han de complir amb les regulacions i les lleis de protecció de dades. La violació d'aquestes regulacions pot tenir conseqüències legals i financeres.

- **Gestió de claus:** La gestió de les claus d'encriptació és un component crític per a la seguretat. Les empreses han de garantir que les claus estiguin ben protegides i que només les persones autoritzades tinguin accés a elles.

3.4.2. Privacitat en la Computació al Núvol

La privadesa en la computació al núvol fa referència a la protecció de les dades personals emmagatzemades en aquest entorn. Algunes de les preocupacions de privadesa més importants en la computació al núvol inclouen:

- **Consentiment de l'usuari:** És essencial que les empreses obtinguin el consentiment dels usuaris abans de recopilar i emmagatzemar les seves dades personals. Les empreses han de ser transparents sobre com s'utilitzaran aquestes dades.

- **Accés a dades personals:** Les empreses han de garantir que les dades personals dels usuaris no siguin compartides o venudes a tercers sense el seu consentiment.

- **Dret a l'oblit:** Els usuaris han de tenir el dret de sol·licitar la supressió de les seves dades personals. Això implica eliminar totes les seves dades dels serveis de núvol.

- **Encriptació:** L'encriptació de les dades personals és una mesura clau per protegir la privadesa. Les dades han de ser inintel·ligibles per a tercers no autoritzats.

- **Conformitat amb les regulacions de privadesa:** Les empreses han de garantir que estiguin en conformitat amb les regulacions de privadesa aplicables, com el Reglament General de Protecció de Dades (RGPD) a la Unió Europea.

3.5. El futur de la computació al núvol

La computació al núvol ha experimentat un creixement significatiu des de la seva introducció, i el futur d'aquesta tecnologia promet continuar transformant la manera com les empreses i els usuaris emmagatzemen, gestionen i accedeixen a les dades i les aplicacions. Algunes de les tendències i les innovacions que es preveuen pel futur de la computació al núvol inclouen:

- **Edge Computing i Núvol Híbrid:** L'edge computing implica processar dades a prop dels dispositius o les fonts d'origen en lloc d'enviar-ho a un centre de dades central. Això redueix la latència i millora la velocitat de processament. En el futur, veurem una major integració entre l'edge computing i el núvol, creant entorns híbrids que oferiran una combinació de les dues tecnologies.

- **Computació Quàntica en el Núvol:** La computació quàntica és una àrea d'investigació que promet revolucionar la capacitat de processament. En el futur, podrem veure serveis de computació quàntica oferts a través del núvol, permetent la resolució de problemes complexos en àmbits com la criptografia, la medicina i la intel·ligència artificial.

- **Més Automatització:** L'ús de l'aprenentatge automàtic i l' IoT permetrà una major automatització en la gestió de recursos de núvol. Això inclou l'autoescalat de recursos, la seguretat basada en l'aprenentatge automàtic i la identificació proactiva de problemes de rendiment.

- **Realitat Virtual i Augmentada en el Núvol:** Les aplicacions de realitat virtual i realitat augmentada estan en augment. En el futur, les empreses podrien optar per emmagatzemar i processar les dades de RV i RA en el núvol, permetent una experiència d'usuari més rica i col·laborativa.

- **Compromís amb la Sostenibilitat:** La indústria de la computació està compromesa amb la sostenibilitat i la reducció de la petjada de carboni. En el futur, veurem núvols que operen de manera més ecològica mitjançant l'ús d'energies renovables i la millora de l'eficiència energètica.

3.6. Visió general d'AWS com a plataforma de núvol

Amazon Web Services (AWS) és una de les plataformes de núvol més prominents i àmplies disponibles avui dia. Fundada per Amazon.com el 2006, AWS ofereix una àmplia gamma de serveis de computació, emmagatzematge, xarxes, bases de dades, analítica i molts altres, que permeten a empreses i desenvolupadors crear i gestionar aplicacions de manera eficient i escalable. AWS és coneguda per la seva fiabilitat i escalabilitat, i és àmpliament utilitzada per empreses de tot tipus, des de petites startups fins a grans corporacions.

Els serveis d'AWS es proporcionen en centres de dades repartits per tot el món, el que permet l'expansió global i l'alta disponibilitat. Això també significa que els usuaris poden triar la ubicació del centre de dades que millor s'ajusti a les seves necessitats i requisits normatius.

3.7. Eines i serveis essencials d'AWS per a la gestió de la infraestructura

Entrant en matèria, a continuació farem un recorregut per els serveis i eines més populars del proveïdor AWS que permeten la gestió eficient de la infraestructura de núvol i l'execució d'aplicacions a escala.

- **Amazon Elastic Compute Cloud (EC2):** Ofereix serveis d'infraestructura per a la creació de màquines virtuals escalables i flexibles. És un dels serveis més utilitzats d'AWS, permetent als usuaris executar aplicacions en màquines virtuals amb diferents configuracions i sistemes operatius.

- **Amazon Simple Storage Service (S3):** És un servei d'emmagatzematge en línia que permet l'emmagatzematge segur i altament escalable d'arxius i dades. És àmpliament utilitzat per emmagatzemar i gestionar dades, realitzant còpies de seguretat i distribució de continguts.

- **Amazon Relational Database Service (RDS):** RDS proporciona serveis de bases de dades gestionades, incloent MySQL, PostgreSQL, Oracle i altres. Això permet als usuaris gestionar bases de dades sense haver de preocupar-se de la infraestructura subjacent.

- **Amazon Lambda:** AWS Lambda és un servei de computació sense servidors que permet als desenvolupadors executar codi en resposta a esdeveniments sense la necessitat de gestionar servidors. És ideal per a aplicacions amb càrregues de treball variables.

- **Amazon Virtual Private Cloud (VPC):** Permet als usuaris crear una xarxa privada virtual aïllada a AWS. Això ofereix un control granular sobre la xarxa i permet la connexió segura de recursos en núvol amb xarxes on-premises.

- **Amazon Route 53:** Aquest servei ofereix serveis de DNS escalables i fiables, juntament amb equilibri de càrrega i rutes optimitzades per a aplicacions.

- **Amazon CloudWatch:** És una eina de monitoratge i generació de registres que permet als usuaris seguir les mètriques i registres dels seus recursos en AWS. És essencial per garantir un rendiment òptim i detectar problemes de manera proactiva.

Figura 6. En la figura és destaquen els serveis principals d'Amazon Web Services. Aquesta imatge proporciona una visió resumida dels serveis essencials, incloent EC2, S3, RDS, Lambda, VPC, Route 53 i CloudWatch, descrits durant la secció.

Font Aallcode.com: Top AWS services.

3.8. AWS vs. Altres proveïdors: Una breu comparativa

A continuació, compararem Amazon Web Services amb altres proveïdors importants de serveis de núvol per ajudar a comprendre com aquesta es situa en el panorama de la computació en núvol.

Encara que hi ha molts proveïdors en aquest mercat, farem una comparació amb dos dels més reconeguts i que es fa competència directa: Microsoft Azure i Google Cloud Platform.

Taula 1. En aquesta comparativa, analitzem breument Amazon Web Services en relació amb altres proveïdors destacats de serveis de núvol, especialment Microsoft Azure i Google Cloud Platform.

	Amazon Web Services	Microsoft Azure	Google Cloud Platform
Gama de serveis	Àmplia varietat de serveis per a diverses aplicacions.	Integració amb productes Microsoft i ofertes per a empreses.	Destacat en anàlisi de dades i intel·ligència artificial.
Ubicacions de Centres de Dades	Centres de dades en nombroses regions al voltant del món.	Presència global significativa amb centres de dades en moltes regions.	Menys regions en comparació amb AWS i Azure.
Facturació i Flexibilitat	Estructura de preus flexible i variada. Pagament només pel que s'utilitza.	Opcions de preus flexibles i descomptes per a compromisos a llarg termini.	Estructura de preus competitiva amb facturació per segon per a molts serveis.
Suport i Servei al Client	Xarxa global de suport i oficines d'atenció al client. Nivells de suport diferents.	Gran presència global amb diverses opcions de suport. Suport per a empreses amb tecnologies Microsoft.	Suport tècnic altament qualificat amb un enfocament a la innovació i la tecnologia.

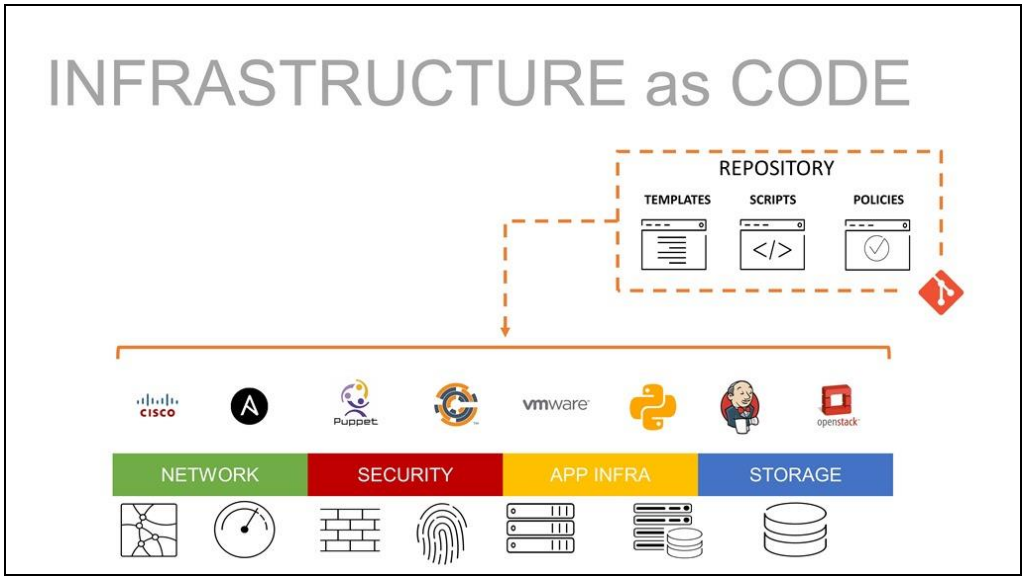
En general, AWS és l' opció preferida per moltes empreses. La seva àmplia gamma de serveis, la robusta seguretat i fiabilitat, la seva àmplia presència global i el compromís continu amb la innovació. Aquestes característiques fan que AWS sigui la plataforma líder en la computació al núvol i segons la meva experiència professional, és la elecció més confiable i completa per a les necessitats empresarials contemporànies.

4. Infraestructura com a Codi

La Infraestructura com a Codi és una pràctica que esdevé cada vegada més crítica en l'àmbit IT i en la gestió de la infraestructura. En aquesta secció, ens centrarem en el concepte per comprendre com està remodelant la forma com les empreses gestionen i implementen la seva infraestructura tecnològica.

Explorarem com aquest enfocament està optimitzant aspectes clau com l'escalabilitat, la gestió, i la consistència de la infraestructura, mitjançant l'ús de codi. Descobrirem que aquesta metodologia no només millora l'eficiència operativa, sinó que també obre portes a la innovació i la flexibilitat.

Figura 7. En aquesta representació gràfica de la Infraestructura com a Codi, s'identifiquen elements clau que exemplifiquen la integració i la gestió eficient dels recursos. La imatge simbolitza la sincronització harmoniosa entre el codi i la infraestructura. Aquesta visió captura l'essència de la IaC i proporciona una base visual per explorar aprofundit la secció dedicada a aquest concepte.



Font f5.com: ¿Qué es la infraestructura como código?

4.1. Concepte i utilitat de la IaC

La Infraestructura com a Codi (IaC) és un concepte que ha causat un impacte significatiu en la gestió i l'aprovisionament de la infraestructura. Bàsicament, aquest concepte implica gestionar els centres de dades i els recursos informàtics a través de fitxers de definició llegibles per màquina, en oposició a la configuració manual de maquinari físic o eines de configuració interactives.

Aquest enfocament no només fa referència als components físics, com ara els servidors de metall de tota la vida, sinó que també inclou màquines virtuals i els recursos de configuració associats. Una característica distintiva és que aquestes definicions d'infraestructura sovint es gestionen a través de sistemes de control de versions, facilitant el seguiment de canvis i la col·laboració en projectes d'infraestructura.

La IaC ofereix la flexibilitat d'utilitzar tant scripts com definicions declaratives per descriure els components de la infraestructura desitjada. Aquest enfocament elimina la necessitat de processos manuals, fomentant l'ús de codi com a mitjà per descriure i gestionar els components d'infraestructura. En termes generals, aquesta pràctica és particularment adequada per a la computació en núvol.

4.2. Origen i popularitat de la IaC

El concepte d'Infraestructura com a Codi va sorgir com una resposta als reptes derivats de dues importants evolucions tecnològiques. El llançament de l' Elastic Compute Cloud (EC2) d'Amazon Web Services i la versió 1.0 de Ruby on Rails l'any 2006 va provocar problemes d'escalabilitat significatius per a moltes organitzacions. A mesura que la tecnologia va evolucionar per abordar aquestes creixents demandes, la IaC es va establir com una solució lògica.

La idea d'expressar la infraestructura com a codi, amb la capacitat de dissenyar, implementar i desplegar la infraestructura seguint les millors pràctiques del desenvolupament de programari, va atraure l'atenció dels desenvolupadors i administradors d'infraestructura. Aquest enfocament va permetre gestionar la infraestructura amb la mateixa agilitat que en altres projectes de programari, impulsant ràpides implementacions d'aplicacions.

La popularitat de la IaC ha crescut de forma exponencial en els últims anys, gràcies als seus avantatges en eficiència, escalabilitat i gestió. Moltes organitzacions han abraçat aquest enfocament per millorar els seus processos i estàndards d'infraestructura, la qual cosa ha contribuït a la seva difusió generalitzada en el sector de la tecnologia.

4.2.1. Quin Problema Resol la IaC?

Ara que sabem què és la Infraestructura com a Codi, centrarem la nostra atenció en la raó per la qual és necessària i els problemes que ajuda a resoldre. Abans de la IaC, la gestió de la infraestructura era un procés manual, costós i poc escalable. El desplegament d'aplicacions era lent i sovint inconsistent, i no es garantia la disponibilitat.

Tot i que la computació a la núvol va alleujar alguns d'aquests problemes, està lluny de ser una panacea. Si bé et permet configurar ràpidament les teves necessitats d'infraestructura, resol problemes importants com l'alta disponibilitat i l'escalabilitat, però no fa res per solucionar els problemes de coherència. Quan més d'una persona realitza les configuracions, les discrepàncies són inevitables.

És aquí on entra la IaC, aquest enfocament revolucionari permet gestionar la infraestructura com si fos codi, oferint una solució a aquests desafiaments i millorant la gestió de la infraestructura. Aprofundirem en com ho fa a continuació.

4.3. Eines més conegudes i utilitzades per a IaC

El món de les eines d'Infraestructura com a Codi ofereix un conjunt de possibilitats per aprovisionar i gestionar la infraestructura. Navegar per aquesta àmplia gamma pot ser una tasca dura, especialment quan s'han de considerar els requisits del projecte, les plataformes de destinació i les habilitats de l'equip.

En aquesta secció, farem un breu viatge per descobrir les característiques de les principals eines d'Infraestructura com a Codi sense entrar en detalls de comparació. Cada eina té el seu propi conjunt d'avantatges i àrees d'aplicació, i la elecció de l'eina adequada dependrà dels requisits particulars de l'organització i del projecte.

- **Terraform**: Desenvolupada per HashiCorp, Terraform és reconeguda com l'orquestrador universal d'Infraestructura com a Codi (IaC). La seva característica principal rau en la capacitat d'aprovisionar i gestionar infraestructura de manera declarativa, proporcionant una abstracció efectiva per treballar amb múltiples proveïdors de núvol, com AWS, Azure i Google Cloud.

  La força de Terraform resideix en la seva facilitat per descriure i mantenir la infraestructura mitjançant codi, permetent als equips definir recursos, dependències i configuracions amb eficàcia i consistència.

Font: Pàgina oficial https://www.terraform.io/

- **AWS CloudFormation:** Dirigida específicament a l'ecosistema d'Amazon Web Services (AWS), AWS CloudFormation destaca per la seva utilització de plantilles declaratives en YAML o JSON anomenades stacks. Aquest enfocament simplifica la definició i gestió de recursos d'AWS, proporcionant un control detallat i la capacitat d'establir canvis de conjunts i disparadors de retrocés.

  La seva integració nativa amb els serveis d'AWS ofereix una solució robusta per als equips que es basen en aquest núvol.

Font: Pàgina oficial https://aws.amazon.com/es/cloudformation/

- **Azure Resource Manager:** Amb l'objectiu de simplificar el desplegament i la gestió de la infraestructura a Microsoft Azure, Azure Resource Manager utilitza plantilles declaratives en JSON per descriure i coordinar recursos. Aquesta eina ofereix una visió integral dels recursos i les seves dependències, permetent als usuaris gestionar la infraestructura amb eficiència i consistència.

  La seva estreta integració amb Azure facilita l'aprovisionament i la gestió de recursos en aquesta plataforma.

  Font: Pàgina oficial https://azure.microsoft.com/resource-manager/

- **Google Cloud Deployment Manager:** Com a eina de Google Cloud, Deployment Manager es destaca per la seva automatització eficient en la creació, desplegament i gestió de recursos a la plataforma Google Cloud.

  Utilitzant Python o YAML com a llenguatge de configuració declaratiu, aquesta eina ofereix flexibilitat per definir recursos i proporciona una única font de veritat per a la configuració. Aquest enfocament simplifica el cicle de vida dels recursos i facilita la seva gestió eficient.

  Font: Pàgina oficial: https://cloud.google.com/deployment-manager/

- **Puppet:** És una eina destacada d'automatització i gestió de configuració que utilitza el seu llenguatge de domini declaratiu conegut com a Puppet Code. Amb una arquitectura centralitzada, emmagatzema el codi al servidor principal, i els agents de Puppet l'executen als sistemes objectiu.

  La seva singularitat rau en l'habilitat per definir l'estat desitjat de la infraestructura de manera eficient i escalable, proporcionant processos de configuració controlats i reproduïbles. Encara que no es limita a un proveïdor específic de núvol, la seva aplicació abasta més enllà de la gestió d'infraestructura en núvol.

  Font: Pàgina oficial https://www.puppet.com/

- **Ansible:** És una eina centrada en l'orquestració i gestió de la configuració, oferint als usuaris la capacitat de definir tasques d'infraestructura mitjançant playbooks basats en YAML imperatiu.

  La seva operació sense agents i la connexió temporal via SSH o Windows Remote Management proporcionen simplicitat i seguretat. Ansible és una eina cloud agnostic, el que significa que és versàtil i no està vinculada a un proveïdor de núvol específic, permetent als usuaris gestionar infraestructures en diversos entorns sense restriccions.

  Font: Pàgina oficial https://www.ansible.com/

- **Chef:** Principalment destaca com a eina d'automatització i gestió de configuració que permet als usuaris definir, configurar i gestionar la infraestructura com a codi.

  Utilitza un model de receptes i desitjos per assegurar que la infraestructura segueixi les configuracions desitjades. Chef és una eina cloud agnostic, oferint flexibilitat per gestionar infraestructures en diferents proveïdors de núvol, adaptant-se a les necessitats diverses dels usuaris.

  Font: Pàgina oficial https://chef.io/

4.4. Per què utilitzar Terraform com a eina principal?

La selecció de la millor eina d'IaC depèn de diversos factors. En aquest context, Terraform destaca com una elecció preferida, i aquí s'expliquen les raons principals:

Terraform és reconeguda com una eina agnòstica de plataformes, el que significa que pot gestionar i aprovisionar recursos a través de múltiples proveïdors de núvol. Aquesta característica fa que sigui una elecció versàtil per a empreses que operen en diversos núvols o que estan considerant la migració a nous proveïdors.

A més, utilitza el seu propi llenguatge de configuració conegut com a HashiCorp Configuration Language (HCL). Aquest llenguatge és llegible per humans i permet als equips descriure la infraestructura de manera concisa i comprensible. La simplicitat de l'HCL facilita el desenvolupament, la col·laboració i la revisió del codi d'infraestructura.

Terraform incorpora una funcionalitat clau anomenada gestió d'estat que permet als equips visualitzar l'estat actual de la infraestructura i comparar-lo amb la configuració desitjada. Aquesta funció ajuda a prevenir canvis no autoritzats i assegura que la infraestructura estigui sempre en conformitat.

La comunitat activa i la gran base d'usuaris fan de que sigui una eina fortament recolzada. Això es tradueix en una àmplia gamma de mòduls predefinits, exemples i recursos disponibles. També implica que hi ha un suport actiu per part d'altres usuaris que poden ajudar a resoldre problemes i compartir coneixements.

Un punt fort addicional de Terraform és la seva capacitat de coexistir amb altres eines. En lloc de competir amb les eines de gestió de configuració, ja que s'integra amb elles, permetent als equips utilitzar-les en sistemes individuals mentre mantenen una visió d'infraestructura més àmplia i orquestrada.

4.5. Instal·lació de Terraform i eines complementàries

En aquesta secció, ens enfocarem en les passes necessàries per preparar l'entorn de treball i procedirem amb la instal·lació de Terraform i algunes de les eines que utilitzarem més endavant per tal de garantir la integritat, la seguretat o la documentació.

4.5.1. Terraform

Es distribueix com un únic binari que es pot descarregar des de la pàgina de Terraform i utilitzarem la guia d'instal·lació recomanada que s'ajusti al nostre sistema operatiu.
https://developer.hashicorp.com/terraform/tutorials/aws-get-started/install-cli

```
sudo apt-get install terraform
```

Comprovem que la instal·lació s'ha produït correctament:

```
ujvaro@walter-rocksun: /mnt/j/Jairo/UOC/05.648 - TFG - Administració de xarxes i sistemes operatius/PAC 2/terraform
$ terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.
```

4.5.2. Tfswitch

Font: Pàgina oficial https://tfswitch.warrensbox.com/

És una eina de línia de comandes que facilita canviar entre diferents versions de Terraform. La instal·lació és simple i mínima. Després de la instal·lació, només cal seleccionar la versió desitjada des del menú desplegable i començar a utilitzar Terraform. https://tfswitch.warrensbox.com/Install/

```
curl -L https://raw.githubusercontent.com/warrensbox/terraform-switcher/release/install.sh | bash
```

Comprovem que la instal·lació s'ha produït correctament:

```
ujvaro@walter-rocksun: /mnt/j/Jairo/UOC/05.648 - TFG - Administració de xarxes i sistemes operatius/PAC 2/terraform
$ tfswitch --version

Version: 0.13.1218
```

4.5.3. Tfsec

Font: Pàgina oficial https://aquasecurity.github.io/tfsec

És una eina d'anàlisi estàtica de seguretat per al codi de Terraform. Ofereix una sortida amigable per als desenvolupadors i comprovacions documentades, facilitant la detecció i correcció ràpida i eficient dels problemes de seguretat.
https://aquasecurity.github.io/tfsec/v1.28.1/guides/installation/

```
curl -s https://raw.githubusercontent.com/aquasecurity/tfsec/master/scripts/install_linux.sh | bash
```

Comprovem que la instal·lació s'ha produït correctament:

```
ujvaro@walter-rocksun: /mnt/j/Jairo/UOC/05.648 - TFG - Administració de xarxes i sistemes operatius/PAC 2/terraform
$ tfsec --version

==================================================
v1.28.4
```

4.5.4. Terraform docs

Font: Pàgina oficial: https://terraform-docs.io/

És una utilitat que genera documentació a partir de mòduls de Terraform en diversos formats de sortida. Per configurar-lo de manera consistent, es pot utilitzar un fitxer .terraform-docs.yml.

Un cop configurat, cada vegada que volguem regenerar la documentació, simplement s'ha d' executar terraform-docs amb la ruta corresponent. Aquesta eina és útil per mantenir la documentació actualitzada i coherent amb el codi dels mòduls de Terraform.

https://terraform-docs.io/user-guide/installation/

```
curl -Lo terraform-docs.tar.gz https://github.com/terraform-docs/terraform-docs/releases/download/v0.16.0/terraform-docs-v0.16.0-linux-amd64.tar.gz
```

Comprovem que la instal·lació s'ha produït correctament:

```
ujvaro@walter-rocksun: /mnt/j/Jairo/UOC/05.648 - TFG - Administració de xarxes i sistemes operatius/PAC 2/terraform
$ terraform-docs --version
terraform-docs version v0.16.0 1f686b1 linux/amd64
```

4.6. Proveïdor de Terraform i selecció de versions

En Terraform, els proveïdors són les interfícies que gestionen la interacció amb les APIs de diferents plataformes, com ara proveïdors de núvol, serveis d'emmagatzematge o xarxes. Cada proveïdor és responsable de traduir les configuracions declaratives de Terraform en crides específiques a l'API del servei que representa.

Quan es treballa amb Terraform, és essencial entendre el concepte de selecció de versions de proveïdors. Aquesta selecció es refereix a la capacitat de Terraform de gestionar quina versió d'un proveïdor s'utilitza en un determinat projecte. Això és important per assegurar la coherència i la compatibilitat entre la configuració del projecte i les versions dels proveïdors amb els quals interactua.

Per especificar la versió d'un proveïdor en Terraform, es pot utilitzar la clàusula ***required_providers*** als fitxers de configuració.

Aquesta especificació permet indicar la versió exacta o un rang de versions amb el qual el projecte és compatible. Gestionar les versions dels proveïdors és crucial per garantir la fiabilitat i la seguretat dels desplegaments, ja que assegura que les noves característiques i correccions de seguretat es puguin aplicar quan sigui necessari, mentre es minimitzen les possibles interrupcions del servei.

https://developer.hashicorp.com/terraform/language/providers

Per exemple:

```
terraform {

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 4"
    }
  }
}
```

4.7. Versió de Terraform i selecció de versions

En Terraform, la gestió de versions és un aspecte crític per garantir la consistència i la coherència dels projectes. Terraform permet especificar la versió del mateix Terraform que s'utilitzarà en un projecte particular.

Per especificar la versió de Terraform per en un projecte, es pot utilitzar la clàusula terraform amb la propietat ***required_version*** al fitxer de configuració. Aquesta declaració estableix la versió mínima necessària de Terraform per a un projecte determinat.

Gestionar les versions de Terraform és fonamental per assegurar que les característiques i la funcionalitat específiques de les versions es mantinguin, i per garantir la compatibilitat amb les configuracions del projecte.

Mantenir-se actualitzat amb les versions més recents de Terraform també és essencial per aprofitar noves funcionalitats, millores de rendiment i correccions de seguretat. No obstant això, cal ser prudent en actualitzar les versions en entorns de producció, ja que canvis significatius poden afectar el comportament dels teus desplegaments.

https://developer.hashicorp.com/terraform/tutorials/configuration-language/versions

Seguint l'exemple de la secció anterior podem declarar la versió de terraform juntament amb el proveidor, generalment això es fa un fitxer anomenat versions.tf, per exemple:

```
terraform {

  required_version = "~> 1.5.0"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = ">= 4"
    }
  }
}
```

4.8. Desenvolupament de mòduls de Terraform

En aquesta secció, explorarem de forma practica el desenvolupament de mòduls de Terraform. Amb els mòduls, podem encapsular i reutilitzar fragments de codi d'infraestructura, fent que la nostra configuració sigui més modular, llegible i mantenible.

Com a preparació per a les seccions futures, anomenades "***5. Control de versions, Cicle de vida i Desenvolupament de Codi.***" i "***7. Elaboració de Laboratoris amb Exemples Reals***", s'han creat mòduls específics de Terraform. Aquests mòduls serveixen com a base per als laboratoris, demostrant les millors pràctiques de desenvolupament de codi i proporcionant exemples pràctics de la utilització d'eines complementàries a Terraform.

Explorarem els principis fonamentals del desenvolupament de mòduls, incloent-hi la seva estructura, la definició de variables i l'ús d'output. A més, veurem com els mòduls poden ser incorporats en configuracions més àmplies per proporcionar una arquitectura d'infraestructura reutilitzable i escalable.

4.8.1. Estructura Bàsica d'un Mòdul de Terraform

Un mòdul de Terraform segueix una estructura bàsica que facilita la seva comprensió i ús. A continuació, es descriu breument la disposició típica d'un mòdul:

- **main.tf:** Aquest és el fitxer principal que conté la configuració principal del mòdul. Aquí es defineixen els recursos i paràmetres que formaran part de la infraestructura.

- **variables.tf:** En aquest fitxer, es declaren les variables que es faran servir al mòdul. Això permet la flexibilitat en la configuració, ja que les variables poden ser personalitzades en cada ús del mòdul.

- **outputs.tf:** Defineix les sortides del mòdul, és a dir, els valors que es volen exposar perquè altres configuracions o mòduls els puguin utilitzar.

- **README.md:** Pot contenir documentació sobre l'ús i la configuració del mòdul. Això és especialment útil quan altres persones han de comprendre i utilitzar el mòdul.

```
Nom mòdul
|____main.tf
|____outputs.tf
|____README.md
|____variables.tf
```

Per demostrar els conceptes que s'expliquen en aquest TFG, ampliarem posteriorment aquesta estructura.

Això inclourà la implementació d'eines addicionals com “tfsec” per validar la seguretat i “terraform docs” per generar documentació en el fitxer README.md. Aquestes eines específiques han estat detallades anteriorment a l’apartat:

- 4.5. Instal·lació de Terraform i eines complementaries.

4.8.2. Creació de repositoris i llista inicial de mòduls per als laboratoris

Per a aquesta secció, he establert una llista de mòduls essencials necessaris per a la execució exitosa de la secció de laboratoris.

L'objectiu principal és proporcionar una mostra del contingut dels mòduls i destacar la seva estructura, preparant el material per a les seccions futures:

- 5. Control de versions, Cicle de vida i Desenvolupament de Codi.
- 7. Elaboració de Laboratoris amb Exemples Reals.

Aquesta llista inicial de mòduls s'ha desenvolupat amb l'enfocament en els serveis essencials d’AWS, tal com es destaca a l’apartat 3.7, assegurant que cobrim els serveis que seran fonamentals en les pròximes etapes del treball.

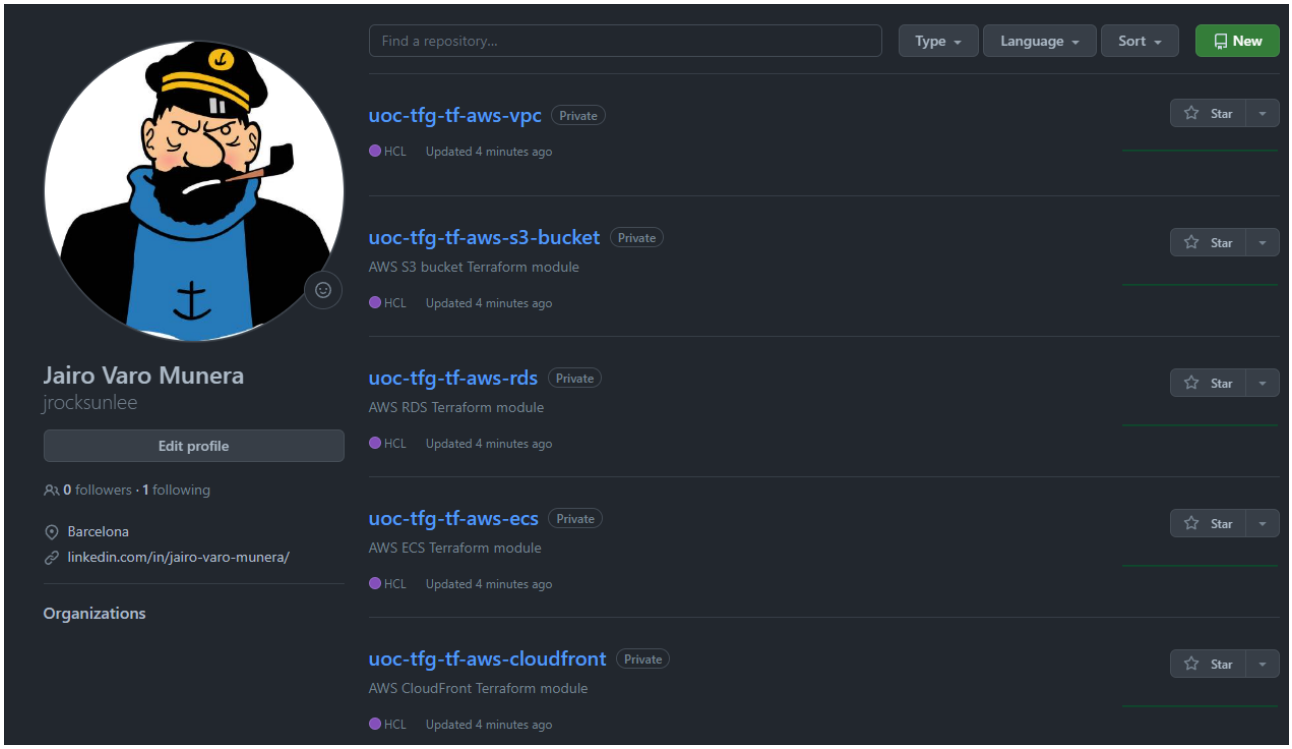
Aquesta visió detallada del contingut dels mòduls es troba al directori annex: Terraform.

(*) Els mòduls que es mostren en aquest TFG s’han extret dels repositoris oberts que hi han disponibles per a la comunitat i adaptats per a mostrar i desenvolupar els objectius i conceptes d’aquest treball.

- https://github.com/terraform-aws-modules/

Dit això, aquesta es la llista inicial dels mòduls de Terraform. Aquests seràn emmagatzemats en repositoris privats de GitHub del meu compte personal.

- https://github.com/jrocksunlee/uoc-tfg-tf-aws-vpc
- https://github.com/jrocksunlee/uoc-tfg-tf-aws-s3-bucket
- https://github.com/jrocksunlee/uoc-tfg-tf-aws-rds
- https://github.com/jrocksunlee/uoc-tfg-tf-aws-ecs
- https://github.com/jrocksunlee/uoc-tfg-tf-aws-cloudfront



5. Control de versions, Cicle de vida i Desenvolupament de Codi

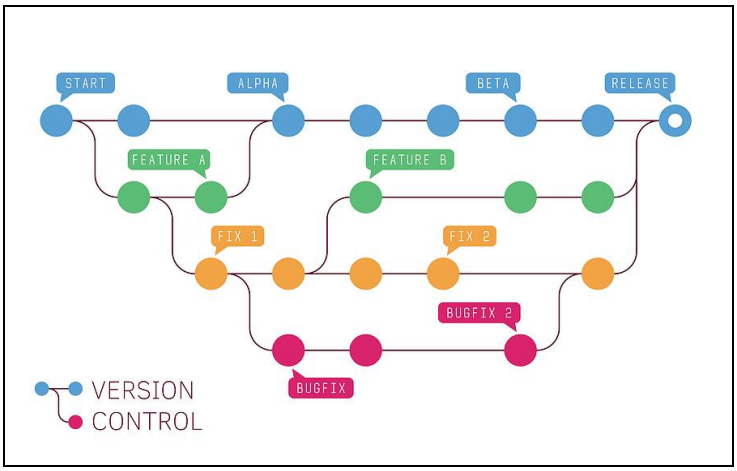
En aquesta secció, explorarem el concepte fonamental del control de versions, el qual implica la gestió dels diferents canvis realitzats en els elements d'un producte o configuració determinada. Cada versió, revisió o edició d'un producte representa l'estat en què es troba en un moment concret del seu desenvolupament o modificació.

Tot i que es pot realitzar el control de versions manualment, és altament recomanable utilitzar eines que facilitin aquesta gestió, donant lloc als anomenats sistemes de control de versions o VCS. Aquestes eines simplifiquen l'administració de les diferents versions de cada producte desenvolupat, així com les possibles especialitzacions realitzades.

Algunes de les eines representatives d'aquest tipus inclouen CVS, Subversion, SourceSafe, ClearCase, Darcs, Bazaar, Plastic SCM, Git, SCCS, Mercurial, Perforce, Fossil SCM, Team Foundation Server, entre altres.

El control de versions es du a terme principalment a la indústria informàtica per gestionar les diverses versions del codi font, donant lloc als sistemes de control de codi font o SCM.

Figura 8. Explorant el rol essencial del control de versions, el cicle de vida del desenvolupament de codi i les eines destacades en aquest procés. Aprofundirem en els conceptes fonamentals del control de versions i les diverses etapes del cicle de vida del desenvolupament, amb un enfocament en les pràctiques i eines clau.



Font Cadence.com: *What is a Version Control System?*

5.1. Introducció al control de versions Git

Una de les eines més influents i àmpliament utilitzades en el control de versions és Git. Els fonaments d'aquest sistema han revolucionat la forma com els desenvolupadors gestionen i col·laboren en projectes de codi font. L'eina ofereix una estructura distribuïda, eficiència en la gestió de branques, i un sistema robust de fusió que el converteix en una elecció destacada en el desenvolupament de software.

Git és un sistema de control de versions distribuït que permet als equips de desenvolupament tenir múltiples còpies locals del codi font del projecte, independents les unes de les altres.

Aquestes branches, es poden crear, fusionar i eliminar ràpidament, permetent als usuaris experimentar amb poc cost computacional abans de la fusió amb la branch principal. Conegut per la seva velocitat, compatibilitat amb fluxos de treball i la seva base de codi obert, Git és essencial per a un desenvolupament eficient i col·laboratiu.

Aquest sistema de control de versions no només és crucial per al desenvolupament de programari, sinó que també és aplicable a altres àmbits com la gestió de documents, imatges i llocs web. El seu ús proporciona una comunicació sense friccions, facilita la gestió de canvis i assegura la reproductibilitat en els projectes de software i altres àmbits.

5.2. Eines més conegudes basades en Git

Actualment existeixen diverses eines que s'han desenvolupat amb la base de Git per millorar i simplificar diferents aspectes del desenvolupament de software. Aquestes eines amplien les funcionalitats inicials i proporcionen solucions específiques per a temes com la gestió de tasques, revisió de codi i integració contínua.

A continuació, es destaquen algunes d'aquestes eines mes populars, sense entrar en profunditat en les seves característiques:

- **GitHub:** Es una plataforma col·laborativa amb funcionalitats avançades com gestió de projectes, control d'accés i integració amb altres eines. Disposa de llicència de caràcter gratuït i està disponible per a repositoris públics, amb opcions de pagament per a repositoris privats. És tracta de la plataforma més gran i àmpliament utilitzada amb una gran comunitat i suport.

Font: Pàgina oficial: https://github.com/

- **GitLab:** Similar a GitHub, es tracta d'una altra opció influent, proporciona control de versions amb característiques addicionals com integració CI/CD, registre de contenidors i opcions d'autoallotjament. Ofereix llicència gratuïta per a repositoris públics i privats, amb opcions autoallotjades i gestionades.

Font: Pàgina oficial https://gitlab.com/

- **Bitbucket:** La eina forma part de la família Atlassian, ofereix serveis d'hostatge de codi amb integració amb altres eines com Jira. Amb una llicència gratuïta per a petits equips i opcions de pagament per a equips més grans, destaca per integrar-se amb altres eines Atlassian, oferint una gestió completa del desenvolupament.

Font: Pàgina oficial https://bitbucket.org/

- **GitKraken:** És un client Git amb una interfície gràfica intuïtiva i funcions avançades per a la gestió visual de branques. Amb una versió gratuïta i opcions premium, destaca per la seva interfície amigable i les funcions de visualització de branques.

Font: Pàgina oficial https://www.gitkraken.com/

Aquestes eines aporten funcionalitats addicionals per millorar l'eficiència del desenvolupament, fomentar la col·laboració entre membres de l'equip i integrar-se amb altres serveis essencials per a la construcció i desplegament de software.

La selecció d'una eina concreta dependrà de les necessitats específiques del projecte, les preferències de l'equip de desenvolupament, així com del cost de les llicències i el tipus de companyia

5.3. Beneficis del control de versions en projectes de desenvolupament de codi

El control de versions és una pràctica essencial en el desenvolupament de codi, oferint una multitud de beneficis que transformen la gestió del projecte i milloren la qualitat del software. A continuació, explorarem en detall com aquesta pràctica contribueix al progrés i la robustesa dels projectes de desenvolupament.

En el marc actual cada vegada més complex del desenvolupament de software, on múltiples desenvolupadors poden treballar simultàniament en diferents aspectes del projecte, el control de versions emergeix com una eina clau. Aquesta pràctica proporciona una estructura ordenada i eficient per gestionar canvis, col·laborar de manera harmoniosa i garantir la consistència del codi en tot moment.

Els beneficis del control de versions s'estenen més enllà de la simple gestió d'arxius i revisions. A través d'aquesta pràctica, els equips de desenvolupament poden experimentar amb noves funcionalitats, revertir canvis no desitjats i mantenir un historial exhaustiu de l'evolució del projecte.

Alguns dels avantatges més rellevants són:

- **Historial detallat:** El control de versions manté un historial meticulós de cada canvi realitzat al codi. Aquesta funcionalitat permet als desenvolupadors revisar, entendre i documentar l'evolució del projecte al llarg del temps.

- **Col·laboració eficient:** Amb el control de versions, diversos membres de l'equip poden treballar simultàniament en diferents aspectes del projecte sense preocupar-se per conflictes o confusions. Les branches faciliten la col·laboració ordenada i la integració harmoniosa de les contribucions individuals.

- **Gestió de branques:** Les eines de control de versions permeten la creació de branches independents, proporcionant una manera estructurada de desenvolupar noves funcionalitats o realitzar canvis sense interferir amb la branch principal.

- **Reversió simplificada:** En cas d'errors o canvis no desitjats, es pot revertir fàcilment a versions anteriors del codi. Aquesta capacitat de reversió ofereix una solució àgil per corregir problemes sense comprometre la continuïtat del projecte.

- **Seguretat i control d'accés:** El control de versions millora la seguretat del codi al permetre l'accés controlat als diferents components del projecte. Els permisos són gestionats de manera granular, garantint que només els membres autoritzats puguin realitzar canvis en àrees específiques del codi.

- **Integració contínua:** La pràctica del control de versions encaixa perfectament amb processos d'integració contínua. Això facilita la detecció ràpida d'errors i la distribució eficient de noves versions, contribuint a un desenvolupament més ràpid i fiable.

5.4. Millors pràctiques en la gestió de canvis

Un altre punt clau, és considerar la gestió eficient dels canvis, ja que és fonamental per mantenir la integritat i la funcionalitat d'un projecte de desenvolupament de software.

Adoptar pràctiques exemplars en aquest àmbit contribueix a un flux de treball més ordenat i col·laboratiu. A continuació es citen, abans d'entrar en més detalls, els punts més importants a tenir en compte per a una gaudir d'una correcta experiència en la gestió de canvis:

- **Planificació acurada:** Abans d'iniciar qualsevol canvi substancial, es vital realitzar una planificació detallada que inclogui l'avaluació dels impactes potencials, les dependències i els recursos necessaris. Això ajuda a evitar sorpreses i retards innecessaris.

- **Ús de branques:** És important emprar un sistema de branches per a organitzar el desenvolupament. Les branches faciliten la implementació paral·lela de funcionalitats i correccions sense interferir en altres aspectes del projecte. Les branches poden incloure desenvolupament de funcionalitats, correccions de bugs i versions estables.

- **Commits significatius:** Es recomana realitza commits atòmics amb missatges significatius. Això millora la llegibilitat del codi i facilita la identificació dels canvis realitzats en cada commit. L'ús de Commits Convencionals o altres estàndards pot afavorir una comunicació més clara sobre les modificacions.

- **Revisió del codi:** Un altre punt molt important és la implementació de processos de revisió de codis per a assegurar la qualitat i la coherència dels canvis. Les revisions per part dels companys de desenvolupament ajuden a identificar possibles errors i assegurar-se que el codi compleix amb les directrius del projecte. Uns dels sistemes mes comuns es establir default reviewers en les accions de Pull Request.

- **Integració contínua:** Implementar la integració contínua permet detectar errors i conflictes de manera ràpida. Amb aquesta pràctica, cada canvi és integrat immediatament al repositori principal, assegurant que el codi base roman estable i lliure d'errors. Es posa l'èmfasi en la identificació eficient de possibles problemes, permetent una acció ràpida per part de l'equip de desenvolupament.

- **Automatització de tests:** És important desenvolupar tests automatitzats per a validar les funcionalitats existents i garantir que les noves implementacions no interfereixin amb el comportament existent del sistema. Això ajuda a mantenir una alta qualitat del codi i a evitar regressions.

- **Gestió de tags i release notes**: És molt útil utilitzar un sistema d'etiquetes per a identificar clarament els punts de versió importants i acompanyar cada versió amb notes de versió detallades que informin als desenvolupadors sobre els canvis, millores i correccions implementades.

- **Comunicació transparent:** Mantenir una comunicació transparent i regular amb l'equip de desenvolupament és clau. Això inclou reunions periòdiques, informes d'estat i altres formes de col·laboració per a assegurar-se que tots estan al corrent del progrés i dels canvis planificats.

5.5. Conventional Commits: Estàndard per a descripcions de commits

Conventional Commits: https://www.conventionalcommits.org/ és un estàndard per a la redacció de descripcions de commits que proporciona una estructura clara i consistent per comunicar canvis en el codi font.

A diferència d'altres formes de redactar commits, aquest estàndard busca estandarditzar les etiquetes i el format dels missatges per tal de millorar la claredat i l'automatització del procés de gestió de versions.

En el marc de Conventional Commits, cada commit segueix un format específic que inclou una etiqueta seguida d'una descripció opcionalment detallada. Les etiquetes segueixen un conjunt predeterminat que indica la naturalesa del canvi, com "fix" per a correccions de bugs, "feat" per a noves funcionalitats, i altres.

Els commits contenen els següents elements estructurals, entre altres que es poden trobar al portal web.

- **fix:** Commits d'aquest tipus s'utilitzen per corregir errors en el codi font, amb una correlació amb la versió PATCH en Semantic Versioning.

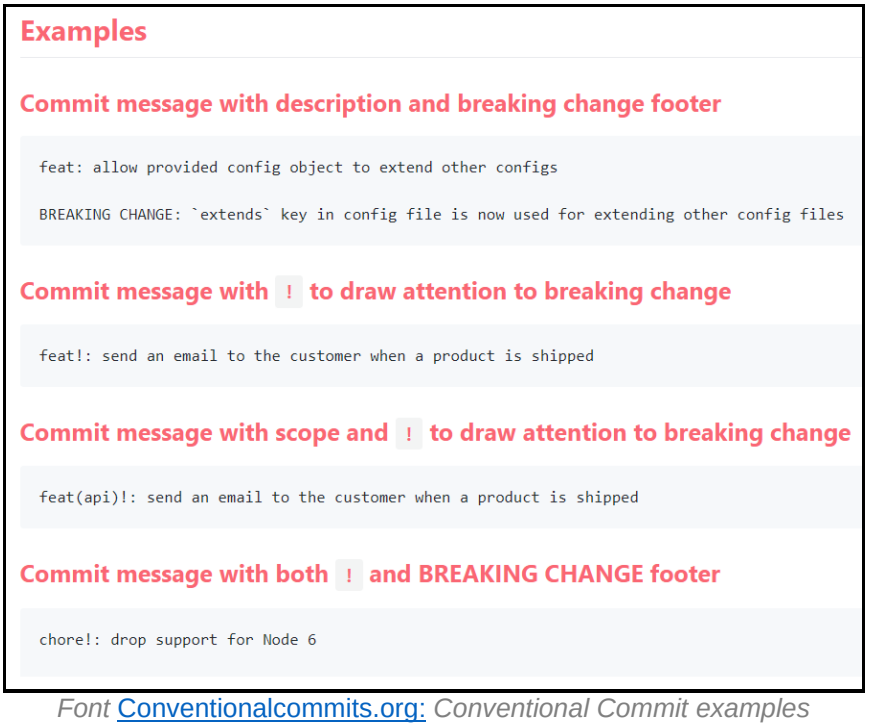
- **feat:** Commits etiquetats com a feat introdueixen noves funcionalitats al codi font, corresponent a la versió MINOR en Semantic Versioning.

- **BREAKING CHANGE:** Commits amb un peu que conté "BREAKING CHANGE:" o que afegeix un "!" després del tipus/àmbit, introdueixen un canvi d'API que trenca la compatibilitat, alineant-se amb la versió MAJOR en Semantic Versioning. Aquest canvi breaking pot ser part de commits de qualsevol tipus.

- **S'admeten tipus diferents de fix i feat:** Per exemple, @commitlint/config-conventional, basat en la convenció Angular, però es recomanen altres tipus com build:, chore:, ci:, docs:, style:, refactor:, perf:, test:, i altres.

- **Es poden proporcionar footers diferents de BREAKING CHANGE:** <descripció>, seguint una convenció.

Figura 9. En aquesta figura, s'analitzen amb exemples els elements estructurals dels commits, proporcionant una guia clara per comprendre les contribucions al codi font.



Font Conventionalcommits.org: Conventional Commit examples

Aquesta pràctica facilita la generació automàtica de notes de versió i permet als desenvolupadors i als col·laboradors comprendre ràpidament els canvis realitzats a través del seguiment de commits.

5.6. Semantic Versioning: Estructura i aplicació

Semantic Versioning: https://semver.org/ és un sistema estandarditzat de numeració de versions que proporciona claredat i consistència en la comunicació dels canvis en el codi font.

En el desenvolupament de software, SemVer emergeix com una eina essencial per evitar el desordre de les dependències. Les regles precises d'aquest sistema guien l'assignació i l'increment dels números de versió, utilitzant el format X.Y.Z (Major.Minor.Patch). Cada part d'aquesta numeració reflecteix canvis específics: l'increment de Patch per correccions, de Minor per funcionalitats noves retrocompatibles, i de Major per canvis d'API incompatibles.

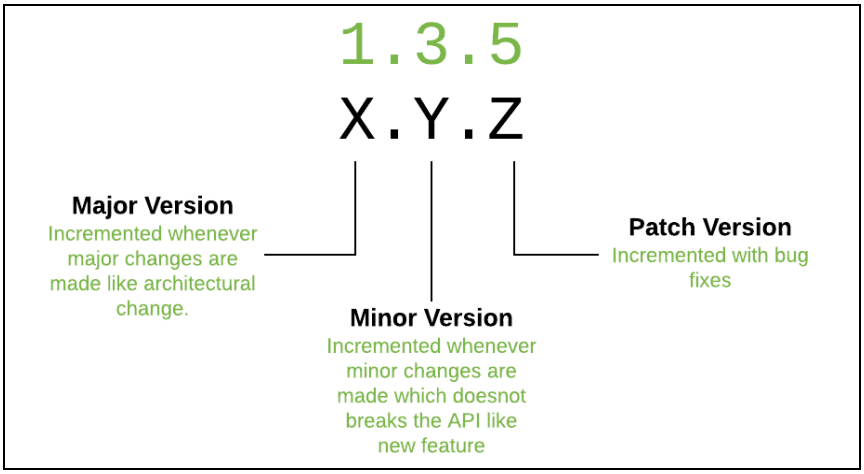
Aquest enfocament permet als desenvolupadors i als usuaris avaluar ràpidament l'impacte dels canvis:

- **Major (X.0.0):** Augmenta quan es realitzen canvis compatibles amb versions anteriors de forma no retrocompatible. Aquesta versió incrementa quan es realitza una funcionalitat important o canvis estructurals que afecten la compatibilitat enrere.

- **Minor (0.X.0):** Augmenta quan es realitzen afegiments de funcionalitats noves de manera retrocompatible. Aquesta versió incrementa amb l'addició de funcionalitats, sense afectar la compatibilitat enrere.

- **Patch (0.0.X):** Augmenta quan es realitzen correccions de bugs retrocompatibles. Aquesta versió incrementa amb correccions de bugs i millores menors que no afecten la compatibilitat enrere.

Figura 10. És detallen els principis del Semantic Versioning, aquesta convenció ajuda a estandarditzar la numeració de versions en el desenvolupament de programari.



Font Devopedia.org: *About Semantic Versioning*

L'aplicació del Semantic Versioning implica decisions significatives sobre la categorització dels canvis, millorant la comunicació i la comprensió compartida de l'estat del software.

Encara que les directrius de SemVer siguin clares, aplicar-les pot resultar complex.

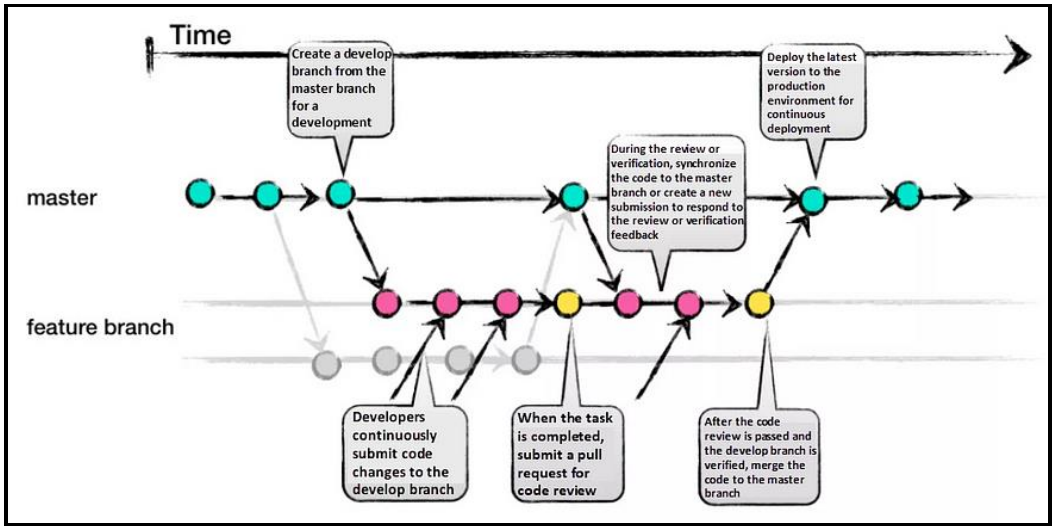
Algunes crítiques inclouen la dificultat en diferenciar canvis, la manca d'automatització per determinar compatibilitat, i la rigidesa en considerar canvis menors com a importants.

SemVer no ofereix protecció completa contra canvis i destaca la necessitat d'actualitzar dependències amb cautela. Malgrat aquestes consideracions, SemVer continua essent una eina útil per comunicar canvis en el desenvolupament de software.

5.7. GitHub Flow: Estratègia de branching

GitHub Flow: https://githubflow.github.io/ és un model de branching lleuger basat en branches. Aquest flux de treball és útil per a tots, no només per als desenvolupadors. Per exemple, a GitHub s'utilitza una versió Git Flow simplificada per a polítiques de lloc web, documentació i planificació.

Figura 11. En aquesta representació visual, s'exemplifiquen els principis del GitHub Flow, una metodologia que destaca la col·laboració, la integració contínua i la implementació regular al llarg del desenvolupament de programari.



Font Medium.com: *Github flow example.*

L'objectiu principal és oferir un procés senzill i flexible per a la col·laboració contínua sense generar massa complexitat.

5.7.1. Fonaments principals i com utilitzar GitHub Flow amb alguns exemples

Branques: Les branchessón utilitzades per aïllar el treball en funcions o correccions específiques. S'utilitza una branch nova per a cada funcionalitat o correcció.

Figura 12. Exemple per crear una branch de treball nova per cada funcionalitat o correcció a implementar.

```
git switch -c nova-funcionalitat
```

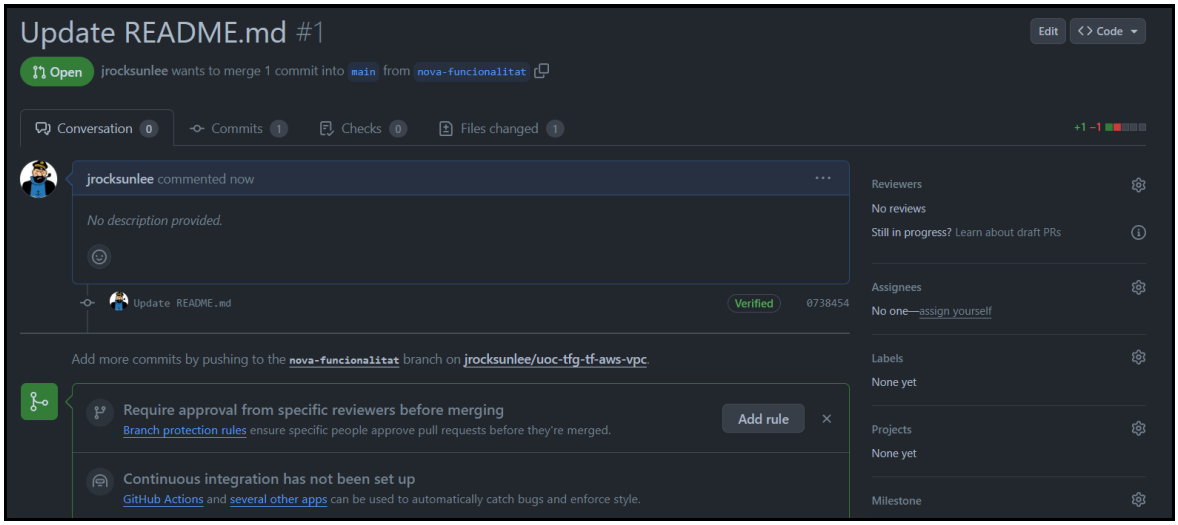
Canvis i commits: Els canvis i commits es fan a la branch de treball i cada commit hauria de representar un canvi lògic i complet. També s'han de proporcionar missatges de commit descriptius per entendre els canvis realitzats.

Figura 13. Exemple per realitzar canvis i commits a la branch de treball.

```
git commit -m "Implementa nova funcionalitat"
```

- **Pull Requests:** Un cop els canvis estan realitzats i commits fets, obre una sol·licitud de tirada. La sol·licitud de tirada serveix per revisar els canvis, discutir-los i sol·licitar aprovació.

Figura 14. Exemple per pujar els canvis i obrir una Pull Request a través de la interfície de GitHub.



- **Revisió i Comentaris:** Els col·laboradors poden revisar la pull request, deixar comentaris i suggeriments. Es clau mantenir una comunicació oberta i clara per millorar els canvis.

- **Merge:** Després d'aprovar la pull request, es fa merge contra la branch principal normalment anomenada "main" o "master". Aquesta acció incorpora els canvis a la branch principal.

Figura 15. Exemple de merge després de rebre l'aprovació.

```
git checkout main
git merge nova-funcionalitat
```

- **Neteja:** Després de la acció de merge, s'ha d' eliminar la branch de treball per mantenir la neteja i evitar confusions.

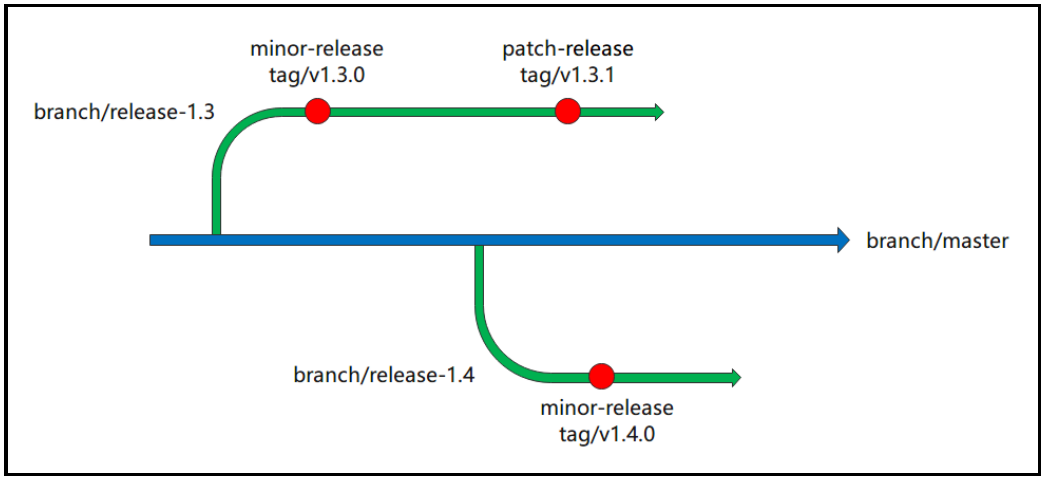
Figura 16. Exemple per esborrar la branch una vegada fet el merge.

```
git checkout main
git merge nova-funcionalitat
```

5.8. Gestió de tags i release notes

La gestió d'etiquetes i release notes és una pràctica essencial per organitzar i comunicar canvis significatius dels projectes. Les etiquetes i les release notes ofereixen una manera estructurada de seguir el progrés del codi i faciliten la comprensió dels usuaris i dels col·laboradors.

Figura 17. A la visualització es destaca la pràctica essencial de la gestió d'etiquetes i release notes. Aquest procés organitza i comunica canvis significatius en els projectes de manera estructurada, facilitant la comprensió tant per als usuaris com per als col·laboradors.



Font Karmada.io: Releasing example.

A continuació, es detallen els passos per gestionar adequadament aquest aspecte en un projecte:

- **Tags:** Les etiquetes són descriptors associats a punts específics en el teu codi, com ara funcionalitats, correccions de bugs o millores. Crea tags significatius que reflecteixin la naturalesa dels canvis, com "Funcionalitat Nova," "Correcció de Bugs," o "Millora de Rendiment."

- **Ús consistent dels tags:** Es clau mantenir una convenció clara per utilitzar etiquetes i assegurar que tots els col·laboradors les entenguin i les utilitzin de manera consistent. Definir les etiquetes en la documentació del projecte per mantenir una referència clara per a tothom.
- **Semàntic commits i tags:** Integrar etiquetes amb el sistema de semàntic commits per automatitzar la generació d'etiquetes i release notes.

- **Release notes:** Les notes de release són resums detallats de les millores, funcionalitats i correccions incloses en cada nova versió del teu software. Incloure informació rellevant com canvis d'API, funcionalitats afegides i correccions de seguretat.

- **Generació automàtica:** Utilitzar eines automàtiques per generar release notes basades en els commits. Això facilita la creació d'informació clara i precisa sense haver de fer-ho manualment.

- **Estratègia de versions:** Mantenir una estratègia clara per a la numeració de versions com per exemple SemVer. Això ajuda els usuaris a comprendre l'impacte dels canvis en el software.

La correcta gestió d'etiquetes i release notes millora la visibilitat del progrés del projecte, facilita la comunicació entre col·laboradors i usuaris, i estableix una base sòlida per a futures versions del software.

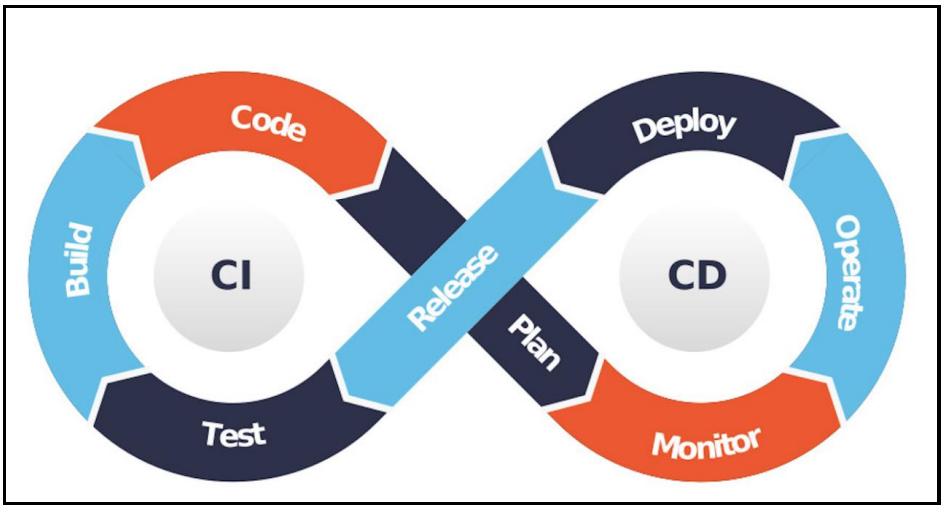
6. Eines de CI/CD

Degut a la transformació del paisatge del desenvolupament de programari, la implementació de pràctiques modernes ha canviat la manera com les organitzacions gestionen els seus cicles de vida de desenvolupament. Conèixer com funcionar la Integració Contínua i Entrega Contínua (CI/CD), es fonamental per millorar l'eficiència i la qualitat dels projectes de desenvolupament.

Aquesta secció comença explorant el concepte essencial de CI/CD, detallant-ne la importància i com aquestes pràctiques poden transformar la forma en què els equips de desenvolupament construeixen, proven i publiquen el seu codi. Posteriorment, analitzarem les diferències crítiques entre la Integració Contínua, la Entrega Contínua i el Desplegament Continu, oferint claredat sobre quan i com s'apliquen aquests processos segons les necessitats específiques d'un projecte.

A més es tractarà de destacar els passos crítics per assegurar la consistència i la fiabilitat del codi al llarg termini. Per últim, analitzarem dues de les eines més rellevants en aquest àmbit, Jenkins i GitHub Actions, explorant-ne les característiques, la utilització efectiva i la implementació com a solucions eficaces dins d'un entorn de desenvolupament.

Figura 18. En aquesta figura es destaquen les fases clau de CI/CD. La comprensió d'aquestes etapes és crucial per a la millora de l'eficiència i la qualitat en el desenvolupament de programari.



Font linkedin.com: What CI/CD mean in the context of DevOps?

6.1. Introducció al concepte de CI/CD

La Integració Contínua (CI) i l'Entrega Contínua (CD) són dues de les pràctiques més importants de DevOps. Aquestes permeten construir programari més ràpidament, reduint el risc i millorant la qualitat.

Aquest dos processos tenen com a objectiu crear un canal de desenvolupament, proves i desplegament. Amb la CI/CD, tots els desenvolupadors poden treballar simultàniament en

el seu codi amb poca sobrecàrrega. Això significa que hi ha menys errors en el producte final, ja que cada programador sempre treballa amb una versió actualitzada de la base de codi del projecte.

6.1.1. Quins són els beneficis de la CI/CD?

L'adopció d'aquesta pràctica permet a l'equip de DevOps assolir una major qualitat en el desenvolupament de programari. Això implica establir procediments de proves automatitzades que s'executen amb cada construcció del codi, contribuint a mantenir un alt nivell de qualitat en el producte.

Un sistema d'integració contínua detectarà problemes d'integració o compilació, donant als equips l'oportunitat de resoldre errors tan aviat com sigui possible, sense haver d'esperar proves manuals al final d'un cicle de projecte. Aquest enfocament millora la velocitat de desenvolupament de programari, ja que permet als desenvolupadors abordar problemes de forma immediata, en comptes d'esperar fins a la finalització del cicle per abordar-los.

El CI/CD contribueix a augmentar la velocitat de desenvolupament, ja que possibilita l'execució de proves de manera contínua en un sistema en viu, en lloc de realitzar-les manualment en entorns separats i esperar fins a la finalització abans de passar a una altra tasca o etapa del procés. Aquesta pràctica estalvia temps, especialment quan hi ha diverses accions simultànies, com és comú en les iteracions àgils, on diverses persones treballen en diferents parts simultàniament.

La integració contínua mitjançant eines d'automatització com Jenkins facilita el manteniment d'una integració fluida, notificant immediatament quan hi ha problemes i permetent que tot l'equip els solucioni sense perdre temps en processos de depuració posteriors. En definitiva, es poden resumir com a principals beneficis, els següents punts:

- Menys temps dedicat a l'espera de proves manuals gràcies a proves automàtiques que s'executen de manera contínua (CI) o després de cada commit (CD).

- Menys conflictes entre desenvolupadors, ja que tots treballen en la seva còpia del repositori al llarg del desenvolupament, sense haver-hi "el meu" versus "el teu".

- Desplegaments més ràpids gràcies a les eines d'automatització que escriuen i executen scripts quan es compleixen certes condicions durant els processos de construcció.

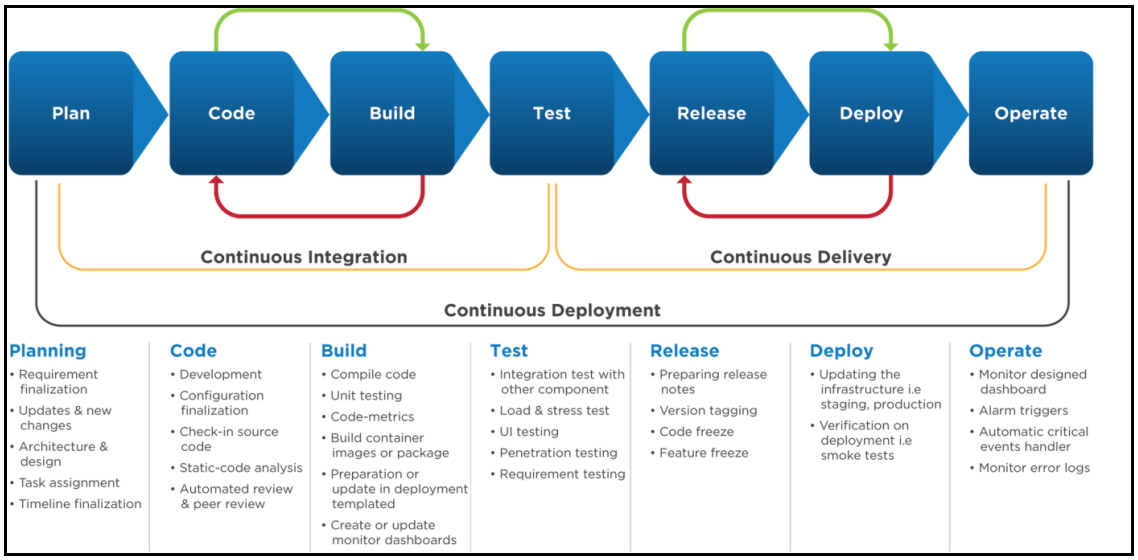
- Augment de la qualitat del programari

6.1.2. Fases i com implementar CI/CD

La implementació del CI/CD constitueix un element clau en la gestió del cicle de vida del desenvolupament de programari. A continuació es mostren els passos fonamentals per a la instauració d'aquestes pràctiques, des de la definició d'un canal de desenvolupament fins a la producció.

El CI/CD proporciona una estructura que permet als equips de desenvolupament construir, provar i desplegar codi de manera eficient, accelerant el procés de desenvolupament i millorant la qualitat del producte final. A continuació, detallarem els elements clau d'aquesta implementació per facilitar la comprensió i l'adopció d'aquesta metodologia.

Figura 19. En aquesta figura, es proporciona orientació sobre com implementar CI/CD eficaçment. Les pràctiques i les eines essencials són explorades per garantir una integració i entrega contínua efectives en el desenvolupament de programari.



Font linkedin.com: How to implement CI/CD.

Fases Integració Contínua

- **Planificació**: En aquesta fase, es defineix el pla per al desenvolupament del codi. S'identifiquen els objectius, s'estableixen les tasques i es determina la seqüència d'accions necessària.

- **Codi**: Els desenvolupadors treballen simultàniament en les seves branchesdel codi font. Aquest pas implica la creació, edició i millora del codi de manera col·laborativa.

- **Construcció**: Un cop es realitzen els canvis al codi, s'inicia el procés de construcció per compilar-lo. Aquest pas assegura que el codi es pugui executar i integrar adequadament.

- **Proves**: Les proves automatitzades es duen a terme per verificar la funcionalitat del codi. Això inclou proves unitàries, d'integració i altres proves que asseguren la qualitat del codi.

Fases Desplegament Continu

- **Proves:** Abans de passar a la següent fase, s'executen proves addicionals per garantir que tots els canvis són funcionals i no causen problemes.

- **Alliberament:** Es preparen els artefactes de la versió per a la implementació. Això pot incloure la generació de paquets d'instal·lació o altres elements necessaris per al llançament.

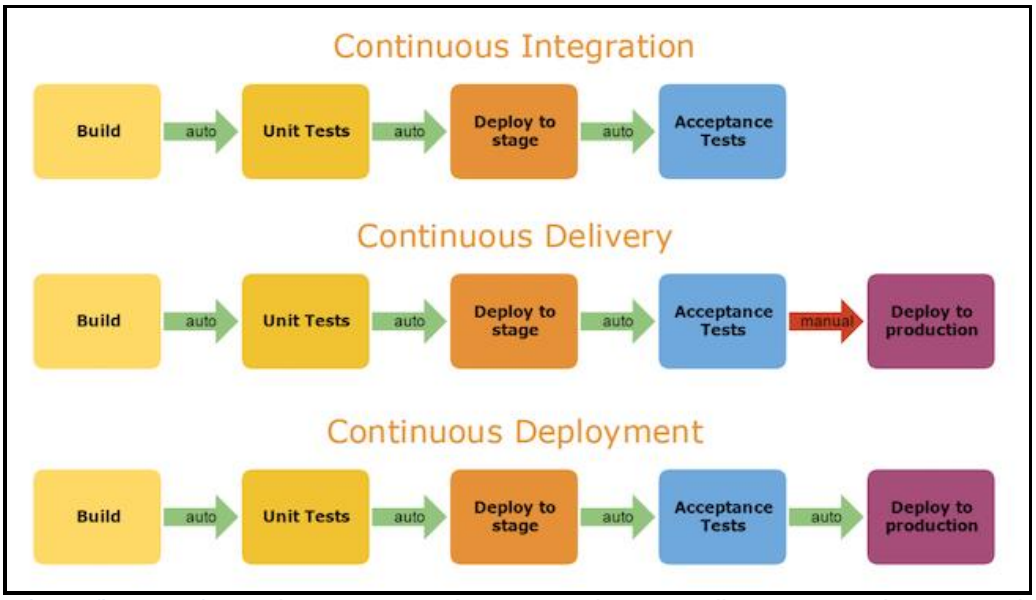
- **Desplegament**: La versió del software es desplega en un entorn real o de producció. Aquest pas permet que els usuaris finals accedeixin a les noves funcionalitats o correccions.

- **Operació:** Un cop desplegat, el software és monitoritzat i gestionat en temps real. Aquesta fase inclou la gestió del rendiment, la detecció d'errors i altres tasques operacionals.

6.2. Diferències entre CI, CD i Continuous Deployment

La integració contínua, la entrega contínua i la implementació contínua són pràctiques essencials per optimitzar el procés de llançament de programari, reduir els cicles de retroalimentació i automatitzar tasques repetitives.

Figura 19. S'estableix una comparativa clara entre Continuous Integration, Continuous Delivery i Continuous Deployment en el context del desenvolupament de programari.



Font stackoverflow.com/: *Continuous Integration vs. Continuous Delivery vs. Continuous Deployment.*

Aquests procediments són crucials per assegurar que els usuaris rebin amb freqüència un programari valuós i funcional. Però sabem diferenciar correctament les seves principals diferències? A continuació es mostra de forma esquemàtica les principals característiques.

6.2.1. Característiques en la Integració Contínua

La Integració Contínua es centra en assegurar una integració suau i sense problemes del codi entre els membres de l'equip. Aquest procés implica l'automatització de la construcció i proves a mesura que es realitzen canvis en el codi.

- **Objectiu Principal:** Evitar conflictes d'integració i assegurar una base de codi actualitzada.
- **Acció Clau:** Automatització de construcció i proves amb cada confirmació de codi.
- **Resultats:** Detecció ràpida d'errors i conflictes d'integració per mantenir la qualitat del codi.
- **Freqüència:** Execució amb cada confirmació de codi.

6.2.2. Característiques en l'Entrega Contínua

L'Entrega Contínua és una extensió lògica de la Integració Contínua, assegurant que el software sigui sempre a punt per ser desplegat en producció. Aquest procés implica l'automatització de proves addicionals, la preparació d'artefactes de versió i la validació de l'entorn de producció.

- **Objectiu Principal:** Assegurar que les versions siguin estables i llestes per al desplegament.
- **Acció Clau:** Executar proves addicionals i preparar artefactes per a la versió.
- **Resultats:** Garantir estabilitat i preparació per al desplegament sense intervenció manual.
- **Freqüència:** Pot ser desencadenada després de cada integració o segons la conveniència del desenvolupador.

6.2.3. Característiques en el Desplegament Continu

El Desplegament Continu representa la culminació dels processos de Integració Contínua i Entrega Contínua, ja que implica la implementació automatitzada d'una nova versió del software en un entorn de producció sense intervenció manual.

- **Objectiu Principal:** Automatitzar completament el desplegament d'una nova versió en un entorn de producció sense cap intervenció manual.
- **Acció Clau:** Desplegar automàticament la versió aprovada sense cap interrupció humana.

- **Resultats:** L'entorn de producció es manté sempre actualitzat amb les últimes versions del software.

- **Freqüència:** Cada vegada que es compleixen els criteris d'èxit de la fase de desplegament.

6.3. Jenkins: Anàlisi, exploració i utilització en CI / CD

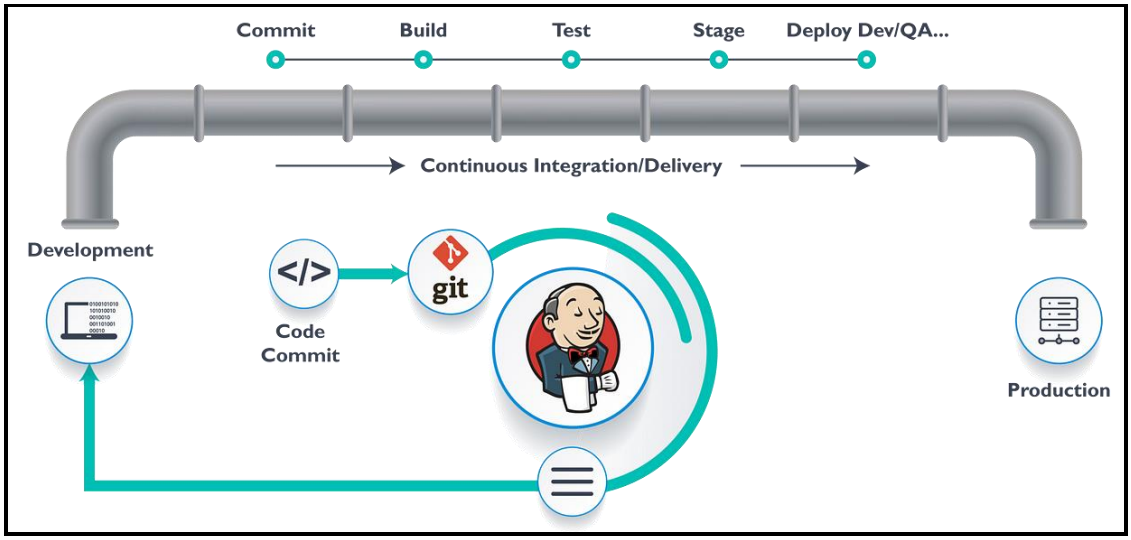
Jenkins, un servidor d'automatització open source basat en Java que es deriva del projecte Hudson (antecessor a Jenkins com el coneixem), és una eina crucial per a la integració contínua i la entrega contínua.

A continuació, explorarem Jenkins, analitzant-ne les característiques, funcionalitats i la seva utilització com a plataforma CI/CD. Jenkins juga un paper clau en la automatització dels processos de desenvolupament de software, ja que facilita la integració contínua amb suport per a diverses eines de control de versions com CVS, Subversion, Git, Mercurial, Perforce i Clearcase.

A més, és capaç d'executar una àmplia gamma de projectes, incloent aquells basats en Apache Ant i Apache Maven, així com scripts de consola i prog branches per lots de Windows. Sota la llicència MIT, Jenkins és una eina de codi obert desenvolupada principalment per Kohsuke Kawaguchi.

Aquesta es la seva pagina oficial on es pot trobar tota la documentació referent a la eina: https://www.jenkins.io/doc/

Figura 20. S'il·lustra el procés d'Automatització de CI/CD mitjançant l'ús de l'eina Jenkins. La imatge proporciona una visió gràfica del flux de treball automatitzat, destacant com Jenkins facilita la integració del codi, la construcció, els tests i el desplegament, contribuint a una execució eficient dels processos de desenvolupament.



Font medium.com: Automated CI/CD with Jenkins.

6.3.1. Utilitzant Jenkins per CI/CD

Abans de disposar d'eines com Jenkins, la integració contínua es trobava amb diversos desafiaments, com ara:

- **Desenvolupament i prova seqüencials:** Es desenvolupava tot el codi abans de provar-lo, cosa que feia que els desplegaments i les proves fossin poc freqüents, allargant els temps de lliurament i dificultant la identificació i correcció d'errors.

- **Espera per provar millores:** Els desenvolupadors havien d'esperar a que es completés tot el codi per provar les seves millores, frenant el procés.

- **Processos manuals propensos a errors:** El desenvolupament i les proves es realitzaven manualment, augmentant la probabilitat d'errors.

No obstant això, amb Jenkins i la integració contínua que proporciona, la situació canvia ja que gràcies a les possibilitats que ofereix, es poden posar solució a una sèrie de tasques de desenvolupament com ara:

- **Desenvolupament i verificació per commit:** Cada commit es desenvolupa i verifica individualment, permetent als desenvolupadors centrar-se en corregir errors específics en lloc de revisar tot el codi.

- **Coneixement immediat dels resultats:** Els desenvolupadors coneixen els resultats de les proves dels seus canvis durant l'execució.

- **Automatització de desplegaments i proves:** Jenkins automatitza el desplegament i les proves, estalviant temps i evitant errors.

- **Cicle de desenvolupament més ràpid:** Es lliuren més funcionalitats de manera més freqüent, maximitzant els beneficis.

6.3.2. Què es pot fer amb Jenkins?

Jenkins permet automatitzar diverses tasques per reduir el temps de llançament al mercat de productes digitals o noves versions. Això inclou las possibilitat d'automatitzar les següents accions:

- Automatització de compilació i proves de programari.

- Notificació d'errors als equips corresponents.

- Desplegament de canvis validats al codi.

- Seguiment de la qualitat del codi i la cobertura de proves.

- Generació de documentació de projectes.

A més, les funcionalitats de Jenkins es poden ampliar mitjançant plugins comunitaris dissenyats per a diverses tasques a les etapes del desenvolupament.

6.3.3. Beneficis i desavantatges de Jenkins com a eina

Avantatges de Jenkins

Jenkins destaca com a eina d'integració contínua per diverses raons que la converteixen en una opció atractiva per a molts equips de desenvolupament:

- Fàcil instal·lació: Jenkins és senzill de instal·lar, facilitant la seva posada en marxa en un entorn de desenvolupament.
- Codi obert i comunitat activa: Al ser una eina de codi obert, Jenkins compta amb una àmplia comunitat de desenvolupadors que donen suport, resolen problemes i milloren contínuament la eina.
- Gratuïta: Jenkins és gratuïta, el que la fa accessible per a petits i grans equips de desenvolupament sense la necessitat de despeses significatives.
- Versàtil i modular: Amb centenars de plugins disponibles, Jenkins és una eina altament versàtil que pot ser adaptada a les necessitats específiques de cada projecte.
- Compatibilitat amb Java i diverses plataformes: El desenvolupament en Java fa que Jenkins sigui compatible amb les principals plataformes, oferint una flexibilitat important als equips de desenvolupament.

Desavantatges de l'ús de Jenkins

Malgrat les seves virtuts, Jenkins no està exempta de certs desavantatges que cal tenir en compte abans de decidir la seva implementació:

- Interfície d'usuari antiquada: L'aparença de l'interfície d'usuari de Jenkins és considerada antiquada i poc intuïtiva. Tot i això, aquesta limitació pot ser mitigada mitjançant l'ús de plugins com Blue Ocean, que millora significativament la interfície.
- Complexitat de les pipelines: La creació de pipelines a Jenkins pot ser una tasca complexa que requereix temps i dedicació. Això pot representar un repte per als equips novells o amb menys experiència en l'ús de la eina.
- Plugins desfasats: Algunes de les extensions de tercers (plugins) poden no estar actualitzades, la qual cosa pot generar incompatibilitats amb versions més recents de Jenkins o altres eines.
- Requeriments de servidor i configuració: L'ús de Jenkins implica la necessitat d'un servidor d'allotjament, que pot requerir configuracions detallades i coneixements tècnics específics, cosa que pot ser un obstacle per als usuaris menys tècnics.

- Documentació limitada en alguns àmbits: Algunes àrees de Jenkins podrien requerir una millora en la documentació per ajudar als usuaris a navegar millor en la configuració i utilització de la eina.

6.4. GitHub Actions: Anàlisi, exploració i utilització en CI / CD

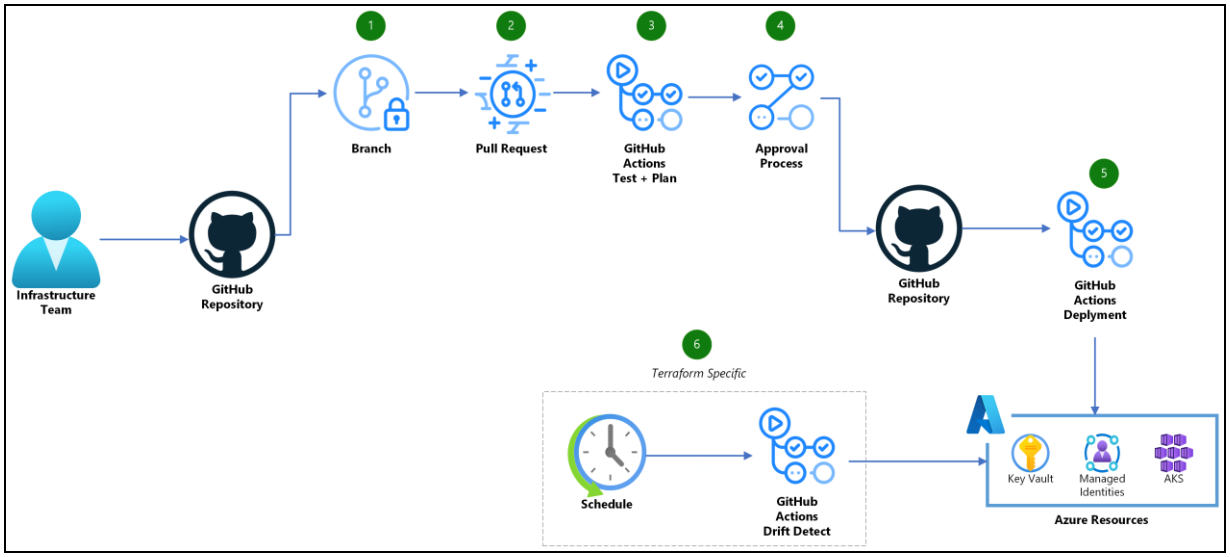
GitHub Actions és una eina innovadora i potent que proporciona funcionalitats integrades per a la integració contínua i entrega contínua. GitHub Actions juga un paper clau en l'automatització dels processos de desenvolupament de software i destaca per la seva estreta integració amb el repositori de codi GitHub.

L'eina facilita la integració contínua amb suport per a diverses eines de control de versions, com Git, i proporciona una plataforma flexible per a la creació de pipelines d'automatització.

GitHub Actions és part integrant de l'economia GitHub, oferint als desenvolupadors una solució completa dins de l'entorn en el qual ja gestionen els seus projectes. Amb una interfície d'usuari intuïtiva i una profunda integració amb les funcionalitats de GitHub, GitHub Actions simplifica la configuració i gestió dels fluxos de treball CI/CD.

Aquesta es la seva pagina oficial on es pot trobar tota la documentació referent a la eina: https://docs.github.com/es/actions

Figura 21. GitHub Actions és una eina integrada per a la integració contínua i entrega contínua. A la imatge es destaquen casos pràctics com Azure-Samples i l'ús de Terraform. La seva estreta integració amb els repositoris de codi de GitHub simplifica la configuració i gestió dels fluxos de treball CI/CD.



Font github.com: GitHub - Azure-Samples and terraform github actions.

6.4.1. Utilitzant GitHub Actions per CI/CD

Abans de l'aparició de GitHub Actions, les pràctiques de CI/CD es basaven sovint en eines separades i processos complicats. GitHub Actions ha canviat aquesta dinàmica, oferint una solució integrada i altament personalitzable per a la integració contínua i la lliurament contínua dins de l'entorn familiar de GitHub. Aquestes són les raons fonamentals per les quals utilitzar GitHub Actions per a CI/CD:

- **Integració nativa amb GitHub:** GitHub Actions es troba totalment integrat dins de l'entorn de GitHub. Això significa que els equips de desenvolupament poden configurar i gestionar fluxos de treball d'integració contínua i lliurament contínua directament dins del mateix entorn on gestionen els seus codis i projectes.

- **Configuració senzilla amb arxius YAML:** La configuració dels fluxos de treball en GitHub Actions es realitza mitjançant arxius de configuració YAML. Aquest format simple i llegible facilita la definició de pasos, triggers i altres configuracions, oferint una manera eficient de gestionar els processos de CI/CD.

- **Ampli suport per a diverses llenguatges i entorns:** GitHub Actions ofereix suport per a una àmplia gamma de llenguatges de programació i entorns d'execució. Això permet als desenvolupadors implementar fluxos de treball CI/CD per a diferents tipus de projectes, independentment del llenguatge utilitzat.

- **Amplia col·lecció d'Actions predefinides:** GitHub Actions proporciona una col·lecció de "Actions" predefinides, que són passos modulars d'automatització que poden ser utilitzats en els fluxos de treball. Aquesta biblioteca extensa permet als desenvolupadors incorporar fàcilment funcionalitats comunes als seus processos de CI/CD.

- **Desplegament directe a entorns de GitHub:** Amb GitHub Actions, és possible desplegar directament a entorns com GitHub Packages, GitHub Pages i altres integracions de GitHub, simplificant encara més el desplegament continu i la lliurament als usuaris finals.

- **Execució paral·lela de tasques:** GitHub Actions permet l'execució paral·lela de diverses tasques, accelerant així els temps d'execució dels fluxos de treball i proporcionant resultats més ràpids als desenvolupadors.

- **Gestió de secrets i seguretat integrada:** GitHub Actions ofereix una gestió segura de secrets, permetent als desenvolupadors emmagatzemar dades sensibles de manera segura i utilitzar-les en els fluxos de treball sense comprometre la seguretat.

- **Flexibilitat i personalització:** GitHub Actions és altament personalitzable i adaptable a les necessitats específiques de cada projecte. La flexibilitat que ofereix permet als equips de desenvolupament crear fluxos de treball que s'ajustin al seu entorn i processos de desenvolupament.

6.4.2. Què es pot fer amb GitHub Actions?

GitHub Actions ofereix una àmplia gamma de possibilitats per a la integració contínua i la lliurament contínua (CI/CD). Amb aquesta eina, els desenvolupadors poden automatitzar:

- **Compilació i proves:** Automatitzar la construcció i proves del codi font.
- **Desplegament de canvis:** Configurar i executar desplegaments automàtics a diferents entorns.
- **Notificacions i comunicació:** Enviar alertes i informació sobre l'estat dels fluxos de treball.
- **Gestió de dependències:** Automatitzar la gestió de dependències del projecte.
- **Creació de paquets i publicació:** Generar paquets i publicar-los en registres o repositoris.
- **Validació de la qualitat del codi:** Executar anàlisis de qualitat i cobertura del codi.
- **Desplegament directe a entorns de GitHub:** Implementar directament a GitHub Pages, GitHub Packages i altres integracions.
- **Personalització amb actions prèviament construïdes:** Aprofitar una àmplia col·lecció d'Actions" prèviament construïdes per a diverses tasques.

6.4.3. Beneficis i desavantatges de GitHub Actions com a eina

Avantatges de GitHub Actions

- **Integració amb GitHub:** Està totalment integrada amb la plataforma GitHub, facilitant la configuració i la gestió dels fluxos de treball directament des del repositori.
- **Configuració amb codi:** Utilitza arxius de configuració en format YAML directament al repositori, el que fa que els processos de CI/CD siguin transparents i controlats pel codi.
- **Catàleg d'accions predefinides:** Ofereix una àmplia gamma d'accions predefinides i fàcilment personalitzables per a les diferents necessitats dels desenvolupadors.
- **Desplegament flexible:** Suporta la integració amb múltiples entorns i plataformes, permetent desplegar aplicacions en diversos serveis i amb diferents configuracions.
- **Escala de forma eficient: Pot** gestionar grans quantitats de treballs simultanis i ofereix la possibilitat de paral·lelitzar diverses tasques, optimitzant el temps d'execució.

Desavantatges de GitHub Actions

- **Costos potencials:** Encara que moltes funcionalitats són gratuïtes, certes característiques avançades o usos intensius poden implicar costos addicionals.
- **Manca d'algunes funcionalitats avançades:** Comparada amb altres eines establertes, GitHub Actions pot mancar de certes funcionalitats avançades i mòduls específics.
- **Aprenentatge inicial:** La configuració inicial pot resultar complexa per a desenvolupadors nous, especialment per aquells menys familiaritzats amb el format YAML.
- **Limitacions en l'ús de màquines pròpies:** L'ús de màquines pròpies en els agents de GitHub Actions pot tenir limitacions, ja que aquestes màquines estan gestionades pel mateix servei.
- **Dependència de la connectivitat a Internet:** Algunes accions poden requerir connexió a Internet, la qual cosa pot ser una limitació en entorns restringits o amb restriccions de xarxa.

6.5. Conclusió comparativa Jenkins Vs. GitHub Actions

La elecció entre Jenkins i GitHub Actions com a eina de CI/CD depèn de diverses consideracions.

Taula 2. Descriu una comparativa entre Jenkins i GitHub Actions.

Category	Jenkins	Github Actions
Environment	Linux/macOS/Windows	Linux/macOS/Windows
Configuration as Code/GitOps	Yes	Yes
Parallelism	Yes	Yes
Branching Support	Yes	Yes
Authentication	Github	Github
Asynchronous Integration	No	Yes
Serverless	No	Yes
Status Visibility	Yes	Yes
Ad Hoc Workflow	Yes	No
Cost	Based on Cloud Provider	Based on Github + Cloud Provider

Font k21academy.com: DevOps Foundation: github-actions vs jenkins

Jenkins és una opció consolidada amb una gran comunitat i una notable versatilitat, malgrat la seva interfície antiga. Amb dues maneres de crear pipelines, ja sigui a través de Pipeline Declaratives o Pipeline de Scripts, ofereix una ampla flexibilitat per als equips de desenvolupament. No obstant això, l'implementació de Jenkins sovint implica un model d'autoservei, on els usuaris supervisen els servidors als seus propis centres de dades.

D'altra banda, GitHub Actions presenta una integració més estreta amb GitHub, facilitant la configuració amb codi i demostrant eficiència en la gestió de tasques. Utilitzant YAML per construir fluxos de treball i establir fitxers, GitHub Actions adopta una estratègia híbrida en el núvol, permetent als usuaris allotjar els seus propis corredors de tasques.

A més, actualment GitHub Actions ofereix gratuïtament els seus serveis per a repositoris públics, amb un sistema de pagament segons ús disponible per a repositoris privats.

La meva conclusió personal és que GitHub Actions emergeix com una opció superior a Jenkins, principalment per la seva versatilitat i pel fet que no requereix manteniment. Això fa que la transició cap a GitHub Actions sigui una decisió lògica per a aquells que inicien nous projectes o utilitzen GitHub com a plataforma de CI/CD.

7. Elaboració de Laboratoris amb Exemples Reals

La següent secció té com a objectiu posar en pràctica alguns dels conceptes abordats al llarg del TFG. L'objectiu principal és demostrar la teoria amb exemples reals i pràctics, facilitant una comprensió més profunda dels continguts teòrics. Amb la intenció d'estandarditzar l'explicació dels laboratoris, he creat una fitxa de requeriments que detalla els requisits, els processos i les necessitats específiques per dur a terme cada laboratori.

Els laboratoris estan dissenyats amb una duració i dificultat que s'ajusten al marc de temps previst dins del propi treball. Cada laboratori s'acompanyarà d'una explicació detallada dels passos, així com dels objectius específics que es pretenen assolir.

(*) Com a pas previ abans de començar la realització dels laboratoris 1 i 2 s'ha creat un petit mòdul per tal de configurar el terraform backend. Aquest pas es dona per suposat en totes les proves i s'inclou dins del directori terraform-modules.

Aquest mòdul bàsicament crea el necessari per al s3 remote backend: https://developer.hashicorp.com/terraform/language/settings/backends/s3

Crear el backend de AWS en Terraform: Per configurar un backend d'AWS a Terraform, cal crear un bucket d'S3. Aquest bucket d'S3 emmagatzemarà el fitxer d'estat de Terraform, que registra tots els recursos creats mitjançant Terraform.

- Per això crearem un fitxer anomenat backend.tf amb següent contingut:

```
terraform {
  required_version = "~> 1"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 5"
    }
  }
}

provider "aws" {
  profile = "default"
  region = local.region
}

terraform {
  backend "s3" {
    bucket         = "terraform-state-693946155914-us-east-1"
    key            = "terraform-backend/terraform.tfstate"
    region         = "us-east-1"
  }
}
```

- i un fitxer anomenat main.tf amb següent contingut:

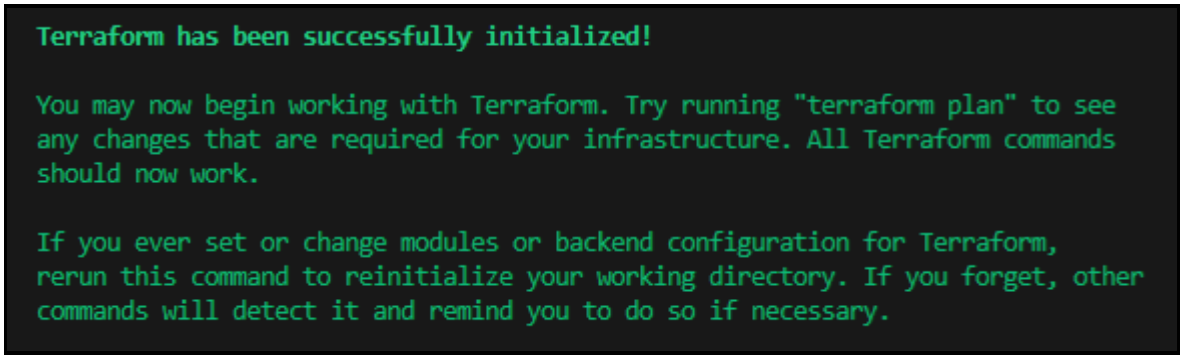
```
################################################################################
# Locals
################################################################################
locals {
  account_id          = "693946155914"
  region              = "us-east-1"
  bucket_name         = "terraform-state-${local.account_id}-${local.region}"
  dynamodb_table      = "terraform-state-${local.account_id}-${local.region}"
  kms_key_alias       = "terraform-states"
}

################################################################################
# Data Policies
################################################################################
data "aws_iam_policy_document" "bucket_policy" {
  version = "2012-10-17"

  statement {
    sid       = "DenyPublishingUnencryptedResources"
    effect    = "Deny"
    actions   = ["s3:PutObject"]
    resources = ["arn:aws:s3:::${local.bucket_name}/*"]

    condition {
      test     = "Null"
      variable = "s3:x-amz-server-side-encryption"
      values   = ["true"]
    }
    principals {
      type        = "*"
      identifiers = ["*"]
    }
  }

  statement {
    sid       = "DenyIncorrectEncryptionHeader"
    effect    = "Deny"
    actions   = ["s3:PutObject"]
    resources = ["arn:aws:s3:::${local.bucket_name}/*"]

    condition {
      test     = "ForAllValues:StringNotEquals"
      variable = "s3:x-amz-server-side-encryption"
      values   = ["aws:kms"]
    }
```

```
    principals {
      type        = "*"
      identifiers = ["*"]
    }
  }

  statement {
    sid     = "AllowSSLRequestsOnly"
    effect  = "Deny"
    actions = ["s3:*"]
    resources = [
      "arn:aws:s3:::${local.bucket_name}",
      "arn:aws:s3:::${local.bucket_name}/*",
    ]

    condition {
      test     = "Bool"
      variable = "aws:SecureTransport"
      values   = ["false"]
    }
    principals {
      type        = "*"
      identifiers = ["*"]
    }
  }
}

################################################################################
# Terraform Backend
################################################################################
module "terraform_backend" {
  source = "../terraform-modules/uoc-tfg-tf-aws-terraform-backend/"

  bucket_name = local.bucket_name
  dynamodb_table = local.dynamodb_table
  kms_key_alias = local.kms_key_alias
  bucket_policy = data.aws_iam_policy_document.bucket_policy.json
}

################################################################################
# Outputs
################################################################################
output "backend_bucket" {
  description = "The name of the S3 bucket used for storing the Terraform state"
  value = module.terraform_backend.backend_bucket
}
```

Inicialitzar Terraform: Executem la comanda terraform init per inicialitzar el backend amb la configuració especificada.(terraform-backend-initial-setup)

```
$ terraform init
Initializing the backend...
```

Després d'inicialitzar el backend, s'obtindrà el missatge d'èxit, el qual indica que el backend s'ha inicialitzat correctament. Ara, Terraform utilitzarà S3 com a backend per emmagatzemar el fitxer d'estat.



Aplicar Terraform: Una vegada s'ha iniciat, passem a aplicar el Terraform per a crear els recursos.

```
$ Terraform will perform the following actions:

  # module.terraform_backend.aws_dynamodb_table.this will be created
  + resource "aws_dynamodb_table" "this" {
      + arn              = (known after apply)
      + billing_mode     = "PAY_PER_REQUEST"
      + hash_key         = "LockID"
      + id               = (known after apply)
      + name             = "terraform-state-693946155914-us-east-1"
      + read_capacity    = (known after apply)
      + stream_arn       = (known after apply)
      + stream_label     = (known after apply)
      + stream_view_type = (known after apply)
      + tags_all         = (known after apply)
      + write_capacity   = (known after apply)

      + attribute {
          + name = "LockID"
          + type = "S"
        }

      + server_side_encryption {
          + enabled     = true
          + kms_key_arn = (known after apply)
        }
```

```
  }

# module.terraform_backend.aws_kms_alias.this will be created
+ resource "aws_kms_alias" "this" {
    + arn           = (known after apply)
    + id            = (known after apply)
    + name          = "alias/terraform-states"
    + name_prefix   = (known after apply)
    + target_key_arn = (known after apply)
    + target_key_id  = (known after apply)
  }

# module.terraform_backend.aws_kms_key.this will be created
+ resource "aws_kms_key" "this" {
    + arn                              = (known after apply)
    + bypass_policy_lockout_safety_check = false
    + customer_master_key_spec         = "SYMMETRIC_DEFAULT"
    + description                      = "This KMS key is used to encrypt the Terraform state file"
    + enable_key_rotation              = true
    + id                               = (known after apply)
    + is_enabled                       = true
    + key_id                           = (known after apply)
    + key_usage                        = "ENCRYPT_DECRYPT"
    + multi_region                     = (known after apply)
    + policy                           = (known after apply)
    + tags_all                         = (known after apply)
  }

# module.terraform_backend.aws_s3_bucket.this will be created
+ resource "aws_s3_bucket" "this" {
    + acceleration_status         = (known after apply)
    + acl                         = (known after apply)
    + arn                         = (known after apply)
    + bucket                      = "terraform-state-693946155914-us-east-1"
    + bucket_domain_name          = (known after apply)
    + bucket_prefix               = (known after apply)
    + bucket_regional_domain_name = (known after apply)
    + force_destroy               = false
    + hosted_zone_id              = (known after apply)
    + id                          = (known after apply)
    + object_lock_enabled         = (known after apply)
    + policy                      = (known after apply)
    + region                      = (known after apply)
    + request_payer               = (known after apply)
    + tags_all                    = (known after apply)
    + website_domain              = (known after apply)
    + website_endpoint            = (known after apply)
  }
```

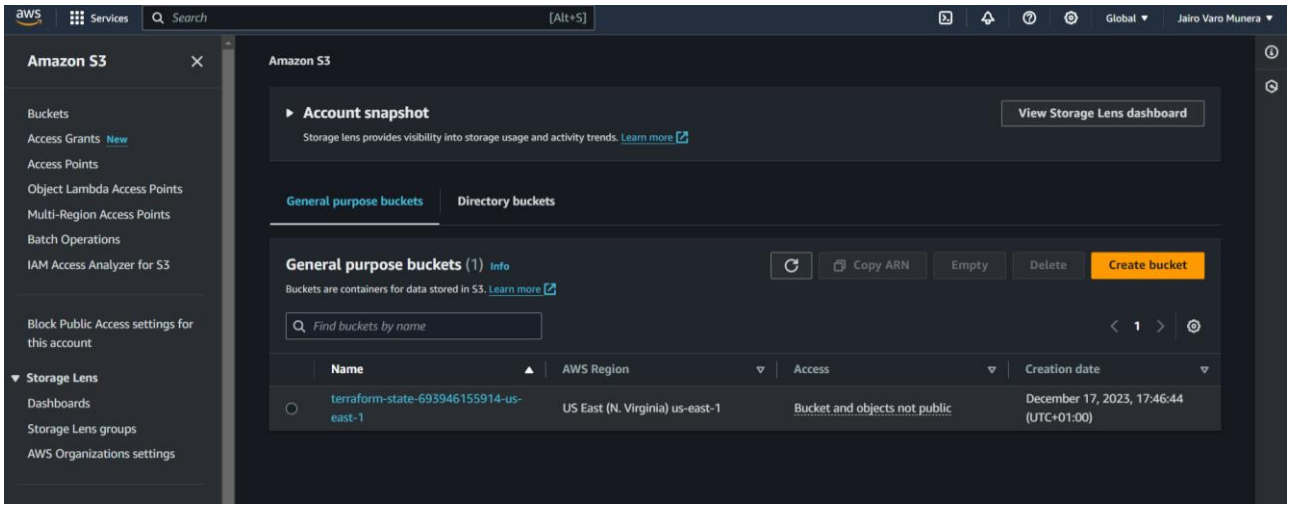
```
# module.terraform_backend.aws_s3_bucket_policy.this will be created
+ resource "aws_s3_bucket_policy" "this" {
    + bucket = (known after apply)
    + id     = (known after apply)
    + policy = jsonencode(
        {
          + Statement = [
              + {
                  + Action    = "s3:PutObject"
                  + Condition = {
                      + Null = {
                          + "s3:x-amz-server-side-encryption" = "true"
                        }
                    }
                  + Effect    = "Deny"
                  + Principal = "*"
                  + Resource  = "arn:aws:s3:::terraform-state-693946155914-us-east-1/*"
                  + Sid       = "DenyPublishingUnencryptedResources"
                },
              + {
                  + Action    = "s3:PutObject"
                  + Condition = {
                      + "ForAllValues:StringNotEquals" = {
                          + "s3:x-amz-server-side-encryption" = "aws:kms"
                        }
                    }
                  + Effect    = "Deny"
                  + Principal = "*"
                  + Resource  = "arn:aws:s3:::terraform-state-693946155914-us-east-1/*"
                  + Sid       = "DenyIncorrectEncryptionHeader"
                },
              + {
                  + Action    = "s3:*"
                  + Condition = {
                      + Bool = {
                          + "aws:SecureTransport" = "false"
                        }
                    }
                  + Effect    = "Deny"
                  + Principal = "*"
                  + Resource  = [
                      + "arn:aws:s3:::terraform-state-693946155914-us-east-1/*",
                      + "arn:aws:s3:::terraform-state-693946155914-us-east-1",
                    ]
                  + Sid       = "AllowSSLRequestsOnly"
                },
            ]
          + Version   = "2012-10-17"
        }
```

```
      )
  }

  # module.terraform_backend.aws_s3_bucket_public_access_block.this will be created
  + resource "aws_s3_bucket_public_access_block" "this" {
      + block_public_acls       = true
      + block_public_policy     = true
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = true
      + restrict_public_buckets = true
    }

  # module.terraform_backend.aws_s3_bucket_server_side_encryption_configuration.this
will be created
  + resource "aws_s3_bucket_server_side_encryption_configuration" "this" {
      + bucket = (known after apply)
      + id     = (known after apply)

      + rule {
          + apply_server_side_encryption_by_default {
              + kms_master_key_id = (known after apply)
              + sse_algorithm     = "aws:kms"
            }
        }
    }

  # module.terraform_backend.aws_s3_bucket_versioning.this will be created
  + resource "aws_s3_bucket_versioning" "this" {
      + bucket = (known after apply)
      + id     = (known after apply)

      + versioning_configuration {
          + mfa_delete = (known after apply)
          + status     = "Enabled"
        }
    }

Plan: 8 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + backend_bucket = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes
```

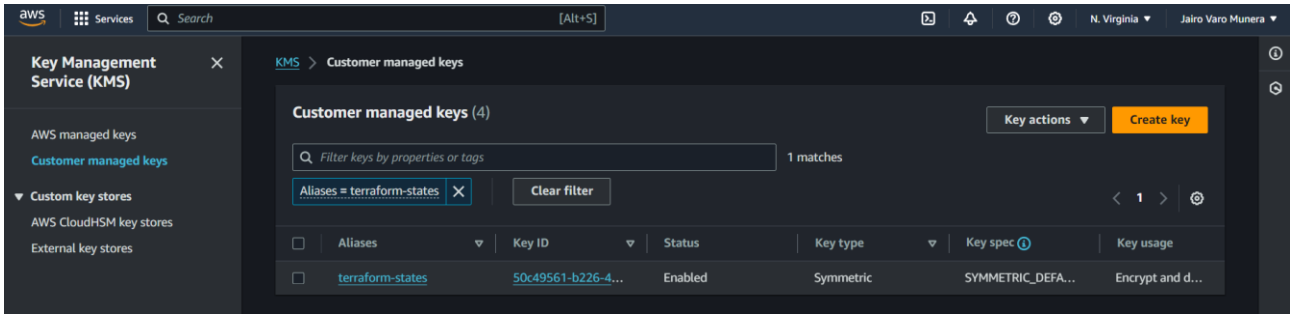
```
module.terraform_backend.aws_kms_key.this: Creating...
module.terraform_backend.aws_s3_bucket.this: Creating...
module.terraform_backend.aws_kms_key.this: Creation complete after 6s [id=50c49561-
b226-41cf-bbab-f9f5cde72a48]
module.terraform_backend.aws_kms_alias.this: Creating...
module.terraform_backend.aws_dynamodb_table.this: Creating...
module.terraform_backend.aws_kms_alias.this: Creation complete after 1s
[id=alias/terraform-states]
module.terraform_backend.aws_s3_bucket.this: Still creating... [10s elapsed]
module.terraform_backend.aws_dynamodb_table.this: Creation complete after 9s
[id=terraform-state-693946155914-us-east-1]
module.terraform_backend.aws_s3_bucket.this: Still creating... [20s elapsed]
module.terraform_backend.aws_s3_bucket.this: Creation complete after 22s [id=terraform-
state-693946155914-us-east-1]
module.terraform_backend.aws_s3_bucket_public_access_block.this: Creating...
module.terraform_backend.aws_s3_bucket_server_side_encryption_configuration.this:
Creating...
module.terraform_backend.aws_s3_bucket_versioning.this: Creating...
module.terraform_backend.aws_s3_bucket_policy.this: Creating...
module.terraform_backend.aws_s3_bucket_public_access_block.this: Creation complete
after 1s [id=terraform-state-693946155914-us-east-1]
module.terraform_backend.aws_s3_bucket_server_side_encryption_configuration.this:
Creation complete after 1s [id=terraform-state-693946155914-us-east-1]
module.terraform_backend.aws_s3_bucket_policy.this: Creation complete after 1s
[id=terraform-state-693946155914-us-east-1]
module.terraform_backend.aws_s3_bucket_versioning.this: Creation complete after 3s
[id=terraform-state-693946155914-us-east-1]

Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

backend_bucket = "terraform-state-693946155914-us-east-1"
```

Resum dels recursos: Finalment tenim els recursos necessaris creats, bàsicament hem creat els següents recursos.

- Un bucket S3: Emmagatzema l'estat de Terraform, que registra tots els recursos creats i és la ubicació on es guarda la informació de configuració, estat i històric dels recursos de Terraform.



- Una clau KMS: Proporciona seguretat en la gestió de claus de xifrat i es utilitzada per encriptar i protegir la informació sensible emmagatzemada al bucket S3.



- Una taula de DynamoDB: Actua com a taula de bloqueig per a Terraform i garanteix la coherència i prevé conflictes durant les operacions concurrents de Terraform, mantenint un estat segur i controlat.

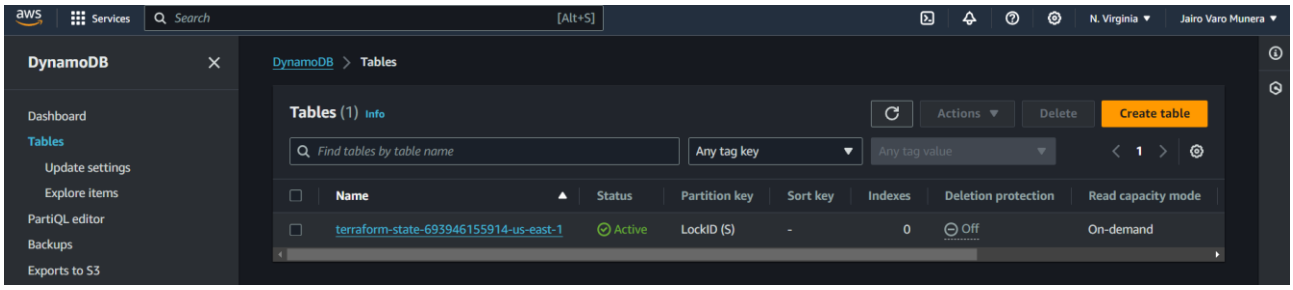
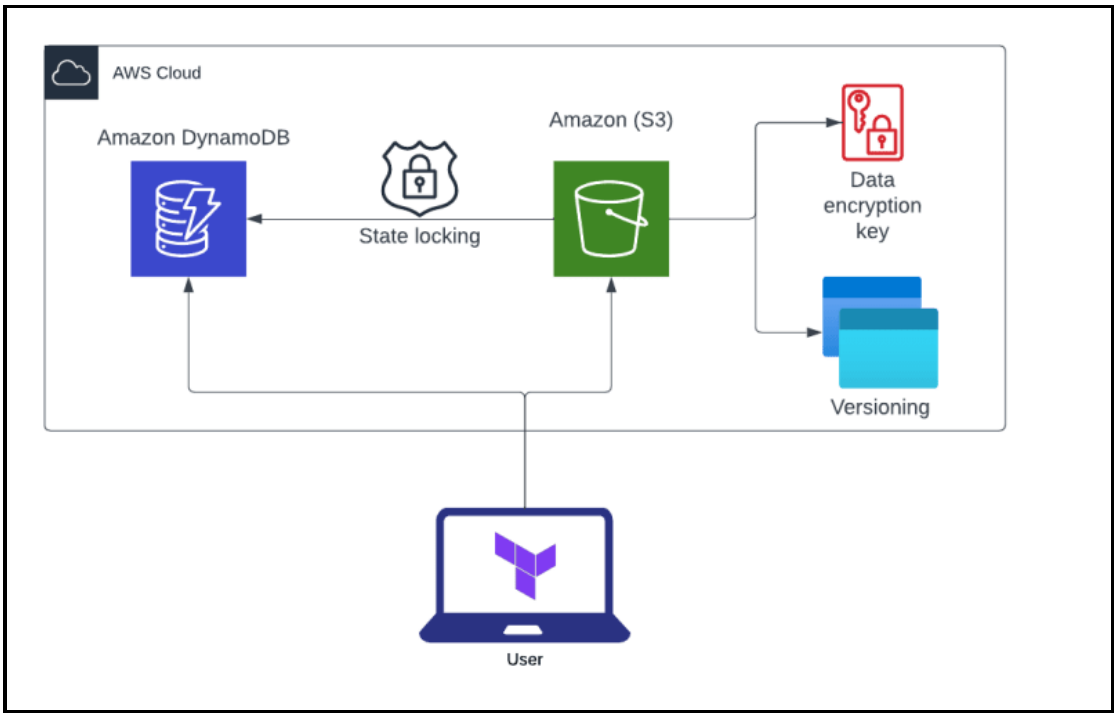


Figura 22. Esquema de la utilització del Terraform Backend amb Amazon S3 com a emmagatzematge remot de l'estat i DynamoDB per al bloqueig d'estat. Aquesta configuració assegura una gestió eficient i segura dels recursos d'infraestructura, proporcionant una base robusta per a la col·laboració i la gestió compartida del codi d'infraestructura amb Terraform en entorns AWS.



Font dev.to: Terraform backend with AWS S3 and DynamoDB state locking.

7.1. Laboratori 1: Desplegament d'una aplicació web estàtica amb Terraform

Aquest laboratori té com a objectiu guiar el lector a través del procés de desplegament d'una aplicació web senzilla i estàtica utilitzant Terraform.

Tots els fitxers necessaris per a la implementació d'aquesta demostració es troben dins del directori Laboratori 1.

Figura 23. A la imatge es mostren els logotips d'Amazon Web Services i Terraform. Aquesta representació simbolitza la integració i la utilització conjunta d'aquestes dues plataformes en el context del desplegament d'aplicacions web estàtiques amb Terraform a AWS.

Font medium.com: Host Static Website on AWS S3 Using Terraform.

Objectius
- Realitzar el desplegament d'una aplicació web estàtica utilitzant Terraform.
- Comprendre els fonaments del desplegament amb Terraform.
- Aplicar les millors pràctiques en la configuració de Terraform.
- Validar el desplegament de l'aplicació web estàtica.

Eines, programari i versions
- Terraform versió ~> 1.
- Editor de codi Visual Studio Code.
- AWS CLI.

Recursos
- Compte actiu en un proveïdor de serveis en el núvol: AWS.

Passos del Laboratori
1) Punt de partida: Es considera que ja disposem dels requisits bàsics per realitzar aquest laboratori.

- Un compte d'AWS.
- La interfície de línia de comandes AWS CLI instal·lada i configurada amb les credencials d'accés.
- Terraform instal·lat.
- Visual Studio com a editor de codi.

2) Crear els recursos necessaris per al lloc web estàtic: Crearem un bucket S3 per allotjar el nostre lloc web estàtic. Per això crearem un fitxer nou anomenat main.tf i amb el següent codi:

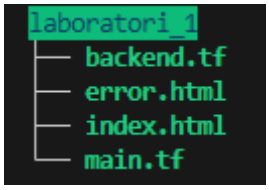
```
################################################################################
# Locals
################################################################################
locals {
  account_id          = "693946155914"
  region              = "us-east-1"
}

################################################################################
# S3 Static Website Bucket Resources
################################################################################
module "laboratori_1" {
    source = "../terraform-modules/uoc-tfg-tf-aws-s3-bucket/"

    create_bucket = true
    bucket = "laboratori-1-${local.account_id}-${local.region}"

    website = {
        index_document = "index.html"
        error_document = "error.html"
    }

    versioning = {
        status     = true
        mfa_delete = false
    }

    tags = {
        Name = "my-static-website"
    }
}
```

3) Aplicar Terraform: Una vegada hem generat el codi, inicialitzem i executem “terraform apply”, escrivim “yes” i els recursos es crearan.

```
Terraform will perform the following actions:

  # module.laboratori_1.aws_s3_bucket.this[0] will be created
  + resource "aws_s3_bucket" "this" {
      + acceleration_status         = (known after apply)
      + acl                         = (known after apply)
      + arn                         = (known after apply)
      + bucket                      = "laboratori-1-693946155914-us-east-1"
      + bucket_domain_name          = (known after apply)
      + bucket_prefix               = (known after apply)
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + object_lock_enabled         = false
      + policy                      = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags                        = {
          + "Name" = "my-static-website"
        }
      + tags_all                    = {
          + "Name" = "my-static-website"
        }
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)
    }

  # module.laboratori_1.aws_s3_bucket_public_access_block.this[0] will be created
  + resource "aws_s3_bucket_public_access_block" "this" {
      + block_public_acls       = true
      + block_public_policy     = true
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = true
      + restrict_public_buckets = true
    }

  # module.laboratori_1.aws_s3_bucket_versioning.this[0] will be created
  + resource "aws_s3_bucket_versioning" "this" {
      + bucket = (known after apply)
      + id     = (known after apply)

      + versioning_configuration {
          + mfa_delete = "Disabled"
```

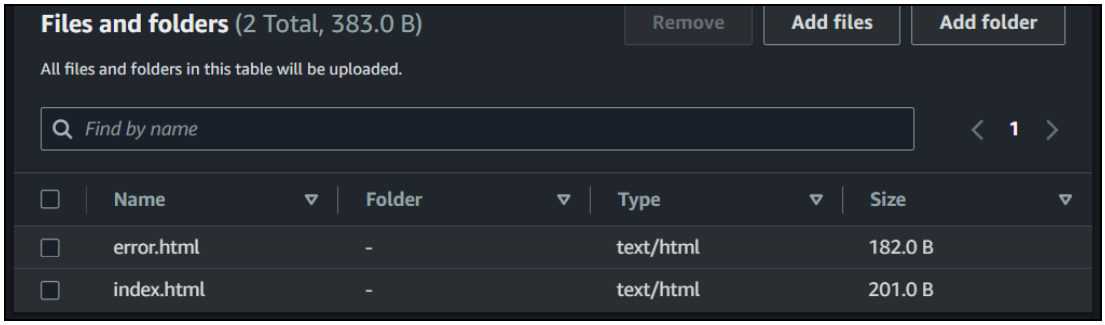
```
        + status     = "Enabled"
      }
  }

  # module.laboratori_1.aws_s3_bucket_website_configuration.this[0] will be created
  + resource "aws_s3_bucket_website_configuration" "this" {
      + bucket           = (known after apply)
      + id               = (known after apply)
      + routing_rules    = (known after apply)
      + website_domain   = (known after apply)
      + website_endpoint = (known after apply)

      + error_document {
          + key = "error.html"
        }

      + index_document {
          + suffix = "index.html"
        }
  }

Plan: 4 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

module.laboratori_1.aws_s3_bucket.this[0]: Creating...
module.laboratori_1.aws_s3_bucket.this[0]: Still creating... [10s elapsed]
module.laboratori_1.aws_s3_bucket.this[0]: Still creating... [20s elapsed]
module.laboratori_1.aws_s3_bucket.this[0]: Creation complete after 23s [id=laboratori-1-
693946155914-us-east-1]
module.laboratori_1.aws_s3_bucket_public_access_block.this[0]: Creating...
module.laboratori_1.aws_s3_bucket_versioning.this[0]: Creating...
module.laboratori_1.aws_s3_bucket_website_configuration.this[0]: Creating...
module.laboratori_1.aws_s3_bucket_public_access_block.this[0]: Creation complete after
0s [id=laboratori-1-693946155914-us-east-1]
module.laboratori_1.aws_s3_bucket_website_configuration.this[0]: Creation complete after
1s [id=laboratori-1-693946155914-us-east-1]
module.laboratori_1.aws_s3_bucket_versioning.this[0]: Creation complete after 2s
[id=laboratori-1-693946155914-us-east-1]

Apply complete! Resources: 4 added, 0 changed, 0 destroyed.
```

4) Mostra de l'arbre de directoris: Aquests son els fitxers necessaris per al desenvolupament del laboratori.

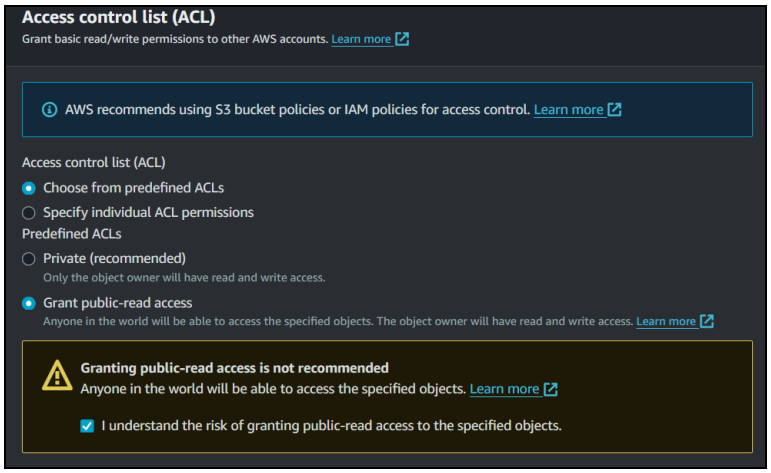


5) Crear una política per permetre l'accés públic: Aquest es un exemple molt senzill i no ho complicarem per tal de demostrar l'objectiu. Ara crearem una política per al bucket S3 que permetrà a qualsevol persona recuperar objectes del cubell.

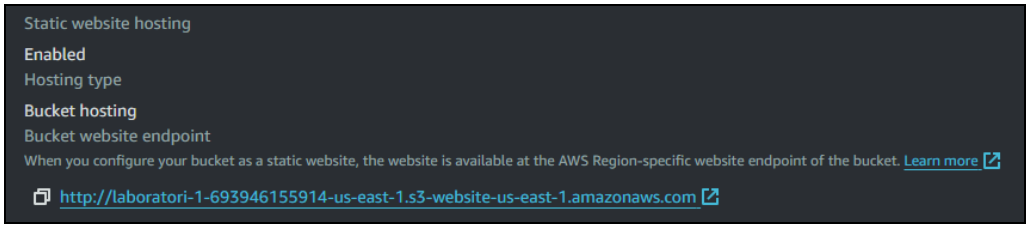
Via AWS Console fem upload dels fitxers index.html i error.html:



I en el mateix moment seleccionem el desplegable de permissions i escollim les següents opcions:



6) Comprovem l'accés al lloc web estàtic: Per accedir a la URL del lloc web estàtic de S3 després de ser creat. Podem veure dins de la Consola d'AWS la seva URL:



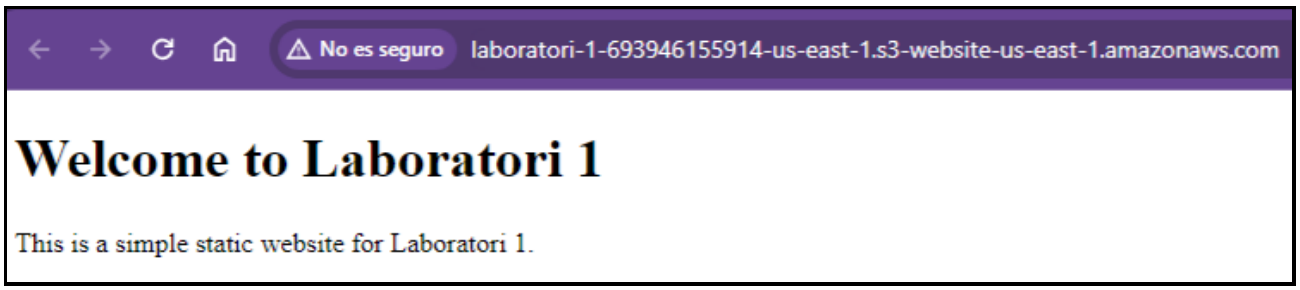
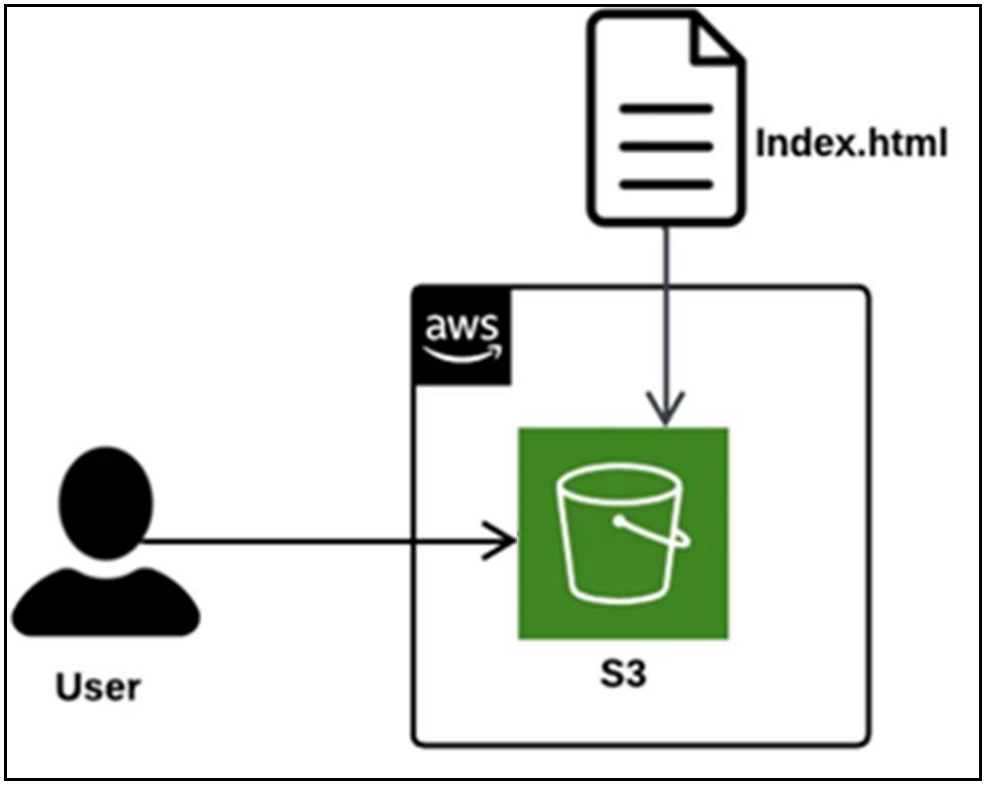


Figura 24. A la imatge, es mostra un diagrama que il·lustra el desplegament d'una aplicació web estàtica utilitzant Terraform i l'emmagatzematge d'Amazon S3.



Font medium.com: Creating an S3 bucket to host a static website.

Conclusió del Laboratori
L''experiència de desplegar un lloc web estàtic amb Amazon S3 mitjançant Terraform ha resultat enriquidora. Aquest enfocament, caracteritzat per la seva simplicitat i eficiència, demostra ser una elecció raonable per a projectes web senzills.

No obstant això, cal reconèixer la necessitat de continuar explorant millores i evolucions futures. Es podrien aprofundir en les polítiques de seguretat aplicades al bucket S3 i considerar integracions addicionals amb serveis com AWS Certificate Manager o CloudFront per a una millor seguretat i rendiment.

L'aprenentatge derivat d'aquest treball va més enllà de les habilitats tècniques, proporcionant una visió més ampla de l'ús estratègic de la informàtica en núvol.

Punts de millora i comentaris
En un context real i professional, aquest laboratori presenta algunes oportunitats d'optimització i millora:

- Millora de la Política del S3 Bucket: Refinar i ajustar la política del cubell S3 per assegurar el compliment dels estàndards de seguretat específics de l'organització.

- Seguretat del Bucket amb KMS: Seria recomanable incorporar una capa addicional de seguretat utilitzant AWS Key Management Service per a la gestió de claus. Això milloraria la protecció dels objectes emmagatzemats al bucket.

- Domini de Route53: Introduir un domini personalitzat amb Amazon Route 53 permetria tenir una URL més amigable i milloraria la identitat del lloc web.

- Certificat ACM: Incloure un certificat d'AWS Certificate Manager habilitaria HTTPS per al lloc web, garantint una connexió segura per als visitants.

- Distribució de CloudFront: Afegir una distribució de CloudFront milloraria l'eficiència i la velocitat del lloc web, proporcionant una CDN global per al contingut estàtic.

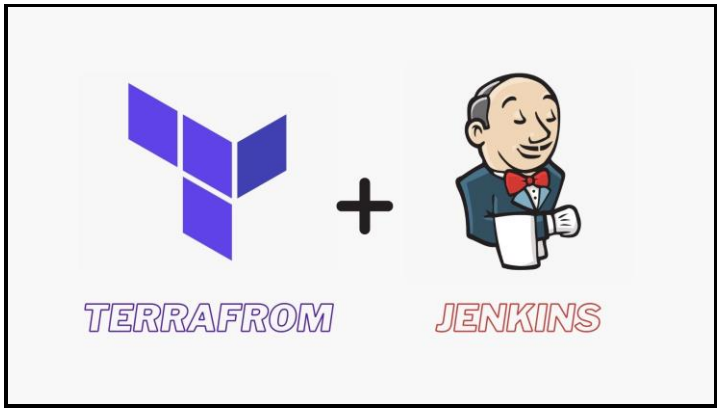
Aquests millores proporcionarien una base més robusta per a la seguretat, rendiment i fiabilitat del lloc web estàtic en un entorn de producció. La seva implementació dependrà de les necessitats específiques i dels requisits de l'aplicació.

7.2. Laboratori 2: Implementació d'un servidor Jenkins amb Terraform

Aquest laboratori està basat en una implementació simple d'un servidor Jenkins. Utilitzant Terraform, es desplegarà una instància EC2 i amb un script de Bootstrap en bash s'instal·larà i es posarà en marxa un servidor Jenkins dins de la pròpia instància.

Tots els fitxers necessaris per a la implementació d'aquesta demostració es troben dins del directori Laboratori 2.

Figura 25. A la imatge es visualitzen els logotips de Terraform i Jenkins, representant la integració d'aquestes dues eines en el context del laboratori.



Font medium.com: Terraform for build a Jenkins instance.

Objectius
- Implementar amb èxit un servidor Jenkins mitjançant Terraform.
- Comprendre com utilitzar Terraform per implementar infraestructures.
- Aprendre a configurar un servidor Jenkins mitjançant Terraform.

Eines, programari i versions
- Terraform versió ~> 1.
- Editor de codi Visual Studio Code.
- AWS CLI.

Recursos
- Compte actiu en un proveïdor de serveis en el núvol: AWS.

Passos del Laboratori
1) Punt de partida: Es considera que ja disposem dels requisits bàsics per realitzar aquest laboratori.

- Un compte d'AWS.
- La interfície de línia de comandes AWS CLI instal·lada i configurada amb les credencials d'accés.
- Terraform instal·lat.
- Visual Studio com a editor de codi.

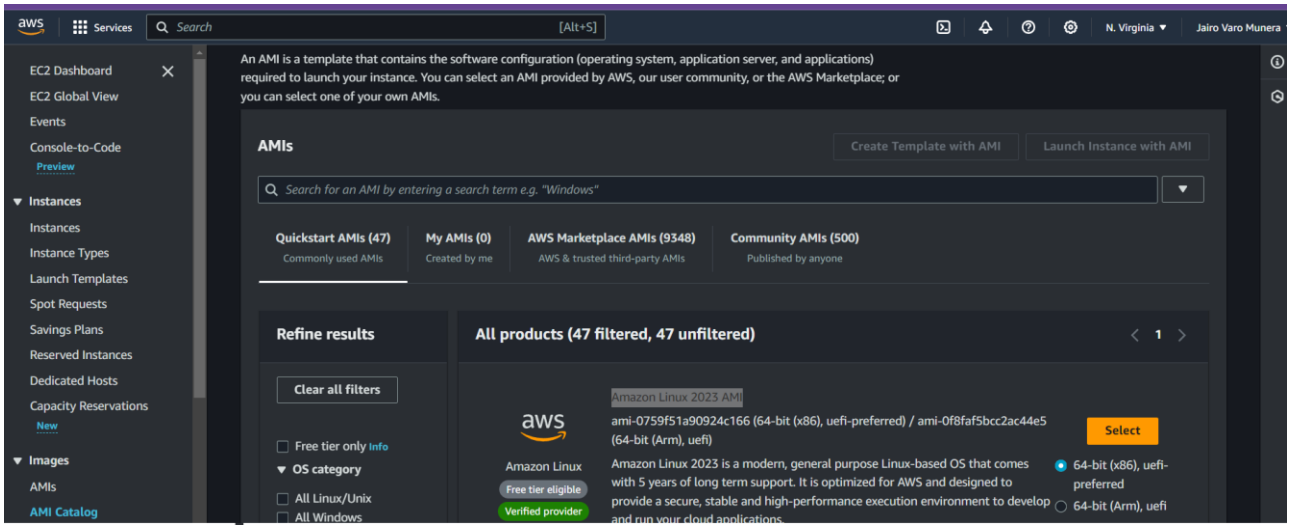
2) Crear l'script d'arrancada (boostrap.sh) per instal·lar i posar en marxa Jenkins a l'EC2: Després de revisar la documentació, creem un simple script per iniciar l'instancia amb tots els requisits necessaris.

https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

```
#!/bin/bash
sudo yum update -y
sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key
sudo yum upgrade -y
sudo dnf install java-11-amazon-corretto -y
sudo yum install jenkins -y
sudo systemctl enable jenkins
sudo systemctl start jenkins
```

3) Crear el fitxer main.tf amb el codi de Terraform: Per tal de simplificar la posada en marxa d'aquest laboratori, no utilitzarem mòduls de Terraform per al desplegament de Jenkins, ho farem directament a través dels propis resources de Terraform.

(*) Utilitzarem de base la AMI de Amazon Linux 2023: Bàsicament es tracta d'una instantània d'una màquina virtual EC2 que inclou el sistema operatiu i configuracions, usada per crear noves instàncies.




```
################################################################################
# Locals
################################################################################
locals {
  account_id          = "693946155914"
  region              = "us-east-1"
}
```

```
################################################################################
# Resources
################################################################################
resource "aws_instance" "jenkins_ec2" { // Aquesta és la instància EC2 que
s'utilitzarà per desplegar Jenkins
  ami           = "ami-0759f51a90924c166"
  instance_type = "t2.micro"
  key_name                    = "laboratori_2"
  associate_public_ip_address = true
  vpc_security_group_ids      = [aws_security_group.jenkins_security_group.id]
  user_data                   = file("bootstrap.sh") // Aquest es el script que
s'executarà al iniciar la instància

  tags = {
    Name = "Jenkins-Server"
  }
}

resource "aws_security_group" "jenkins_security_group" { // Aquesta és la regla de
seguretat que s'aplicarà a la instància
  name        = "jenkins_security_group"
  description = "Allows Port SSH and HTTP Traffic"

  ingress {
    description = "Allow SSH Traffic"
    from_port   = 22
    to_port     = 22
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "Allow HTTPS Traffic"
    from_port   = 443
    to_port     = 443
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    description = "Allow 8080 Traffic"
    from_port   = 8080
    to_port     = 8080
    protocol    = "TCP"
    cidr_blocks = ["0.0.0.0/0"]
  }
```

```
  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

resource "aws_s3_bucket" "jenkins_laboratori_2" { // Aquest és el bucket de S3 que
s'utilitzarà per guardar els artefactes de Jenkins
  bucket = "laboratori-2-${local.account_id}-${local.region}"

  tags = {
    Name = "Jenkins S3 Bucket"
  }
}

resource "aws_s3_bucket_acl" "s3_bucket_acl" {
  bucket = aws_s3_bucket.jenkins_laboratori_2.id
  acl    = "private"
  depends_on = [aws_s3_bucket_ownership_controls.s3_bucket_acl_ownership]
}

resource "aws_s3_bucket_ownership_controls" "s3_bucket_acl_ownership" {
  bucket = aws_s3_bucket.jenkins_laboratori_2.id
  rule {
    object_ownership = "ObjectWriter"
  }
}
```

4) Aplicar Terraform: Una vegada hem generat el codi, inicialitzem i apliquem terraform i els recursos es crearan.

```
Terraform will perform the following actions:

  # aws_instance.jenkins_ec2 will be created
  + resource "aws_instance" "jenkins_ec2" {
      + ami                          = "ami-0759f51a90924c166"
      + arn                          = (known after apply)
      + associate_public_ip_address  = true
      + availability_zone            = (known after apply)
      + cpu_core_count               = (known after apply)
      + cpu_threads_per_core         = (known after apply)
      + disable_api_stop             = (known after apply)
      + disable_api_termination      = (known after apply)
      + ebs_optimized                = (known after apply)
      + get_password_data            = false
```

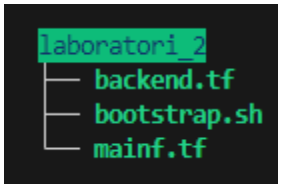
```
    + host_id                              = (known after apply)
    + host_resource_group_arn              = (known after apply)
    + iam_instance_profile                 = (known after apply)
    + id                                   = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
    + instance_lifecycle                   = (known after apply)
    + instance_state                       = (known after apply)
    + instance_type                        = "t2.micro"
    + ipv6_address_count                   = (known after apply)
    + ipv6_addresses                       = (known after apply)
    + key_name                             = "laboratori_2"
    + monitoring                           = (known after apply)
    + outpost_arn                          = (known after apply)
    + password_data                        = (known after apply)
    + placement_group                      = (known after apply)
    + placement_partition_number           = (known after apply)
    + primary_network_interface_id         = (known after apply)
    + private_dns                          = (known after apply)
    + private_ip                           = (known after apply)
    + public_dns                           = (known after apply)
    + public_ip                            = (known after apply)
    + secondary_private_ips                = (known after apply)
    + security_groups                      = (known after apply)
    + source_dest_check                    = true
    + spot_instance_request_id             = (known after apply)
    + subnet_id                            = (known after apply)
    + tags                                 = {
        + "Name" = "Jenkins-Server"
      }
    + tags_all                             = {
        + "Name" = "Jenkins-Server"
      }
    + tenancy                              = (known after apply)
    + user_data                            = "75a54c113dd525524465f515ddbf101a42fee221"
    + user_data_base64                     = (known after apply)
    + user_data_replace_on_change          = false
    + vpc_security_group_ids               = (known after apply)
  }

  # aws_s3_bucket.jenkins_laboratori_2 will be created
  + resource "aws_s3_bucket" "jenkins_laboratori_2" {
    + acceleration_status         = (known after apply)
    + acl                         = (known after apply)
    + arn                         = (known after apply)
    + bucket                      = "laboratori-2-693946155914-us-east-1"
    + bucket_domain_name          = (known after apply)
    + bucket_prefix               = (known after apply)
    + bucket_regional_domain_name = (known after apply)
    + force_destroy               = false
```

```
    + hosted_zone_id               = (known after apply)
    + id                           = (known after apply)
    + object_lock_enabled          = (known after apply)
    + policy                       = (known after apply)
    + region                       = (known after apply)
    + request_payer                = (known after apply)
    + tags                         = {
        + "Name" = "Jenkins S3 Bucket"
      }
    + tags_all                     = {
        + "Name" = "Jenkins S3 Bucket"
      }
    + website_domain               = (known after apply)
    + website_endpoint             = (known after apply)
  }

# aws_s3_bucket_acl.s3_bucket_acl will be created
+ resource "aws_s3_bucket_acl" "s3_bucket_acl" {
    + acl    = "private"
    + bucket = (known after apply)
    + id     = (known after apply)
  }

# aws_s3_bucket_ownership_controls.s3_bucket_acl_ownership will be created
+ resource "aws_s3_bucket_ownership_controls" "s3_bucket_acl_ownership" {
    + bucket = (known after apply)
    + id     = (known after apply)

    + rule {
        + object_ownership = "ObjectWriter"
      }
  }

# aws_security_group.jenkins_security_group will be created
+ resource "aws_security_group" "jenkins_security_group" {
    + arn                    = (known after apply)
    + description            = "Allows Port SSH and HTTP Traffic"
    + egress                 = [
        + {
            + cidr_blocks      = [
                + "0.0.0.0/0",
              ]
            + description      = ""
            + from_port        = 0
            + ipv6_cidr_blocks = []
            + prefix_list_ids  = []
            + protocol         = "-1"
            + security_groups  = []
            + self             = false
```

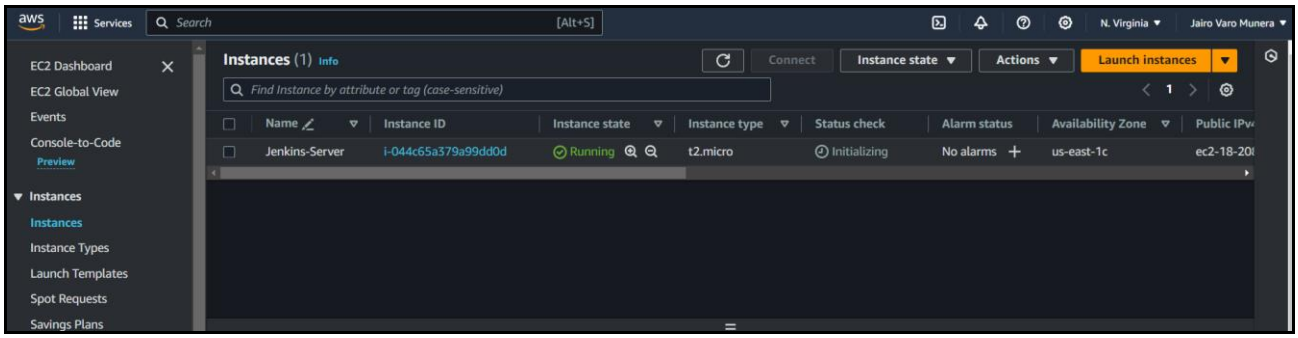
```
      + to_port         = 0
      },
  ]
+ id                    = (known after apply)
+ ingress               = [
    + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Allow 8080 Traffic"
      + from_port        = 8080
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 8080
    },
    + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Allow HTTPS Traffic"
      + from_port        = 443
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 443
    },
    + {
      + cidr_blocks      = [
          + "0.0.0.0/0",
        ]
      + description      = "Allow SSH Traffic"
      + from_port        = 22
      + ipv6_cidr_blocks = []
      + prefix_list_ids  = []
      + protocol         = "tcp"
      + security_groups  = []
      + self             = false
      + to_port          = 22
    },
  ]
+ name                  = "jenkins_security_group"
+ name_prefix           = (known after apply)
+ owner_id              = (known after apply)
+ revoke_rules_on_delete = false
```

```
      + tags_all              = (known after apply)
      + vpc_id                = (known after apply)
    }

Plan: 5 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

aws_s3_bucket.jenkins_laboratori_2: Creating...
aws_security_group.jenkins_security_group: Creating...
aws_security_group.jenkins_security_group: Creation complete after 4s [id=sg-
0e9d23efb6ee8180d]
aws_instance.jenkins_ec2: Creating...
aws_s3_bucket.jenkins_laboratori_2: Still creating... [10s elapsed]
aws_instance.jenkins_ec2: Still creating... [10s elapsed]
aws_s3_bucket.jenkins_laboratori_2: Still creating... [20s elapsed]
aws_s3_bucket.jenkins_laboratori_2: Creation complete after 22s [id=laboratori-2-
693946155914-us-east-1]
aws_s3_bucket_ownership_controls.s3_bucket_acl_ownership: Creating...
aws_s3_bucket_ownership_controls.s3_bucket_acl_ownership: Creation complete after 1s
[id=laboratori-2-693946155914-us-east-1]
aws_s3_bucket_acl.s3_bucket_acl: Creating...
aws_s3_bucket_acl.s3_bucket_acl: Creation complete after 0s [id=laboratori-2-
693946155914-us-east-1,private]
aws_instance.jenkins_ec2: Still creating... [20s elapsed]
aws_instance.jenkins_ec2: Still creating... [30s elapsed]
aws_instance.jenkins_ec2: Creation complete after 34s [id=i-044c65a379a99dd0d]

Apply complete! Resources: 5 added, 0 changed, 0 destroyed.
```

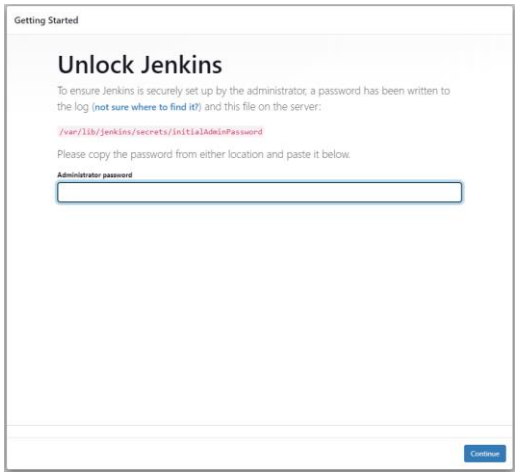
4) Mostra de l'arbre de directoris: Aquests són els fitxers necessaris per al desenvolupament del laboratori.



5) Validació dels recursos: Com podem comprovar, la nostra instancia EC2 està funcionant i per tant el servidor de Jenkins disponible.



Accedim a la instancia via web a través de la seva IP publica i port 8080 i completem el formulari inicial de Jenkins:



Per últim podem observar com tenim el nostre servidor de Jenkins disponible:

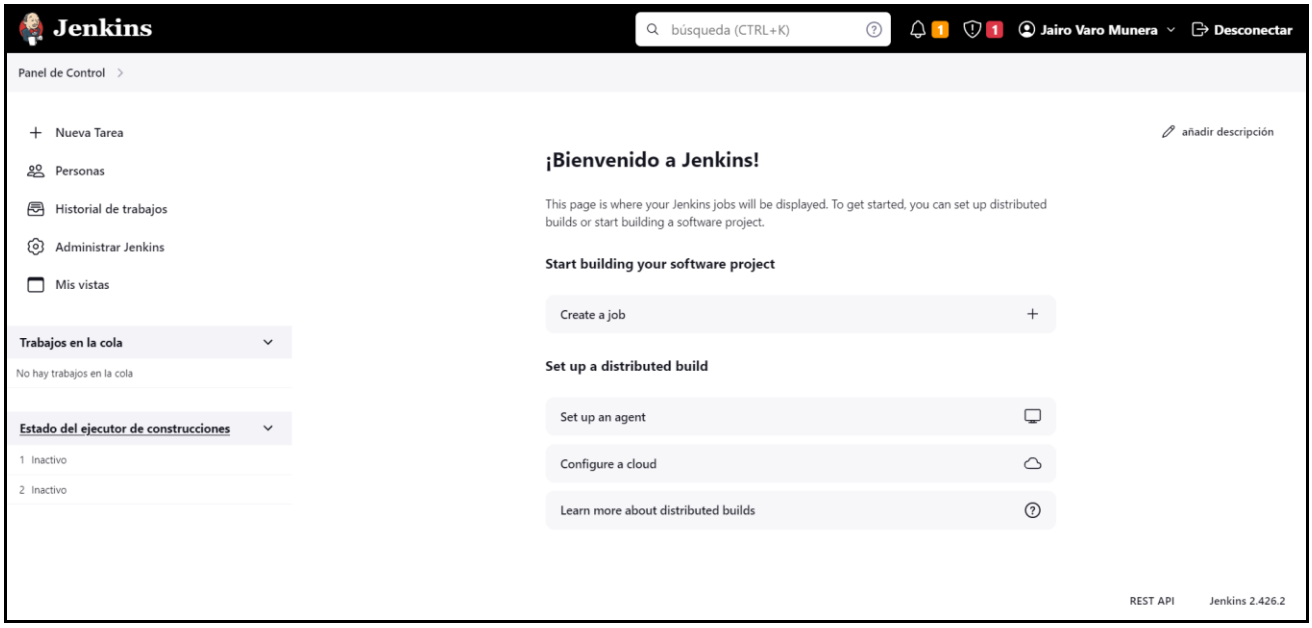
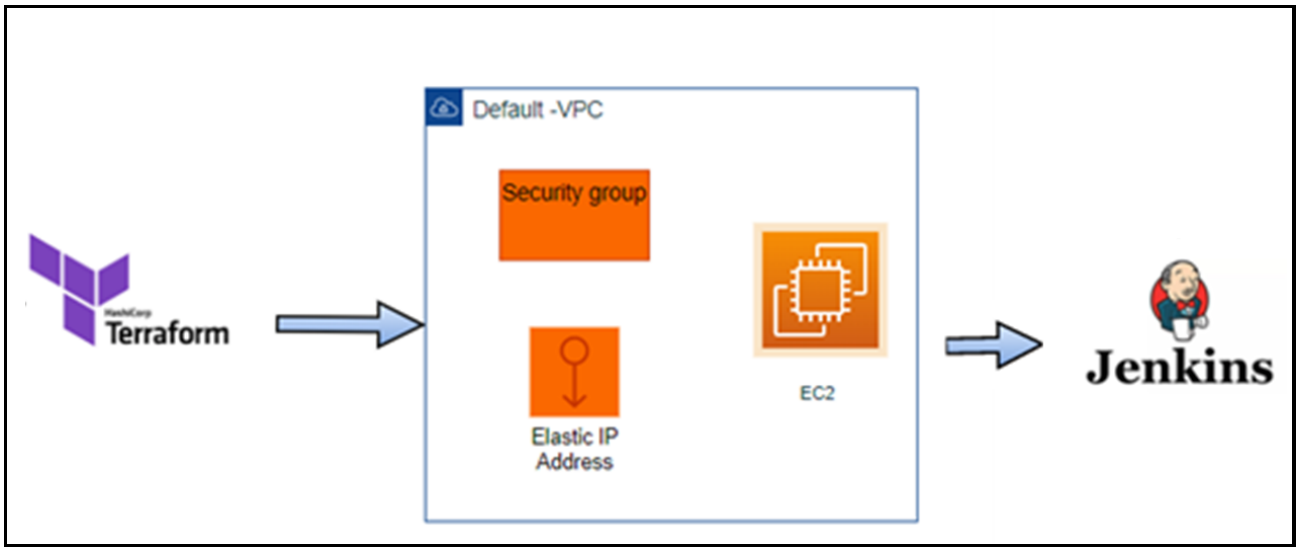


Figura 26. En aquesta imatge es presenta un diagrama amb la configuració de l'entorn EC2 mitjançant Terraform i Jenkins.



Font medium.com: *Diagram EC2 setup using Terraform and Jenkins.*

Conclusió del Laboratori
L'experiència d'implementar un servidor Jenkins amb Terraform ha estat una oportunitat d'aprenentatge valuosa. La utilització d'aquesta eina per desplegar una instància EC2 amb Jenkins ha destacat per la seva eficiència i simplicitat, demostrant la facilitat amb què Terraform pot gestionar tot el cicle de vida de la infraestructura.

No obstant això, reconec que hi ha possibilitats de millora. En futures iteracions, es podrien explorar alternatives d'instàncies EC2 per a una escalabilitat més gran i considerar l'ús de configuració com a codi (CASC) per gestionar la configuració de Jenkins de manera més eficient.

Aquest laboratori ha proporcionat una comprensió pràctica de l'ús estratègic de les eines de la informàtica en núvol i ha servit com a fonament per a futures exploracions en l'àmbit de la implementació d'infraestructures amb Terraform.

Punts de millora i comentaris
En un context real i professional, aquest laboratori presenta algunes oportunitats d'optimització i millora:

- Elecció d'Instància EC2: Per a necessitats de gran escala, pot ser més apropiat explorar alternatives com Fargate per a una millor escalabilitat.

- Configurabilitat amb CASC: S'optaria per la configuració com a codi (CASC) per gestionar la configuració de Jenkins de manera eficient. https://www.jenkins.io/projects/jcasc/

- Persistència de Dades: Considerar l'ús d'un volum EBS per garantir la persistència de dades crucials de Jenkins.

- Backup i Recuperació: Implementar polítiques adequades de backup i recuperació d'AWS per garantir la continuïtat del sistema.

- Seguretat: Revisar la configuració del grup de seguretat i els ports d'accés a la instància per assegurar una implementació segura del servei Jenkins.

- Domini personalitzat: Explorar l'ús de serveis com Route 53, certificats d'Amazon Certificate Manager i Amazon CloudFront per aconseguir un domini personalitzat amb HTTPS.

7.3. Laboratori 3: Integració de Repositoris de Codi i .githooks amb GitHub Actions

Aquest laboratori es centrarà en la integració de repositoris de codi amb GitHub Actions, amb especial atenció als fitxers .githooks. A través del procés d'implementació, es presentaran els fitxers i es destacaran les millors pràctiques per garantir una integració eficient.

Tots els fitxers necessaris per a la implementació d'aquesta demostració es troben dins del directori Laboratori 3.

Figura 27. La figura presenta únicament el logotip de GitHub, simbolitzant la integració de repositoris de codi amb GitHub Actions.

Font github.com: GitHub Actions Logo.

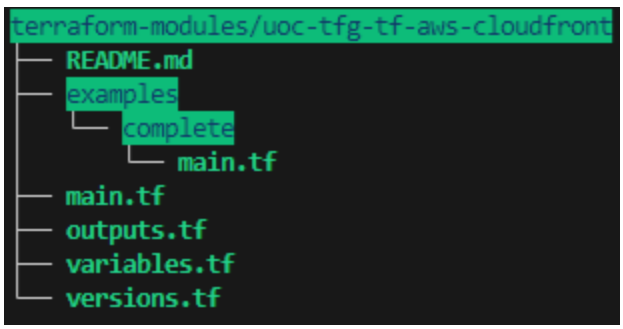
Objectius
- Integrar amb èxit un repositori de codi amb GitHub Actions.
- Configurar i utilitzar .githooks en la integració contínua amb GitHub Actions.
- Entendre la implementació i l'ús dels fitxers .githooks.

Eines, programari i versions
- Repositori de codi en GitHub amb un projecte vàlid (mòduls de Terraform)

Recursos
- Compte actiu de GitHub.

Passos del Laboratori

1) Punt de partida: Es considera que ja hem creat els repositoris necessaris amb el codi referent als mòduls de Terraform i l'estructura de fitxers típica. Per exemple:



2) Crear el fitxer de .githooks: L'objectiu es millorar la qualitat del mòdul de Terraform.

A més, integrarem l'ús d'eines addicionals esmentades a la secció "4.5. Instal·lació de Terraform i eines complementàries": Tfsec i Terraform Docs.

- **.githook/terraform-tools:** Aquest script realitza dos tasques::
    - Funció de f_terraform-docs: Actualitza el fitxer README.md utilitzant terraform-docs per mantenir la documentació actualitzada. Verifica si terraform-docs està instal·lat i executa la comprovació amb les opcions configurades, afegint els canvis a README.md si tot és correcte.
    - Funció de f_tfsec: Comprova la seguretat del codi Terraform amb tfsec i verifica si tfsec està instal·lat i executa la comprovació amb una severitat mínima establerta com a High.

```
#!/bin/bash

function f_terraform_docs {
  printf "👉 Executant terraform-docs i actualitzant fitxer README.md... \n"

  TERRAFORM_DOCS_CMD="terraform-docs"
  if ! command -v $TERRAFORM_DOCS_CMD &>/dev/null; then
    printf " -> ${TERRAFORM_DOCS_CMD} no s'ha trobat, si us plau instal·la-ho %s\n"
"❌"
    printf "https://terraform-docs.io/user-guide/installation/ \n"
    exit 1
  fi

  TERRAFORM_CMD_DOCS="$TERRAFORM_DOCS_CMD -c .terraform-docs.yml ."
  if $TERRAFORM_CMD_DOCS &>/dev/null; then
    git add README.md
    printf "  ->  %s ☑\n" "$TERRAFORM_CMD_DOCS"
  else
    printf "  ->  %s ERROR: ha fallat! ❌\n" "$TERRAFORM_CMD_DOCS"
    exit 1
  fi
}

function f_tfsec {
  printf "👉 Executant tfsec i comprovant la seguretat... \n"
  TFSEC_CMD="tfsec"
  TFSEC_CMD_ARG="--minimum-severity HIGH"
  if ! command -v "$TFSEC_CMD" &>/dev/null; then
    printf " -> ${TFSEC_CMD} no s'ha trobat, si us plau instal·la-ho %s\n" "❌"
    printf "https://tfsec.dev/docs/installation/ \n"
    exit 1
  fi
```

```
  TFSEC_CMD_EXEC="${TFSEC_CMD} ${TFSEC_CMD_ARG}"
  if $TFSEC_CMD_EXEC . &>/dev/null; then
    printf "  ->  %s ☑\n" "$TFSEC_CMD_EXEC"
  else
    printf "  ->  %s ERROR: ha fallat! ❌\n" "$TFSEC_CMD_EXEC"
    exit 1
  fi
}

# Si el primer argument és un nom de funció, crida aquesta funció amb els arguments
restants
if declare -f "$1" &>/dev/null; then
  "$@"
else
  # Si el primer argument no és un nom de funció, executa el hook per defecte
  printf "👉 Executant el .githook/terraform-tools ...\n"

  f_terraform_docs
  f_tfsec
fi
```

3) Crear el fitxer .github/workflows: Amb l'objectiu de gestionar el codi de manera eficient, introduïm un fitxer específic per a GitHub Actions que realitzi tasques de validació.

- **.github/workflows/pull-request.yml:** Aquesta action s'activa en cada pull request a la branch principal i s'executa en servidor de GitHub. Cadascun dels passos d'aquest flux de treball té una funció específica, des de la instal·lació de les eines necessàries fins a l'execució de les validacions de seguretat i documentació.

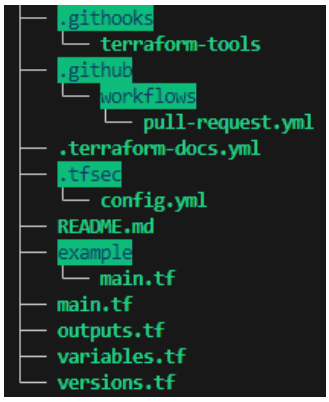
  Aquests passos s'utilitzen amb els scripts definits anteriorment (.githooks/terraform-tools), assegurant una execució coherent i automatitzada per mantenir la consistència i la integritat del codi.

```
name: Validacions

on:
  workflow_dispatch:
  pull_request:
    branches:
      - main

jobs:
  validate:
    runs-on: ubuntu-latest
    steps:
```

```
      - name: Checkout del codi font
        uses: actions/checkout@v4

      - name: Instal·la les eines necessàries del sistema operatiu
        run: sudo apt-get install -y curl git tree

      - name: Instal·lació de Terraform
        run: |
          export TERRAFORM_OS_ARCH="linux_amd64"
          export TERRAFORM_VERSION="1.5.7"
          export
TERRAFORM_URL="https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraf
orm_${TERRAFORM_VERSION}_${TERRAFORM_OS_ARCH}.zip"
          curl -L -O -J $TERRAFORM_URL
          unzip terraform_${TERRAFORM_VERSION}_${TERRAFORM_OS_ARCH}.zip
          chmod +x terraform
          mv terraform /usr/local/bin/terraform
          terraform --version

      - name: Instal·lació de Terraform Docs
        run: |
          export TERRAFORM_DOCS_OS_ARCH="linux-amd64"
          export TERRAFORM_DOCS_VERSION="0.16.0"
          export TERRAFORM_DOCS_URL="https://terraform-
docs.io/dl/v${TERRAFORM_DOCS_VERSION}/terraform-docs-v${TERRAFORM_DOCS_VERSION}-
${TERRAFORM_DOCS_OS_ARCH}.tar.gz"
          curl -sSLo ./terraform-docs.tar.gz $TERRAFORM_DOCS_URL
          tar -xzf terraform-docs.tar.gz
          chmod +x terraform-docs
          mv terraform-docs /usr/local/bin/terraform-docs
          terraform-docs --version

      - name: Instal·lació de tfsec
        run: |
          export TFSEC_OS_ARCH="linux_amd64"
          export TFSEC_VERSION="latest"
          export TFSEC_URL=$(curl -s
"https://api.github.com/repos/aquasecurity/tfsec/releases/${TFSEC_VERSION}" \
                             | grep
"browser_download_url.*${TFSEC_OS_ARCH}.tar.gz" \
                             | cut -d : -f 2,3 \
                             | tr -d \" \
                             | sed 's/^ *//;s/ *$//')
          curl -sSLo ./tfsec.tar.gz $TFSEC_URL
          tar -xzf tfsec.tar.gz
          chmod +x tfsec
          mv tfsec /usr/local/bin/tfsec
```

```
        tfsec --version

    - name: Executa la validació de seguretat
      run: bash .githooks/terraform-tools f_tfsec

    - name: Executa la validació de la documentació
      run: |
        bash .githooks/terraform-tools f_terraform_docs
```

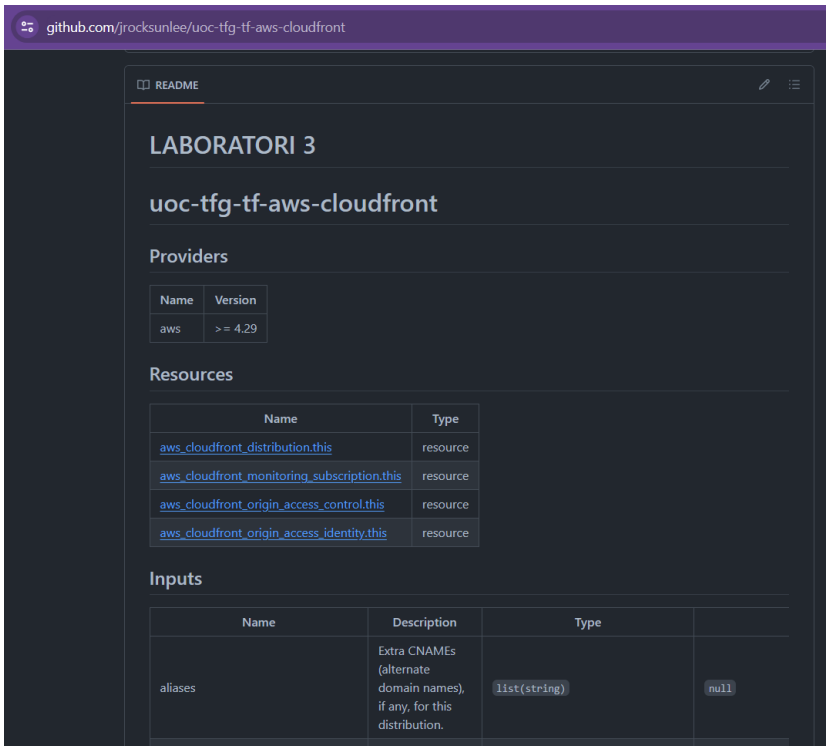
4) Mostra de l'arbre de directoris: Aquests son els fitxers necessaris per al desenvolupament del laboratori. Exemple d'un mòdul de Terraform:



5) Validació del laboratori:

- **Creació automatitzada del fitxer README.md amb terraform-docs:** Aquest pas s'executa de forma local i es en la GitHub action quan es fa push del codi que es comprova que el fitxer conté el resultat d'executar terraform-docs.

(*) Per tant, hem de considerar que el .githook s'executa dues vegades, la primera en local i es quan es persisteix al fitxer de README.md i la segona de forma remota en el servidor de github on es comprova que el resultat es el mateix.



- **Revisió automatitzada de la seguretat amb tfsec:**

```
$ .githooks/terraform-tools
👉 Executant el .githook/terraform-tools ...
👉 Executant terraform-docs i actualitzant fitxer README.md...
  ->  terraform-docs -c .terraform-docs.yml . ✅
👉 Executant tfsec i comprovant la seguretat...
  ->  tfsec --minimum-severity HIGH ERROR: ha fallat! ❌
FAIL
```

La següent imatge mostra per que es dona aquest error, la causa es que tfsec ha detectat una vulnerabilitat que s'ha de resoldre, per tant no es permet fer push del commit.

```
 27    resource "aws_cloudfront_distribution" "this" {
 ..
256 [     minimum_protocol_version = lookup(var.viewer_certificate, "minimum_protocol_version", "TLSv1") ("TLSv1")
...
282    }

        ID aws-cloudfront-use-secure-tls-policy
    Impact Outdated SSL policies increase exposure to known vulnerabilities
Resolution Use the most modern TLS/SSL policies available

More Information
- https://aquasecurity.github.io/tfsec/v1.28.4/checks/aws/cloudfront/use-secure-tls-policy/
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudfront_distribution#minimum_protocol_version


timings
─────────────────────────────────────────
disk i/o             639.2µs
parsing              11.7426ms
adaptation           225.8µs
checks               9.2222ms
total                21.8298ms

─────────────────────────────────────────
modules downloaded   0
modules processed    1
blocks processed     49
files read           4

results
─────────────────────────────────────────
passed               2
ignored              1
critical             0
high                 1
medium               0
low                  0

  2 passed, 1 ignored, 1 potential problem(s) detected.
FAIL
```

Una vegada es soluciona la vulnerabilitat aplicant el canvi que el propi tfsec suggereix, podem fer commit i push sense cap problema. En el següent pas ja hem fer la acció de push.

- **Validació del .githook amb una action de GitHub:**

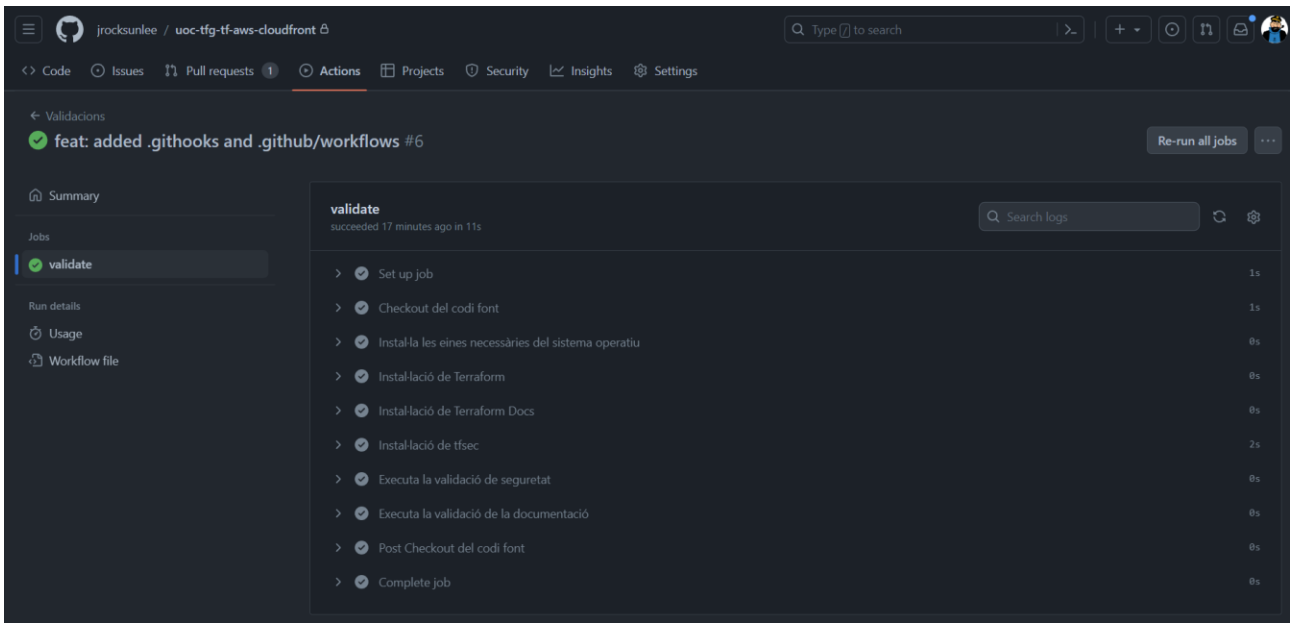
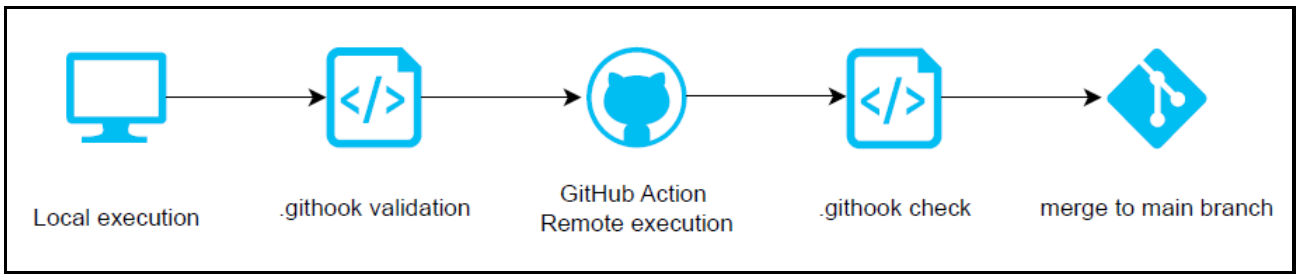


Figura 28. En aquesta figura, es mostra el procés d'execució del .githook, validant els commits localment abans de permetre l'acció de push. Posteriorment, GitHub realitza una comprovació similar. En cas que tots els passos es completin amb èxit, s'autoritza l'acció de merge a la branch principal.



Font app.diagrams.net .githooks flow for validate commits.

Conclusió del Laboratori

La implementació de la integració de repositoris de codi amb GitHub Actions i l'ús dels fitxers .githooks ha estat una experiència positiva i educativa. Aquesta pràctica ha aprofundit en els coneixements prèvis sobre GitHub Actions, oferint una perspectiva pràctica de com automatitzar tasques crucials en el desenvolupament de projectes amb mòduls de Terraform.

L'aplicació d'eines com GitHub Actions i .githooks per millorar la qualitat i la consistència dels mòduls de Terraform es fàcil d'integrar. Aquesta experiència no només ha enfortit les competències tècniques, sinó que també ha subratllat la importància de la integració contínua i l'automatització en el desenvolupament de projectes.

Punts de millora i comentaris
En un context real i professional, aquest laboratori presenta algunes oportunitats d'optimització i millora:

- Gestió de dependències: Podria considerar-se la inclusió d'un sistema de gestió de dependències per als scripts i eines utilitzades, assegurant que tothom que contribueixi al projecte disposi de les mateixes versions d'eines.

- Configuració d'actions: Per a GitHub Actions, es podrien afegir més detalls de configuració com ara la inclusió d'informació d'entorn, gestió de secrets i notificacions específiques.

- Gestió de errors més detallada amb logs: Afegir una gestió més detallada d'errors als scripts, proporcionant informació addicional sobre per què una tasca ha fallat per facilitar la identificació i correcció d'errors.

- Validació dels missatges de commits: Afegir la validació dels missatges de commit semàntic podria ser una millora valuosa per assegurar-se que les contribucions es realitzen de manera coherent.

8. Valoració Personal

Aquest treball, amb un enfocament teòric, introdueix una àmplia gamma de conceptes sense profunditzar de manera exhaustiva. S'aborden nombroses eines i metodologies de treball, que han resultat essencials per al meu creixement personal i desenvolupament professional.

Els laboratoris, tot i ser exemples pràctics, presenten casos senzills que serveixen com a introducció als conceptes de les seccions 2 a 6. L'objectiu principal era oferir una visió bàsica i clara d'aquests temes.

Amb l'ajustament a les restriccions de format i extensió requerides, he prioritzat l'exposició de continguts essencials. No obstant això, estic satisfet amb el resultat final, i crec que aquest document pot ser una guia útil per a aquells que vulguin adquirir coneixements bàsics sobre DevOps, Infraestructura com a Codi i el cicle de desenvolupament associat.

9. Agraïments

A la meva estimada companya Diandra, li expresso el meu profund agraïment pel seu suport constant, la seva ànima inspiradora i la motivació inestimable que m'ha proporcionat al llarg d'aquesta intensa etapa acadèmica. Als meus estimats pares, Diego, Trini, i al meu germà Cristian, els qui sempre han valorat la importància de completar els estudis universitaris, mostrant-me que mai és massa tard.

A més, agraeixo profundament als meus bons amics Ruben, Gerard i Erik per la seva implicació i suport quan els he demanat ajuda i opinió tant pel treball com en altres temes relacionats amb el grau.

La meva gratitud també es dirigeix als meus antics companys de feina, Cèsar, David O, Moha, David V i Jonay, que, d'alguna manera, han estat fonts d'inspiració per a la meva aspiració de millorar professionalment, des dels temps que vam compartir en el mateix entorn laboral. Als meus actuals companys de feina, dels quals continuo aprenent constantment i que contribueixen al meu creixement professional: Christian, Carlos i Eloy.

Finalment, desitjo expressar el meu reconeixement al meu tutor del TFG, Fernando, per la seva orientació i suport durant les diferents fases d'aquest treball.

Com m’agrada dir: *Un rockandrolla sempre vol el pack complert.*

10. Annexos

L'apartat d'Annexos conté diversos directoris relacionats amb la elaboració d'aquest TFG, especialment per a la part dels laboratoris. En cada apartat del treball està especificat el seu propòsit.

```
├── GranttTFG_Varo_Munera_Jairo.mpp
├── Makefile
├── laboratori_1
├── laboratori_2
├── laboratori_3
├── laboratoris_terraform_backend
└── terraform-modules
```

11. Referències

- [W] Wikipedia, Article de DevOps, Consultat el 23 de desembre de 2023. https://es.wikipedia.org/wiki/DevOps
- [O] O'Reilly Radar, What is DevOps?, Consultat el 23 de desembre de 2023. http://radar.oreilly.com/2012/06/what-is-devops.html
- [S] Somic, The Rise of DevOps, Consultat el 23 de desembre de 2023. http://www.somic.org/2010/03/02/the-rise-of-devops/
- [I] IT Revolution, DevOps Culture (Part 1), Consultat el 23 de desembre de 2023. https://itrevolution.com/articles/devops-culture-part-1/
- [Q] InfoQ, DevOps Toolchain, Consultat el 23 de desembre de 2023. https://www.infoq.com/articles/devops-toolchain/
- [P] Paradigma Digital, DevOps: cómo ha evolucionado (o desvirtuado), Consultat el 23 de desembre de 2023. https://www.paradigmadigital.com/dev/devops-como-ha-evolucionado-desvirtuado/
- [I] IT User, Las 5 etapas de la evolución de DevOps, Consultat el 23 de desembre de 2023. https://discoverthenew.ituser.es/devops/2018/10/las-5-etapas-de-la-evolucion-de-devops
- [K] KnowledgeHut, DevOps Infrastructure, Consultat el 23 de desembre de 2023. https://www.knowledgehut.com/blog/devops/devops-infrastructure
- [C] CircleCI Blog, Platform Engineering: DevOps at Scale, Consultat el 23 de desembre de 2023. https://circleci.com/blog/platform-engineering-devops-at-scale/

- [W] Wikipedia, [D] Cloud Computing, Consultat el 23 de desembre de 2023 https://en.wikipedia.org/wiki/Cloud_computing
- [A] Amazon Web Services, What is Cloud Computing?, Consultat el 23 de desembre de 2023 https://aws.amazon.com/es/what-is-cloud-computing/
- [Z] ZDNet, What is Cloud Computing: Everything you need to know about the cloud, Consultat el 23 de desembre de 2023 https://www.zdnet.com/article/what-is-cloud-computing-everything-you-need-to-know-about-the-cloud/
- [G] Gartner, Cloud Service Providers, Consultat el 23 de desembre de 2023. https://www.gartner.com/document/4019559?ref=solrAll&refval=383079209&
- [A] All Code, Top AWS Services, Consultat el 23 de desembre de 2023. https://allcode.com/top-aws-services/
- [E] EC-Council, Biggest Cloud Service Providers, Consultat el 23 de desembre de 2023. https://www.eccouncil.org/cybersecurity-exchange/cloud-security/biggest-cloud-service-providers/
- [S] Stackify, What is Infrastructure as Code (IaC): A Complete Guide, Consultat el 23 de desembre de 2023. https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/
- [W] Wikipedia, Infrastructure as Code, Consultat el 23 de desembre de 2023. https://en.wikipedia.org/w/index.php?title=Infrastructure_as_code
- [F] F5 Networks, Infrastructure as Code (IaC), Consultat el 23 de desembre de 2023. https://www.f5.com/es_es/glossary/infrastructure-as-code-iac
- [I] Medium, Extensive Comparison of IaC Tools, Consultat el 23 de desembre de 2023. https://ibatulanand.medium.com/extensive-comparison-of-iac-tools-49118e962ef8
- [H] HashiCorp, Terraform vs. Other IaC Tools, Consultat el 23 de desembre de 2023. https://developer.hashicorp.com/terraform/intro/vs
- [T] GitHub, terraform-docs, Consultat el 23 de desembre de 2023. https://github.com/terraform-docs/terraform-docs
- [T] GitHub, tfsec, Consultat el 23 de desembre de 2023. https://github.com/aquasecurity/tfsec
- [T] GitHub, terraform-switcher, Consultat el 23 de desembre de 2023. https://github.com/warrensbox/terraform-switcher

- [L] LinkedIn, Caso Real: Terraform y Jenkins, Consultat el 23 de desembre de 2023. https://www.linkedin.com/pulse/caso-real-terraform-jenkins-juan-ignacio-paz/?originalSubdomain=es

- [W] Wikipedia, Control de versiones, Consultat el 23 de desembre de 2023. https://es.wikipedia.org/wiki/Control_de_versiones

- [D] Drauta, 5 softwares de control de versiones que debes conocer, Consultat el 23 de desembre de 2023. https://www.drauta.com/5-softwares-de-control-de-versiones

- [Y] YouTube, Qué es GIT y cómo usar un sistema de control de versiones, Consultat el 23 de desembre de 2023. https://www.youtube.com/watch?v=hN8zaUu_k-k: Qué es GIT y como usar un sistema de control de versiones

- [C] Cadence PCB, What is a Version Control System?, Consultat el 23 de desembre de 2023. https://resources.pcb.cadence.com/blog/what-is-a-version-control-system

- [F] freeCodeCamp, What is Git? Learn Git Version Control, Consultat el 23 de desembre de 2023. https://www.freecodecamp.org/news/what-is-git-learn-git-version-control/

- [G] Tower, Git Best Practices, Consultat el 23 de desembre de 2023. https://www.git-tower.com/learn/git/ebook/en/command-line/appendix/best-practices

- [H] HubSpot, Herramientas DevOps: 20 herramientas para mejorar tu desarrollo, Consultat el 23 de desembre de 2023. https://blog.hubspot.es/website/herramientas-devops

- [R] Red Hat, What is CI/CD?, Consultat el 23 de desembre de 2023. https://www.redhat.com/es/topics/devops/what-is-ci-cd

- [L] LinkedIn, What Does Continuous Integration/Delivery (CI/CD) Mean?, Consultat el 23 de desembre de 2023. https://www.linkedin.com/pulse/what-does-continuous-integration-delivery-cicd-mean-context-devops-/

- [J] JetBrains, Continuous Integration vs. Delivery vs. Deployment, Consultat el 23 de desembre de 2023. https://www.jetbrains.com/es-es/teamcity/ci-cd-guide/continuous-integration-vs-delivery-vs-deployment/

- [S] Stack Overflow, Continuous Integration vs. Continuous Delivery vs. Continuous Deployment, Consultat el 23 de desembre de 2023. https://stackoverflow.com/questions/28608015/continuous-integration-vs-continuous-

delivery-vs-continuous-deployment

- [W] Wikipedia, [D] Jenkins, Consultat el 23 de desembre de 2023. https://es.wikipedia.org/wiki/Jenkins
- [S] Sentrio, ¿Qué es Jenkins?, Consultat el 23 de desembre de 2023. https://sentrio.io/blog/que-es-jenkins/
- [J] Jorge Rodríguez, CI/CD con GitHub Actions, Consultat el 23 de desembre de 2023. https://jorgehrj.medium.com/ci-cd-con-github-actions-66a7c325f45f
- [G] GitHub Docs, Understanding GitHub Actions, Consultat el 23 de desembre de 2023. https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions
- [G] GitHub Blog, How we build a CI/CD pipeline with GitHub Actions in four steps, Consultat el 23 de desembre de 2023. https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/
- [K] K21 Academy, GitHub Actions vs. Jenkins, Consultat el 23 de desembre de 2023. https://k21academy.com/devops-foundation/github-actions-vs-jenkins
- [A] AWS Tips, Host a Static Website on AWS S3 Using Terraform, Consultat el 23 de desembre de 2023. https://awstip.com/host-static-website-on-aws-s3-using-terraform-60cdbb7e0702
- [A] AWS Plain English, Using Terraform to Install and Launch Jenkins on an EC2 Instance, Consultat el 23 de desembre de 2023. https://aws.plainenglish.io/using-terraform-to-install-and-launch-jenkins-on-an-ec2-instance-151f676fc5c6