Desenvolupament teòric / pràctic d'una CI / CD amb la creació d'un microservei per a reserves d'espais eventuals.

Memòria de Projecte Final de Màster

Enginyeria Informàtica

Desenvolupament d'aplicacions

Autor: David Hidalgo Muñoz

Consultor: Joan Giner Miguelez

Professor: David García Solórzano

6 de gener de 2020

Crèdits/Copyright

Dedicatòria/Cita

A la meva família, sobretot al meu fill, l'Edgar, que s'ha acostumat a veure al seu pare treballant quan ell només vol jugar a totes hores; i a la meva dona Iolanda, per tanta paciència i comprensió.
Us estimo.

Als meus companys de feina que comprenen que els dies propers als lliuraments són dies amb poca nit i amb molta cafeïna. Al David T. ("el capi") per tot el recolzament que ens hem donat, a Juan C. ("vestúqueyonopuedo") per ensenyar-me tant i a en Marc P. ("mestre Jedi") per suportar-me els dies més fluixos i aprendre coses noves cada dia i per tota l'ajuda i facilitats. En definitiva, a tot el departament de TI de Fira de Barcelona. Sous els millors!.

Abstract

Aquest projecte consta de tres parts principals:
Estudi teòric dels nous marcs de treball emergents com són les metodologies Agile, DevOps, CI/CD, testing i versionat així com l'impacte econòmic i de recursos en un entorn de negoci real.
Creació d'una infraestructura Cloud per a un desplegament continu, que utilitza una varietat de serveis proporcionats per aquest nou paradigma.
Creació d'una aplicació consistent en una api i un front-end, que permet posar en pràctica els resultats de la infraestructura.
L'objectiu final del projecte és el de l'explotació d'aquestes noves tecnologies donant molta importància a la seguretat de l'entorn, per comprovar la seva viabilitat en un entorn productiu d'empresa.

Paraules clau: CI/CD, integració continua, desplegament continu, DevOps, Agile, Cloud.

Abstract (english version)

This project consists of three main parts:
Theoretical study of emerging frameworks such as Agile, DevOps, CI/CD methodologies, testing and versioning as well as the economic and resource impact in a real business environment.
Creating a Cloud infrastructure for continuous deployment, which uses many of the services provided by this new paradigm.
Creation of an application consisting of an API and a front-end, which allows to implement the results of the infrastructure.
The main purpose of the project is to bring these new technologies to the extreme, taking also into account the cybersecurity, to check its viability in a productive business environment.

Keywords: CI/CD, continuous integration, continuous deployment, DevOps, Agile, Cloud.

Notacions i Convencions

El codi font es presenta amb un fons negre i colors, copia de l'estil de l'IntelliJ d'IDEA, amb tipografia Courier New, mida 8. A vegades s'utilitza [...] per denotar que existeix més codi però que no es rellevant en l'explicació.
Labels en groc, textos en verd, paraules reservades taronges, variables i enumeracions en lila, etc.
Exemple:

```
@Entity
@Table(name = "events")
@EntityListeners(AuditingEntityListener.class)
public class Event {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "description", nullable = false)
    private String description;
    @Column(name = "start_date", nullable = false)
    @JsonFormat(pattern="yyyy-MM-dd'T'HH:mm:ss")
    private Date startDate;
    @Column(name = "end_date", nullable = false)
    @JsonFormat(pattern="yyyy-MM-dd'T'HH:mm:ss")
    private Date endDate;

[…]
}
```

Les comandes en terminals, tant powershell, com cmd, com consoles de Linux, es presenten amb fons negre i lletra blanca, tipografia Courier New amb mida 9.
Exemple:

```
npm install -g @angular/cli
```

Totes les imatges porten un peu d'imatge amb una numeració i un descriptiu centrat.
Les imatges porten un marc negre per delimitar la mida d'aquesta i no ser confosa pel fons blanc del document.

S'utilitza negreta en les enumeracions per distingir l'element de l'explicació a continuació.
Exemple:

- **Linux**. Sistema operatiu nascut en....

S'utilitza negreta en els paràgrafs d'explicacions per a remarcar algun terme especialment.

S'utilitza cursiva en els paràgrafs d'explicacions per a denotar algun terme o per "sortir-nos" de la quarta paret, per exemple quan s'explica que algun punt queda fora de l'abast del projecte.
Exemple:
El desenvolupament del front-end *surt for de l'abast del projecte.*

Índex

Figures i taules

Índex de figures

Índex de taules

1. Introducció/Prefaci

1.1. Integració, lliurament i desplegament continu

Les metodologies Agile ja són un fet en el nostre país. Amb aquestes metodologies i marcs de treball s'arrosseguen unes formes de treballar diferents a les que estem acostumats.
Vivim en una època de continus canvis. L'evolució tecnològica i la transformació digital continuen fent que les empreses creïn entorns digitals progressivament més sofisticats i complexes que fan que la gestió TI es converteixi en un desafiament cada vegada major. ITIL, referent de les bones practiques en els departaments de TI, va publicar el 28 de febrer de 2019 una nova versió (v4). En aquesta nova versió es desprèn de l'excessiva rigidesa. En aquesta nova versió es centra en la creació de valor afegit i en el lliurament ràpid de serveis d'alta qualitat. Ara ITIL abraça molts dels valors bàsics trobats en Agile, Lean i DevOps.

Els departaments de TI han de poder donar respostes a l'evolució tecnològica, han de ser competitius. El mercat (i el negoci) demana múltiples lliuraments en curts períodes de temps i amb garanties. DevOps es una resposta a aquestes necessitats, assegura el llançament en terminis raonables, permetent que el codi estigui preparat per a producció i donant valor al client.
Històricament els projectes es desenvolupaven, es realitzaven els tests i les proves i es lliuraven, ara es vol el mateix en petites fites que donin valor, no es poden utilitzar els mètodes anteriors, els fluxos de treball eren molt rígids i això faria que perdessin agilitat en els lliuraments.
Una de les respostes tecnològiques que trenca amb els fluxos unidireccionals tradicionals, és la integració continua.

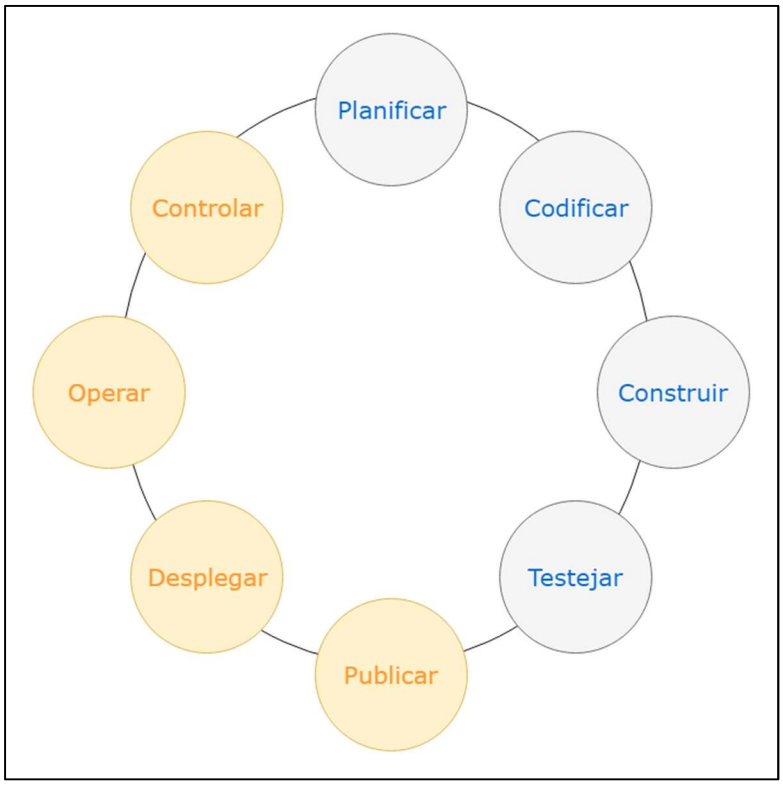


Figura 1: Cicle de vida iteratiu de la CI/CD

Integració continua

La integració continua es tracta d'una practica amb la qual els desenvolupadors poden combinar els canvis realitzats en el codi d'un projecte de manera periòdica i en un repositori central per a després realitzar proves, descobrir errades i corregir possibles errors en el menor temps possible.

Es poden destacar 5 de les principals avantatges de la integració continua.

1. **Reduir riscos i temps**. La integració continua de codi evita els problemes de les integracions a la finalització de projectes o d'evolutius grans, on s'havia d'integrar una gran quantitat de canvis. Al integrar contínuament es disminueixen la quantitat de canvis, el risc de que surtin errors i el temps per a solucionar-los.
2. **Reduir processos repetitius manuals**. Les instal·lacions manuals repetitives són una pèrdua de temps i una font d'errors.
3. **Creació d'una versió de software mitjançant un procés conegut, confiable, provat, versionat i repetible**. Un procés integrat el podem executar a qualsevol lloc, moment i amb menor dependència de persones.
4. **Millorar la visibilitat de l'estat del projecte.** Si es te un procés i un entorn d'integració continua, es te un procés automàtic d'integració, compilat, testeig, mesurament de la qualitat, etc. Com que aquest procés es llança freqüentment, sabem amb la mateixa freqüència l'estat del projecte i no a la finalització del projecte que ja es massa tard per fer alguna cosa. Amb analítiques de la integració continua es poden obtenir estadístiques i tendències de l'estat del desenvolupament.
5. **Aconseguir una major auto-confiança i seguretat en l'equip de desenvolupament.** Un equip que utilitza integració continua aconsegueix una major confiança interna al comprovar constantment l'estat del projecte. No només es confiança de l'equip de desenvolupament sinó també de l'entorn (usuaris, gestors, etc). Un equip més segur és un equip més motivat i més productiu.

Lliurament continu

El lliurament continu es el pas següent de la integració continua i fa referencia a les practiques dissenyades per a lliurar actualitzacions de software als usuaris sobre una base sòlida i constant assegurant que les aplicacions funcionin segons l'esperat.

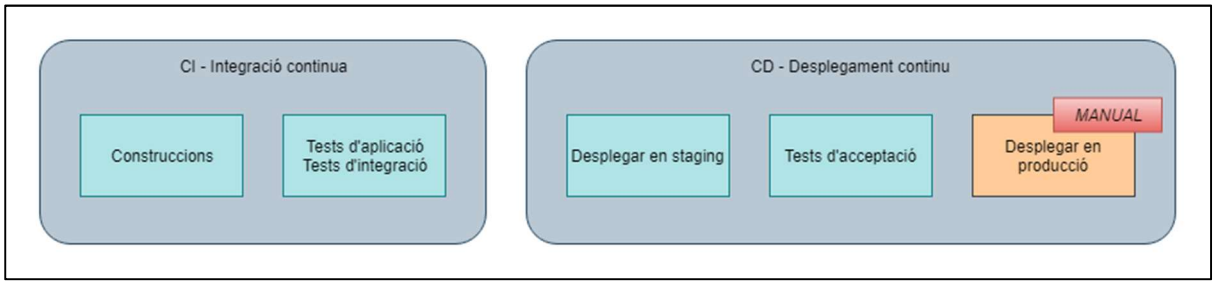


Figura 2: Lliurament continu

Desplegament continu

El desplegament continu és una tècnica més avançada que el lliurament continu. En aquest punt ja no existeix la intervenció humana, l'automatització es l'eix central.

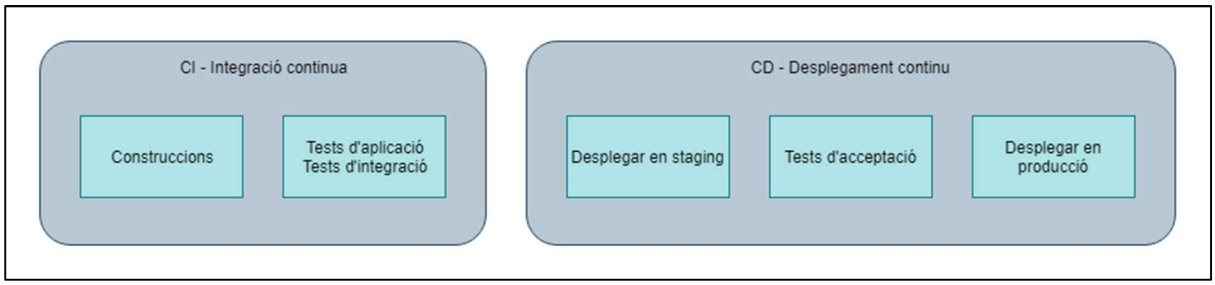


Figura 3: Desplegament continu

Conclusions

L'aplicació d'aquestes tècniques es tradueixen en flexibilitat a l'hora d'incloure canvis en el desenvolupament de software i eficiència en els equips tant de desenvolupament com d'operacions a través de cultures organitzatives com DevOps o frameworks Agile. A la mateixa vegada possibilita el lliurament al client d'un producte de qualitat en el menor temps possible.

1.2. Microservei de reserva d'espais eventuals de reunions

Els microserveis representen un patró, un mode de programar software. Amb els microserveis les aplicacions es divideixen en els seus components més petits, sent independents entre ells.
A diferencia de l'enfocament tradicional i monolític de les aplicacions, en el que tot es compila en una sola peça, els microserveis són independents i funcionen en conjunt per portar a terme les tasques.
Es pot dir que cadascun dels elements o processos d'una tasca és un microservei.
Aquest enfocament d'implementació de components premia el nivell de detall, la senzillesa i la capacitat de compartir un procés similar en varies aplicacions.

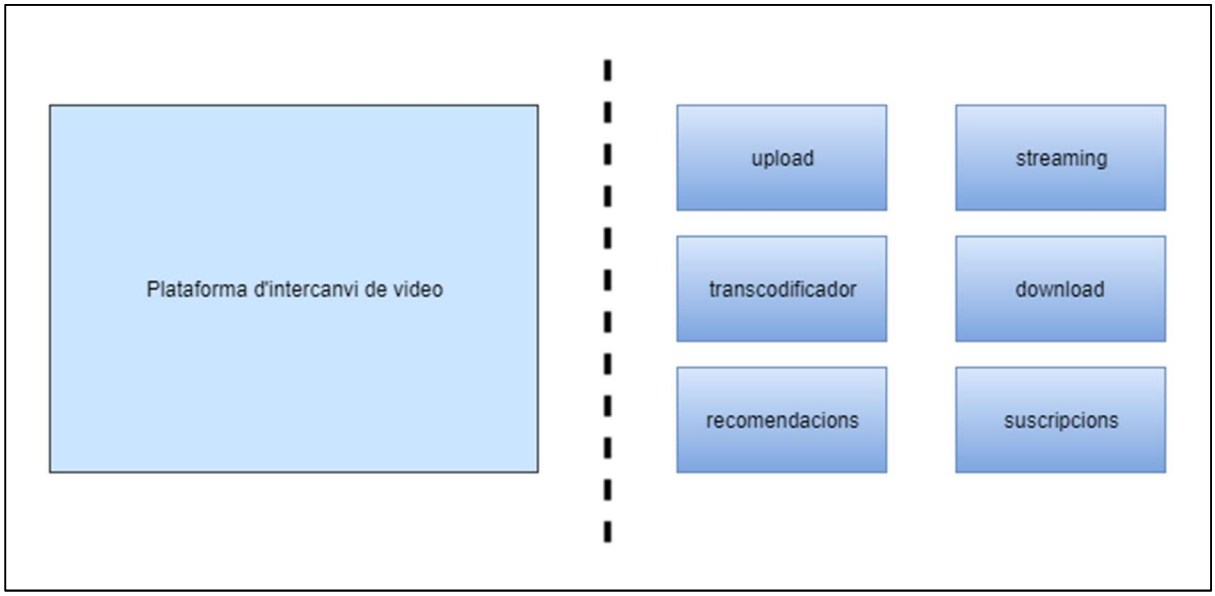


Figura 4: Estructura monolítica vs. microserveis

Fàcilment es poden veure les avantatges potencials dels microserveis:

- **Desenvolupament independent**. Al ser petits e independents el components es poden desenvolupar per equips independents. La quantitat de temps que es necessita per aprendre sobre un component es redueix i és més fàcil desenvolupar noves funcionalitats.

- **Desplegament independent**. Permet que les noves funcionalitats s'alliberin amb major velocitat i menys riscos.
- **Escalabilitat independent**. Cadascun dels components poden escalar independentment un de l'altre.
- **Reusabilitat**. Els components compleixen una funció petita i específica.

Desavantatges dels microserveis

Però no tot son avantatges a l'hora de programar amb un patró de microserveis.

- Major complexitat per als desenvolupadors.
- Major complexitat per als operadors.
- Requereixen un nivell de perícia molt alt.
- Complexitat en el versionat.
- Transaccions distribuïdes.
- Poden ser monolítics disfressats.

Quan i per que escollir els microserveis

Els microserveis no son la resposta absoluta a la construcció d'una arquitectura, hem de tenir en compte el nostre estat actual i les nostres necessitats en la decisió de l'ús de microserveis. Com ja s'ha descrit anteriorment, tenen moltes avantatges i desavantatges.

Hi ha diferents consideracions a tenir en compte per a decidir-se per un desenvolupament de microserveis:

1. Mida de l'equip
Si la mida de l'equip no és gran, els microserveis poden arribar a ser una solució a un problema que no existeix ja que els reptes es poden gestionar amb una bona comunicació.
Si en canvi la mida de l'equip és gran o existeixen varis equips, imposar límits mitjançant la separació de components en serveis aïllats poden ajudar.

2. Estat
Si es treballa sense estat, es pot considerar per prescindir dels microserveis i optar per la via sense servidor.
Si es treballa amb estat, es pot considerar treballar amb microserveis però tenint en compte que no seran fàcils d'implementar ni gestionar.

3. Consumidors
Si només existeix un únic consumidor (una única aplicació) els microserveis poden ser vàlids però potser que només amb una única funcionalitat s'estén actualitzant molts microserveis.
Si existeixen varis consumidors, els microserveis poden ser un patró molt eficient que poden incorporar ràpidament noves funcionalitats o nous consumidors ràpidament.

4. Dependències

Si es tenen dependències monolítiques els serveis escalables de manera independent, no són probablement una avantatge ja que depenen del rendiment de les dependències.
Si no es tenen las limitacions de dependències monolítiques, es pot aconseguir un alt grau d'independència per a realitzar un escalat efectiu dels microserveis.

5. Experiència

Si es compta amb experts amb contenidors, orquestració, DevOps, etc, es te la destresa necessària per a gestionar les complexitats que poden sorgir, llavors es poden capitalitzar totes les avantatges.
Si no es compta amb l'experiència necessària, saltar als microserveis pot ser molt ambiciós.

2. Descripció/Definició/Hipòtesi

Es vol enfocar aquest treball final en dues columnes troncals però molt relacionades entre elles. Es vol realitzar un estudi i una implantació d'una arquitectura per a la realització d'un lliurament continu i un servei basat en microserveis que s'integri en una arquitectura real i que doni serveis a varies aplicacions en productiu.

2.1. Infraestructura per a un lliurament continu

El propòsit d'aquesta primera part es el muntatge des de cero d'un entorn per al lliurament continu. Es pretén crear un entorn totalment funcional amb la finalitat concreta de portar-lo al mon real. Per aquest motiu, es realitzarà amb maquines dockeritzades que es desplegaran sobre una arquitectura cloud pública (Amazon Web Services).
Com a eines necessàries mínimes necessitem una eina per a gestionar el versionat del software, una eina per a l'orquestració de la integració continua, una eina per a l'avaluació del codi font.
Des de l'eina d'orquestració agafaren el codi font, es compilarà, es llençaran els tests, s'avaluarà el codi i si tot es correcte, es desplegarà automàticament en entorn productiu.

2.2. Microservei de reserva d'espais eventuals de reunions

Aquesta segona part surt d'una necessitat en un entorn productiu.
Des de diferents plataformes es poden reservar uns espais durant la durada d'un esdeveniment. Ara mateix, aquesta reserva dispara un e-mail i el control es fa manualment mitjançant un Excel.
La resposta a la proposta de reserva es un altre e-mail amb una acceptació o una disculpa si no es possible, o be perquè l'espai ja ha estat reservat o be perquè no queden espais de mitja jornada o jornada sencera o altres raons.
Això te problemes actuals evidents, errors humans, lentitud, dret de replica als correus, etc.
L'objectiu és automatitzar aquest procés, per això es planteja la realització d'un microserveis.
Després d'avaluar les avantatges i desavantatges dels microserveis, per les raons de desenvolupament independent, desplegament independent, escalabilitat independent (serà cridada de varis llocs, el servei necessitarà ser escalable) i per la reusabilitat (serà cridada per varies aplicacions). Totes aquestes avantatges indiquen que el microserveis es el més òptim, per sobre de la manca d'experiència en el desenvolupament de microserveis i de la major complexitat.

Per aquest TFM, es desplegarà mitjançant la infraestructura mencionada anteriorment en el cloud d'AWS i es cridarà externament amb mètodes manuals. Les dades es guardaran en una base de dades relacional.

El desenvolupament es realitzarà mitjançant la pràctica TDD (Test-driven Development) que ella mateixa involucra la pràctica Test First Development i el Refactoring.
Així mateix, es desenvoluparà seguint bones practiques del codi net i sota els tres factors essencials que ***Robert C. Martin*** sosté en el seu llibre *Codi Net: Manual d'estil per al desenvolupament àgil de software*:

1. **Llegibilitat de la font**. Capacitat del codi d'autoexplicar-se sense la necessitat de recorre a res més que el codi.
2. **Organització dels elements**. Estructuració correcta dels elements en un sistema.
3. **Certesa de funcionament**. S'obté mitjançant la implementació de diferents proves que verifiquin el nostre sistema. "*Els veritables enginyers de software proven el seu codi i la millor manera de garantir això és implementant TDD*".

2.3. Front-end de reserva d'espais eventuals de reunions

Surt fora de l'abast del projecte una part front-end, però ha semblat una bona idea aportar una part visual a l'aplicació encara que incompleta; però sobretot ens servirà per a completar la part de la integració continua i el desplegament continu.
Aquest és el principal objectiu d'aquesta part front-end, el seu desenvolupament no és llavors un objectiu del treball ni es posarà com a objectius el seu desenvolupament, si però serà un objectiu la integració d'aquest amb la infraestructura de la integració continua i el lliurament continu.
A més, es realitzà un login en la part front-end i s'autenticarà l'API del servei de reserves.

3. Objectius

3.1. Principals

Els objectius d'aquest treball es oferir un entorn totalment funcional per a CI/CD i un software per a realitzar reserves d'espais en esdeveniments, tot en un entorn Cloud públic utilitzant els serveis que ofereix.

- OBJP1 – Conèixer i operar amb garanties en un Cloud públic com és Amazon Web Services (AWS).
- OBJP2 – Creació d'un microservei per a reserves de sales eventuals.
- OBJP3 – Creació d'un entorn complert per a la integració i el desplegament continu.
- OBJP4 – Configurar amb garanties i seguretat la infraestructura necessària.
- OBJP5 – Creació dels jobs necessaris en Jenkins per a la integració i desplegament continu del servei i del front-end.
- OBJP6 – Brindar un marc teòric per a conèixer i integrar una CI/CD en un negoci.
- OBJP7 – Configurar escalat horitzontal en el microservei.
- OBJP8 – Mètriques dels servidors EC2 i de la base de dades RDS mitjançant Cloudwatch i amb lambda notificacions SNS.

3.2. Secundaris

- OBJS1 – Microservei compleixi amb les bones pràctiques del codi net.
- OBJS2 – Snapshots dels servidors de Jenkins i GitLab mitjançant lambda

3.3. Extres

- OBJEX1 – Petit front-end per tal de desplegar amb CI/CD.

4. Marc teòric/Escenari

En quantitat d'empreses es defineixen unes finestres per pujada a entorns productius i inclús a entorns d'staging o entorns QA. Aquestes finestres poden ser des d'un cop a la setmana fins i tot finestres molt més amples.
Això te una raó de ser, proves manuals, pujades manuals, alt risc per no tenir automatitzats els processos.
Els inconvenients són obvis, molt baix time-to-market, molt risc, incertesa, desenvolupadors amb incomoditats en el seu treball, etc.

Problema: Lentitud i incertesa en les posades a producció.
Objectiu: Desplegaments amb qualitat, seguretat i rapidesa a l'entorn productiu.
Pregunta: Com podem aconseguir aquest objectiu sense un increment de desenvolupadors, gent de QA i operadors de sistemes i sense que ens suposi un projecte amb un elevat cost?
Sub-preguntes:

1. Existeix una metodologia o un marc de treball que ens permeti arribar als nostres objectius i que estigui provat que funciona?
2. Existeixen eines que ens ajudin a arribar als nostres objectius?

Es tracta de canviar els processos de l'empresa a un model DevOps (teòric i pràctic) impulsat pel departament de TI gracies a la implementació d'una CI/CD.

DevOps
En 2008, Andrew Shafer va protagonitzar una conferencia amb només un assistent, Patrick Debois. El mateix Shafer va cancel·lar la seva conferencia, però això no va impedir que Debois el trobés. Van formar l'anomenat *Agile Systems Administration Group*.
En 2009, John Allspaw i Paul Hammond van protagonitzar la famosa conferencia "*10 Deploy a Day: Dev and Ops Cooperation al Flickr*"[(1)] . Feia referencia als malentesos entre l'equip d'operacions i l'equip de desenvolupament basat en el tradicional pensament de que els desenvolupadors han d'afegir noves funcionalitats i els operaris de sistemes han de mantenir els sistemes estables. Proposava solucions sota la premissa de que el negoci requereix un canvi i l'objectiu de disminuir els riscos del canvis mitjançant eines i cultura[(1)]:

- Eines:
    - Infraestructura automatitzada.
    - Control de versions compartit.
    - Construcció i desplegament amb un sol pas.
    - Utilitzar branques i flags en el codi.
    - Mètriques compartides.
    - IRC and IM robots. (ara podríem parlar d'Slack per exemple).
- Cultura:
    - Respecte.
    - Confiar.
    - Bona actitud davant les fallides.
    - Evitar les culpes.

Patrick Debois va lamentar no poder-hi ser en aquesta conferencia.

A finals del mateix any, 2009, Patrick Debois va convocar una conferencia anomenada *DevOpsDays*. Va ser la primera vegada que es va utilitzar el terme DevOps (3 primeres lletres de Development i 3 primeres lletres d'Operations).
Al finalitzar la conferencia, les discussions es van traslladar a Twitter i Debois va crear un hashtag que ja ha passat a l'historia de la informàtica: *#DevOps*.

Que entenem actualment per DevOps?
Sembla que els inicis del moviment DevOps ens respon a les dues preguntes, existeix una cultura i existeixen unes eines per ajudar-nos al nostre objectiu.
Bass, Weber i Zhu van proposar com a definició: DevOps és un conjunt de pràctiques destinades a reduir el temps entre el compromís d'un canvi en un sistema i el canvi que es col·loca en la producció normal, al temps que garanteix una alta qualitat.

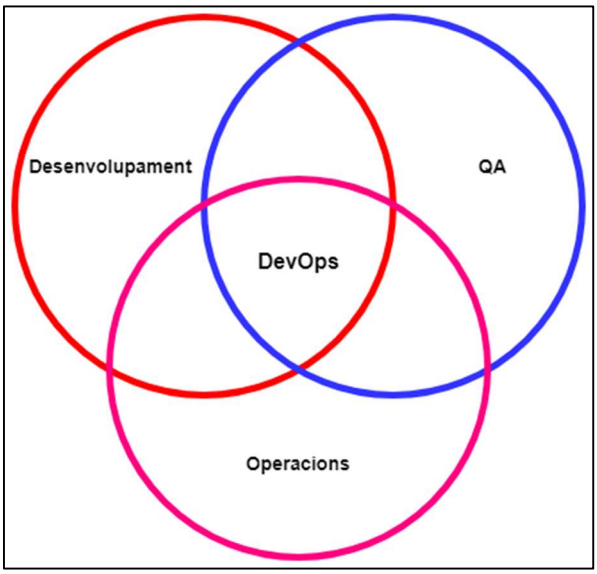


Figura 5: DevOps

Eines
Com que DevOps pretén ser un mode de treball interfuncional en lloc d'una sola eina DevOps, existeixen conjunts de múltiples eines. S'espera que les eines de DevOps encaixin en una o més de les següents categories que reflecteixen els aspectes clau del procés de desenvolupament i lliurament:

- Codi: desenvolupament i revisió de codi, eines d'administració de codi, fusió de codi.
- Construcció: eines d'integració continua, estat de la compilació.
- Prova: eines de proves continues.
- Paquet: repositori d'artefactes.
- Llançament: gestió de canvis, aprovacions de versions, automatització de versions.
- Configurar: configuració i gestió de la infraestructura (infraestructura com a codi).
- Monitor: monitoratge del rendiment de les aplicacions, experiència de l'usuari final.

Objectius del DevOps
Els objectius del DevOps abasten tot el procés de lliurament.

- Freqüència millorada en els desplegaments.
- Time-to-market més ràpid.
- Baixa taxa d'errors en noves aplicacions.
- Temps de lliurament més curt entre fixs.
- Temps de recuperació més ràpid (en cas de fallada d'una nova versió).

DevOps te com objectiu maximitzar la predicibilitat, eficiència, seguretat i manteniment dels processos operatius.

La integració del DevOps s'enfoca en el lliurament de productes, proves continues, proves de qualitat, desenvolupament de característiques i versions de manteniment per millorar la confiabilitat i la seguretat i proporcionar cicles de desenvolupament i implementació més ràpids.

Relació DevOps i Agile

La necessitat de DevOps va sorgir del creixent èxit del desenvolupament de software àgil, ja que això va portar a les empreses a voler llançar el software més ràpid i amb major freqüència. A mesura que es tractava de superar la tensió que això suposava amb els processos de gestió de versions, havien d'adoptar patrons com l'automatització del llançament d'aplicacions, les eines d'integració continua i l'entrega continua.

Entrega Continua

L'entrega continua i DevOps tenen objectius comuns i amb freqüència s'usen en comú però són dos conceptes diferents[(2)].

L'entrega continua es centra en l'automatització dels processos d'entrega de software, DevOps va més enllà i també es centra en el canvi de l'organització per admetre una gran col·laboració entre les moltes funcions involucrades.[(3)]

DevOps i entrega continua comparteixen una base comú en mètodes àgils: canvis petits i freqüents amb valor focalitzat en el client final.

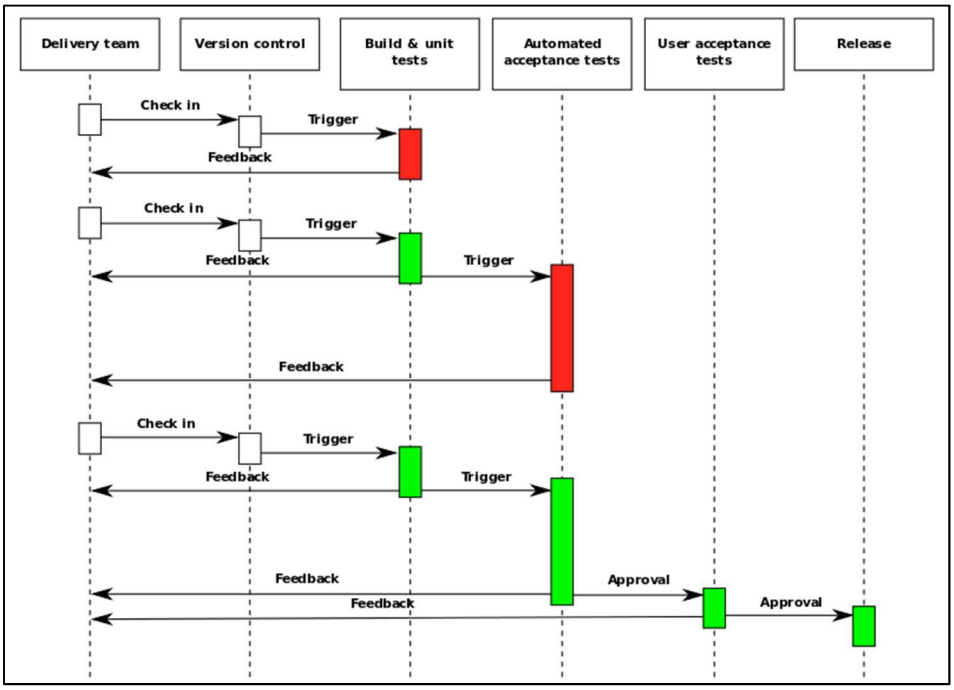


Figura 6: Continuous Delivery[(4)]

Microserveis

Aquest tipus d'enfocament permet a les empreses digitals brindar alta disponibilitat i estabilitat a les seves aplicacions; això es deu a que totes les parts de les aplicacions (base de dades, backend, front-end, etc.) són independents i, si una falla, no implica que tota l'aplicació caigui, en lloc d'això, les altres parts continuen treballant fins que es restaura el component afectat. Els enginyers DevOps requereixen de microserveis per a optimitzar els seus desenvolupaments i deixar endarrere arquitectures monolítiques.

(1) https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr

(2) Hammond, Jeffrey (2011) "The Relationship between DevOps and Continous Delivery"

(3) Humble, Jez, Farley, David (2011). Continuous Delivery: reliable software releases through build, test, and deployment automation.

(4) By Grégoire Détrez, original by Jez Humble - This file was derived from: Continuous Delivery process diagram.png, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=43977816

5. Continguts

5.1. Requisits funcionals Reserva d'espais eventuals de reunions

- RF1 – Gestió d' esdeveniments.
    - RF1.1 – Creació d'un nou esdeveniment delimitat amb dates d'inici i fi.
    - RF1.2 – Assignació de sales a un esdeveniment. Les sales poden ser reutilitzades entre edicions d' esdeveniments o entre esdeveniments diferents.
- RF2 – Gestió de sales.
    - RF2.1 – Creació d'una nova sala.
    - RF2.2 – Llistat de sales per esdeveniment.
    - RF2.3 – Assignació d'equipament a una sala.
- RF3 – Gestió d'equipament.
    - RF3.1 – Creació d'un nou equipament.
    - RF3.2 – Llistat de tots els equipaments.
    - RF3.3 – Llistat de l'equipament assignat a una sala.
- RF4 – Gestió de tipologies de reserva.
    - RF4.1 – Creació d'una nova tipologia de reserva.
- RF5 – Gestió de reserves.
    - RF5.1 – Creació d'una nova reserva.
    - RF5.2 – Llistat de reserves per esdeveniment.

5.2. Requisits no funcionals Reserva d'espais eventuals de reunions

- RNF1 – Alta disponibilitat, escalat horitzontal.
- RNF2 – APIs autenticades.
- RNF3 – Ha d'admetre gran quantitat de peticions recurrents.
- RNF4 – El software ha de ser robust i fiable.
- RNF5 – Login d'usuaris.
- RNF6 – El software ha de ser segur.
- RNF7 – El software s'ha de desplegar en serveis Cloud.

5.3. Model de dades Reserva d'espais eventuals de reunions

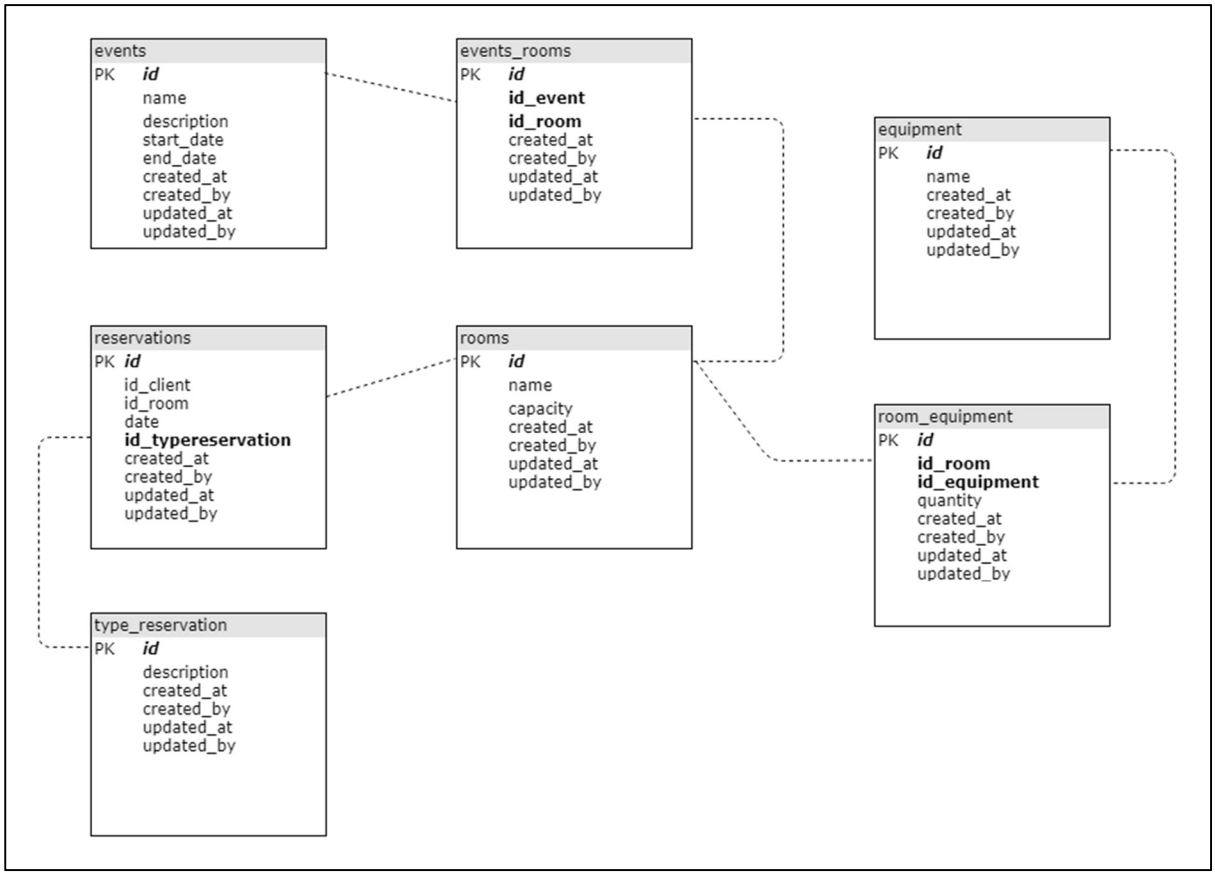


Figura 7: Diagrama model de dades

5.4. Requisits Infraestructura CI/CD

- R1 – Automatització. Es tracta de treballar l'estructura i que per als desenvolupadors li sigui transparent i totalment automatitzat.
- R2 – Segur. La seguretat és una part molt important.
- R3 – Fàcilment ampliable. Es realitzaran les integracions necessàries però es vol que sigui fàcil ampliar aquestes integracions.
- R4 – Backups. Es realitzaran snapshots del servidor de GitLab i del servidor de Jenkins mitjançant AWS lambda.
- R5 – Obtenció de resultats. Mitjançant CloudWatch i notificacions SNS de les lambda.

6. Metodologia

La metodologia escollida per a la realització d'aquest Treball Final de Màster és el model clàssic en cascada. Això pot semblar una mica contradictori si pensem en els objectius DevOps d'aquest treball, però per varis motius es té la certesa que un model clàssic en aquest projecte és més adient:

- No es poden efectuar cerimònies per compartir l'estat actual del projecte.
- Els requisits i les fites estan ben marcades des d'un bon principi per l'entrega de les PACs, no pot haver-hi canvi de requeriments.
- En la primera entrega es demana un anàlisi del producte.
- Es demana una entrega final amb el producte acabat.

Aquestes premisses no neguen la possibilitat de treballar amb un marc Agile, però penso que seria massa forçat.

6.1. Model en cascada

El model en cascada ordena les etapes del procés per al desenvolupament de software, de tal manera que l'inici de cada etapa ha d'esperar a la finalització de l'etapa anterior. A la finalització de cada etapa el model està dissenyat per a portar a terme una revisió final, que s'encarrega de determinar si el projecte està enllestit per a avançar a la següent etapa.

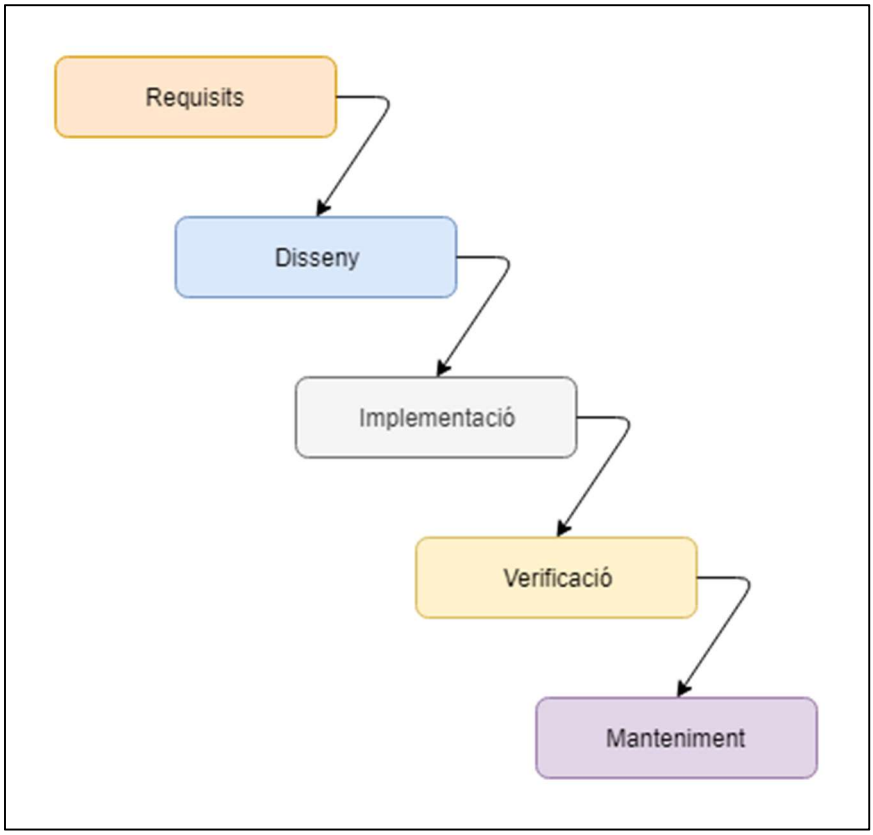


Figura 8: Model en cascada

Anàlisi de requisits del software. En aquesta fase s'analitzen les necessitats dels usuaris finals del software per a determinar quins objectius ha de cobrir.

En aquesta etapa s'ha de consensuar tot el que es requereix del sistema i serà allò el que seguirà en les fases posteriors, no poden requerir-se nous resultats a meitat del procés d'elaboració del software.

Disseny del sistema. Es descomponen i s'organitza el sistema en elements que pugin elaborar-se per separat, aprofitant les avantatges del desenvolupament en equip.
Es convenient distingir entre disseny d'alt nivell o arquitectònic i disseny detallat. El primer te com a objectiu definir l'estructura de la solució. El segon defineix els algoritmes empleats i l'organització del codi per a començar la implementació.

Disseny del programa. Es la fase on es realitzen els algoritmes necessaris per al compliment dels requeriments de l'usuari així com també els anàlisis necessaris per a saber quines eines utilitzar en l'etapa de codificació.

Codificació. És la fase on s'implementa el codi font, fent us de prototips així com de proves i assajos per a corregir errors. Depenent del llenguatge de programació i la seva versió, es creen les biblioteques i components reutilitzables dins del mateix projecte per a fer que la programació sigui un procés molt més ràpid.

Proves. Els elements ja programats s'acoblen per a composar el sistema i es comprova que funcioni correctament. Es busquen sistemàticament i es corregeixen tots els errors abans de ser entregat a l'usuari.

Validació i verificació del producte de software. És la fase on l'usuari final o el client executa el sistema i s'assegura que cobreix les seves necessitats.

Manteniment. Una de les etapes més crítiques. Pot ser que al fer ús l'usuari final pot ser que no compleixi amb totes les expectatives.

6.2. Avantatges del model en cascada per al projecte

- Es un model fàcil d'implementar i entendre.
- Orientat a documents.
- És un model conegut i utilitzat amb freqüència.
- Promou una metodologia de treball efectiva: definir abans que dissenyar, dissenyar abans que codificar.

7. Arquitectura de l'aplicació/sistema/servei

7.1. Arquitectura Reserva d'espais eventuals de reunions

Els microserveis representen un estil d'arquitectura i una manera de programar software. Amb els microserveis, les aplicacions es divideixen en els seus components més petits, independents entre ells i que funcionen en conjunt.
Aquest enfocament privilegia el nivell de detall, la senzillesa i la capacitat de compartir un procés en diverses aplicacions. És un component fonamental de l'optimització del desenvolupament de software cap a un model natiu del Cloud.

El model de desenvolupament és un model per capes. L'objectiu principal que vol aconseguir aquest model, és el desacoplament entre les parts que composen el sistema, fent que les parts o capes compleixen una missió simple i abstracta.
Algunes de les avantatges del model per capes són:

- **Alta escalabilitat**: És a dir, en cas de que les necessitats augmentin podran ampliar-se amb facilitat.
- **Alta mantenibilitat**: L'esforç requerit per realitzar el manteniment del codi serà molt menor.

Les 3 capes que es proposen en l'arquitectura de 3 capes són:

- **Capa de presentació**. Es el que veu l'usuari. Mitjançant aquesta capa, l'usuari realitzarà les accions que aniran a la capa de negoci.
- **Capa de negoci (lògica de l'aplicació)**. Aquesta capa gestiona la lògica de l'aplicació. És a on s'indica que es fan amb les dades.
- **Capa de dades**. En aquesta capa es creen, recuperen i actualitzen les dades.

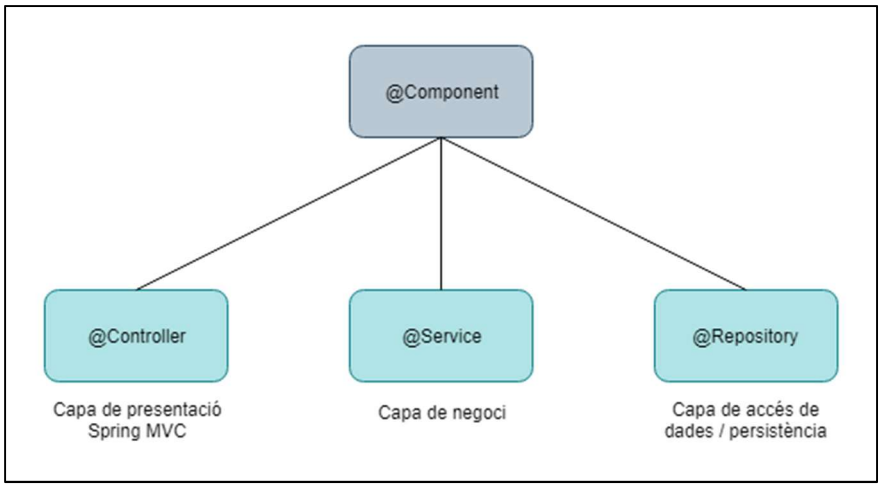


Figura 9: Arquitectura de 3 capes amb Spring MVC

Llibreries utilitzades per al desenvolupament

L'aplicació s'ha desenvolupat en la versió 8 del llenguatge de programació Java.

Spring Framework.

Spring ens permet desenvolupar aplicacions de manera més rapida, eficaç i amb menys codi, sense caure en tasques repetitives i estalviant-nos línies de codi. Spring és un framework molt extens, però podem destacar les funcions bàsiques per les quals és molt conegut; la inversió de control i la injecció de dependència. Una de les característiques principals d'Spring és l'Spring container, que és un contenidor d'inversió de codi i la seva tasca és realitzar injecció de dependències sense la necessitat de realitzar la definició en Java. S'encarrega de crear els objectes, configurar-los i administrar el seu cicle de vida complet, des de la creació fins a la seva destrucció (inversió de control).

Spring Boot.

Tradicionalment les aplicacions realitzades amb Spring Framework són construïdes amb 3 pasos:

1.- Realització d'un projecte Maven o Gradle.

2.- Realització de l'aplicació (desenvolupament del codi).

3.- Desplegar en un servidor.

Spring Boot va néixer amb la intenció de simplificar els passos 1 i 3, que son passos més allunyats del desenvolupament i així poder centrar-nos en el pas 2 que és el propi de desenvolupament.
Els projectes Spring Boot compten a una classe amb l'anotació *@SpringBootApplication* que son les encarregades d'arrencar el projecte.

Maven.

L'objectiu de maven és el de simplificar els processos de compilar i generar executables a partir del codi font. Abans de la utilització de maven, es tenia que analitzar quines parts de codi s'havien de compilar, que llibreries utilitzava i a on influir-les, quines dependències de compilació tenia el projecte etc. Maven ens simplifica aquestes tasques, però no només això; podem dir que maven és una eina capaç de gestionar un projecte de software complet, des de l'etapa en que es comprova que el codi es correcte, fins que es desplega l'aplicació, passant per l'execució de proves, generació d'informes i documentació. Les etapes d'un cicle per defecte serien:

- Validació. Validar que el projecte és correcte.
- Compilació.
- Test. Provar el codi font utilitzant un framework per a proves unitàries.
- Empaquetar. Empaquetar el codi i transformar-lo en algun format tipo .jar o .war.
- Proves d'integració. Processar i desplegar el codi en algun entorn on es pugin realitzar les proves d'integració.
- Verificar. Verificar que el codi empaquetat és vàlid i compleix amb els criteris de qualitat.
- Instal·lar. Els codis empaquetats en el repositori maven per usar-los com a dependència d'altres.
- Desplegar. En un entorn.

Per altre costat amb maven es gestionen les dependències entre mòduls i distintes versions de llibreries molt fàcilment. Només s'han d'indicar els mòduls que composen un projecte o quines llibreries utilitza el software que estem desenvolupant en un arxiu de configuració del projecte anomenat pom.xml. Maven a més proporciona un repositori remot (Maven central) on es troben la majoria de llibreries que s'utilitzen en els desenvolupaments.

Primeres línies de l'arxiu pom.xml de l'aplicació:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.8.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.tfm</groupId>
    <artifactId>reservas</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>reservas</name>
    <description>Proyecto para reserva de salas eventuales</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    […]
```

JPA i Spring Data JPA.

JPA ens permet interactuar amb la base de dades relacional mitjançant objectes, a diferencia amb JDBC que ens permet realitzar consultes directes a la base de dades.

JPA no és un framework, és una especificació, és a dir, són una sèrie de regles que s'han de complir per qualsevol proveïdor que desitgi desenvolupar una implementació JPA. Per tant existeixen moltes implementacions de JPA i Spring Data JPA és una d'aquestes.

Spring Data és un framework dins la plataforma Spring que el seu objectiu es simplificar la persistència de dades contra diferents repositoris d'informació.

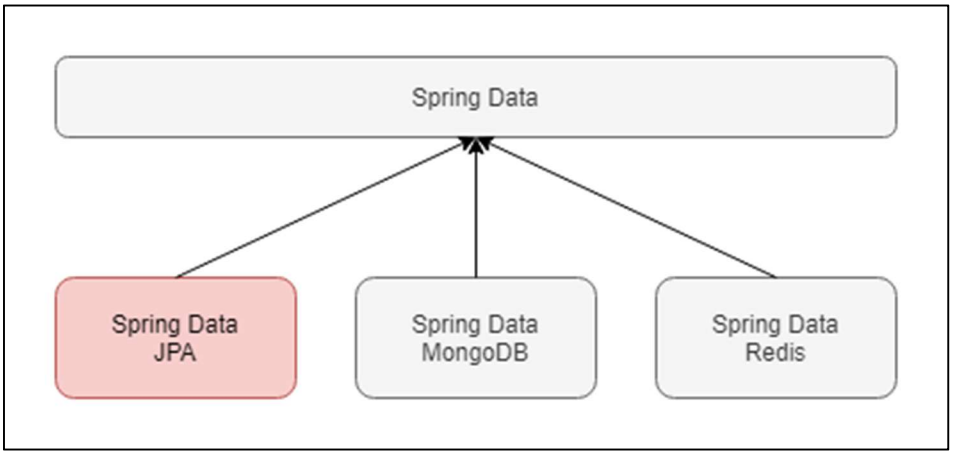


Figura 10: Spring Data

D'aquesta manera en el projecte s'han creat tots els models amb les anotacions JPA i els repositoris que estenen de *JpaRepository*.

Exemple de model:

```
@Entity
@Table(name = "equipment")
@EntityListeners(AuditingEntityListener.class)
public class Equipment {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long id;
    @Column(name = "name", nullable = false)
```

```
    private String name;

    @Column(name = "created_at", nullable = false)
    @CreatedDate
    @JsonFormat(pattern="yyyy-MM-dd'T'HH:mm:ss")
    private Date createdAt;
    @Column(name = "created_by", nullable = false)
    @CreatedBy
    private String createdBy;

[…]
```

Exemple de repositori:

```
package com.tfm.reservas.repository;

import com.tfm.reservas.model.Equipment;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface EquipmentRepository extends JpaRepository<Equipment, Long> {
}
```

Spring test.

Spring MVC test és un framework per a provar classes de tipus Controller. Aquests test es troben en la frontera enter els tests d'integració i els tests unitaris. Amb un test unitari "tradicional" molts aspectes de les especificacions es troben fora de les proves:

- Capçaleres de negociació.
- Codis de resposta.
- Serialització / Deserialització d'objectes a JSON.
- Disponibilitat de capçaleres en les respostes.

Spring Security.

Spring Security és un framework dissenyat per a gestionar els mecanismes de seguretat d'una aplicació com són: autenticació, autorització, protecció, etc.

Les seves característiques principals són:

- Suport complet i extensiu tant per a l'autenticació com per a l'autorització.
- Protecció contra atacs com *session fixation*, *clickjacking*, *cross site request forgery*, etc.
- API d'integració.
- Integració opcional amb Spring MVC.

Angular 8.

Angular és un framework per a aplicacions web que s'utilitza per a crear aplicacions web d'una sola pàgina. Utilitza el patró d'arquitectura Model Vista Controlador (MVC).

Angular es basa en classes tipus "Components", que les seves propietats son les usades per a fer enllaç de les dades (*databinding*). En aquestes classes tenim propietats (variables) i mètodes (funcions a cridar).

Base de dades

El model de dades relacional en gestió de base de dades és avui dia, el model més utilitzat per a la gestió de dades dinàmiques, és a dir, dades subjectes a modificacions i actualitzacions).

El **model relacional** és un model d'organització i gestió de base de dades consistent en l'emmagatzematge de dades composades per files i columnes.

Taula: és el nom que rep cadascuna de les relacions que s'estableix entre les dades emmagatzemades; cada nova relació dona lloc a una tabla. Estan formades per **files**, on es descriuen els elements que configuren una tabla, **columnes** amb els atributs i valors corresponents, i el **domini**, concepte que agrupa a tots els valors que poden figurar en cada columna.
Claus: elements que impedeixen la duplicitat de registres, una de les gran desavantatges que presenten altres models d'organització i gestió de base de dades. Existeixen dos grans grups de claus, les **claus primàries** i les **claus secundaries**.
Claus primàries: són els atributs (columnes) segons el tipus de relació que s'ha definit en una taula.
Claus secundaries: son les claus que es defineixen per a cada una de les claus primàries establertes per als elements o entitats d'una relació.
Restriccions d'identitat: límits i restriccions que s'imposen en les relacions. Concepte vinculat amb les **regles d'integritat** pròpies del model relacional, el compliment de les quals està garantit amb les claus.

MySQL és un sistema open source d'administració de base de dades desenvolupat i suportat por Oracle. MySQL utilitza un model client-servidor. La part del servidor es a on les dades resideixen. Per a poder accedir a ells, s'han de demanar, des d'un client. Mitjançant el llenguatge SQL, el client envia una petició al servidor de la base de dades per a les dades que el client necessita.

Diagrama arquitectura microservei

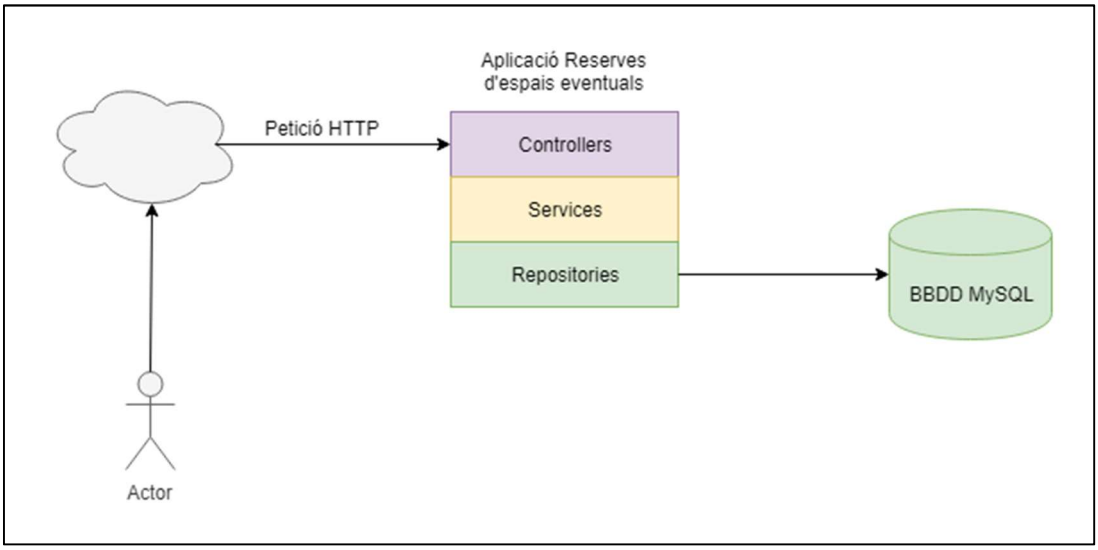


Figura 11: Diagrama arquitectura microservei de Reserva d'espais eventuals de reunions

Diagrama aplicació Angular amb Spring Boot

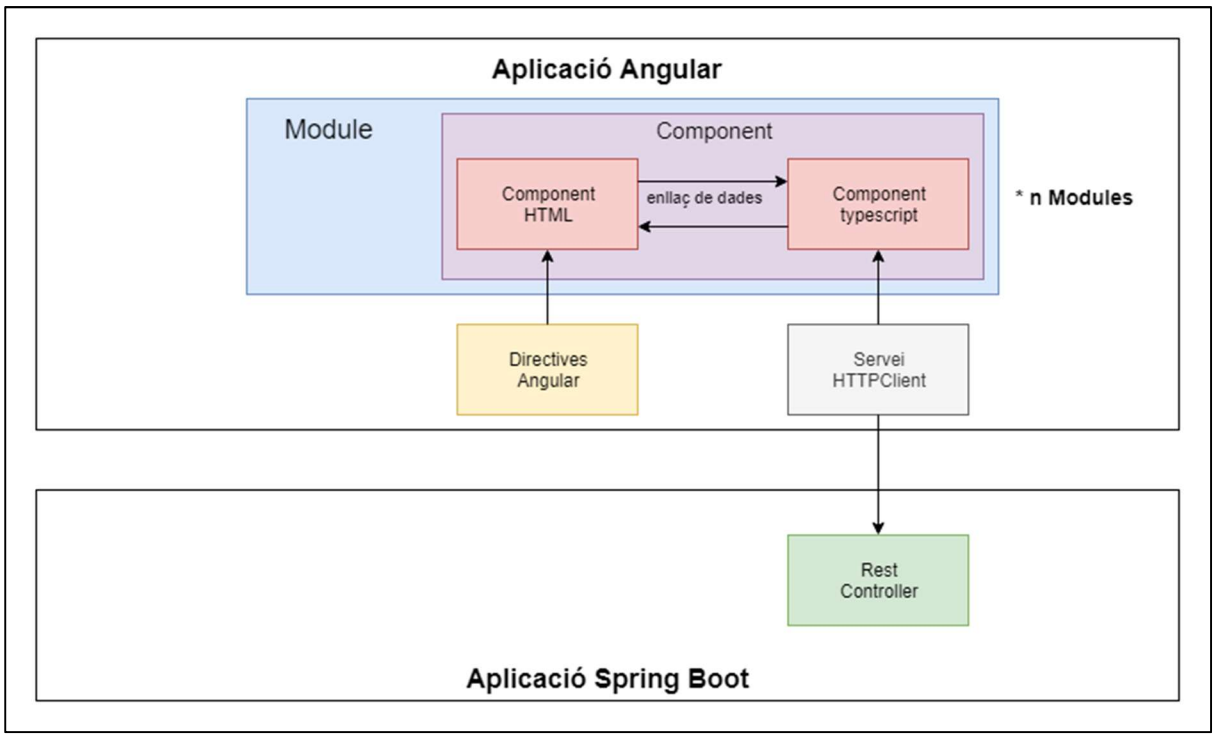


Figura 12: Diagrama aplicació Angular + Spring Boot

7.2. Infraestructura Reserva d'espais eventuals de reunions

Per a la infraestructura per al desplegament de la reserva d'espais eventuals de reunions:

- **Route 53**. Servei de DNS.
- **Application Load Balance**. Equilibrador de càrrega d'aplicacions. Distribueix el tràfic entrant d'aplicacions entre varis destins, instancies EC2 per exemple, en varies zones de disponibilitat.
- **Elastic Container Service (ECS)**. Servei d'organització de contenidors d'alta escalabilitat i rendiment, compatible amb els contenidors Dockers.
- **Elastic Compute Cloud (EC2)**. On es desplegarà el software. Amb alta disponibilitat en diferents zones de disponibilitat i amb auto-escalat horitzontal.
- **Relational Database Service (RDS) MySQL.** És un servei que ofereix AWS. AWS administra la BBDD, copies de seguretat, *fix*, etc. La base de dades que s'opta és MySQL, que és una base de dades relacional de codi obert.

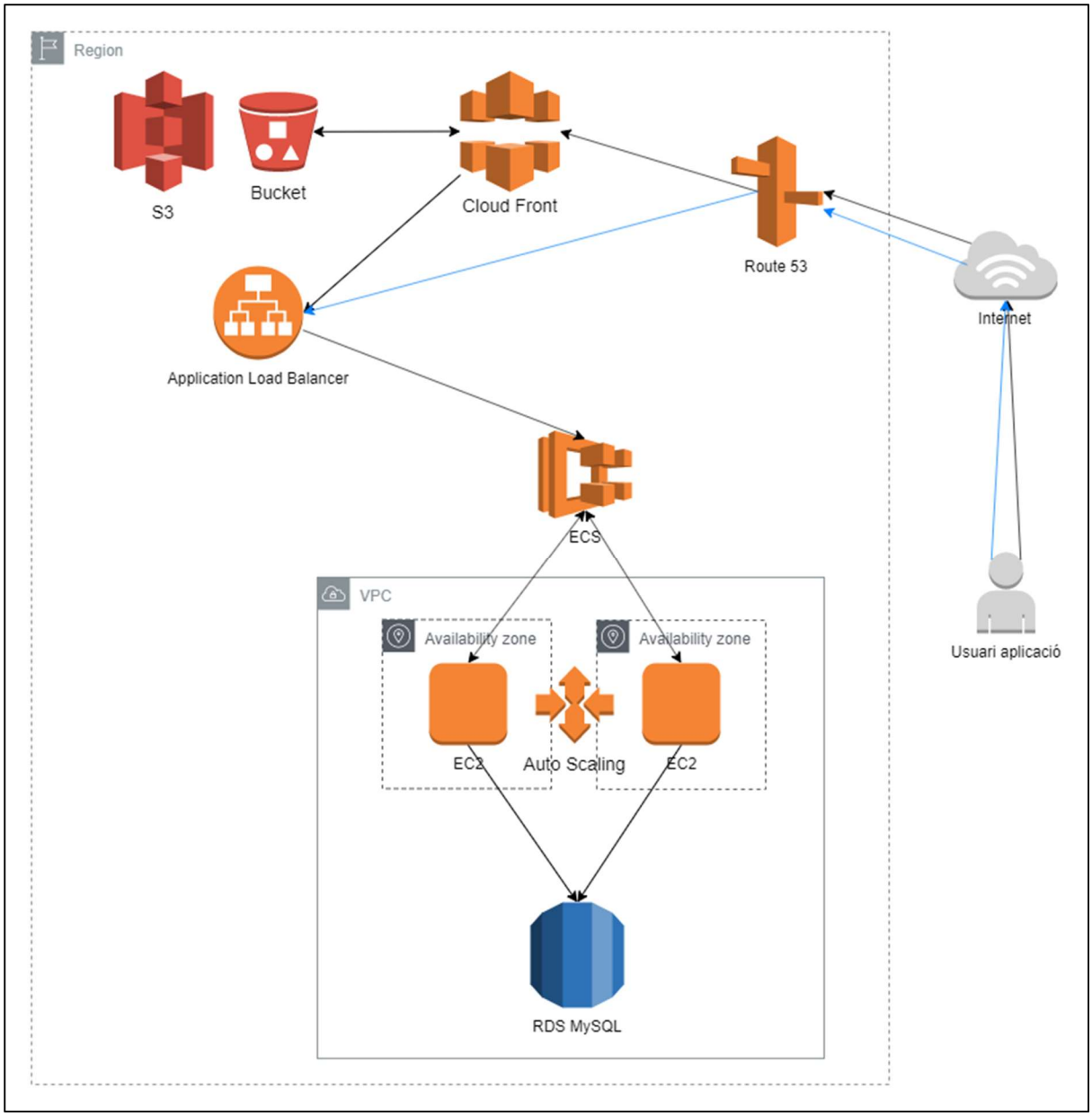


Figura 13: Diagrama infraestructura Reserva d'espais eventuals de reunions

7.3. Arquitectura / Infraestructura CI/CD

Per a la infraestructura per al desplegament de la reserva d'espais eventuals de reunions:

- **Route 53**. Servei de DNS. Resoldrà les peticions i redirigirà a un servei o un altre.
- **Elastic Compute Cloud (EC2)**. És un servei web que proporciona capacitat informàtica en el núvol, de mida modificable. Desplegarem en 3 EC2 Linux diferents, 3 Docker amb diferents software, Jenkins, SonarQube i GitLab.
- **AWS Lambda**. Permet l'execució de codi sense aprovisionar ni administrar servidors (serverless). Farem córrer codi Python en dues Lambdes diferents per a realitzar snapshots de Jenkins i GitLab.
- **Cloudwatch**. És un servei de monitorització i observació. Ofereix dades i informació processable per monitoritzar aplicacions, respondre a canvis de rendiment que afectin a tot el sistema, optimitzar l'ús de recursos i aconseguir una vista unificada de l'estat de les operacions. Monitoritzarem els EC2, crearem alertes i visualitzarem logs.

- **Simple Notification Service (SNS)**. És un servei de missatgeria de publicació/subscripció completament administrat, d'alta disponibilitat, segur i amb durabilitat. L'utilitzarem per a fer arribar missatges en cas d'alertes de Cloudwatch, i en l'execució dels backups.
- **Simple Storage Service (S3).** Servei d'emmagatzematge d'objectes que ofereix escalabilitat, disponibilitat de dades seguretat i gran rendiment. Emmagatzemarem els snapshots creats per les lambdes de Jenkins i GitLab.

GitLab

GitLab és una eina basada en Git. A més del hosting remot per a repositoris, GitLab ofereix una interface web per a controlar el repositori i altres eines. Ofereix la possibilitat d'examinar el codi en qualsevol de les versions, realitzar accions relacionades amb el sistema de repositori como gestionar les "merge requests". La principal diferencia entre GitLab i els seus competidors més directes, és que s'ofereix com a software lliure que es pot instal·lar en qualsevol servidor.

En aquesta infraestructura, s'utilitza GitLab per al versionat del software i és on Jenkins agafarà el codi font per a empaquetar-lo.

Jenkins

Jenkins és un servidor autònom, de codi obert, que es pot utilitzar per automatitzar tot tipus de tasques relacionades amb la integració continua de software; a més, és un producte que pot estendre les seves funcionalitats mitjançant la instal·lació de complements. És altament extensible.

En la nostra infraestructura, Jenkins agafarà el codi font de les aplicacions directament del servidor de GitLab, executarà els tests, empaquetarà, utilitzarà SonarQube per auditoria de codi font i si tot és correcte, el desplegarà mitjançant una Task Definition, en el cas de l'API, o el publicarà en un bucket S3 en el cas del front-end. *Veure apartat 11.9. Jobs de Jenkins per a més informació.*

SonarQube

SonarQube permet realitzar anàlisis estàtics de codi font de forma automàtica, buscant patrons amb errors, males praxis o incidents. A més, realitza un càlcul del deute tècnic. Verifica coses com:

- Detecció de codi duplicat.
- La mancança de proves unitàries, de comentaris.
- Complexitat ciclomàtica, alt acoblament.
- Mida dels mètodes.
- No adequació a estàndards i convencions de codi.
- Vulnerabilitats conegudes de seguretat.

Fem servir el servidor SonarQube integrat amb Jenkins per realitzar anàlisis de codi font abans de realitzar el desplegament.

Diagrama de les eines per a la CI/CD

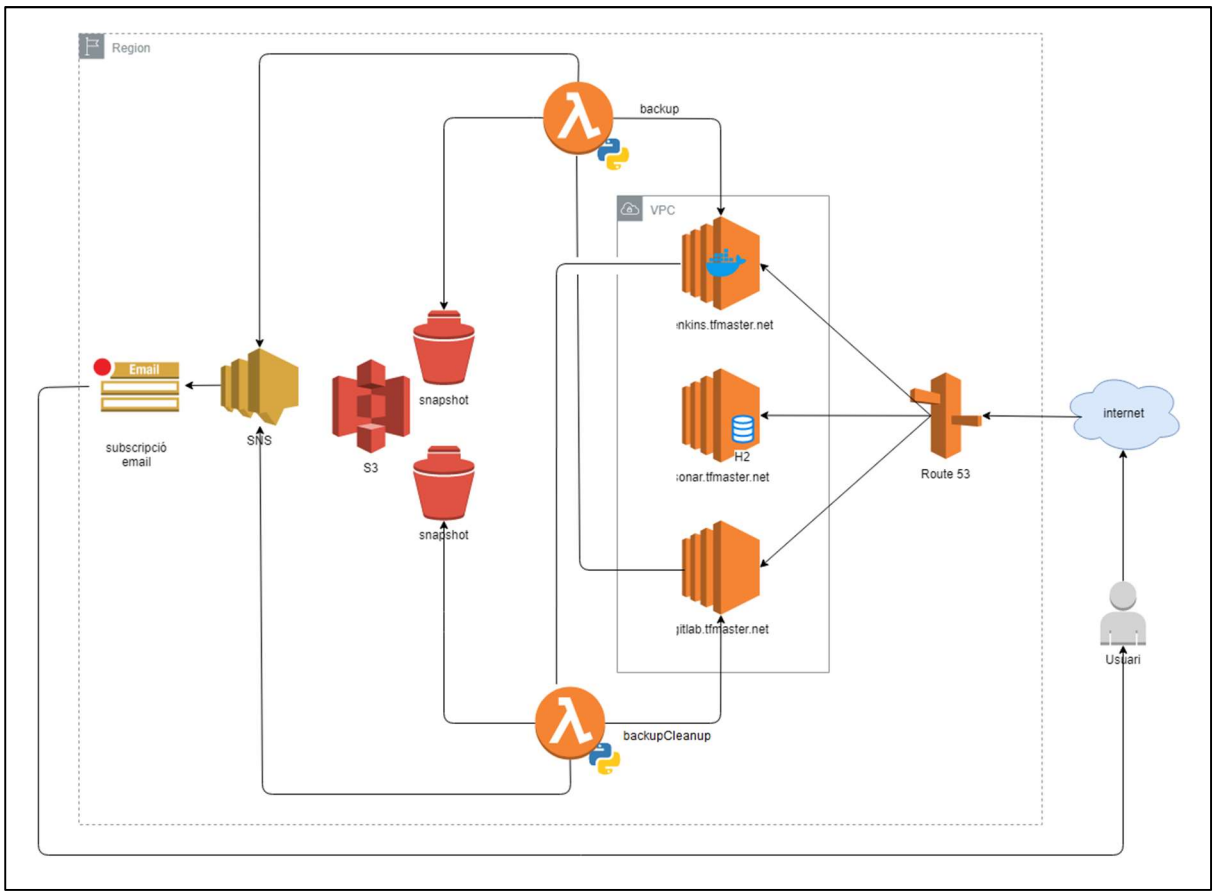


Figura 14: Diagrama infraestructura CI/CD

7.4. Visió completa de la infraestructura

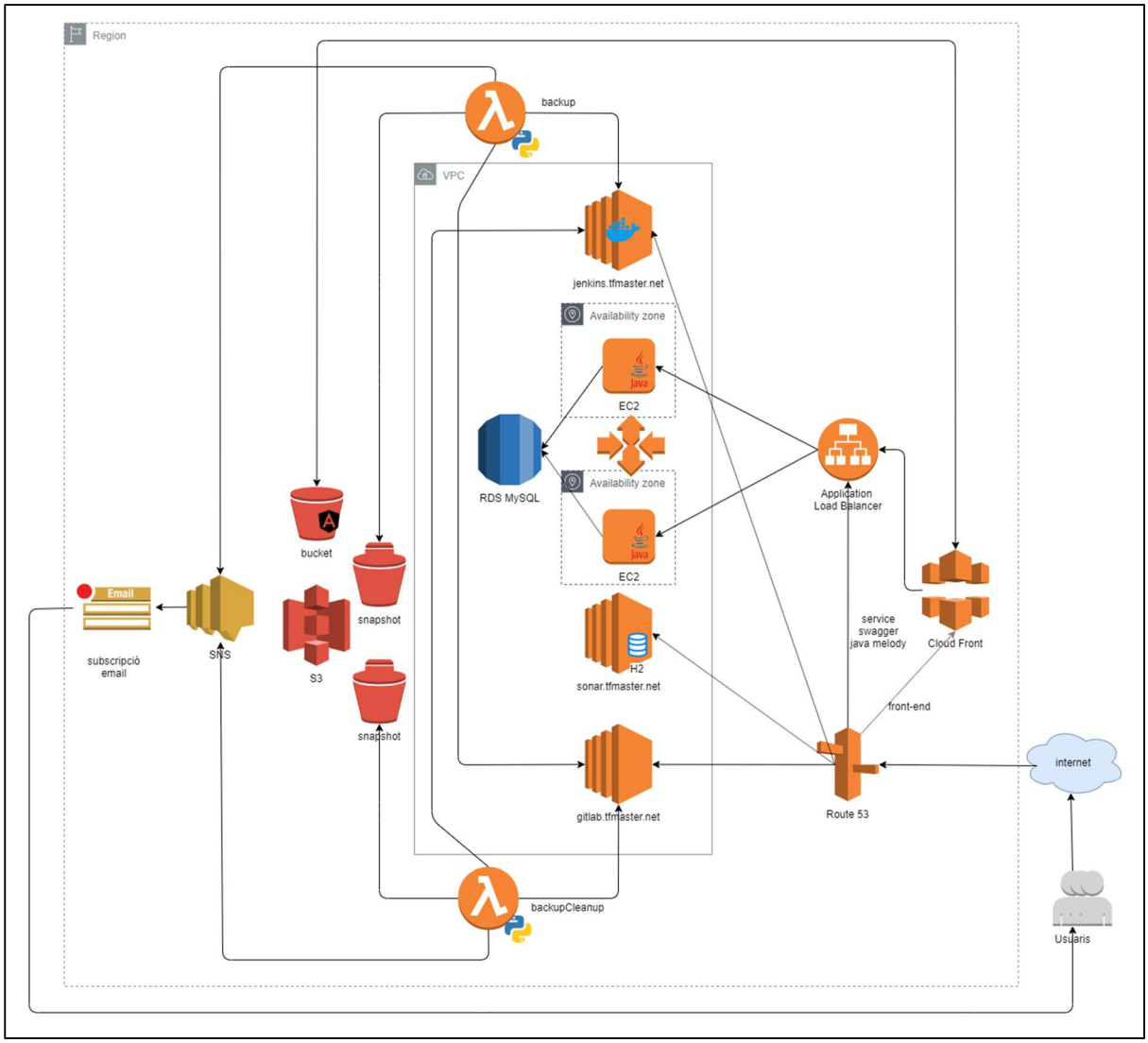


Figura 15: Diagrama infraestructura completa treball

S'ha registrat el domini tfmaster.net. Les nostres entrades a la infraestructura serà per subdominis d'aquest.

Totes les peticions seran ateses pel servei Route 53 que segons regles, ens redirigirà a una de les 5 possibilitats:

- El servidor de GitLab.
- El servidor de SonarQube.
- El servidor de Jenkins.
- A un Application Load Balancer que serveix el contingut de l'API directament.
- A un CloudFront que serveix el contingut front-end de la nostra aplicació emmagatzemat en un bucket S3 (i aquest connecta amb l'API mitjançant el Application Load Balancer).

L'API te una connexió a una RDS MySQL per connexió jdbc.

Existeixen dues Lambda que s'encarreguen dels snapshots dels servidors Jenkins i GitLab. Aquestes tenen un cron programat per executar-se. Quan s'executen, estan subscrites a una subscripció SNS que envia un correu electrònic amb informació del treball realitzat.

8. Plataforma de desenvolupament

8.1. Software

SO Windows 10
IntelliJ IDEA 2019.1 (Ultimate Edition)
JDK 1.8.0.201
MySQL Workbech.
Docker desktop v2.1.0.3
Office 365, Word, Excel, PowerPoint.
Microsoft Expression Encoder 4.
GanttProject.
Trello
Drawn.io
AWS Command Line Interface
WinSCP
node v10.15.3
npm 6.4.1

8.2. Hardware

Portàtil HP i5, 16 gb de RAM i disc dur de 250 gb SSD.
Hardware d'AWS:

- RDS MySQL: 1 instància *db.t2.micro*.
- Servei de l'aplicació de reserves: 1 instància EC2 *t2.micro* escalable a 4 instàncies.
- Servidor Jenkins: 1 instància EC2 *t2.medium*.
- Servidor GitLab CE: 1 instància EC2 *t2.medium*.
- Servidor SonarQube: 1 instància EC2 *t2.medium*.
- Dues AWS Lambda de 128 mb de memòria.

9. Planificació

9.1. Dates clau

01/10/2019 – Lliurament PAC1
30/10/2019 – Lliurament PAC2
08/12/2019 – Lliurament PAC3
06/01/2019 – Lliurament Final

9.2. Fites

02/10/2019 – Kickoff
20/10/2019 – Finalització del desenvolupament del software.
23/10/2019 – Tenir documentat el desenvolupament, llibreries utilitzades, etc.
27/10/2019 – Tenir desplegat en AWS.
30/10/2019 – Tenir documentat la infraestructura d'AWS per al desplegament i documentar la resta de PAC2.
14/11/2019 – Tenir muntada la infraestructura per a la CI/CD.
17/11/2019 – Tenir documentada la infraestructura.
25/11/2019 – Tenir els jobs de Jenkins funcionant i documentats.
08/12/2019 – Tenir finalitzada la PAC3, incloent el vídeo amb el funcionament.
06/01/2020 – Tenir finalitzat el projecte.

9.3. Planificació de les tasques

El pla de treball consta de 4 fases definides amb les dates clau d'entrega de les PACs. En aquestes 4 etapes es defineixen una sèrie de tasques.

1. Pla de treball

- Context i motivació.
- Objectius i requisits.
- Enfocament i metodologia.
- Planificació del projecte.
- Documentació.
- Entrega PAC1.

2. Anàlisis, Disseny, Arquitectura i Implementació del software

- Disseny del software API + front-end.
- Documentació del disseny.
- Preparació de l'entorn.
- Desenvolupament API + front-end.
- Desplegament en entorn de producció.

- Testing.
- Documentació del software.
- Vídeo de funcionament.
- Entrega PAC2.

En aquesta fase es realitzarà en primer lloc el disseny tècnic del sistema definint formalment els casos d'us per establir les funcionalitats del microservei i l'arquitectura del sistema, identificant: les entitats de la base de dades per a realitzar els diagrames de disseny; les classes i els objectes que intervenen en els processos, que es plasmaran en diagrames UML i l'estructura.
Posteriorment es construirà el producte de software, es començarà implementant l'arquitectura base i es continuarà amb la implementació total del microservei.
Es passaran els tests.
Posteriorment es preparà l'entorn real de producció i es desplegarà manualment amb alta disponibilitat.
Es passaran proves d'estres.
Per últim es redactarà tota la documentació necessària per a l'entrega de la PAC2.

3. Anàlisis, Disseny, Arquitectura i Implementació de la infraestructura de CI/CD

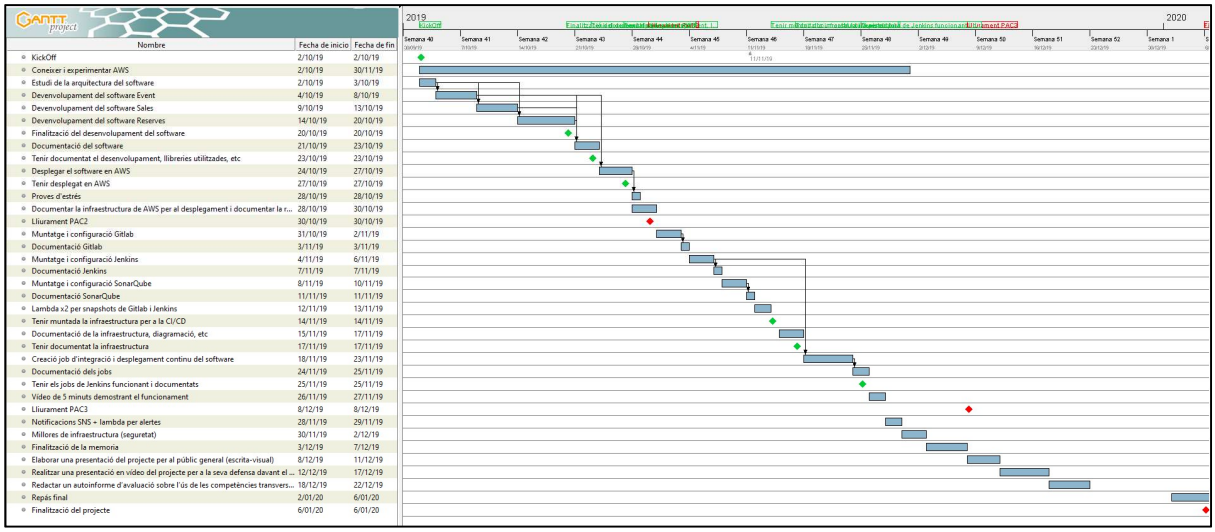
- Disseny de l'arquitectura.
- Documentació del disseny.
- Aprenentatge de l'arquitectura.
- Implementació de l'arquitectura.
- Proves de funcionament.
- Documentació de l'arquitectura.
- Entrega PAC3.

En la fase 3, es dissenyarà l'arquitectura on volem configurar la integració i el desplegament continu.
Es realitzaran diagrames del disseny de l'arquitectura y es documentarà la mateixa.
Posteriorment al disseny, s'implementarà tota l'arquitectura. Es realitzarà diagrames dels serveis i servidors utilitzats i es documentarà tota la configuració dels mateixos.
Es configuraran els jobs de Jenkins necessaris.
Per últim es redactarà tota la documentació necessària per a l'entrega de la PAC3.

4. Entrega final

- Finalització de la memòria.
- Confecció de la presentació.
- Gravació vídeo presentació.
- Preparació del lliurament.
- Entrega final.

9.4. Diagrama de Gantt



Nombre	Fecha de inicio	Fecha de fin
KickOff	2/10/19	2/10/19
Conèixer i experimentar AWS	2/10/19	30/11/19
Estudi de la arquitectura del software	2/10/19	3/10/19
Devenvolupament del software Event	4/10/19	8/10/19
Devenvolupament del software Sales	9/10/19	13/10/19
Devenvolupament del software Reserves	14/10/19	20/10/19
Finalització del desenvolupament del software	20/10/19	20/10/19
Documentació del software	21/10/19	23/10/19
Tenir documentat el desenvolupament, llibreries utilitzades, etc	23/10/19	23/10/19
Desplegar el software en AWS	24/10/19	27/10/19
Tenir desplegat en AWS	27/10/19	27/10/19
Proves d'estrés	28/10/19	28/10/19
Documentar la infraestructura de AWS per al desplegament i documentar la r...	28/10/19	30/10/19
Lliurament PAC2	30/10/19	30/10/19
Muntatge i configuració Gitlab	31/10/19	2/11/19
Documentació Gitlab	3/11/19	3/11/19
Muntatge i configuració Jenkins	4/11/19	6/11/19
Documentació Jenkins	7/11/19	7/11/19
Muntatge i configuració SonarQube	8/11/19	10/11/19
Documentació SonarQube	11/11/19	11/11/19
Lambda x2 per snapshots de Gitlab i Jenkins	12/11/19	13/11/19
Tenir muntada la infraestructura per a la CI/CD	14/11/19	14/11/19
Documentació de la infraestructura, diagramació, etc	15/11/19	17/11/19
Tenir documentat la infraestructura	17/11/19	17/11/19
Creació job d'integració i desplegament continu del software	18/11/19	23/11/19
Documentació dels jobs	24/11/19	25/11/19
Tenir els jobs de Jenkins funcionant i documentats	25/11/19	25/11/19
Vídeo de 5 minuts demostrant el funcionament	26/11/19	27/11/19
Lliurament PAC3	8/12/19	8/12/19
Notificacions SNS + lambda per alertes	28/11/19	29/11/19
Millores de infraestructura (seguretat)	30/11/19	2/12/19
Finalització de la memoria	3/12/19	7/12/19
Elaborar una presentació del projecte per al públic general (escrita-visual)	8/12/19	11/12/19
Realitzar una presentació en vídeo del projecte per a la seva defensa davant el ...	12/12/19	17/12/19
Redactar un autoinforme d'avaluació sobre l'ús de les competències transvers...	18/12/19	22/12/19
Repàs final	2/01/20	6/01/20
Finalització del projecte	6/01/20	6/01/20

Figura 16: Diagrama de Gantt

9.5. Diagrama Pert

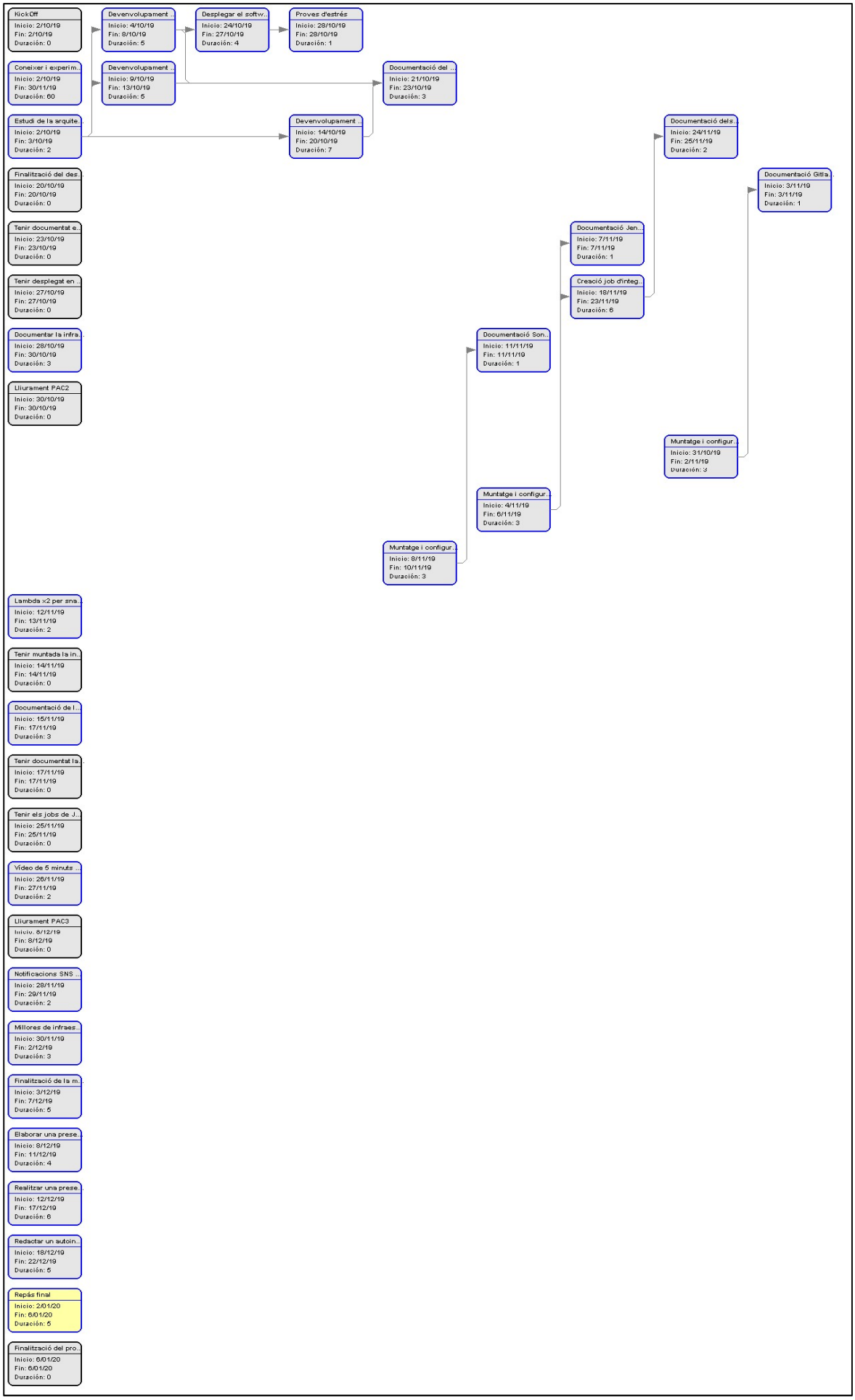


Figura 17: Diagrama Pert

9.6. Riscos del projecte

RISCOS					
Nom	**Descripció**	**Probabilitat**	**Impacte**	**Risc**	**Accions**
Carga laboral	Una carga laboral alta fa que augmenti la jornada laboral, més temps i més dedicació	Molt alta	Alt	Risc major	Avançar tot el treball possible en moments de menys carga
Malalties	Contraure alguna malaltia, fill, familiar o jo mateix	Mitja	Baix	Risc significatiu	Avançar tot el treball possible en moments de menys carga. No posposar tasques. Prestar atenció als primers símptomes i posar remei, visita mèdica, etc.
Pèrdua del projecte	Fallida de software, hardware, robatori de portàtil (el porto cada dia del treball a casa) pot provocar la pèrdua del treball realitzat	Baixa	Alt	Risc major	Emmagatzemar copies de seguretat periòdiques en pen drive i en emmagatzematge Cloud.
Planificació deficient	Error en l'estimació de les tasques a l'hora de planificar el projecte	Mitja	Alt	Risc major	Revisar periòdicament la planificació ajustant-la. Deixa temps de "greix" en les estimacions.
Coneixements insuficients	Augment en el temps de desenvolupament de les tasques al afrontar reptes que poden suposar major temps d'aprenentatge	Alta	Mig	Risc significatiu	Anticipació als reptes documentar-me abans d'hora.

Taula 1: Taula de riscos

9.7. Sumari dels productes obtinguts

Els productes finals obtinguts d'aquest treball son:

- El software desenvolupat en Java, se entregarà compilat en un arxiu .jar.
- El software desenvolupat en Java, s'entregarà un arxiu .zip amb el codi font d'aquest.
- El software desenvolupat en Angular 8, s'entregarà un arxiu .zip amb el codi font d'aquest.
- Lambda per a realitzar snapshots, s'entregarà el codi font en Python.
- Lambda per a realitzar esborrat d'snapshots, s'entregarà el codi font en Python.
- Una memòria del Treball Final de Màster, en la que es descriurà amb exhaustivitat tot el treball realitzat.
- Annexes. Col·lecció PostMan, com desplegar l'aplicació, etc.
- Vídeo de 5 minuts demostrant el funcionament tant del software com de la infraestructura de CI/CD.
- Presentació del projecte per al públic general.
- Vídeo de presentació on s'explicarà el treball desenvolupat i es realitzarà una demo del software i de la infraestructura d'AWS.

10. Procés de treball/desenvolupament de l'aplicació

10.1. Creant un projecte Spring. Des de cero?

Si es vol crear una aplicació amb framework Spring, Spring ens proporciona una eina (Spring Initializr) que ens crea una estructura de projecte buida que després podem importar al nostre IDE. Podem fer ús d'aquesta eina en el següent enllaç: https://start.spring.io/

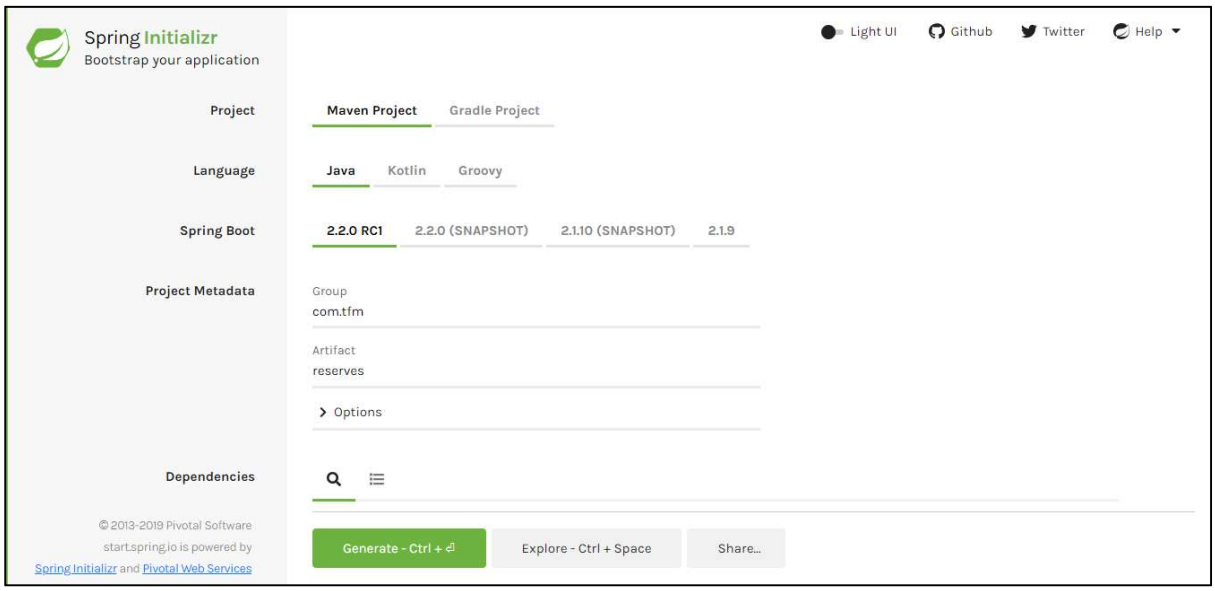


Figura 18: Configuració de l'aplicació de reserves en Spring Initializr

10.2. Patró d'arquitectura per capes

L'aplicació s'estructura en diferents capes:

- **Controllers** – defineixen la transferència de dades d'entrada de l'usuari al servei.
- **Services** – Es poden definir com un "middelware" mitjançant els controllers i els repositories. Reuneixen dades dels controllers, realitzen lògica de validació i lògica de negoci i les crides als repositories per a la manipulació de dades.
- **Repositories** – capa d'interacció amb els models. Realitzen operacions de base de dades.
- **Models** – capa de definició dels models on es defineixen les relacions.

10.3. Test-driven development

Test-driven development o Desenvolupament guiat per proves o TDD, és un procediment molt habitual entre els seguidors de les metodologies àgils.

Es basa en un desenvolupament de proves mitjançant les quals ha de passar el codi. Si està be programat es continua creant codi enfocat a la següent prova, sinó, es continua amb el codi erroni.

Aquest procés agilita el procés de creació de codi net. Amb TDD anem detectant errors i solucionant-los. S'aconsegueix un treball més ràpid, resolutiu, amb menys errors i més enfocat a l'usuari final ja que cada prova que es passa esta enfocada a l'usuari final.

Cicle de desenvolupament en TDD

- Escollim una prova.
- Sotmetem el codi a la prova.
- Si dona error, resolem el problema.
- Tornem a realitzar la prova fins que doni positiu i passem al següent test.
- Si no dona error passem a la següent prova.

Beneficis del TDD:

1. **Usant processos TDD evitem escriure codi innecessari**. En moltes ocasions es crea més codi del que es necessita. Usant TDD, vas solucionant petites tasques, superant test concrets, que ens ajuden a optimitzar el temps i ser molt més concrets durant la creació.
2. **Feedback d'API**. Gràcies a la ràpida resposta sobre la qualitat del codi i els múltiples tests que ens ofereix TDD, el flux d'informació és més gran.
3. **Gran quantitat de documentació**. Durant el procés de creació del codi, anem resolent dubtes i superant dificultats que a la llarga, ens proporcionen una documentació sobre tot el que anem implementant.
4. **Disseny emergent**. Igual que la creació del codi, el disseny és evolutiu i es pot anar modificant al mateix temps que es va desenvolupant el codi. De manera que no és necessari crear un disseny complet des del principi.
5. **Interface independent**. El codi no té perquè influir al disseny, si des d'un principi tots dos coneixen els requisits.
6. **Millora la comunicació**. Com que els problemes arriben en curts períodes de temps, fomenta la comunicació entre l'equip per poder solucionar-los.
7. **Major tranquil·litat**. En seguir el procés de TDD, saps que en acabar la feina, el codi serà pur, clar i net.
8. **Refactorització**. Després de trobar un problema, passem al procés de refactorització del mateix, de manera que no hem d'esperar a tenir el codi finalitzat per poder dur a terme aquesta tasca, sinó que el codi es va reescrivint a mesura que es detecten errors.
9. **Depuració**. TDD és la millor manera per detectar els erros, si apareix un i modifiquem el codi fins que passi el test, ho anem depurant poc a poc i obtenim un codi net.
10. **Escalabilitat**. A mesura que el projecte avança, el codi es fa més complex, i el llançament al mercat més pròxim. TDD permet avaluar en quin punt ens trobem en cada moment.
11. **Aprendre**. Ja que en TDD l'equip treballa junt per resoldre els problemes, el creixement individual també és més gran.
12. **Satisfacció del client**. En tot moment, els tests estan pensats des del punt de vista del client, per tant el producte estarà totalment enfocat a la seva satisfacció.
13. **Desenvolupament àgil**. Un flux continu de tasques, anàlisi i correccions és el que defineix la metodologia àgil. TDD compta amb tots els requisits per ser-ho.
14. **Millor integració amb tercers**. En portar un minuciós seguiment de les tasques, reduïm els costos i el temps.

10.4. Tests Unitaris

Les proves unitàries o Unit testing, formen part dels diferents procediments que es poden portar a terme dins d'una metodologia àgil. Son principalment trossos de codi dissenyats per a comprovar que el codi principal està funcionant com esperàvem.

Característiques principals de les proves unitàries

- **Automatitzable**. Encara que els resultats han de ser específics per a cadascun dels tests unitaris desenvolupats, els resultats es poden automatitzar de manera que es pugin realitzar les proves de forma individual o en grups.
- **Completes**. Son petits tests sobre parts del codi però al final s'ha de comprovar en la seva totalitat.
- **Repetibles**. El resultat ha de ser sempre el mateix encara que l'ordre d'executar els tests sigui diferent.
- **Independents**. És un codi aïllat que s'ha creat amb l'objectiu de comprovar un altre codi molt concret, no ha d'interferir en cap altre cosa.
- **Ràpids de crear**. No han de portar més de 5 minuts en la seva creació.

Principals avantatges de les proves unitàries

- **Proporcionen un treball àgil**. Permet detectar els erros a temps.
- **Qualitat de codi**. Al realitzar proves constantment, el codi resultant és un codi net i de qualitat.
- **Detecció rapida d'errors**.
- **Facilita els canvis i afavoreix la integració**. Ens permeten modificar parts del codi sense afectar al conjunt.
- **Proporcionen informació**.
- **Ajuden al procés de debugging**.
- **Disseny**. Si primer es dissenyen els tests, és molt més fàcil saber amb anterioritat com hem d'enfocar el disseny i veure que necessitats que s'han de complir.
- **Reducció del cost**. La detecció ràpida d'errors implicar tenir que escriure menys codi, optimitzant els temps d'entrega, per tant s'obté una reducció econòmica.

10.5. Entorns de treball. Definició i desenvolupament

Es defineixen dos entorns de treball, **desenvolupament** i **producció**.
Per a habilitar diferents arxius de propietats es defineixen dos profiles en pom.xml

```
<profiles>
    <profile>
        <id>development</id>
        <properties>
            <spring.profiles.active>development</spring.profiles.active>
        </properties>
        <activation>
            <activeByDefault>true</activeByDefault>
```

```
        </activation>
    </profile>
    <profile>
        <id>production</id>
        <properties>
            <spring.profiles.active>production</spring.profiles.active>
        </properties>
    </profile>
</profiles>
```

I s'habilita el filtrat del resource en el mateix fitxer:

```
<build>
    […]

    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
        </resource>
    </resources>
</build>
```

En l'application.properties es defineix un placeholder:

```
spring.profiles.active=@spring.profiles.active@
```

I es creen dos fitxers nous:

- application-development.properties
- application-production.properties

On es defineixen les variables que han de tenir diferents valors en diferents entorns:

```
host = http://localhost
```

10.6. Requisits funcionals

Es mostra com queden solucionats els requisits funcionals de l'aplicació mitjançant extractes del codi font.

Llistat de requisits funcionals:

RF1.1 – Creació d'un nou esdeveniment delimitat amb dates d'inici i fi.

EventServiceImpl:

```
@Override
public Event createEvent(Event event) {
    return eventRepository.save(event);
}
```

RF1.2 – Assignació de sales a un esdeveniment. Les sales poden ser reutilitzades entre edicions d' esdeveniments o entre esdeveniments diferents.

EventRoomServiceImpl:

```
@Override
public EventRoom createEventRoom(EventRoom eventRoom) {
    return eventRoomRepository.save(eventRoom);
}
```

RF2.1 – Creació d'una nova sala.

RoomServiceImpl:

```
@Override
public Room createRoom(Room room) {
    return roomRepository.save(room);
}
```

RF2.2 – Llistat de sales per esdeveniment.

RoomServiceImpl:

```
@Override
public List<Room> getAllRoomsByEventId(Long eventId) throws ResourceNotFoundException  {

    List<EventRoom> eventRoomList = eventRoomService.getAllEventRooms();
    List<Room> roomResultList = new ArrayList<>();
    for (EventRoom eventRoom : eventRoomList) {
        if (eventRoom.getEvent().getId() == eventId){
            Room room = roomRepository.findById(eventRoom.getRoom().getId()).orElseThrow(() ->
new ResourceNotFoundException("Room no encontrada :: "+ eventRoom.getRoom().getId()));;
            roomResultList.add(room);
        }
    }

    return roomResultList;
}
```

RF2.3 – Assignació d'equipament a una sala.

RoomEquipmentServiceImpl:

```
@Override
public RoomEquipment createRoomEquipment(RoomEquipment roomEquipment) {
    return roomEquipmentRepository.save(roomEquipment);
}
```

RF3.1 – Creació d'un nou equipament.

EquipmentServiceImpl:

```
@Override
public Equipment createEquipment(Equipment equipment) {
    return equipmentRepository.save(equipment);
}
```

RF3.2 – Llistat de tots els equipaments.

EquipmentServiceImpl:

```
@Override
public List<Equipment> getAllEquipments() {
    return equipmentRepository.findAll();
}
```

RF3.3 – Llistat de l'equipament assignat a una sala.

EquipmentServiceImpl:

```
@Override
public List<Equipment> getAllEquipmentByRoomId(Long roomId) throws ResourceNotFoundException {

    List<RoomEquipment> roomEquipmentsList = roomEquipmentService.getAllRoomEquipments();
    List<Equipment> equipmentResultList = new ArrayList<>();
    for (RoomEquipment roomEquipment : roomEquipmentsList) {
        if (roomEquipment.getRoom().getId() == roomId){
            Equipment equipment =
equipmentRepository.findById(roomEquipment.getEquipment().getId()).orElseThrow(() -> new
ResourceNotFoundException("Equipment no encontrado :: "+
roomEquipment.getEquipment().getId()));
            equipmentResultList.add(equipment);
        }
    }
```

```
    return equipmentResultList;
}
```

RF4.1 – Creació d'una nova tipologia de reserva.

TypeReservationServiceImpl:

```
@Override
public TypeReservation createTypeReservation(TypeReservation typeReservation) {
    return typeReservationRepository.save(typeReservation);
}
```

RF5.1 – Creació d'una nova reserva.

ReservationServiceImpl:

```
@Override
public Reservation createReservation(Reservation reservation) {
    return reservationRepository.save(reservation);
}
```

RF5.2 – Llistat de reserves per esdeveniment.

ReservationServiceImpl:

```
@Override
public List<Reservation> getAllReservationsByEventId(Long eventId) throws
ResourceNotFoundException {

    List<Room> roomListByEventId = roomService.getAllRoomsByEventId(eventId);
    List<Reservation> reservationList = reservationRepository.findAll();
    List<Reservation> reservationResultList = new ArrayList<>();

    for (Reservation reservation : reservationList) {
        for (Room room : roomListByEventId) {
            if (reservation.getIdRoom().getId() == room.getId()) {
                reservationResultList.add(reservation);
            }
        }
    }

    return reservationResultList;
}
```

10.7. Monitoring

Per al monitoreig del software s'inclou una dependència per a JavaMelody.
L'objectiu de JavaMelody és supervisar les aplicacions Java o Java EE en entorns de QA i producció. No simula les sol·licituds dels usuaris sinó que és una eina per a mesura i calcular estadístiques sobre el funcionament real d'una aplicació en funció de l'ús de l'aplicació per part dels usuaris.

Inclou gràfics de resum que mostren l'evolució al llarg del temps dels indicadors següents:

- Nombre d'execucions, temps d'execució i percentatge d'erros de peticions http, peticions sql, jsf, struts, Spring, etc.
- Memòria Java.
- CPU de Java.
- Nombre de sessions d'usuari.
- Nombre de connexions jdbc.

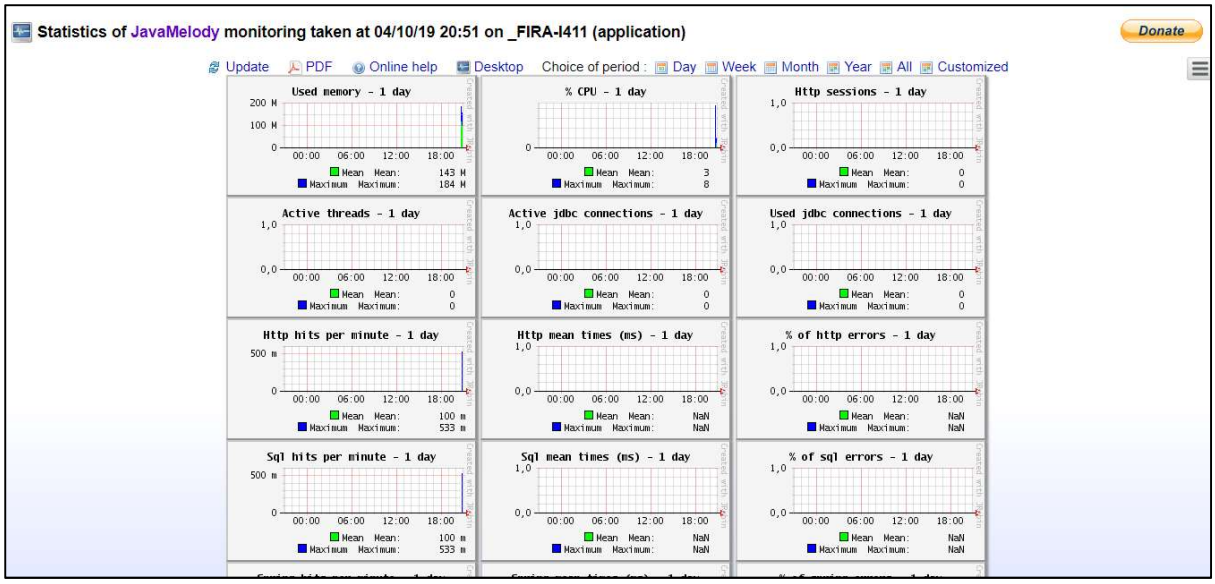


Figura 19: Exemple JavaMelody en execució local

Es pot accedir a les vistes en la següent url: http://<host>/<context>/monitoring

També s'ha inclòs la possibilitat d'exportar els reports en format PDF afegint dependències maven en l'arxiu pom.xml

```
<dependency>
    <groupId>net.bull.javamelody</groupId>
    <artifactId>javamelody-spring-boot-starter</artifactId>
    <version>1.77.0</version>
</dependency>
<!-- itext, option to add PDF export -->
<dependency>
    <groupId>com.lowagie</groupId>
    <artifactId>itext</artifactId>
    <version>2.1.7</version>
    <exclusions>
        <exclusion>
            <artifactId>bcmail-jdk14</artifactId>
            <groupId>bouncycastle</groupId>
        </exclusion>
        <exclusion>
            <artifactId>bcprov-jdk14</artifactId>
            <groupId>bouncycastle</groupId>
        </exclusion>
        <exclusion>
            <artifactId>bctsp-jdk14</artifactId>
            <groupId>bouncycastle</groupId>
        </exclusion>
    </exclusions>
</dependency>
```

10.8. Health / Actuator

En una arquitectura de microserveis, de vegades, una instancia de servei pot no ser capaç de gestionar les sol·licituds encara que continua en funcionament. Per exemple es pot haver quedat sense connexions a base de dades. Quan això es produeix, el sistema de control ha de ser capaç de generar una alerta, a més el balancejador de càrrega no ha d'encaminar les sol·licituds a la instancia de servei que ha fallat.

Per això es freqüent habilitar un endpoint que retorna la salut del servei. El handler de l'endpoint realitza varies comprovacions com ara:

- L'estat de les connexions als serveis d'infraestructura utilitzats per l'instancia del servei.

- L'estat del host (per exemple disc dur, etc.)
- Lògica específica de l'aplicació que podem customitzar.

El mòdul Spring Boot Actuator ens ajuda a controlar i administrar les aplicacions Spring Boot proporcionant funcions com el Health, auditing, mètriques, el tracing HTTP. Aquestes funcions es poden accedir mitjançant endpoints HTTP.

El mòdul Spring Boot Actuator us ajuda a controlar i administrar la vostra aplicació Spring Boot proporcionant funcions prèvies per a la producció com ara la revisió sanitària, l'auditoria, la recopilació de mètriques, el rastreig d'HTTP, etc. Totes aquestes funcions es poden accedir mitjançant endpoints JMX o HTTP.

ID	Descripció
auditevents	Exposa esdeveniments d'auditoria per l'aplicació.
beans	Llista tots els beans registrats per l'aplicació.
caches	Exposa un llistat de cache disponibles.
conditions	Mostra les condicions que van ser auto avaluades en les classes de configuració i auto-configuració i les raons per les quals no van coincidir.
configprops	Una llista de tot el que està anotat amb @ConfigurationProperties.
env	Exposa les propietats en un ConfigurableEnviroment.
flyway	Mostra totes les migracions flyway que han sigut aplicades.
health	Mostra informació sobre la salut de l'aplicació.
httptrace	Mostra informació de rastreig HTTP (per defecte les ultimes 100 crides).
info	Desplega informació de l'aplicació i el codi font.
integrationgraph	Mostra el gràfic d'integració d'Spring.
loggers	Mostra i modifica els valors dels loggers configurats en l'aplicació.
liquibase	Mostra migracions Liquidbase de base de dades que s'han aplicat.
metrics	Mostra informació de mètriques de l'aplicació.
mappings	Mostra una llista de tots els paths anotats amb @RequestMapping.
scheduledtasks	Desplega les tasques programades de l'aplicació.
sessions	Permet obtenir i esborrar sessions d'usuari sempre i quan siguin sessions d'Spring.
shutdown	Permet que l'aplicació s'apagui correctament.
threaddump	Mostra informació sobre els threads.

Taula 2: Taula endpoints actuator

Es pot accedir al health de l'aplicació en la següent url: http://<host>/<context>/actuator/health

Figura 20: Resultat del health endpoint

10.9. Documentació d'API

Swagger és una sèrie de regles, especificacions i eines que ens ajuda a documentar les nostres APIs.

L'eina Swagger UI exposa la documentació de la nostra API, la fa navegable i organitzada per un fàcil accés.

S'han d'afegir les dependències al pom.xml

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

Es crea una class anomenada SwaggerConfig amb l'anotació @EnableSwagger2. Aquesta anotació també podria anar en l'SpringApplication, però volem realitzar més configuracions, però això optem per crear la classe SwaggerConfig.

Només volem exposar la documentació de les APIs que es troben en el package com.tfm.reserves, per això hem de modificar la següent línia:

```
RequestHandlerSelectors.basePackage("com.tfm.reservas")
```

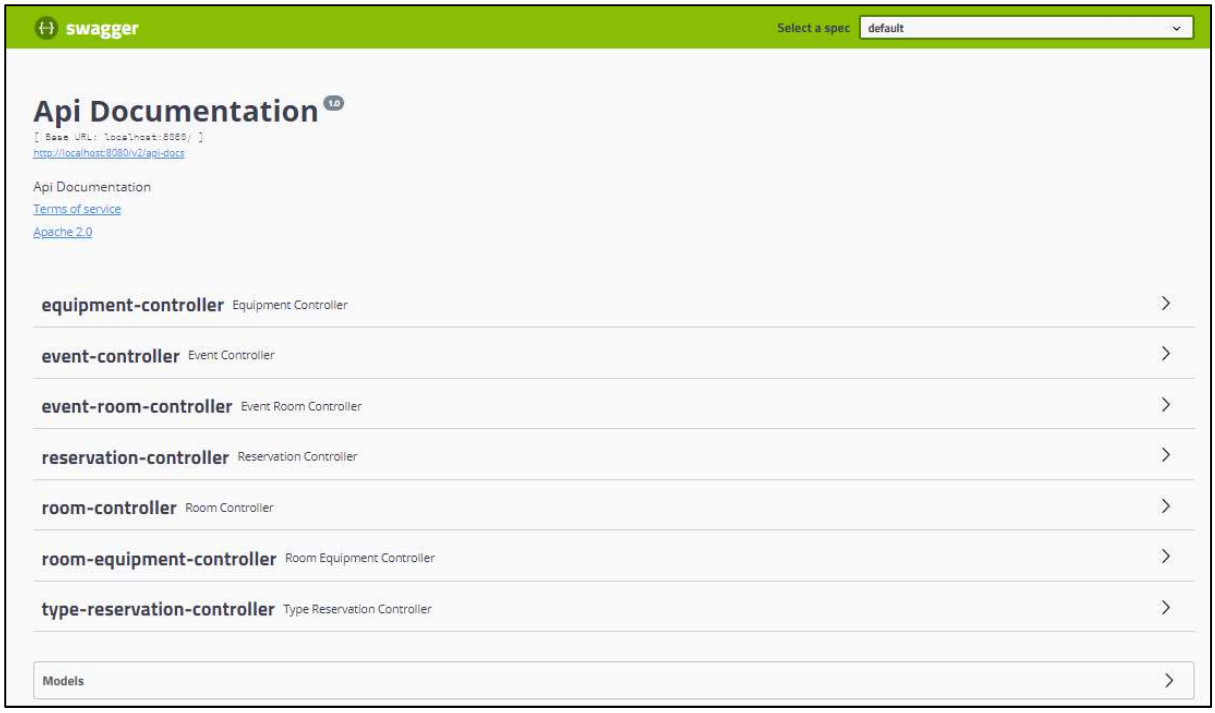


Figura 21: Swagger-ui de l'API

Figura 22: Detall de controllers en Swagger-ui

Podem accedir a la documentació en la següent url: http://<host>/<context>/swagger-ui.html#/
Per a veure el json generat es pot accedir a: http://<host>/<context>/v2/api-docs/

10.10. Front-end

Recordem que la part front-end surt de l'abast del projecte, només la volem per a realitzar un desplegament e integrar-la amb la infraestructura d'entrega continua.

Per començar a treballar amb Angular8, hem d'instal·lar un software previ:

Instal·lació Node.js i npm

Node.js és un entorn d'execució de JavaScript, així que fa que la majoria del codi JS s'executi en part del servidor i no en navegador client.

Necessitem instal·lar Node.js per a desenvolupar en Angular8.
Finalment per a Windows és un paquet instal·lable que el podem descarregar a la seva web:
https://nodejs.org/es/download/
Per comprovar la versió podem utilitzar la següent comanda al cmd:

```
node -v
```

i ens retornarà la versió instal·lada.

Npm és el sistema de gestió de paquets per defecte per a Node.js. Amb la instal·lació de Node.js, s'ha d'instal·lar automàticament npm.
Per comprovar la versió podem utilitzar la següent comanda al cmd:

```
npm -v
```

i ens retornarà la versió instal·lada.

Instal·lació Angular 8

Angular és un framework per a aplicacions web que s'utilitza per a crear aplicacions web d'una sola pàgina. Utilitza el patró d'arquitectura Model Vista Controlador (MVC).
Angular es basa en classes tipus "Components", que les seves propietats son les usades per a fer binding de les dades. En aquestes classes tenim propietats (variables) i mètodes (funcions a cridar).

Aplicacions web d'una sola pàgina.
Single-page application (SPA) és una aplicació web o un site web que cap en una solo pàgina amb el propòsit de donar una experiència més fluida als usuaris com si fos una aplicació d'escriptori. En una SPA tots els codis html, js i css es carreguen d'una vegada. Així passar d'una "pàgina" a una altra no fa falta recarregar el navegador. En realitat dona la sensació de diferents pàgines però en realitat intercanvia diferents "vistes" per donar la sensació de navegació, ens movem en el site web sense sortir-nos d'una única pagina.

Model Vista Controlador (MVC).
És un patró d'arquitectura de software que separa les dades i la lògica de negoci d'una aplicació de la seva representació i el mòdul encarregat de gestionar els esdeveniments i les comunicacions.
Per a això, MVC proposa la construcció de tres components distints que són el **model**, la **vista** i el **controlador**.
Aquest patró es basa en les idees de reutilització de codi i de la separació de conceptes.

Instal·lació Angular8
Instal·lem el client Angular mitjançant el gestor de paquets npm:

```
npm install -g @angular/cli
```

Per a comprovar la versió de la instal·lació:

```
ng --version
```

```
     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/

Angular CLI: 8.3.9
Node: 10.15.3
OS: win32 x64
Angular:
...

Package                      Version
------------------------------------------------------
@angular-devkit/architect    0.803.9
@angular-devkit/core         8.3.9
@angular-devkit/schematics   8.3.9
@schematics/angular          8.3.9
@schematics/update           0.803.9
rxjs                         6.4.0
```

Projecte reservas-frontend

Primer de tot, s'ha de crear un nou projecte en Angular:

```
ng new reservas-frontend
```

Identifiquem els components, serveis i mòduls que es volen crear:

Components:

- event-list
- room-create
- room-list
- room-details
- room-update
- eventroom-create
- eventroom-list
- eventroom-details
- eventroom-update
- equipment-create
- equipment-list
- equipment-details
- equipment-update
- roomequipment-create
- roomequipment-list
- roomequipment-details
- roomequipment-update
- typereservation-create
- typereservation-list
- typereservation-details
- typereservation-update
- reservation-list

Serveis (per a mètodes HTTP)**:**

- event.service.ts
- room.service.ts
- eventroom.service.ts
- equipment.service.ts
- roomequipment.service.ts
- typereservation.service.ts
- reservation.service.ts

Mòduls:

- FormsModule
- HttpClientModule
- AppRoutingModule

Classes:

- event.ts
- room.ts

- eventroom.ts
- equipment.ts
- roomequipment.ts
- typereservation.ts
- reservation.ts

Per a la creació dels components i dels serveis anem a

```
reservas-frontend/src/app
```

I en aquest directori executem les comanda

```
ng g s nom-del-servei
```
Per a cadascun dels serveis, i

```
ng g c nom-del-component
```
Per a cadascun dels components.

Posem com a exemple ***room***:

```
reservas-frontend\src\app>ng g s room
CREATE src/app/room.service.spec.ts (323 bytes)
CREATE src/app/room.service.ts (133 bytes)

reservas-frontend\src\app>ng g c room-create
CREATE src/app/room-create/room-create.component.html (26 bytes)
CREATE src/app/room-create/room-create.component.spec.ts (657 bytes)
CREATE src/app/room-create/room-create.component.ts (288 bytes)
CREATE src/app/room-create/room-create.component.css (0 bytes)
UPDATE src/app/app.module.ts (493 bytes)

reservas-frontend\src\app>ng g c room-list
CREATE src/app/room-list/room-list.component.html (24 bytes)
CREATE src/app/room-list/room-list.component.spec.ts (643 bytes)
CREATE src/app/room-list/room-list.component.ts (280 bytes)
CREATE src/app/room-list/room-list.component.css (0 bytes)
UPDATE src/app/app.module.ts (585 bytes)

reservas-frontend\src\app>ng g c room-details
CREATE src/app/room-details/room-details.component.html (27 bytes)
CREATE src/app/room-details/room-details.component.spec.ts (664 bytes)
CREATE src/app/room-details/room-details.component.ts (292 bytes)
CREATE src/app/room-details/room-details.component.css (0 bytes)
UPDATE src/app/app.module.ts (689 bytes)

reservas-frontend\src\app>ng g c room-update
CREATE src/app/room-update/room-update.component.html (26 bytes)
CREATE src/app/room-update/room-update.component.spec.ts (657 bytes)
CREATE src/app/room-update/room-update.component.ts (288 bytes)
CREATE src/app/room-update/room-update.component.css (0 bytes)
UPDATE src/app/app.module.ts (2549 bytes)
```

Integració de Bootstrap i JQuery

Per a instal·lar Bootstrap i JQuery, altre cop hem d'utilitzar el gestor de paquets:

```
npm install bootstrap jquery --save
```

Hem de canviar les línies següents de l'arxiu angular.json:

```
"styles": [
  "src/styles.css"
],
"scripts": []
```

Per aquestes:

```
"styles": [
  "src/styles.css",
```

```
  "node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
  "node_modules/jquery/dist/jquery.min.js",
  "node_modules/bootstrap/dist/js/bootstrap.min.js"
]
```

Detalls a destacar del codi

Estructura de directoris:

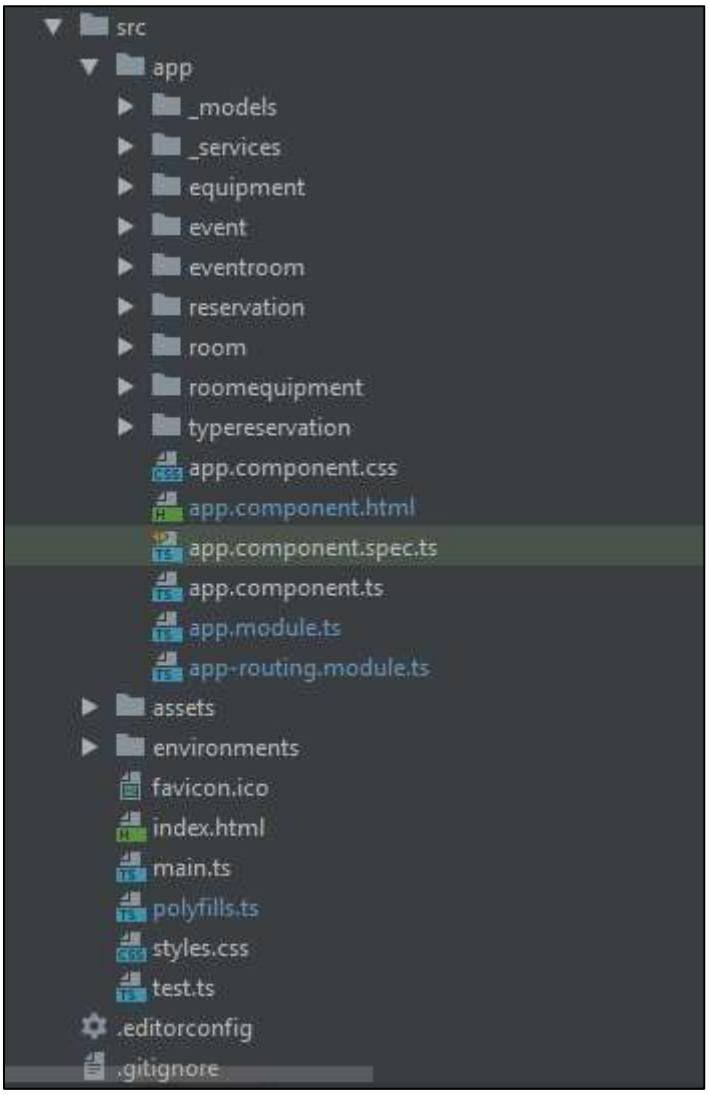


Figura 23: Estructura codi Angular 8

Enviroments:

Creem dos enviroments, en els arxius enviroments/enviroment.ts

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:8080/api/v1/'
};
```

i enviroments/enviroments.prod.ts:

```
export const environment = {
  production: true,
  apiUrl: 'http://lb-reservas-896768456.eu-west-1.elb.amazonaws.com/api/v1/'
};
```

d'aquesta manera podem realitzar un import i utilitzar les variables declarades en els anteriors arxius:

```
private baseUrl = environment.apiUrl + 'equipments';
```

10.11. Afegint seguretat. Login i Basic Auth

Afegint login

Per a afegir un servei de login i seguretat en el nostre front-end, afegim els següents elements:

Components

Serveis

- authentication
- authguard
- basicauthinterceptor

Components

- login
- logout

authentication.ts

```
export class AuthenticationService {

  private url = environment.apiUrl + 'login/validateLogin';

  constructor( private http: HttpClient) { }

  authenticate(username, password) {
    console.log(username);
    console.log(password);
    console.log(this.url);
    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(username + ':' +
password) });
    console.log(headers);
    return this.http.get<User>(this.url, { headers }).pipe(
      map(
        userData => {
          sessionStorage.setItem('username', username);
          const authString = 'Basic ' + btoa(username + ':' + password);
          sessionStorage.setItem('basicauth', authString);
          return userData;
        }
      )

    );
  }

  isUserLoggedIn() {
    const user = sessionStorage.getItem('username');
    console.log(!(user === null));
    return !(user === null);
  }

  logOut() {
    sessionStorage.removeItem('username');
  }
}
```

authguard.ts

```
export class AuthguardService implements CanActivate  {

  constructor(private router: Router,
              private authService: AuthenticationService) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (this.authService.isUserLoggedIn()) {
      return true;
    }

    this.router.navigate(['login']);
    return false;

  }
}
```

basicauthinterceptor.ts

```
export class BasicAuthHtppInterceptorService implements HttpInterceptor {

  constructor() { }

  intercept(req: HttpRequest<any>, next: HttpHandler) {

    if (sessionStorage.getItem('username') && sessionStorage.getItem('basicauth')) {
      req = req.clone({
        setHeaders: {
          Authorization: sessionStorage.getItem('basicauth')
        }
      })
    }

    return next.handle(req);

  }
}
```

login.ts

```
export class LoginComponent implements OnInit {

  username = '';
  password = '';
  invalidLogin = false;

  constructor(private router: Router,
              private loginservice: AuthenticationService) { }

  ngOnInit() {
  }

  checkLogin() {
    (this.loginservice.authenticate(this.username, this.password).subscribe(
        data => {
          this.router.navigate([''])
          this.invalidLogin = false
        },
        error => {
          this.invalidLogin = true

        }
      )
    );
  }
}
```

login.html

```
<div class="container">
  <div>
    <label for="username">User Name:</label>
    <input id="username" type="text" class="form-control" name="username"
[(ngModel)]="username">
    <br>
    <label for="password">Password:</label>
    <input id="password" type="password" class="form-control" name="password"
[(ngModel)]="password">
  </div>
  <div>
  <button (click)=checkLogin() class="btn btn-success">Login</button>
  </div>
</div>
```

logout.ts

```
export class LogoutComponent implements OnInit {

  constructor(
    private authenticationService: AuthenticationService, private router: Router) {
  }

  ngOnInit() {
    this.authenticationService.logOut();
    this.router.navigate(['login']);
  }
}
```

header.ts Afegim el service d'Autentication.

```
export class HeaderComponent implements OnInit {

  constructor(public loginService: AuthenticationService) { }

  ngOnInit() {
  }
}
```

En el header.html on son els links als diferents elements de la web, controlem si l'usuari te login:

```
<li class="nav-item">
  <a *ngIf="loginService.isUserLoggedIn()" routerLink="room/rooms" class="nav-link"
routerLinkActive="active">Room</a>
</li>
```

En el app-routing, s'ha d'afegir:

```
{ path: 'equipment/equipments', component: EquipmentListComponent,
canActivate:[AuthguardService] },
```

En tots els paths que volem que no es pugi arribar directament, redireccionarà a *login*.

En app.module.ts afegim l'interceptor de BasicAuthHTTP.

```
providers: [
  {
    provide: HTTP_INTERCEPTORS, useClass: BasicAuthHtppInterceptorService, multi: true
  }
],
```

Securitzant l'API

S'utilitzarà Basic Auth per a una implantació de seguretat amb les llibreries d'Spring Boot Security. Totes les peticions REST des del front-end fins a l'API seran autenticades usant Basic Authentication. Basic Auth, és un esquema simple d'autenticació construït sobre el protocol HTTP. Quan s'utilitza aquest protocol, les peticions han de portar un Authorization header, format per la paraula Basic seguit d'un espai i un string username:password codificat en base 64.

Afegim les dependència al pom.xml:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

S'ha de crear la classe de configuració SecurityConfig. Deixem fora /actuator/* ja que sinó els heatlh checks des d'AWS donen errors ja que consideren que el servei no està funcionant correctament.

```
@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.csrf().disable().

                authorizeRequests().antMatchers(HttpMethod.OPTIONS, "/**").

permitAll().anyRequest().authenticated().antMatchers("/actuator/**").permitAll()
                .and().httpBasic();
    }

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("dhidalgo").password("{noop}dhidalgo").roles("USER");
    }

    public void configure(WebSecurity web) {
```

```
        web.ignoring().antMatchers("/actuator/**");
    }
}
```

Es crea la classe User, que serà l'objecte que es retornarà al front-end quan realitzi una petició de login:

```
public class User {
    private String status;

    public User(String status) {
        this.status = status;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

I es crea un petit controller que serà el que es cridi des del front-end per a realitzar el login UserController:

```
@CrossOrigin()
@RestController
@RequestMapping({ "/api/v1/login" })
public class UserController {

    @GetMapping(produces = "application/json")
    @RequestMapping({ "/validateLogin" })
    public User validateLogin() {
        return new User("User successfully authenticated");
    }
}
```

Amb això hem realitzat una senzilla implementació de Basic Auth que gracies al servei de login en Angular podem fer crides http securitzades cap a l'API.

10.12. Infraestructura per al desplegament de l'aplicació.

ECS

Amazon ECS utilitza imatges de Docker en les definicions de tasques per a llençar contenidors en instancies d'Amazon EC2 en els clústers.

Per a la creació d'un nou clúster es segueixen les següents passes:
Es crea un clúster nou amb nom reservas-cluster amb la següent configuració:

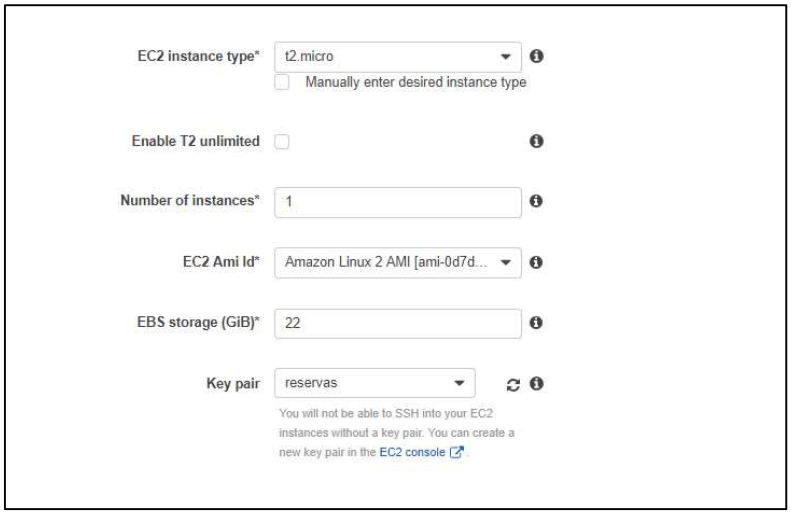


Figura 24: Creació ECS, part 1

Per ara només es crea una instancia, més endavant, en aquest punt, subapartat “Auto escalat” s’explica com realitzar un auto escalat del servei.

Hem de crear un key pair per accedir via SSH al servidor EC2:

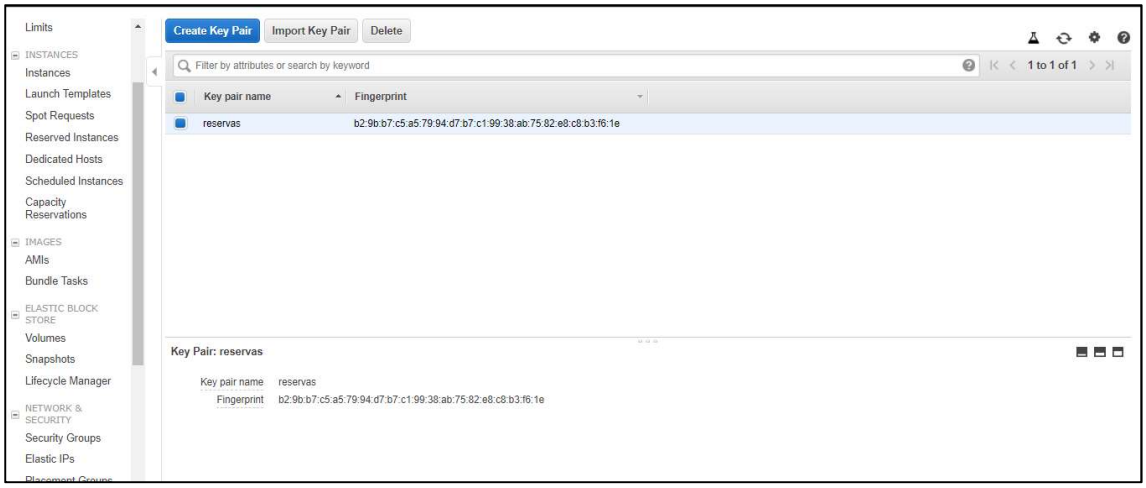


Figura 25: Creació ECS, part 2

En l’apartat networking, escollim l’VPN que ens crea amb la conta y escollim dos subnets per a habilitar amb posterioritat l’alta disponibilitat.

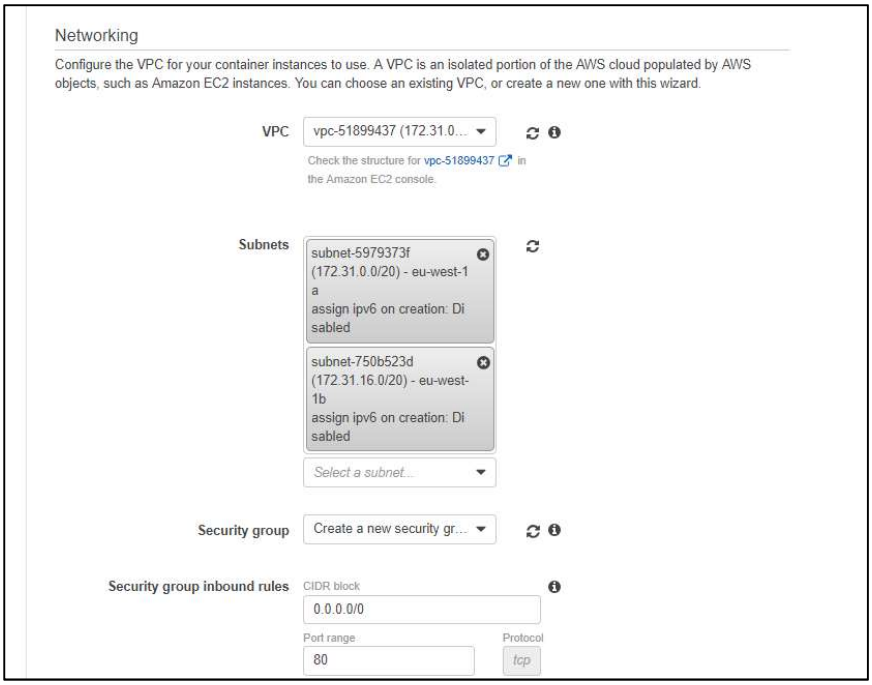


Figura 26: Creació ECS, part 3

S'ha de crear un nou Security group per a administrar el tipus d'accés a les instancies EC2. Per el moment només habilitem l'accés SSH per el port 22 a la meva IP:

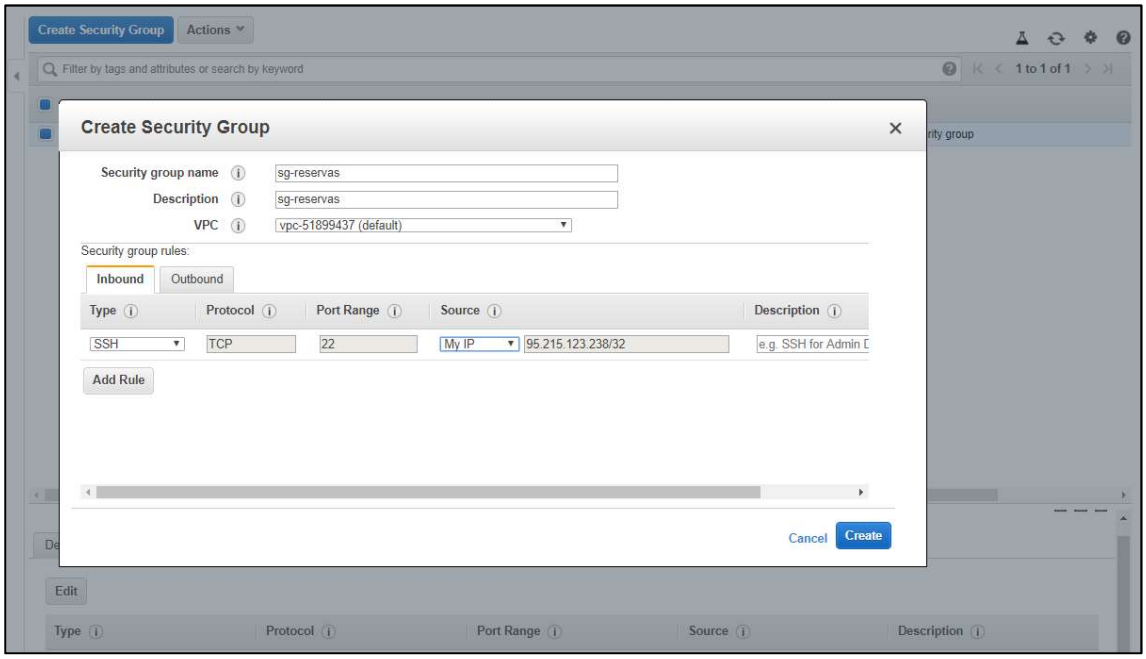


Figura 27: Creació ECS, part 4

Com a resultat, l'apartat networking queda així:

Figura 28: Creació ECS, part 5

I finalitzem la creació del clúster:

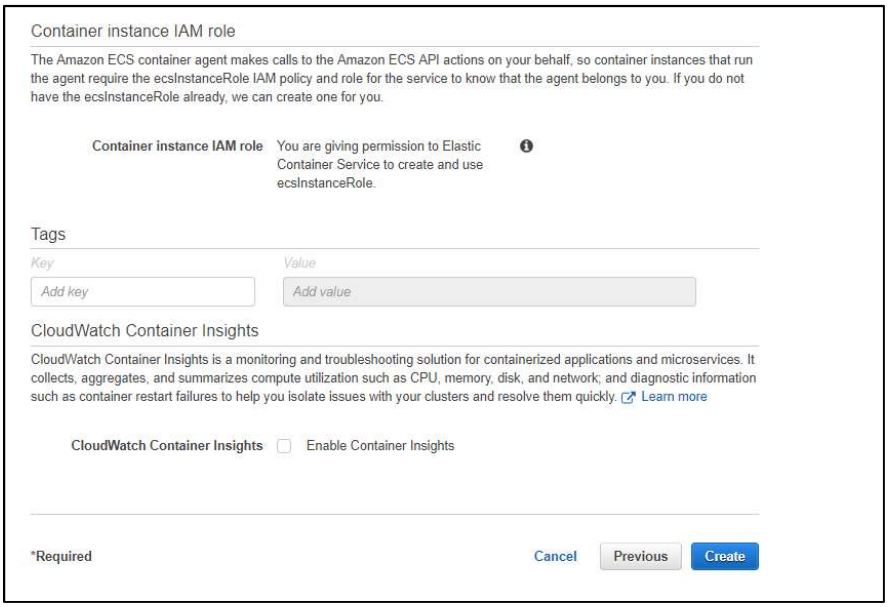


Figura 29: Creació ECS, part 6

En la posterior pantalla veiem com es crea el desitjat:

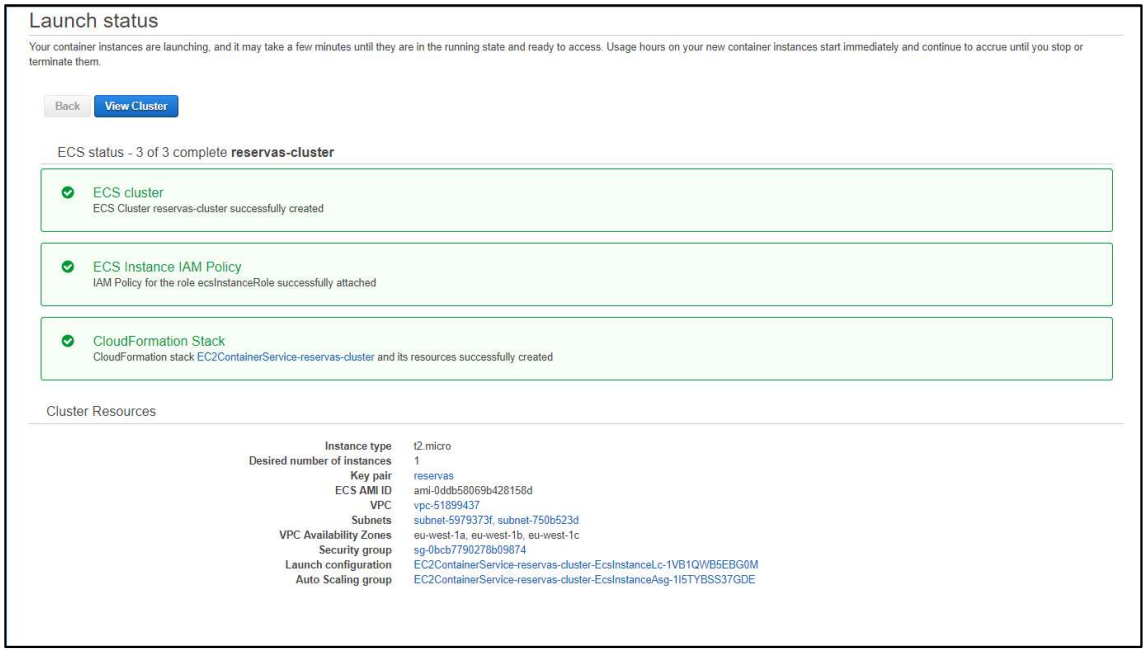


Figura 30: Creació ECS, part 7

Application Load Balancer i target group

Application Load Balancer és un balancejador de càrrega per a aplicacions que equilibra el tràfic HTTP i HTTPS. Dirigeix el tràfic als destins dins del VPC (Virtual Private Cloud) en funció del contingut de la sol·licitud en una o més zones de disponibilitat.

Es crea un nou Load Balancer anomenat lb-reservas.

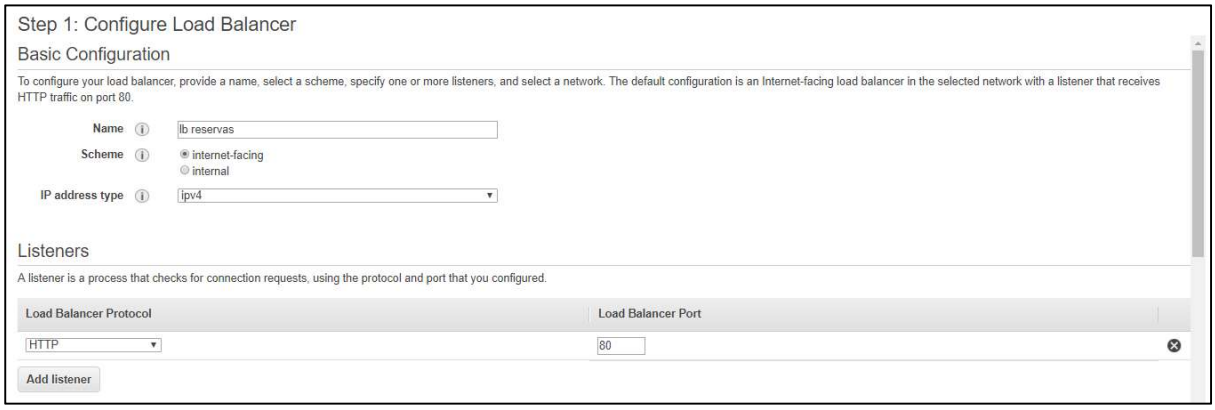


Figura 31: Creació Load Balancer, part 1

S'assignen dues zones de disponibilitat tal i com vam configurar el clúster.

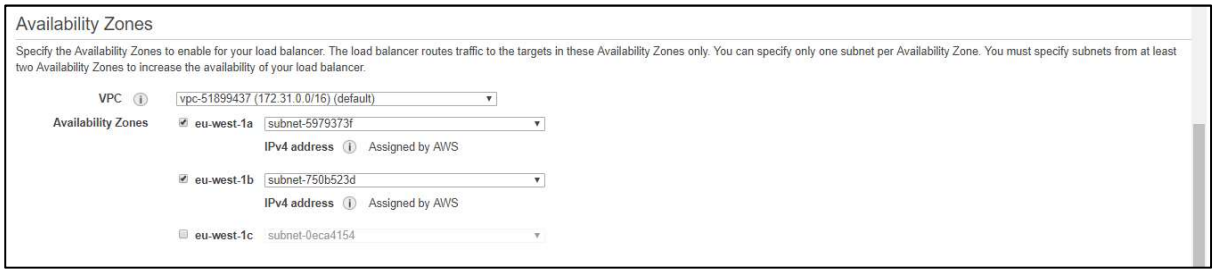


Figura 32: Creació Load Balancer, part 2

Escollim el Security Group creat en el pas anterior amb nom sg-reservas.

Figura 33: Creació Load Balancer, part 3

Es crea un nou target group anomenat tg-reservas, amb protocol HTTP i port 80, del target type instance.

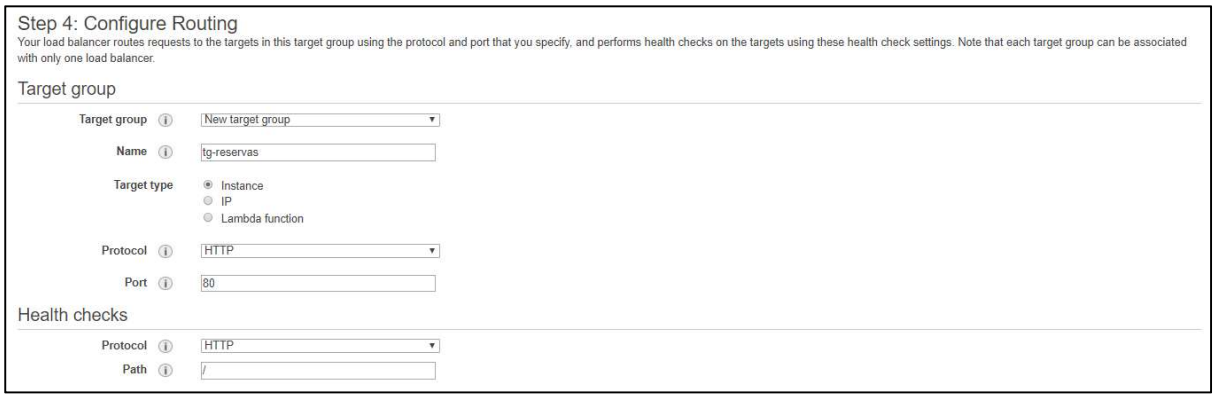


Figura 34: Creació Load Balancer, part 4

No registrem encara cap instancia.

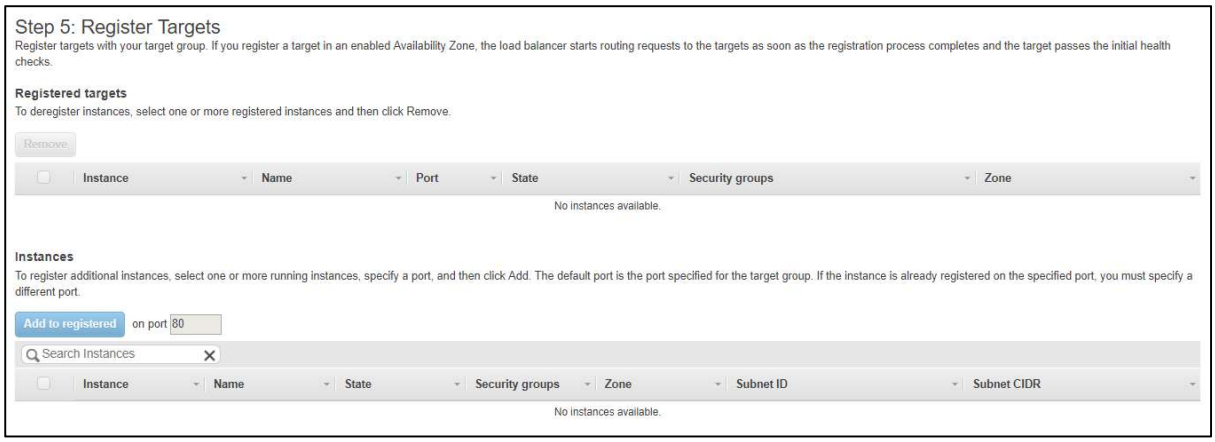


Figura 35: Creació Load Balancer, part 5

Finalment la pantalla amb confirmació dels resultats de la creació del Load Balancer.

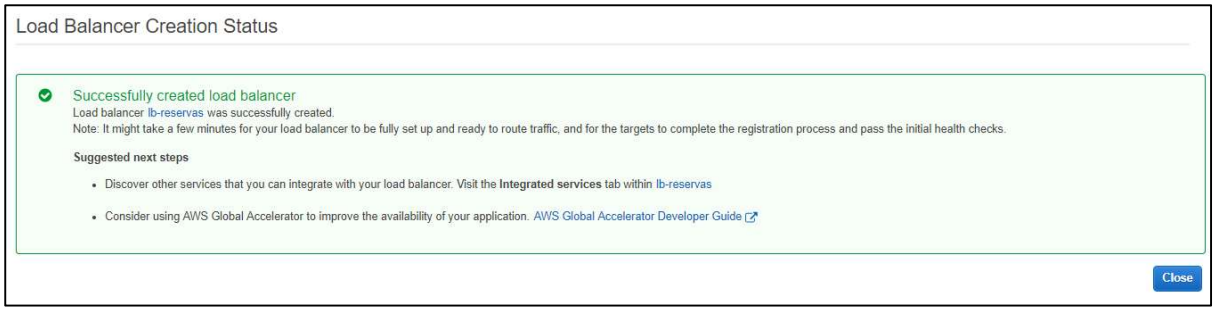


Figura 36: Creació Load Balancer, part 6

ECR i Docker

Amazon Elastic Container Registry és un registre de contenidors de Docker.

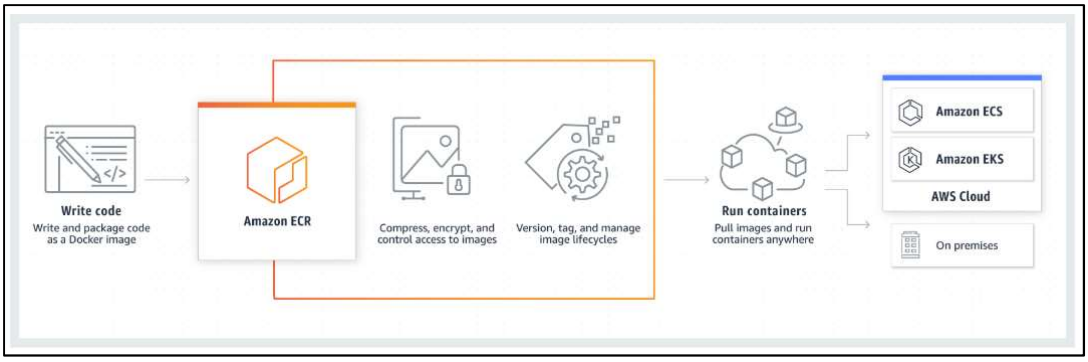


Figura 37: Funcionament d'ECR, extret de https://aws.amazon.com/es/ecr/

Creem un nou repositori ECR que és on gestionarem les imatges de Docker que creem i a on farem push de les mateixes una vegada les construïm en local.

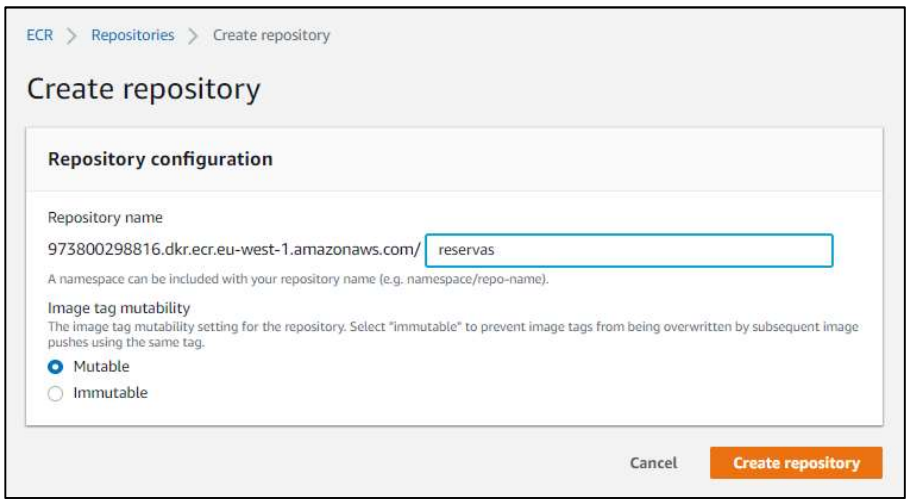


Figura 38: Creació ECR, part 1

En el mateix repositori, ja ens dona la informació que necessitem per a fer el push en aquest:

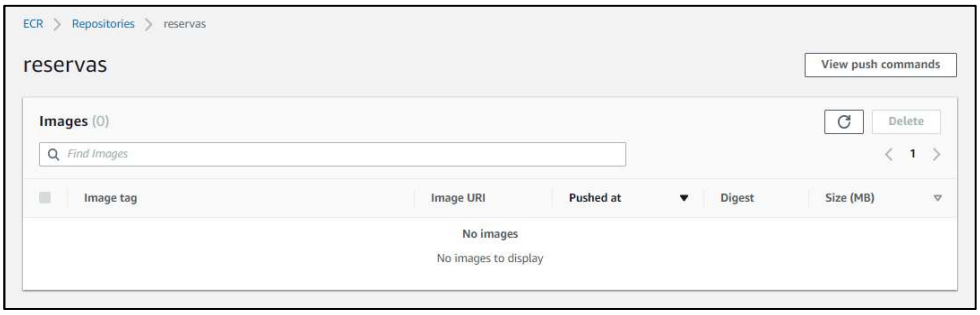


Figura 39: Creació ECR, part 2

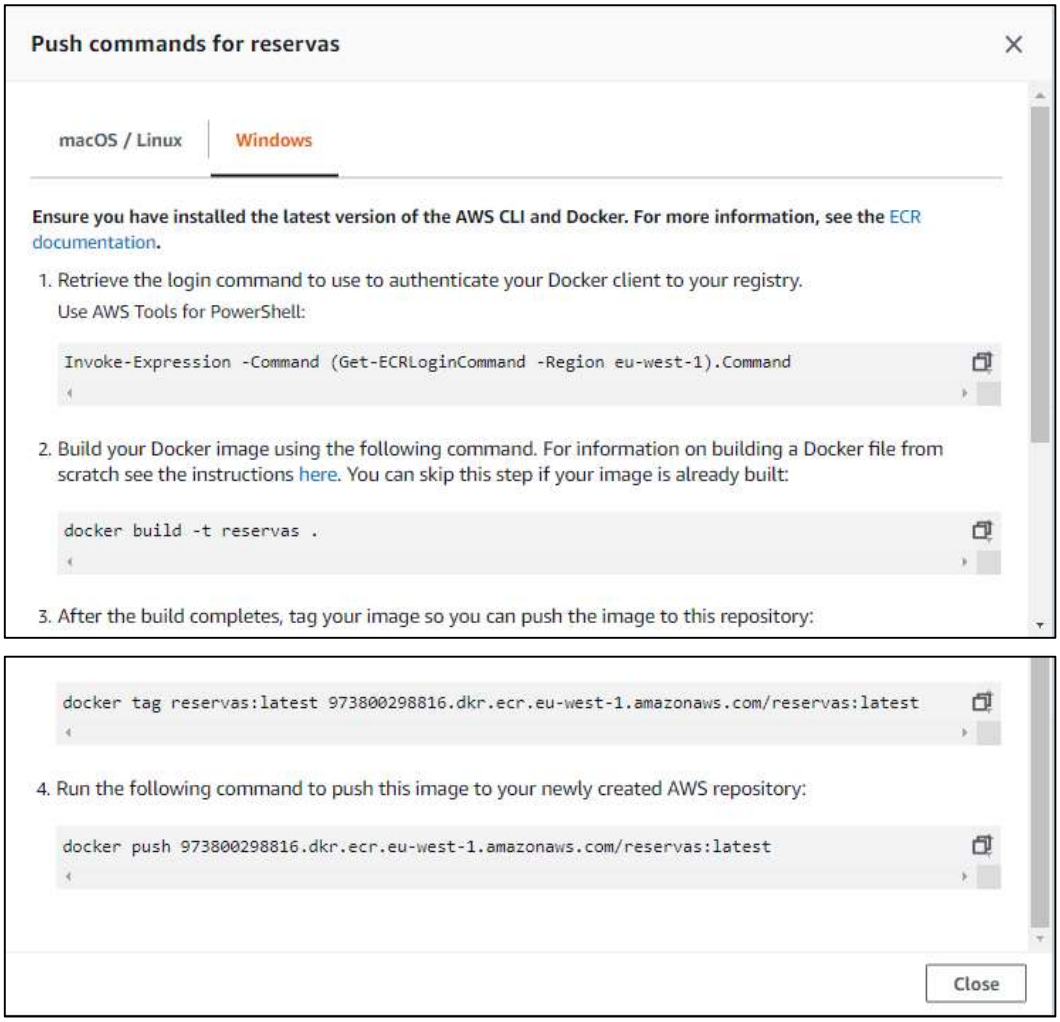


Figura 40: Creació ECR, part 3

Amb una prèvia instal·lació del Docker desktop per a Windows, fem un pull de la imatge desitjada del docker hub, openjdk (https://hub.docker.com/_/openjdk) amb la tag 8-jre.

```
PS C:\Users\dhidalgo> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
PS C:\Users\dhidalgo> docker pull openjdk
Using default tag: latest
latest: Pulling from library/openjdk
a316717fc6ee: Pull complete
809137453b07: Pull complete
b363ad12fddb: Pull complete
Digest: sha256:3d78bbf6043f085db058560493fc77236dde10a4e3b0533215278ca38d0bf42f
Status: Downloaded newer image for openjdk:latest
docker.io/library/openjdk:latest
PS C:\Users\dhidalgo> docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
openjdk             latest              b255bbd4a82d        6 weeks ago         490MB
PS C:\Users\dhidalgo>
```

Figura 41: Creació imatge Docker, part 1

Es crea el fitxer *Dockerfile* amb el següent contingut:

Fitxer *Dockerfile*:

```
FROM openjdk:8-jre
RUN apt-get update && apt-get install -y gettext-base python3
RUN curl -O https://bootstrap.pypa.io/get-pip.py && python3 get-pip.py && pip
install awscli --upgrade
COPY entrypoint.sh /entrypoint.sh
WORKDIR /
RUN chmod +x /entrypoint.sh
```

```
ENTRYPOINT ["/entrypoint.sh"]
```

Fitxer *entrypoint.sh*:

```
#!/bin/bash
if [ -z "$S3_URL" ]; then
    echo "S3_URL is empty"
else
    echo "S3_URL is $S3_URL"
    if [ -n "$AWS_ACCESS_KEY_ID" ]; then
      export AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      export AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
    fi
    export AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}
    aws s3 cp $S3_URL app.jar
    export JAVA_MAX_MEM_RATIO=100
    echo "Using JAVA_OPTS = $JAVA_OPTS"
    java $JAVA_OPTIONS $JAVA_OPTS -jar app.jar
fi
```

Es construeix la imatge amb la següent comanda:

```
docker build --tag="973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas" .
```

Llancem la comanda de login:

```
Invoke-Expression -Command (Get-ECRLoginCommand -Region eu-west-1).Command
```

```
PS C:\WINDOWS\system32> aws configure
AWS Access Key ID [None]: AKIA
AWS Secret Access Key [None]: d9                                        0
Default region name [None]: eu-west-1
```

Figura 42: Creació imatge Docker, part 2

Abans de llençar la comanda, hem de tenir instal·lades les aws cli, per a Power Shell:

```
Install-Module -Name AWSPowerShell
```

Fem un push al ECR amb la següent comanda:

```
docker push 973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas:latest
```

```
PS C:\WINDOWS\system32> Invoke-Expression -Command (Get-ECRLoginCommand -Region eu-west-1).Command
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
Login Succeeded
PS C:\WINDOWS\system32> docker push 973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas:latest
The push refers to repository [973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas]
18388d9062ae: Pushed
467b80542519: Pushed
d5b70e35257b: Pushed
b690201c509b: Pushed
6a00eae1d01d: Pushed
cb5d7b073fde: Pushed
8bacec4e3446: Pushed
26b1991f37bd: Pushed
55e6b89812f3: Pushed
latest: digest: sha256:4006aeec8d689dbe6d64dbd7fe27715cfd8a1344a6513f185fb2674c72d6b4f7 size: 2420
PS C:\WINDOWS\system32>
```

Figura 43: Creació imatge Docker, part 3

Ja tenim la imatge Docker en ECR.

Bucket d'S3

Es necessita un lloc on poder deixar els compilats. AWS ens proporciona un servei d'emmagatzematge d'objectes anomenat Amazon Simple Storage Service (S3) que ens ofereix escalabilitat, disponibilitat de dades, seguretat i alt rendiment.

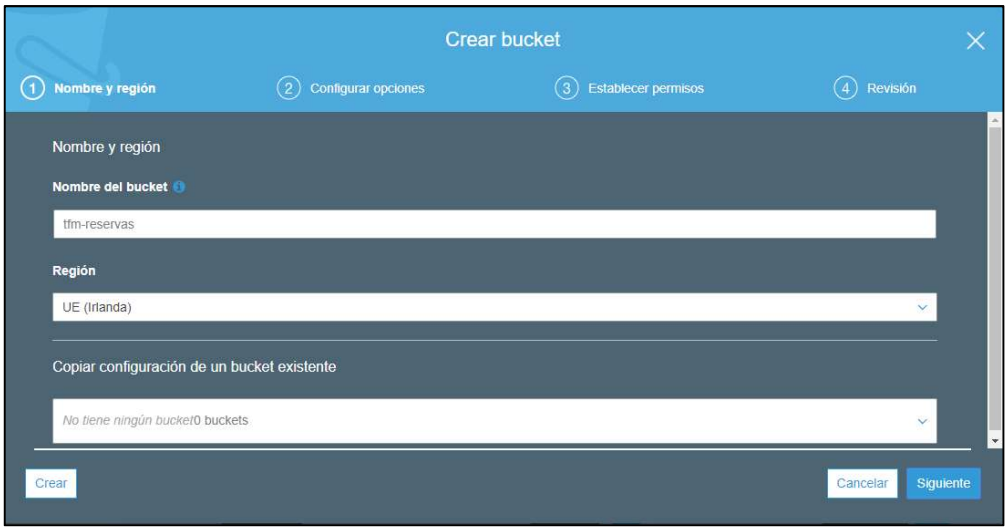


Figura 44: Creació Bucket S3, part 1

Creem un nou usuari amb la política *AmazonS3FullAccess*.

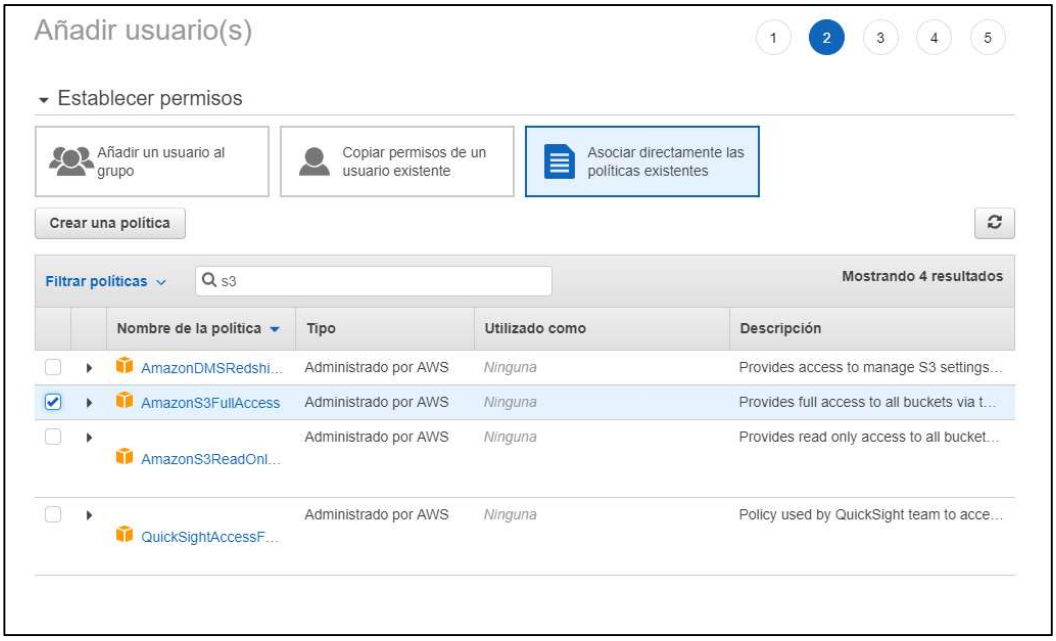


Figura 45: Creació usuari IAM amb política S3 Full Access

Utilitzarem les claus *access-key* i *secret-access-key* d'aquest usuari per a accedir al Bucket S3 per a agafar el compilat.

En aquest punt, pujarem a ma el compilat al Bucket creat i amb la definició de tasca l'agafarà del Bucket i el desplegarà en l'instancia / les instancies que li indicarem.

Task definition i EC2 Service

Per a executar contenidors Dockers en AWS es necessita una Task Definition. En la Task Definition definim una sèrie de paràmetres com a:

- La imatge Docker que s'utilitzarà amb cada contenidor amb la seva tasca.
- La quantitat de CPU de memòria que es vol utilitzar amb cada tasca o cada contenidor dins d'una tasca.
- El tipus de llançament, que determina la infraestructura en la que s'allotja les seves tasques.

- El mode de xarxa del Docker.
- Si la tasca ha de continuar executant-se si el contenidor falla o finalitza.
- La comanda que el contenidor ha d'executar a l'inicialitzar-se.
- Els volums de dades que s'han d'utilitzar amb els contenidors en la tasca.
- El rol IAM que les tasques han d'utilitzar.

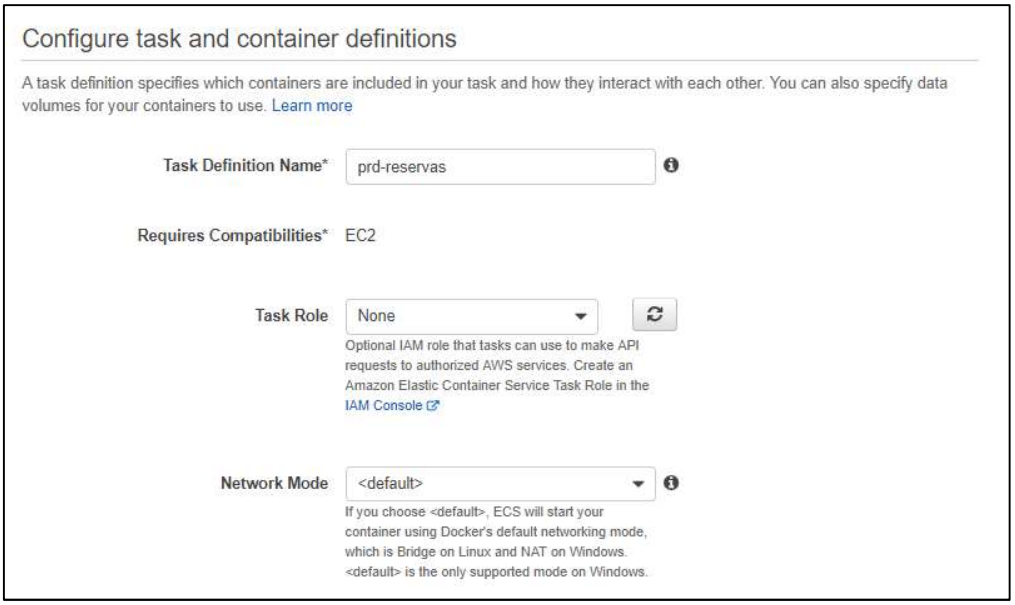


Figura 46: Creació Task Definition, part 1

S'ha de crear un container. En el container es defineixen variables d'entorn que les necessitem per a l'arxiu que hem creat per a la imatge Docker (*entrypoint.sh*).

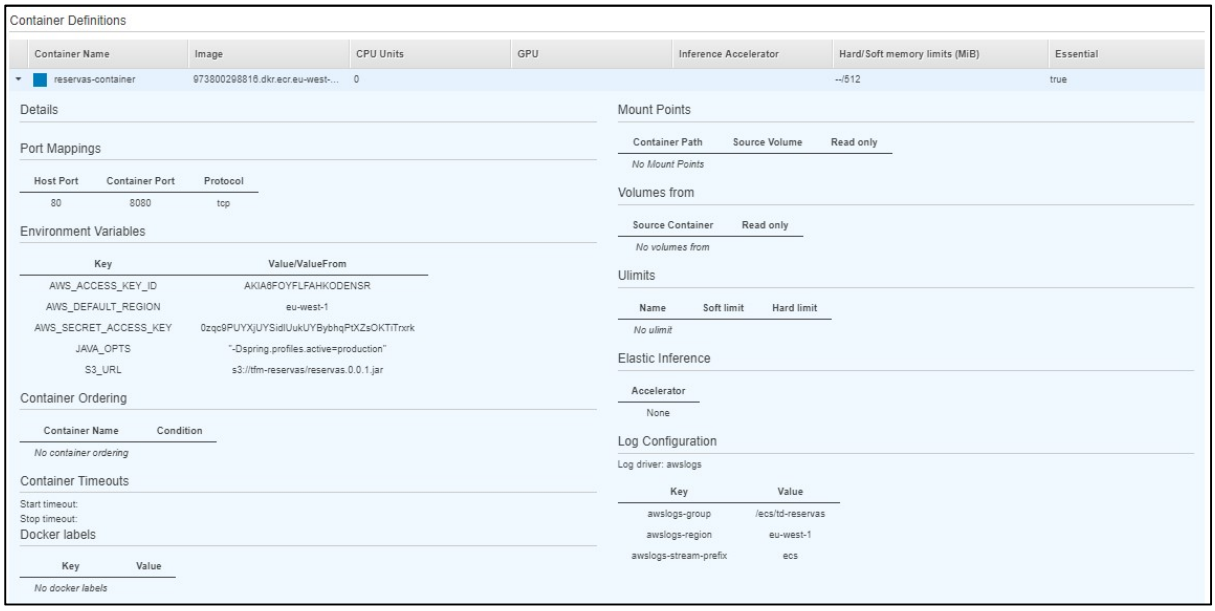
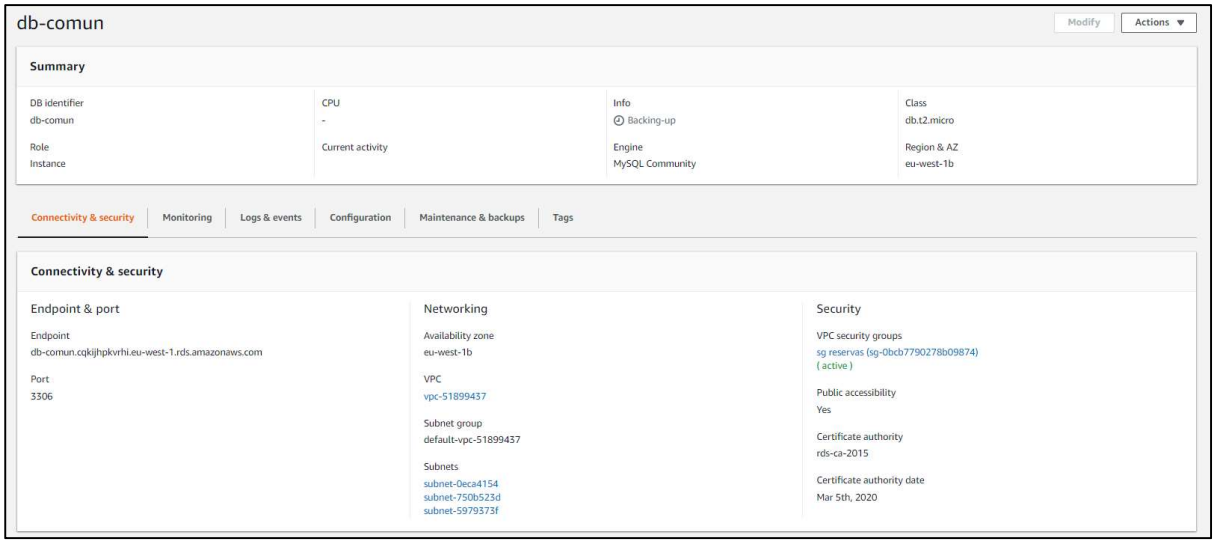


Figura 47: Creació Task Definition, part 2

Amazon RDS

Amazon Relation Database Service, és un servei que subministra capacitat rentable i escalable al mateix temps que automatitza les tasques administratives, com el aprovisionament de hardware, la configuració de base de dades, la implementació de fix i la creació de copies de seguretat.

RDS està disponible per a varies instancies de base de dades i proporciona 6 motors de base de dades. En el nostre cas s'ha escollit MySQL com a motor de base de dades.



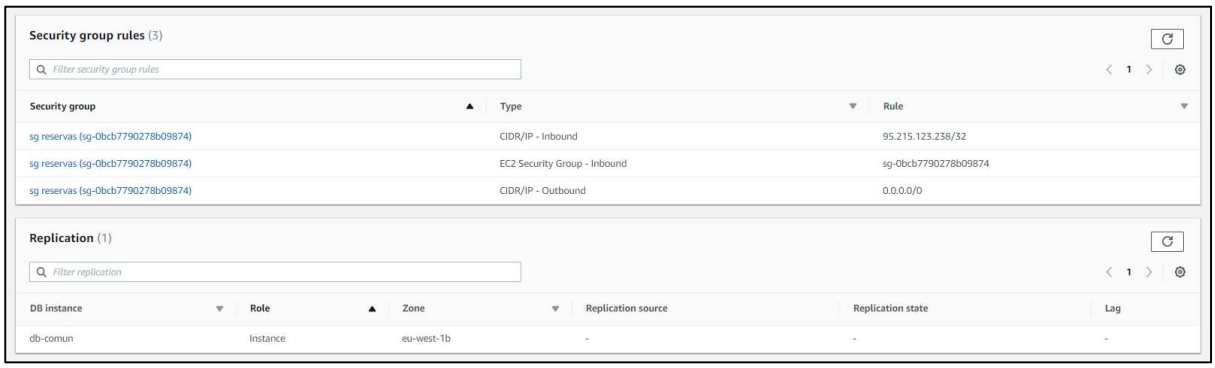


Figura 48: Creació Amazon RDS, part 1

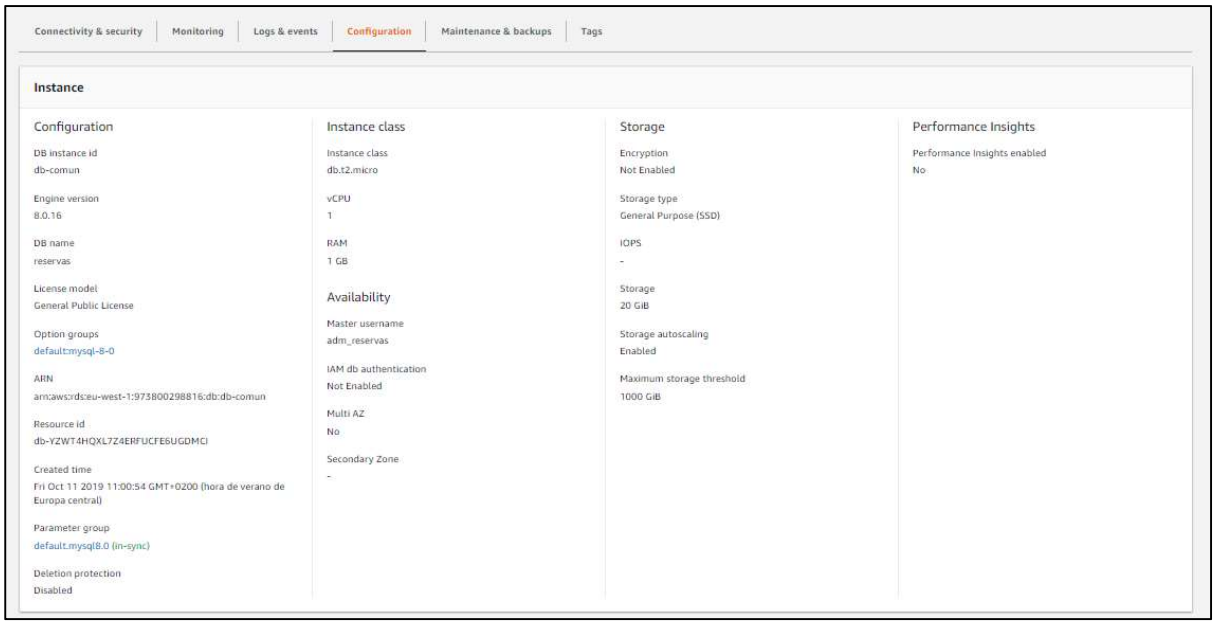


Figura 49: Creació Amazon RDS, part 2

Modificar security group per a autoritzar totes les instancies del security group.

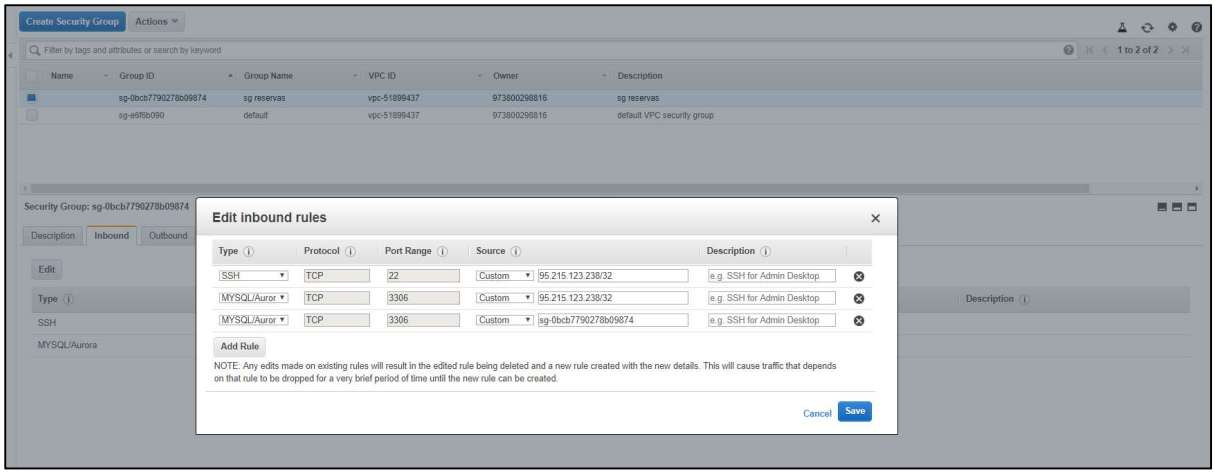


Figura 50: Creació Amazon RDS, part 3 modificació del Security Group

Posada en producció

Configurem una eina FTP (WinSCP) per a la connexió amb el nostre S3.

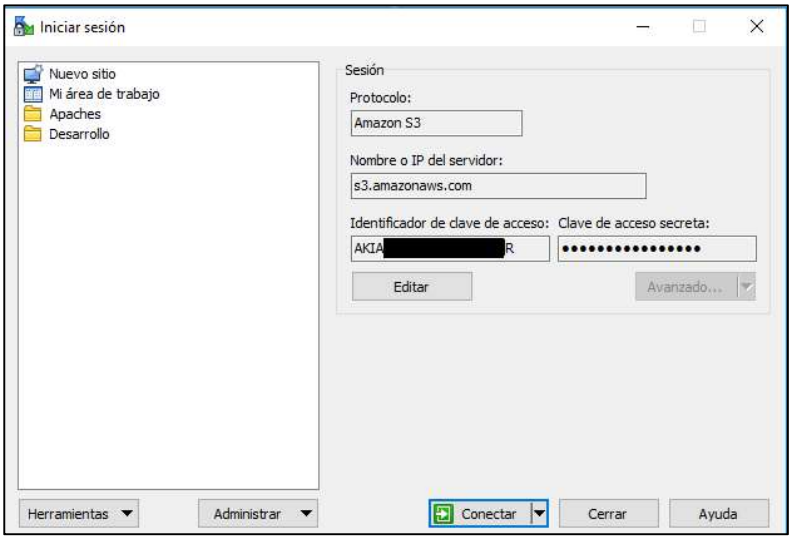


Figura 51: Posada en producció, configuració WinSCP

Es carrega l'arxiu a desplegar al Bucket S3, s'ha d'anar amb compte ja que s'ha de modificar la *Task Definition* amb el nom del jar que es vol desplegar. *Veure variables d'entorn en l'apartat Task Definition*.

Figura 52: Posada en producció, pujada arxiu .jar manualment al Bucket S3

Informem en el target group on pot realitzar el *Health check*.

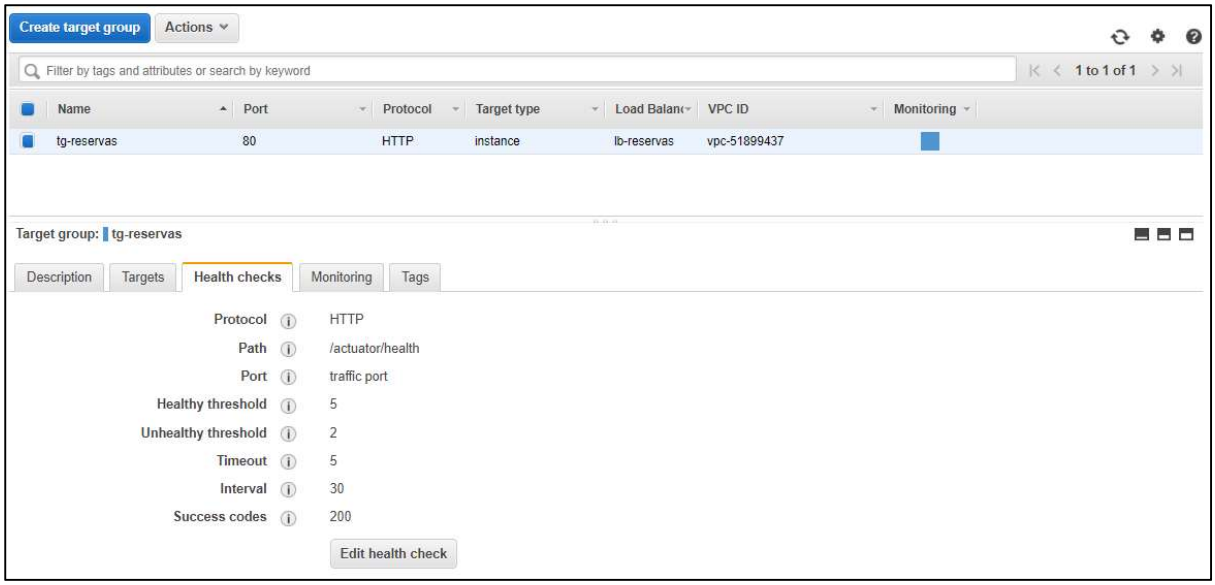


Figura 53: Posada en producció, informant Health check

S'ha d'afegir el Security Group del Load Balancer als Inbounds de l'instancia.

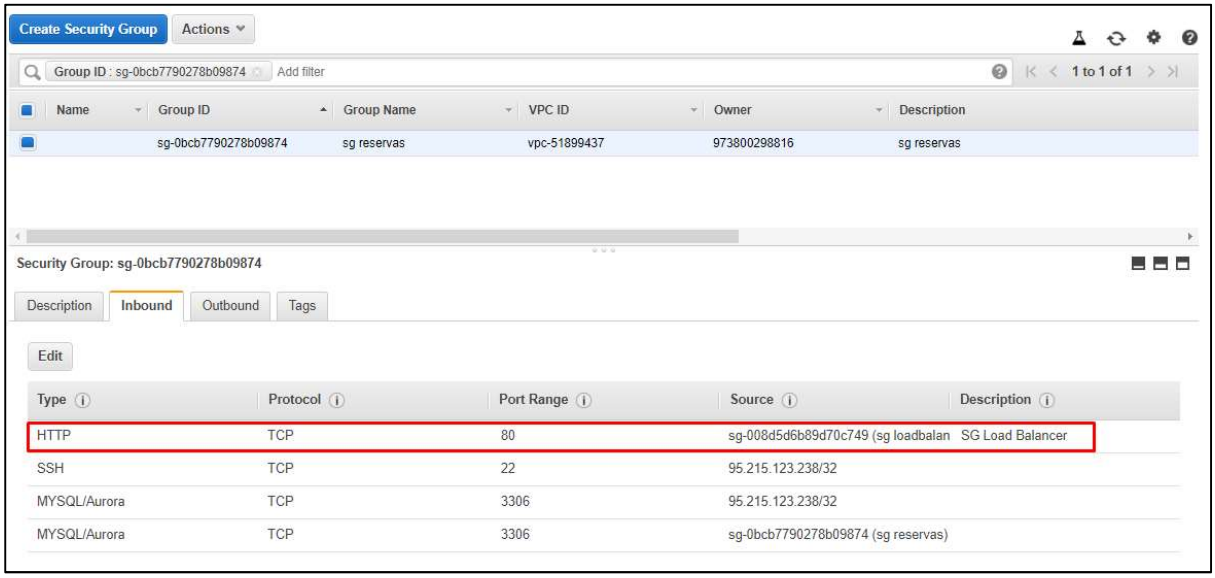


Figura 54: Posada en producció, afegir Security Group a l'instancia.

Comprovem que funciona des del nostre navegador.

http://lb-reservas-896768456.eu-west-1.elb.amazonaws.com/actuator/health

Figura 55: Posada en producció, comprovant funcionament

Route 53 i Certificate Manager

Se ha comprat un domini: **tfmaster.net**

Figura 56: Compra de domini tfmaster.net

En Certificate Manager, afegim els dominis i/o subdominis que volem demanar una certificat SSL

Figura 57: Certificats SSL sobre domini i subdominis tfmaster.net

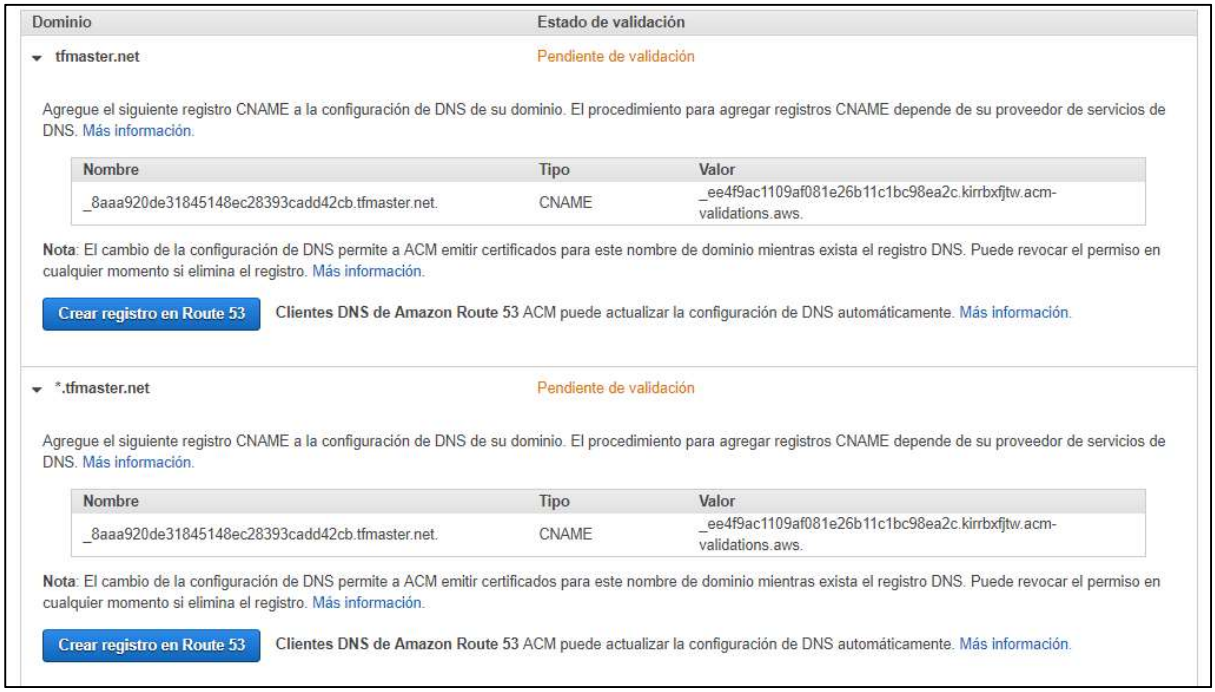


Figura 58: Resultats dels certificats SSL sobre domini i subdominis tfmaster.net

En la consola d'administració de Route 53 es crea un nou conjunt de registres:

Figura 59: Crear un nou conjunt de registres

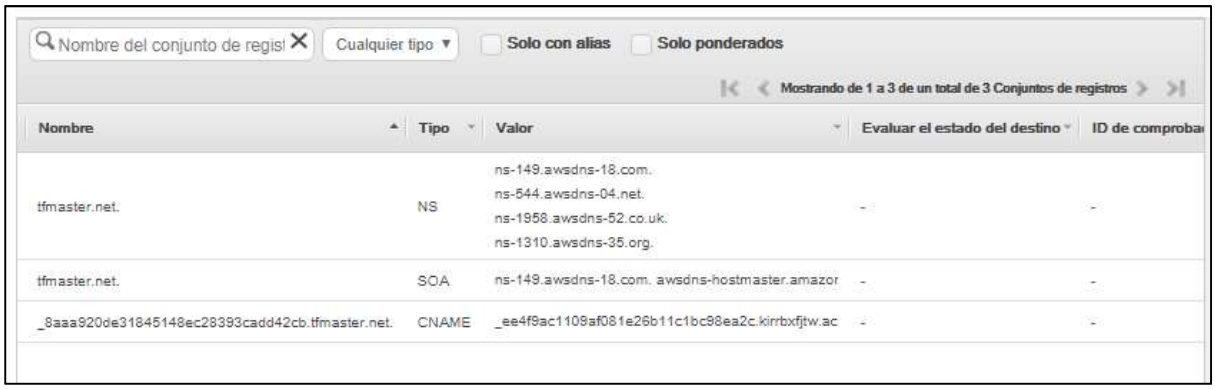


Figura 60: Conjunts de registres resultant

En el Load Balancer que es el nostre punt d'entrada, s'ha d'afegir un nou *Listener*:

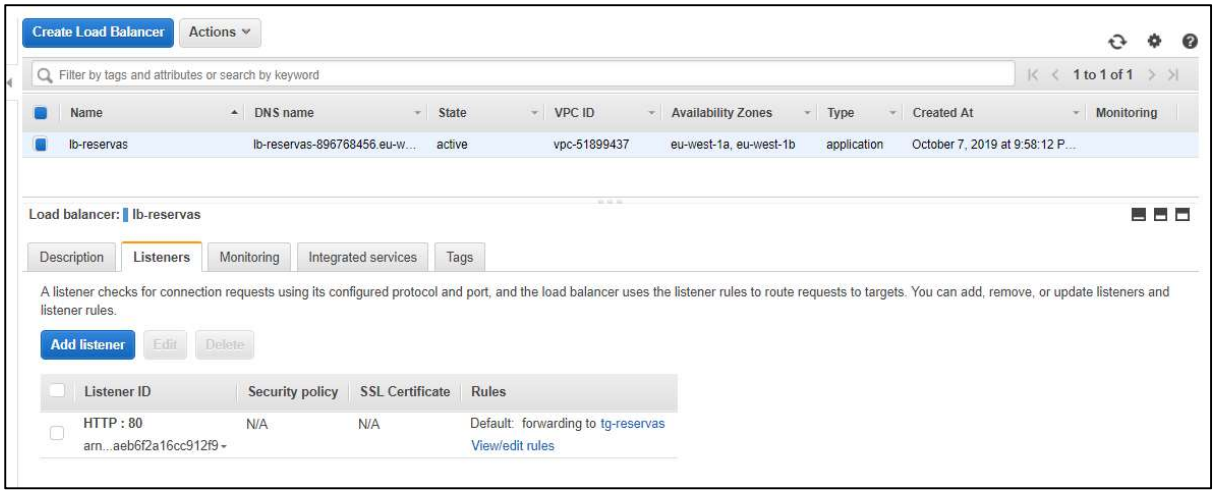


Figura 61: Afegir un nou Listener en el Load Balancer

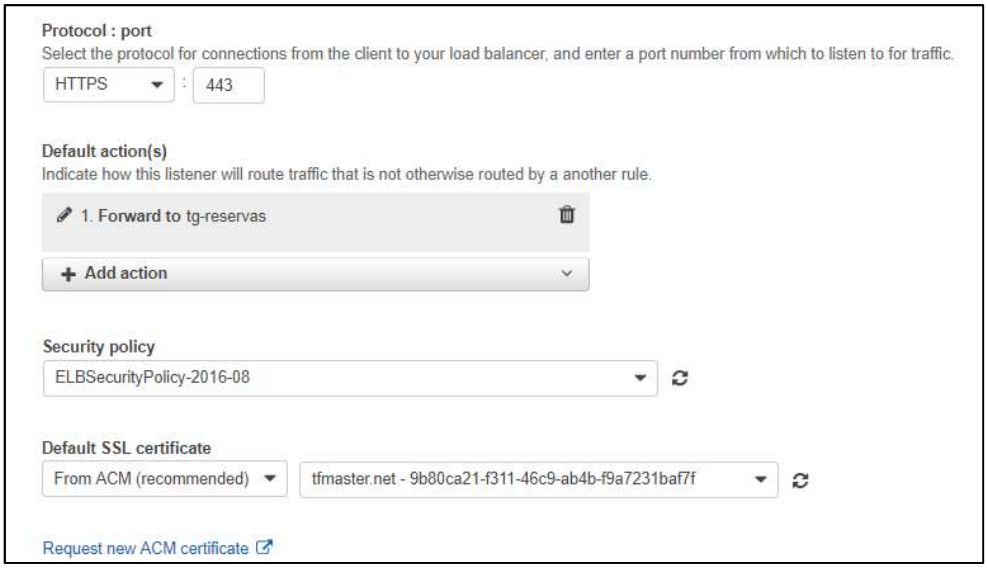


Figura 62: Dades nou *Listener*

Ara, si volem donar un nom de subdomini per a la nostre aplicació, s'ha de crear un nou conjunt de registre per a crear un nou subdomini per a l'aplicació:

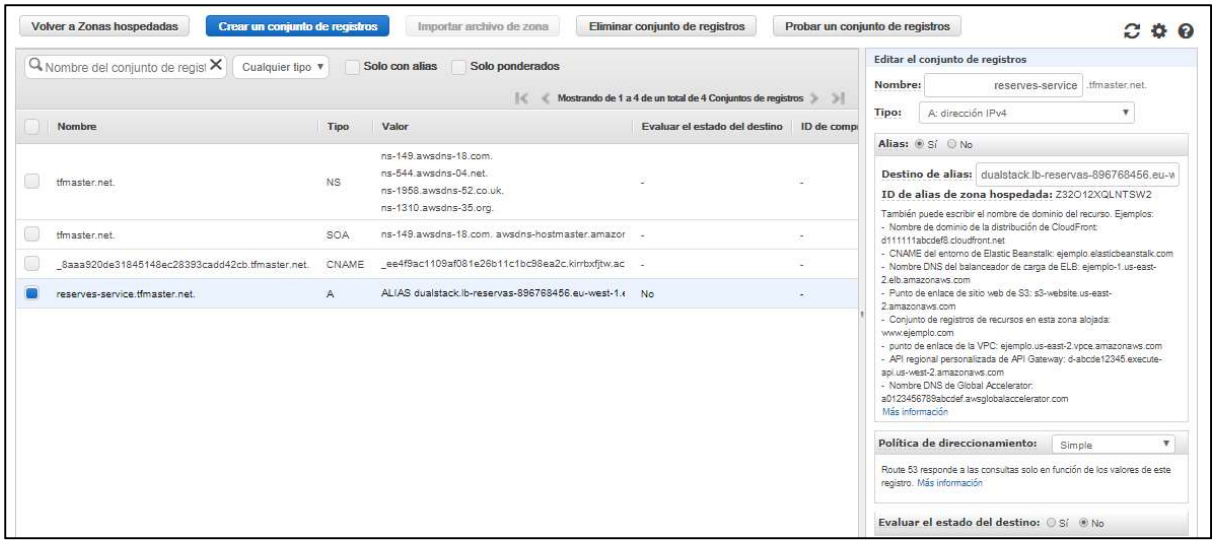


Figura 63: Crear un subdomini per a l'aplicació

Ara podem accedir a l'aplicació per el nom del subdomini i amb el port 443:

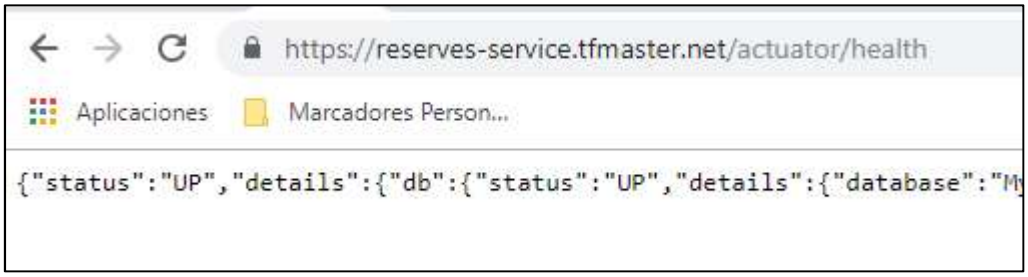


Figura 64: Publicat pel port 443 amb un subdomini

Desplegar Front-end

Hem de crear un bucket privat per a contenir els arxius de la part front-end.

Figura 65: Nou bucket per a contenir part front-end

Abans de crear una nova distribució, hem de crear un certificat en Viriginia, ja que Cloud Front només es allotjat en aquesta zona.

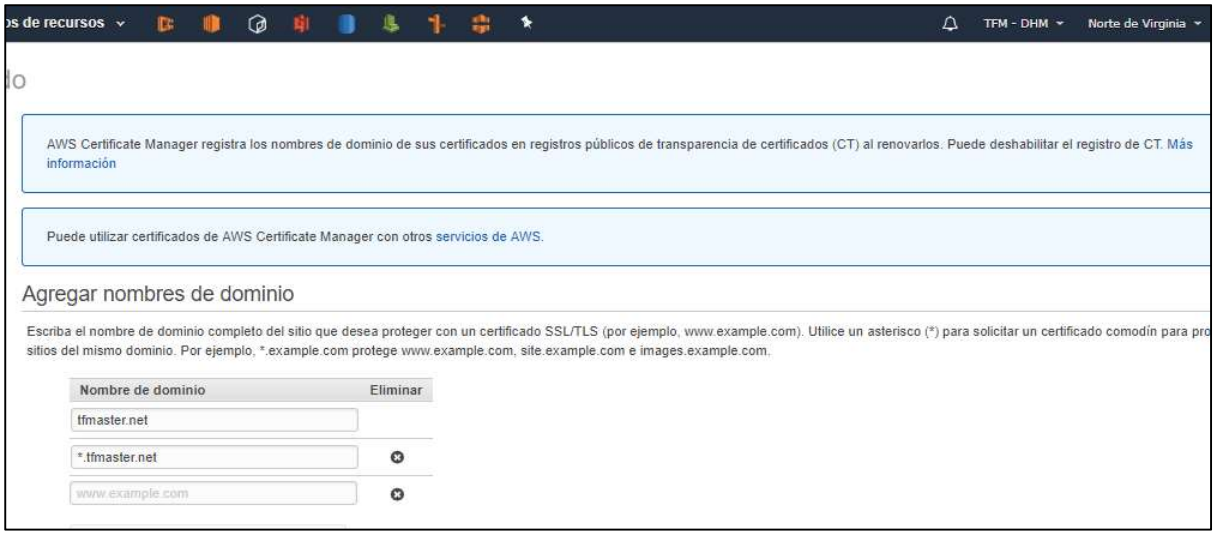


Figura 66: Nou certificat a la zona de Virginia

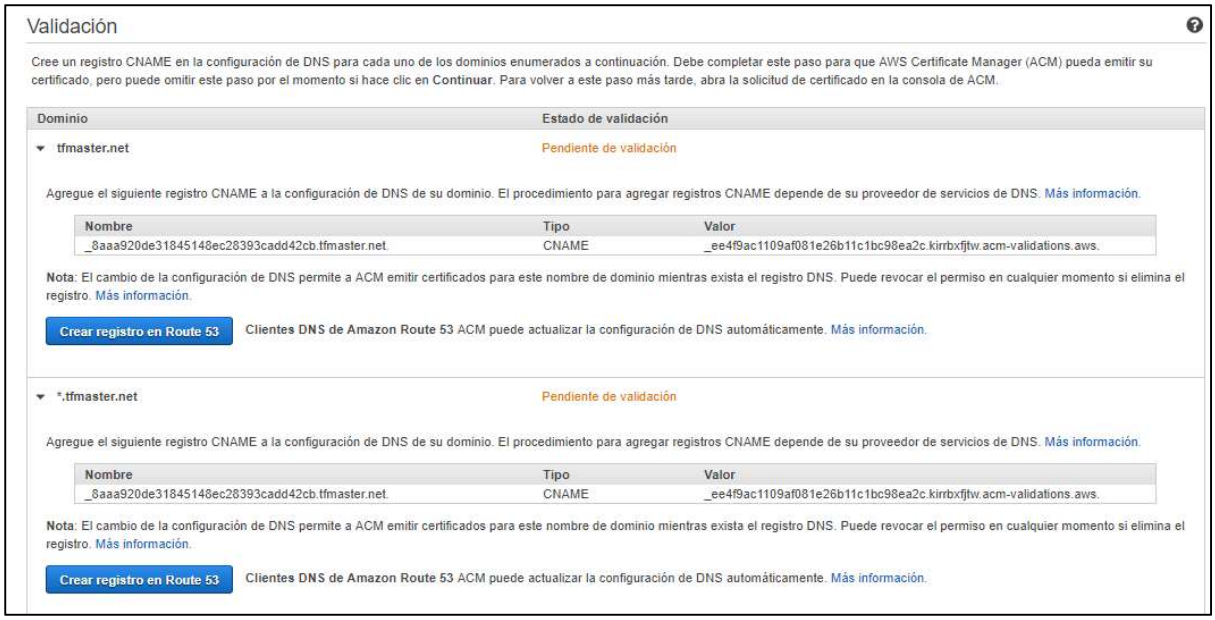


Figura 67: Dades del nou certificat a la zona de Virginia

Es crea amb el mateix CNAME que l'anterior.

Creem una nova distribució web des del servei Cloud Front.

Posem imatges de les dades més rellevants:

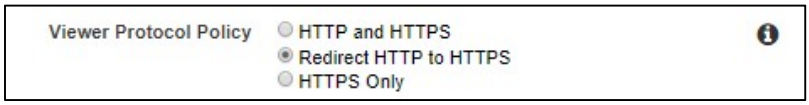


Figura 68: Crear nova distribució, part 1

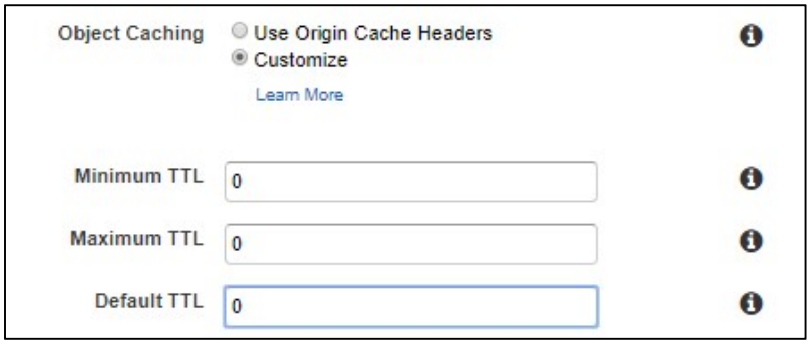


Figura 69: Crear nova distribució, part 2

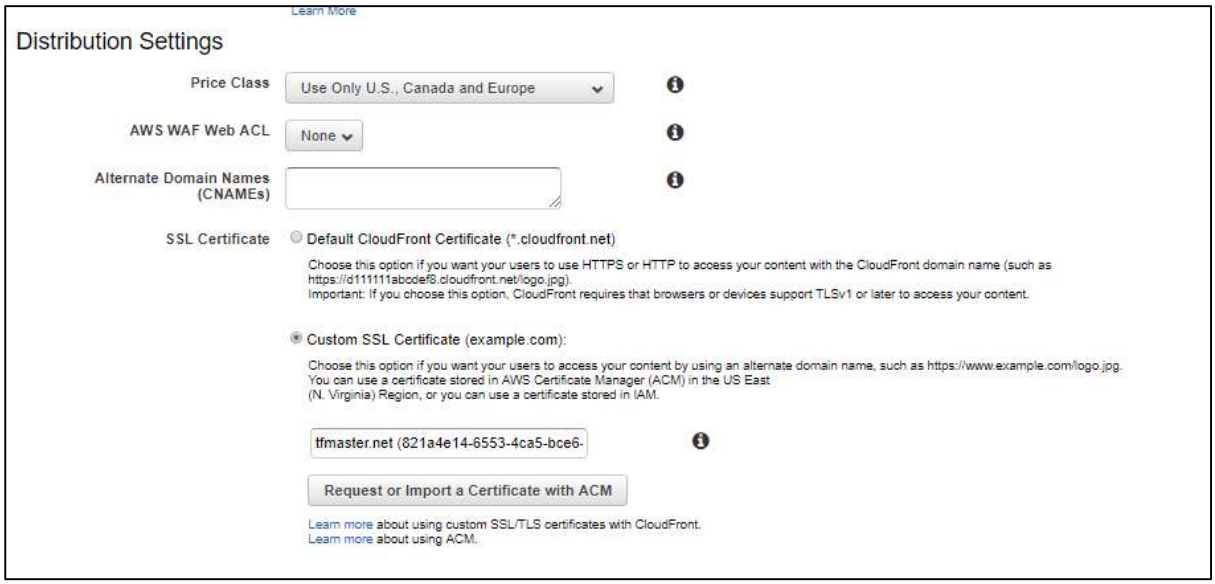


Figura 70: Crear nova distribució, part 3

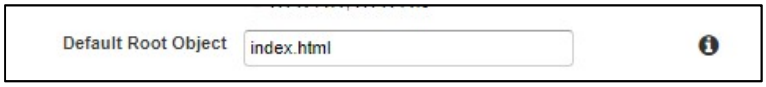


Figura 71: Crear nova distribució, part 4

Figura 72: Crear nova distribució, resultat

Amb WinSCP pugem els arxius del front-end al nou bucket:

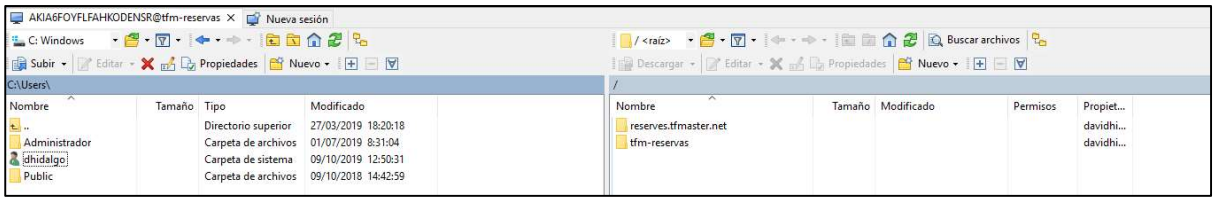


Figura 73: WinSCP per a pujar arxius al bucket privat

Hem de crear un nou Conjunt de Registres:

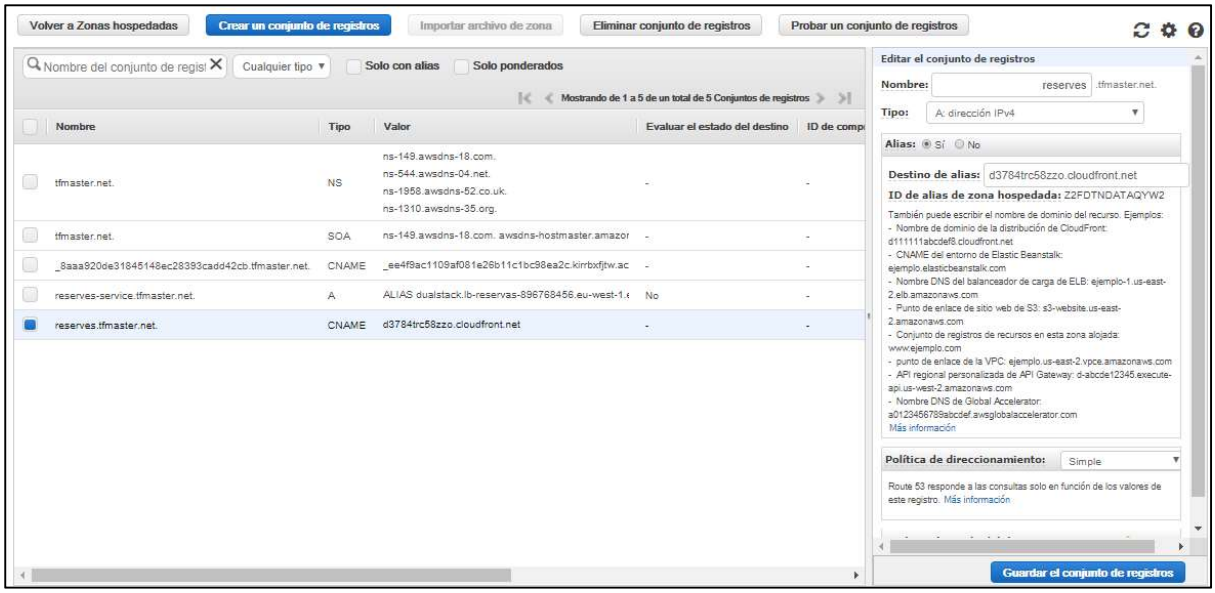


Figura 74: Conjunt de registres per a la distribució

I dirigir-lo a la distribució (Cloud Front) que acabem de crear.

Ja podem accedir per la següent url: https://reserves.tfmaster.net/

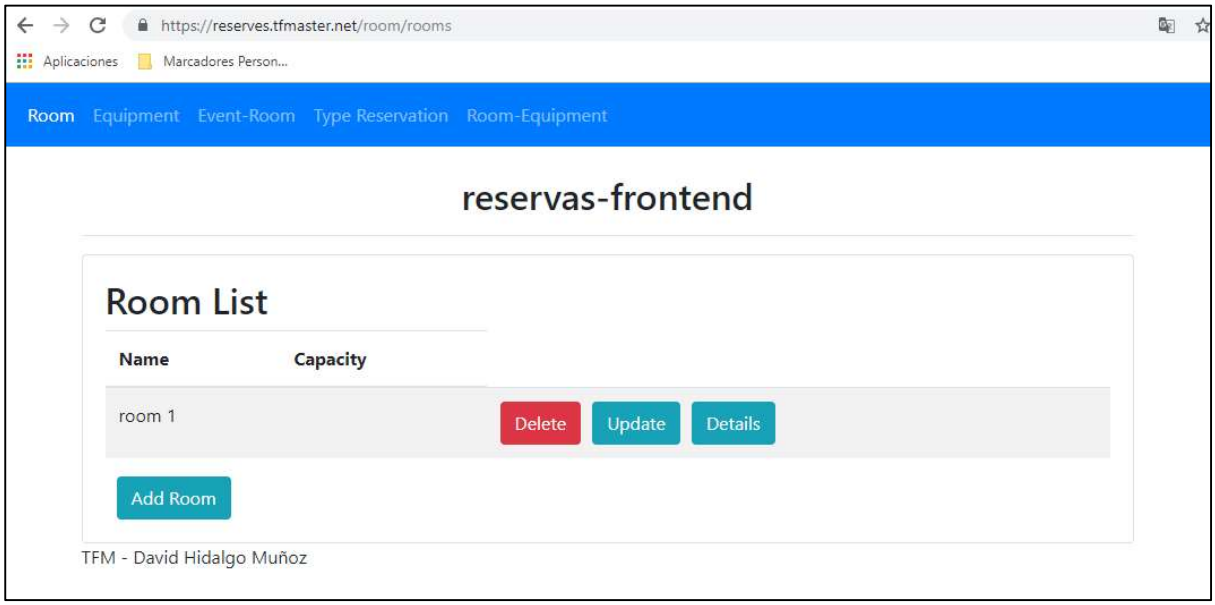


Figura 75: Aplicació en entorn productiu

Auto escalat

Per a la realització de l'auto escalat, hem de crear mètriques de Cloud Watch.

Comencem creant les mètriques del clúster. S'han de crear dues mètriques del clúster, una quan la CPU augmenti, i una altre quan la CPU torni a un estat "normalitzar" per deixar el servei com a un inici.

Per això hem d'anar al servei CloudWatch, a l'apartat mètriques i seleccionar on volem realitzar mètriques, el clúster del servei del tipus *MemoryUtilization*.

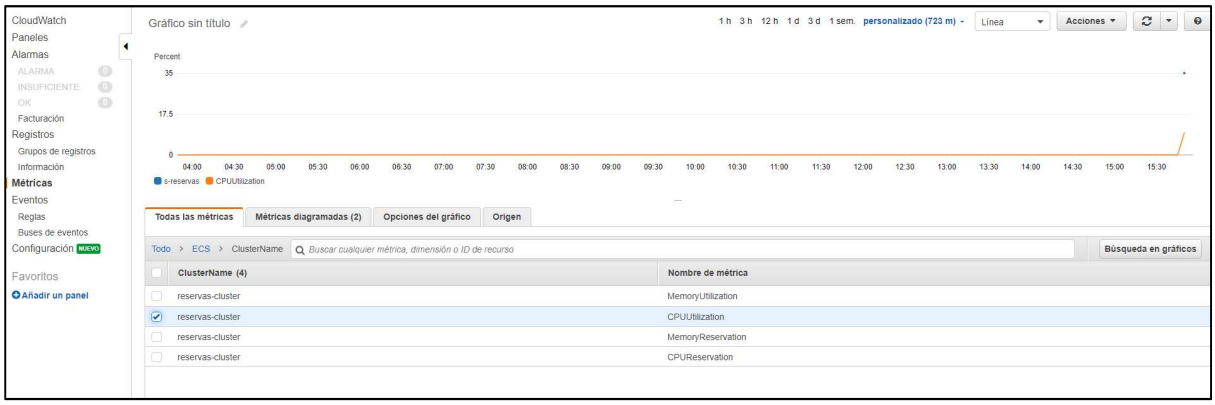


Figura 76: Auto escalat, mètriques CloudWatch part 1

Configurem que la mètrica sigui quan la utilització de la CPU sigui major o igual al 30%

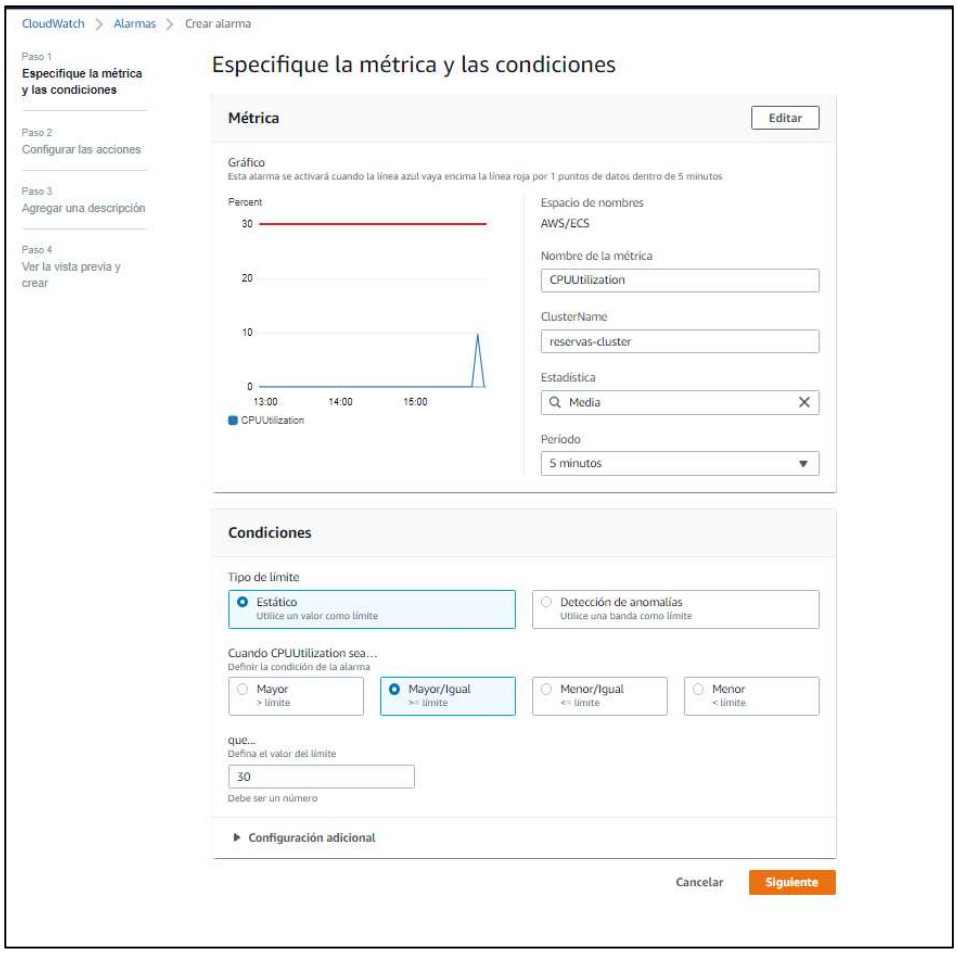


Figura 77: Auto escalat, mètriques CloudWatch part 2

Podem configurar una acció. L'acció seleccionada és una subscripció SNS que envia un correu a una direcció d'email (la configuració de la subscripció SNS s'explica en l'apartat *10.11. AWS Lambdes per a realitzar snapshots*).

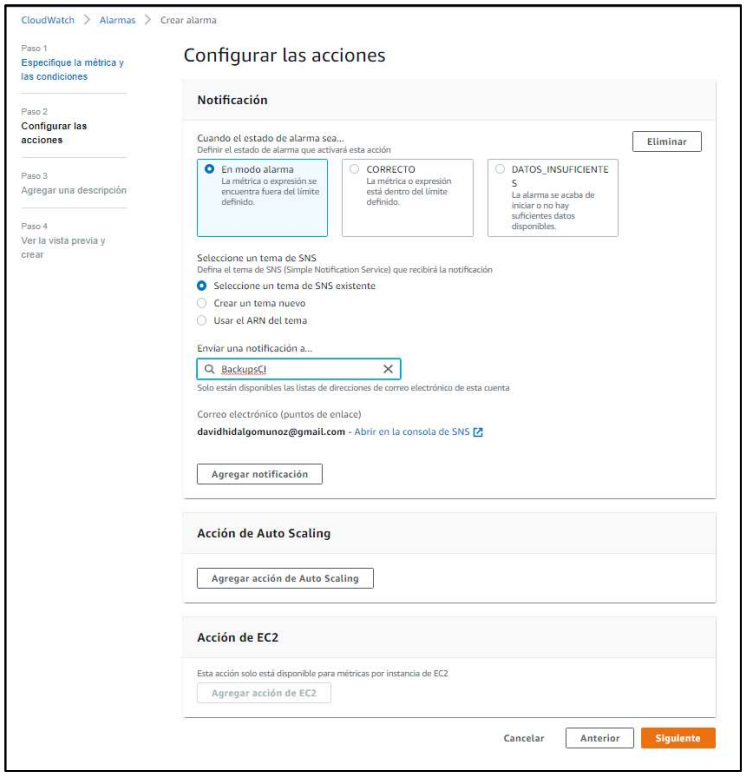


Figura 78: Auto escalat, mètriques CloudWatch part 3

En aquest punt ja tenim una mètrica creada que quan la CPU sigui >= 30%, ens notifica enviant un correu mitjançant una subscripció SNS. Aquesta mètrica l'anomenem *ClusterReservesASGCPUOut*.

Hem de repetir les mateixes passes però amb la diferencia que la mètrica serà quan la CPU sigui <= 15%. Aquesta mètrica l'anomenem *ClusterReservesASGCPUIn*.

Hem de crear altres dues mètriques molt semblants, però ara sobre el service del clúster:

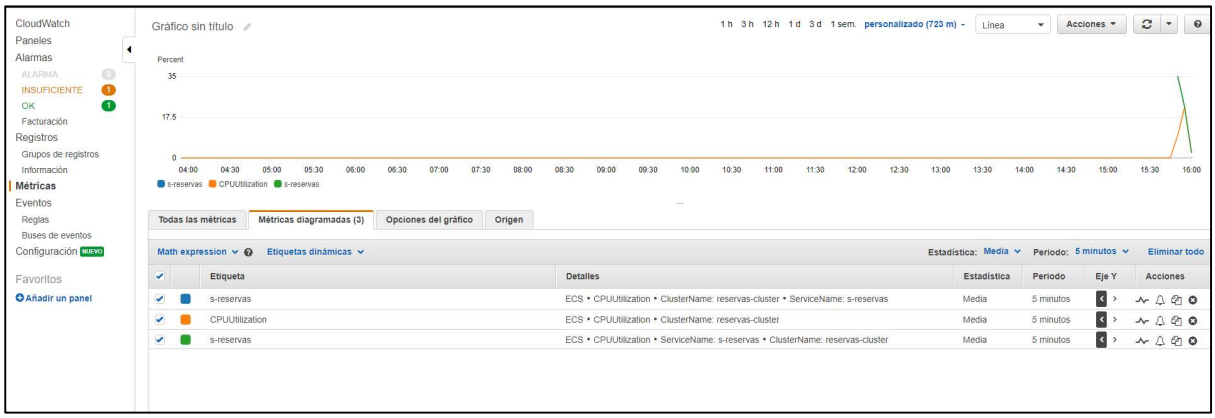


Figura 79: Auto escalat, mètriques CloudWatch part 4

La primera mètrica serà quan la CPU sigui >= al 60%. L'anomenem *ServiceReservesASGCPUOut*. La segona mètric sobre el service serà quan la CPU sigui <= al 30%. L'anomenem *ServiceReservesASGCPUIn*.

En el servei EC2, secció *autoscaling group*, apartat instancies, hem d'activar l'opció de *Configurar la protecció del escalat descendent*.

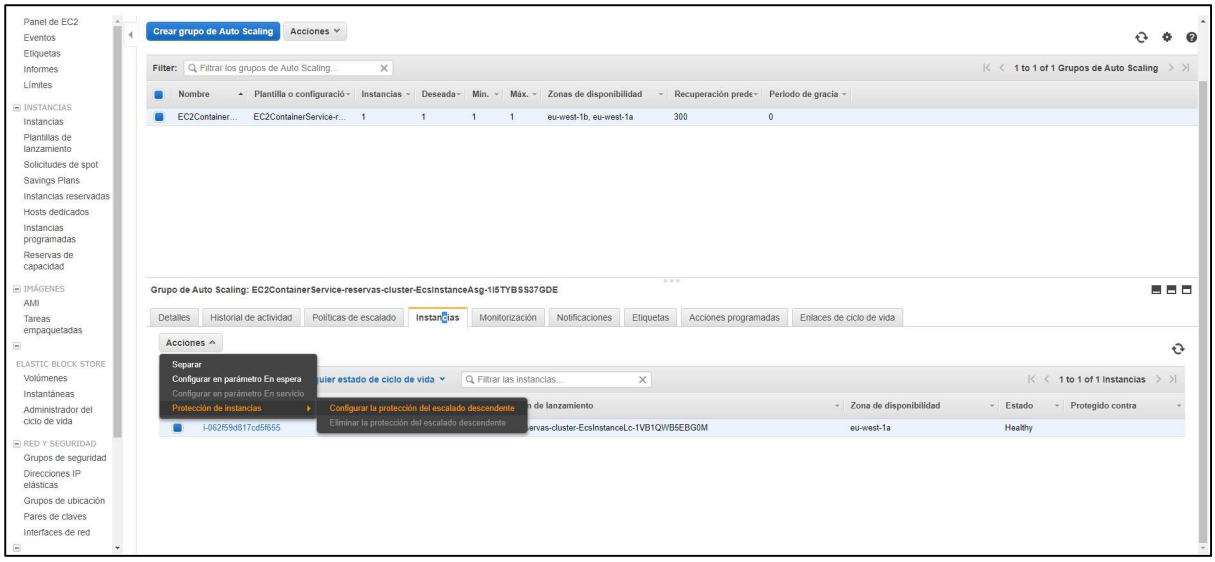


Figura 80: Auto escalat, autoscaling group, part 1

Activant aquesta opció, garantim que sempre hi haurà una instancia, ja que si activem directament les mètriques, farem que quan baixi del 15% d'utilització de CPU ens destruirà totes les instancies i al menys volem que sempre 1 quedi activa sempre.

En el servei EC2, secció *autoscaling group*, anem a la pestanya polítiques d'escalat i afegim una nova política anomenada *policyClusterReservesCPUOut*.

Figura 81: Auto escalat, autoscaling group, part 2

En aquesta política d'escalat configurem en "Executar la política quan" la mètrica del clúster anteriorment creada de CPU >=30 i en "Realitzar l'acció", afegir 1 unitat.
Per tant, quan la CPU del clúster sigui >= 30%, es crearà una instancia nova però encara no l'activarà.

Creem una nova política d'escalat semblant a l'anterior anomenada *policyClusterReservesCPUIn*. En aquesta cas associem la mètrica de quan la CPU <= 15% i l'acció realitzada eliminar 1 unitat.

Figura 82: Auto escalat, autoscaling group, part 3

Amb aquestes accions, la pestanya de Polítiques d'escalat resultaria:

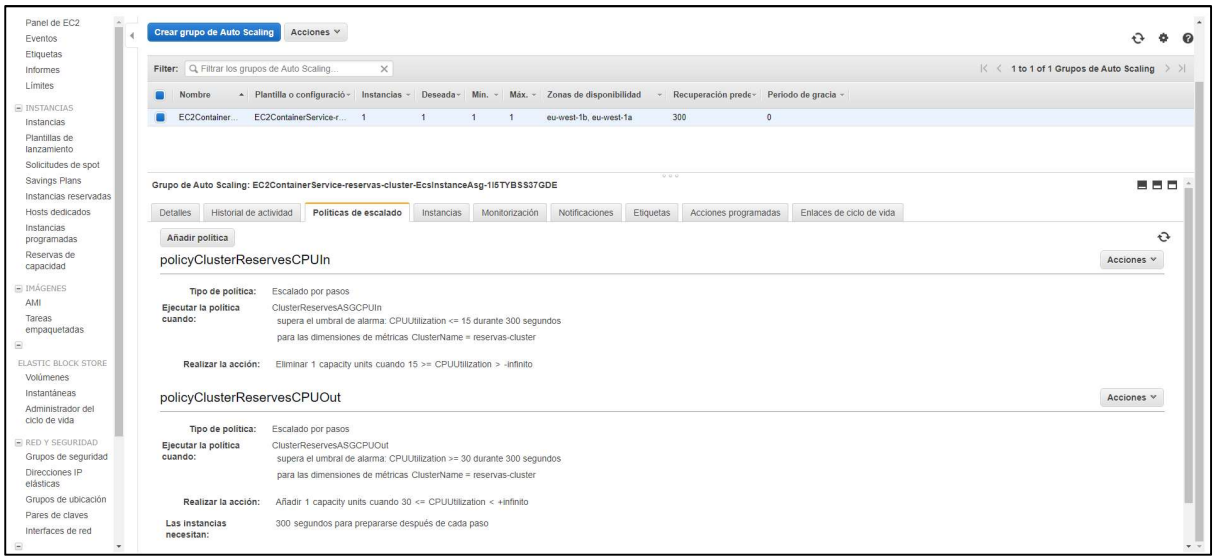


Figura 83: Auto escalat, autoscaling group, part 4

Finalment editem el grup d'autoscaling i posem com a màxim 4 instancies:

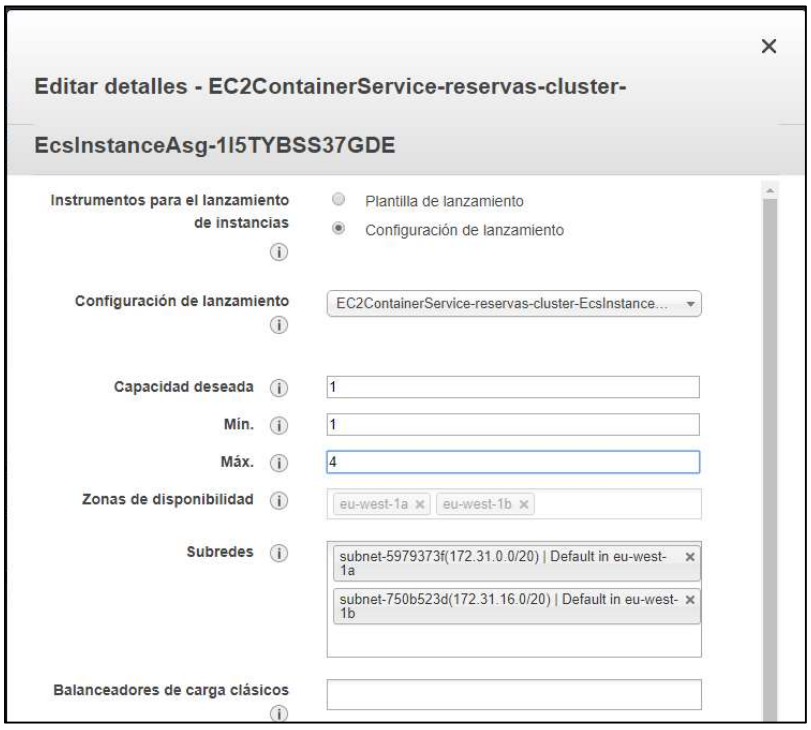


Figura 84: Auto escalat, autoscaling group, part 5

En aquest punt ja tenim que quan la CPU sigui >= 30% ens prepara una nova instancia i la deixa en espera i quan la CPU sigui <= 15%, elimina les instancies creades exceptuant d'una que l'hem protegint amb l'opció de *Configurar la protecció del escalat descendent*. Però tot i havent creant les noves instancies, no s'activen ni donen servei.

Anem al servei ECS, al nostre reserves-cluster i editem el servei s-reserves:

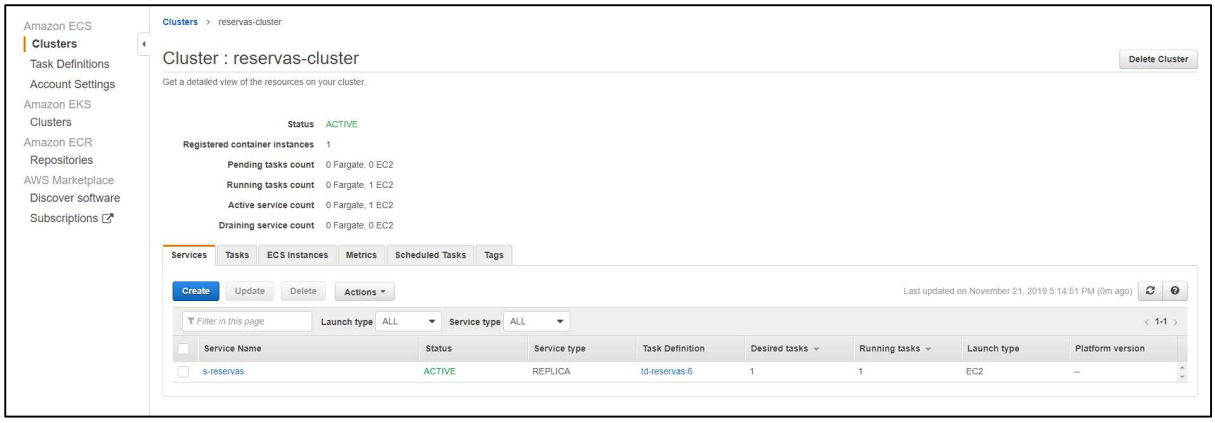


Figura 85: Auto escalat, editant el servei del clúster, part 1

En l'apartat opcional de configurar l'auto escalat del servei, activem l'opció de configuració del servei d'auto escalat. Volem un mínim d'1 instancia, un desitjat d'1 instancia i un màxim de 4 instancies (ja que considerem que amb 4 instancies podrà suportar els pics de demanda).

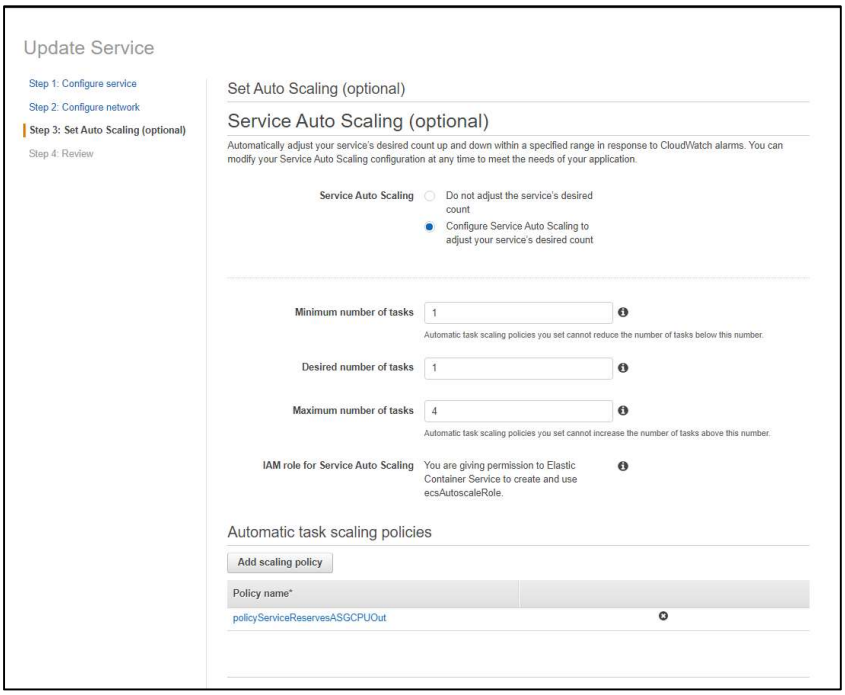


Figura 86: Auto escalat, editant el servei del clúster, part 2

En aquesta mateix punt, en l'apartat d'Automatic task scaling policies, afegir dues polítiques d'escalat.

Creem la política anomenada *policyServiceReservesASGCPUOut*, a la qual li configurem una alarma existent i una acció d'afegir 1 tasca quan la CPU sigui >= 60%.

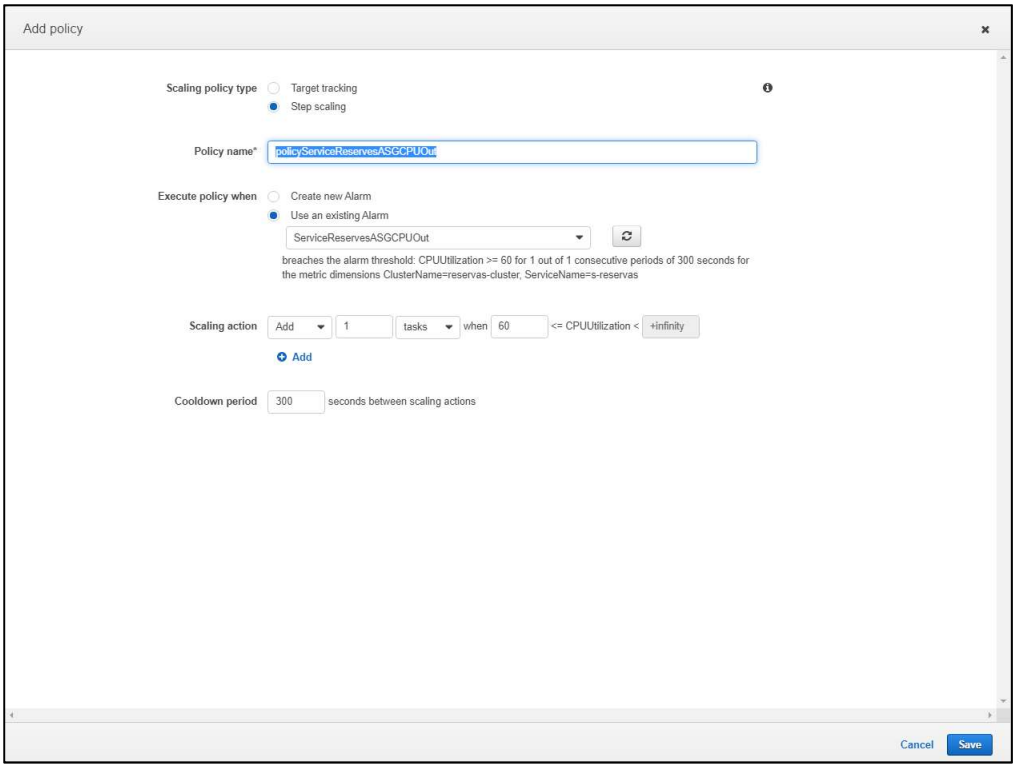


Figura 87: Auto escalat, editant el servei del clúster, part 3

La següent política que hem d'afegir, l'anomenem *policyServiceReservesASGCPUIn* i també l'associem a una de les alarmes que hem creat del servei i com a acció configurem que elimini una tasca quan la CPU sigui <= 30%.

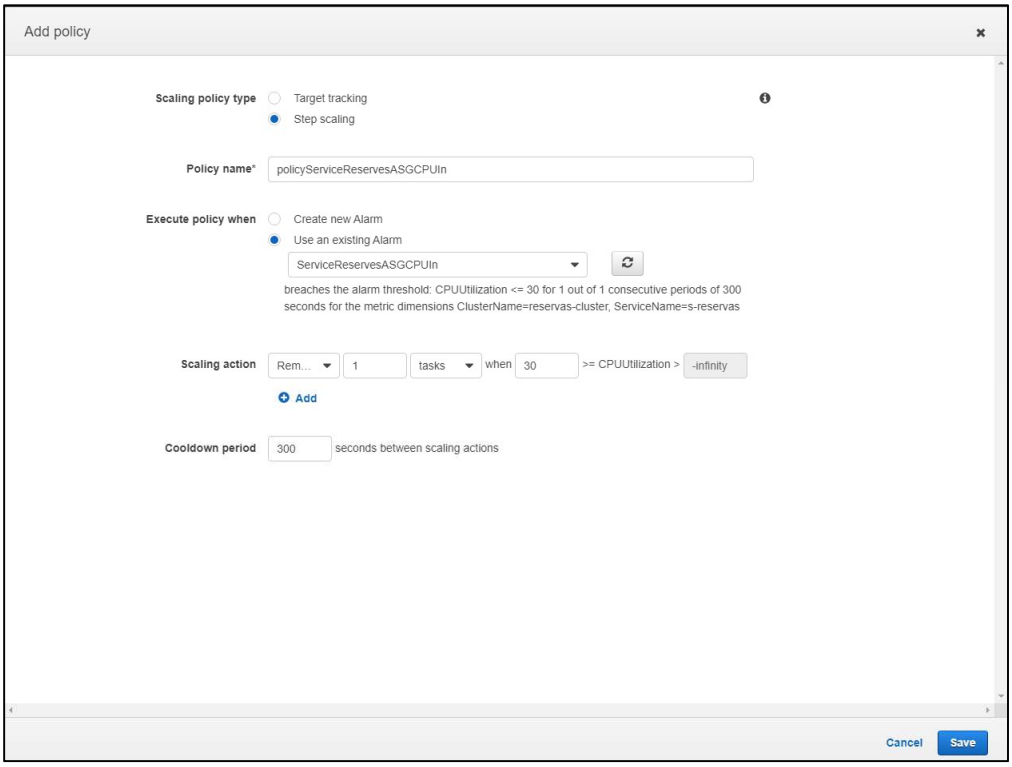


Figura 88: Auto escalat, editant el servei del clúster, part 4

Finalment l'aspecte del pas de configuració de l'escalat del servei amb les dues tasques afegides:

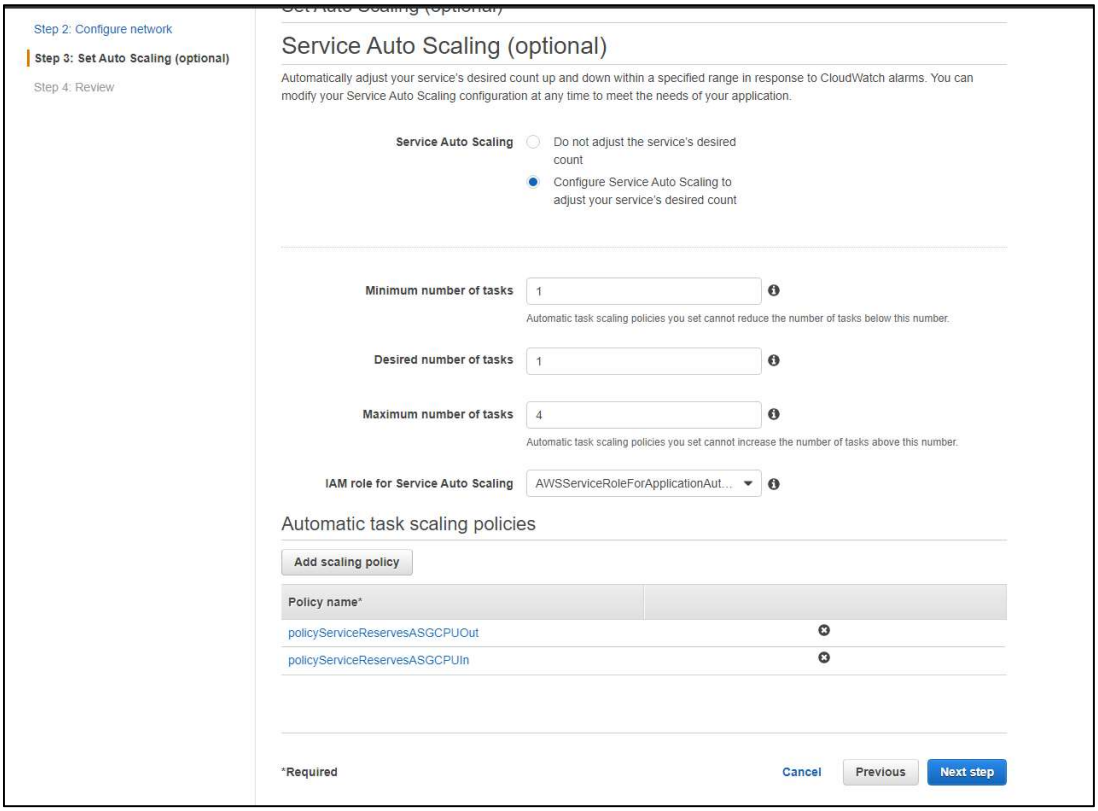


Figura 89: Auto escalat, editant el servei del clúster, part 5

En aquest punt afegim al que teníem que al 60% de CPU, l'instancia creada al 30% es posi activa donant servei. A més, que al 30% de CPU les instancies "extres" deixen de donar servei deixant-les actives, fins al 15% de CPU que les destrueix.

Podem realitzar proves del funcionament des de l'*autoscaling group:*

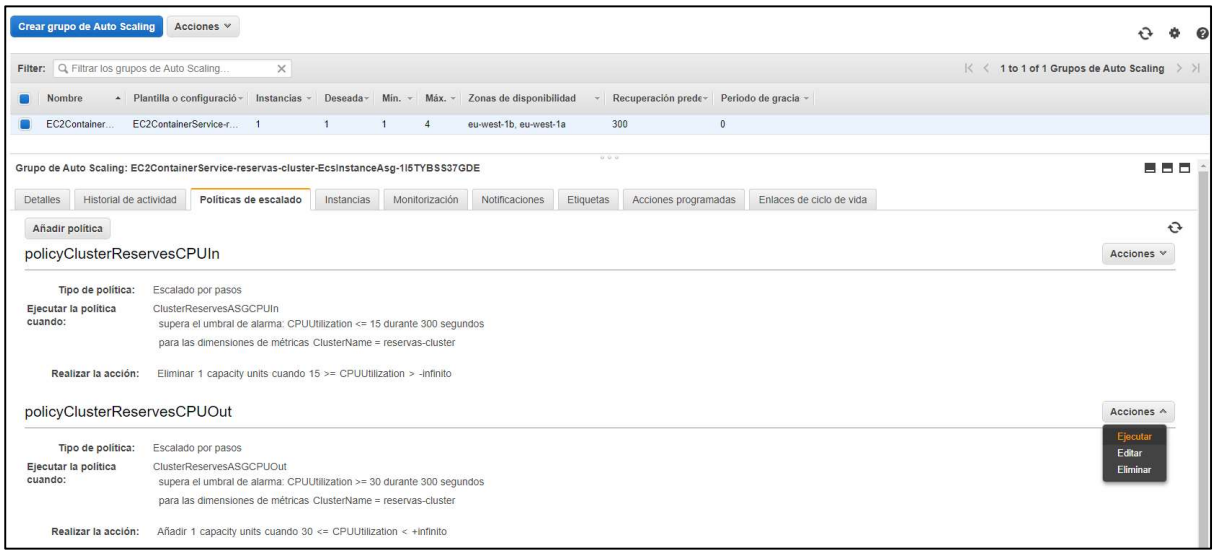


Figura 90: Auto escalat, provant l'escalat

Els resultats són els esperats:

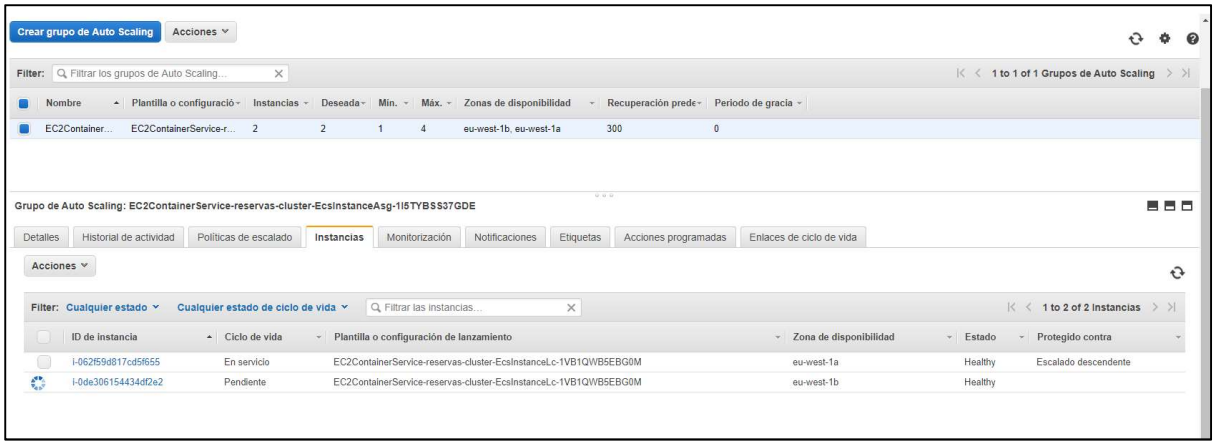


Figura 91: Auto escalat, creant instancia EC2

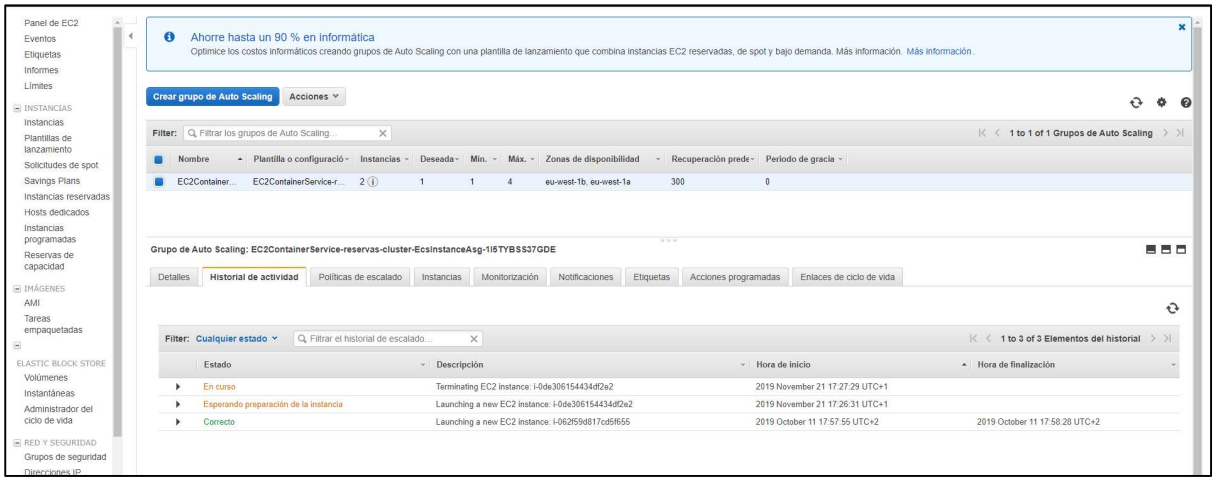


Figura 92: Auto escalat, terminant instancia EC2

Les noves instancies agafen una zona de disponibilitat que encara no s'ha agafat gracies a que vam afegir 3 zones de disponibilitat al clúster ECS.

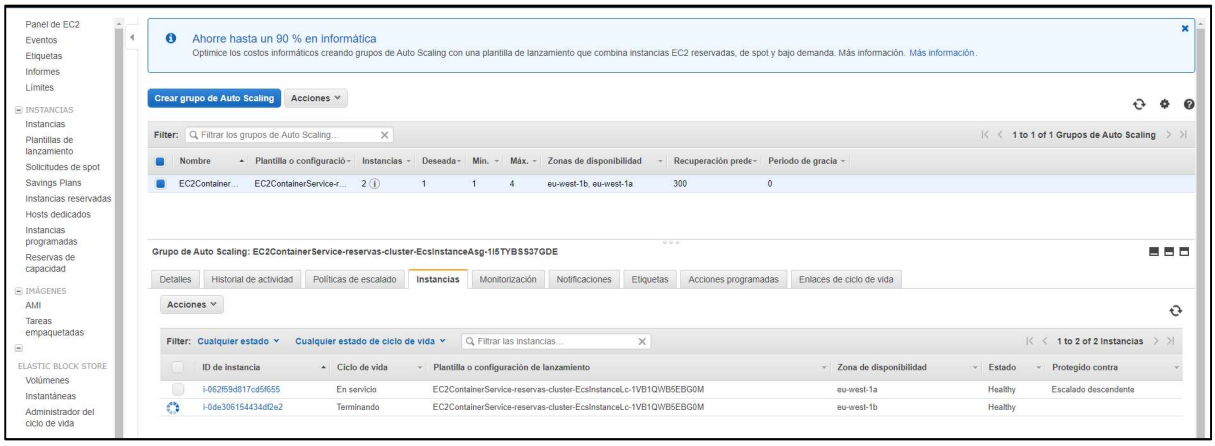


Figura 93: Auto escalat, diferents zones de disponibilitat

Baixant els serveis

Ens intentem moure en el free tier que ofereix AWS, podem consultar l'*Annex 6. Baixant els serveis per estalviar costos* on es descriu una sèrie d'accions a realitzar per estalviar costos (amb fins educatius, es clar que en un entorn productiu no realitzarem la totalitat de les accions recomanades en aquest apartat).

11. Procés de treball/desenvolupament de la infraestructura CI/CD

11.1. Creant les instancies EC2 per als servidors

Aquests passos on comuns per a tots els servidors que volem crear, Jenkins, Gitlab i SonarQube. S'instal·laran aquests serveis sobre una instancia EC2 d'AWS cadascun d'ells.

Primer seleccionen una AMI per a l'instancia EC2:

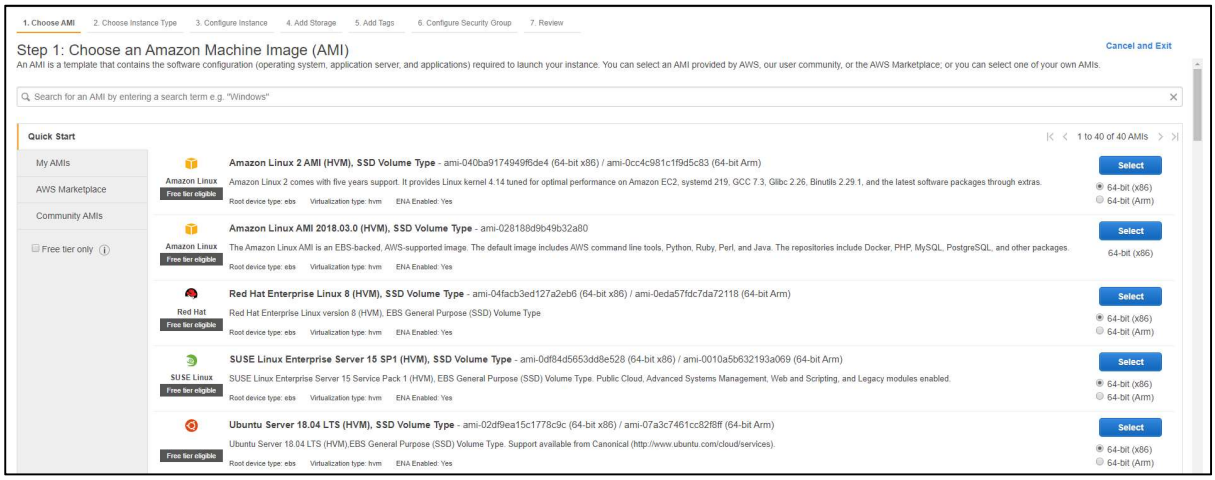


Figura 94: Configurant instancia per als servidors CI, part 1

Escollim la mida de la instancia, escollim per motius econòmics la mida free tier:

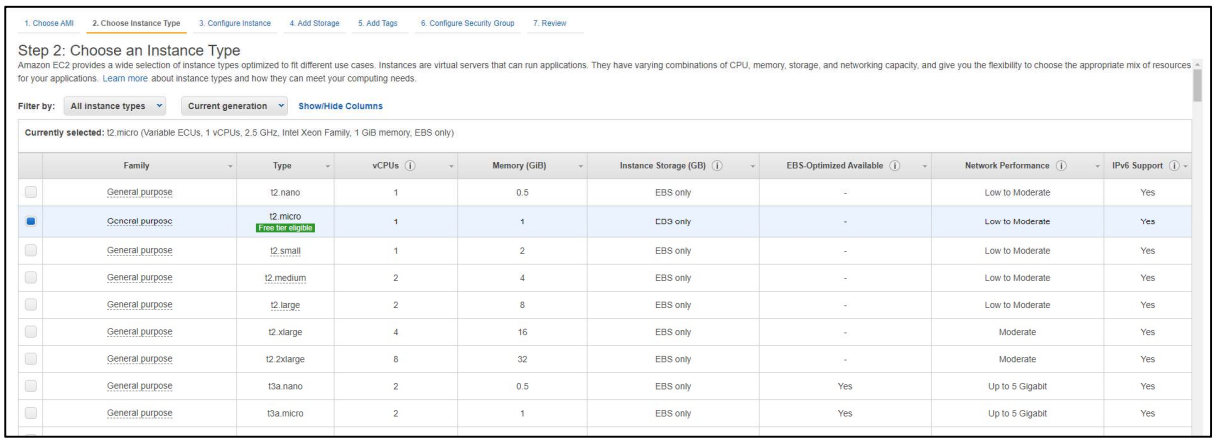


Figura 95: Configurant instancia per als servidors CI, part 2

Continuem amb la configuració de la instancia. Remarquem dos punts importants, *IAM Role* escollim *ecsInstanceRole* i activem la monitorització per CloudWatch:

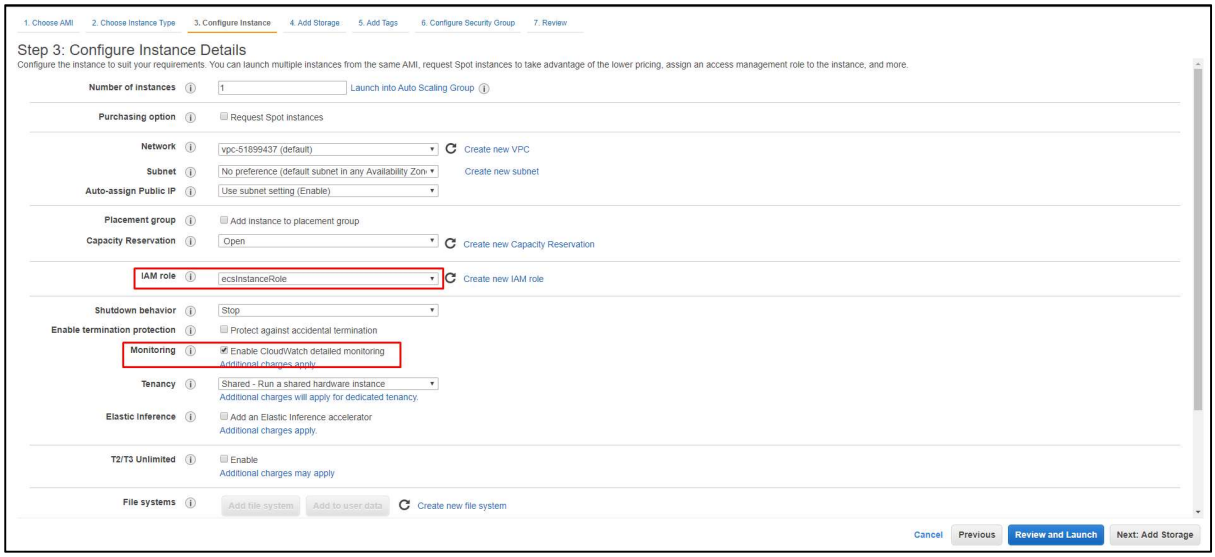


Figura 96: Configurant instancia per als servidors CI, part 3

En storage el deixem per defecte:

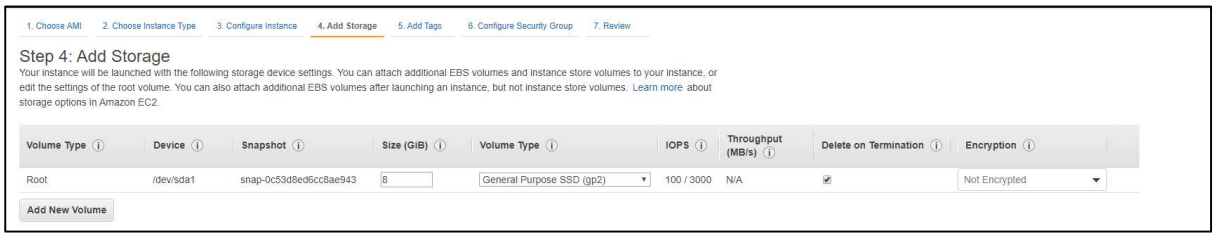


Figura 97: Configurant instancia per als servidors CI, part 4

Tags no les omplim:

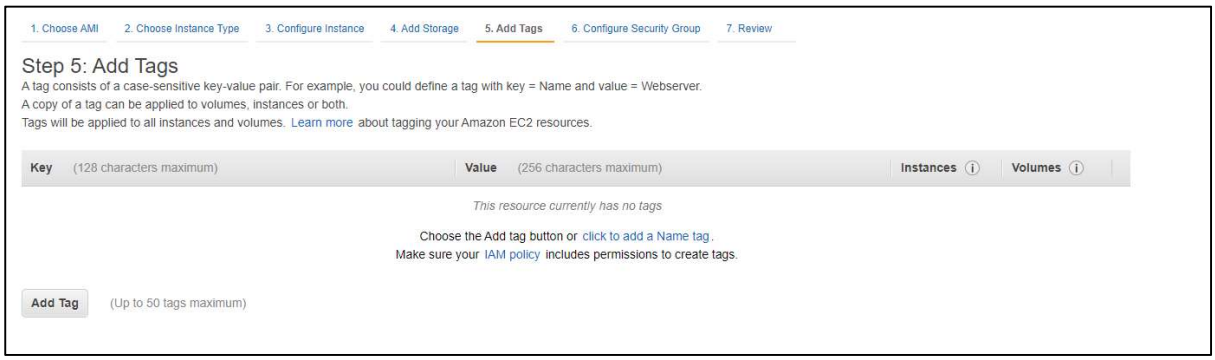


Figura 98: Configurant instancia per als servidors CI, part 5

En Security Group obrim el port d'SSH per a la meva IP:

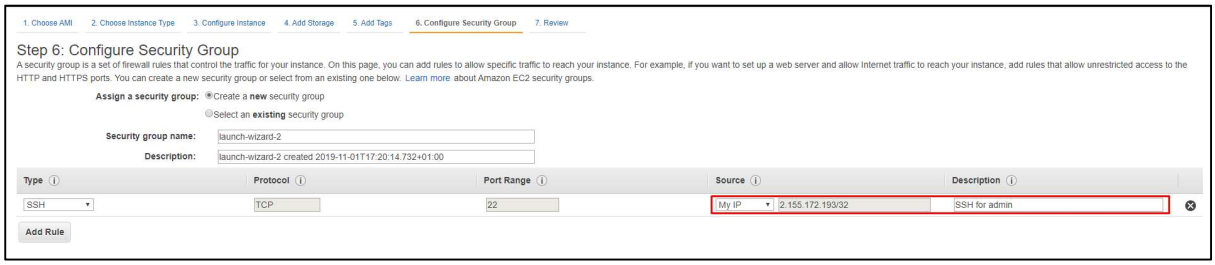


Figura 99: Configurant instancia per als servidors CI, part 6

Creem un nou keypair. Utilitzarem el mateix keypair per a tots els servidors d'aquesta secció:

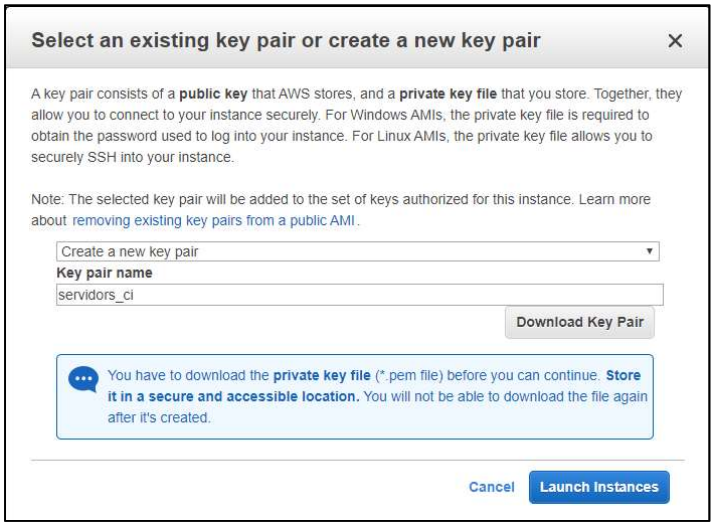


Figura 100: Configurant instancia per als servidors CI, part 7

Consultar l'*Annex 7. Connectar amb una instancia d'AWS* per descobrir com podem connectar a qualsevol instancia creada.

11.2. Instal·lació del servei Jenkins

Connectats per SSH a la instancia, volem instal·lar una imatge Docker amb Jenkins.

Comencem instal·lant Docker; en la consola de Linux executem:

```
sudo yum install docker
```

Arranquem Docker:

```
sudo systemctl start docker
```

Afegir l'usuari ubuntu al grup docker:

```
sudo usermod -a -G docker ec2-user
```

S'ha de sortir de la sessió per aplicar el grup docker.

Per instal·lar una imatge Docker amb Jenkins, trobem informació en:

https://hub.docker.com/r/jenkins/jenkins/

```
sudo docker pull jenkins/jenkins:lts
```

Amb això tenim baixada una nova imatge Docker amb Jenkins.

Creem el directori jenkins_home en arrel:

```
sudo mkdir /jenkins_home
```

y ara podem executar la imatge Docker amb la següent comanda:

```
docker run -d -v jenkins_home:/var/jenkins_home -p 8080:8080 -p 50000:50000
jenkins/jenkins:lts
```

Ja podem accedir via http:

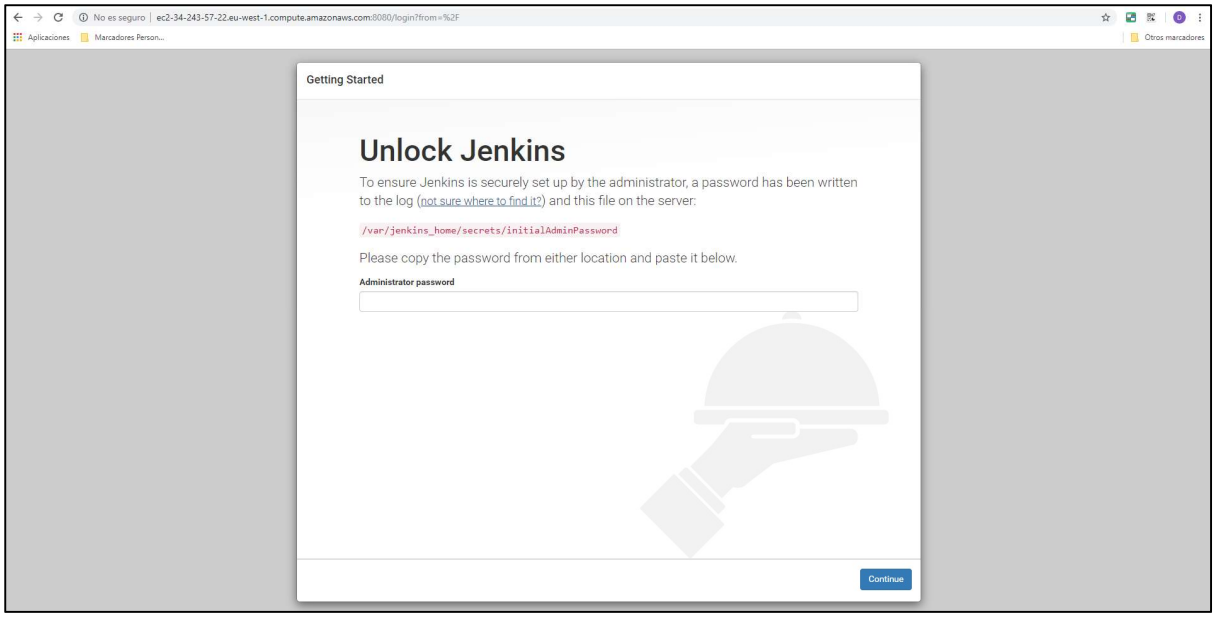


Figura 101: Instal·lació Jenkins, part 1

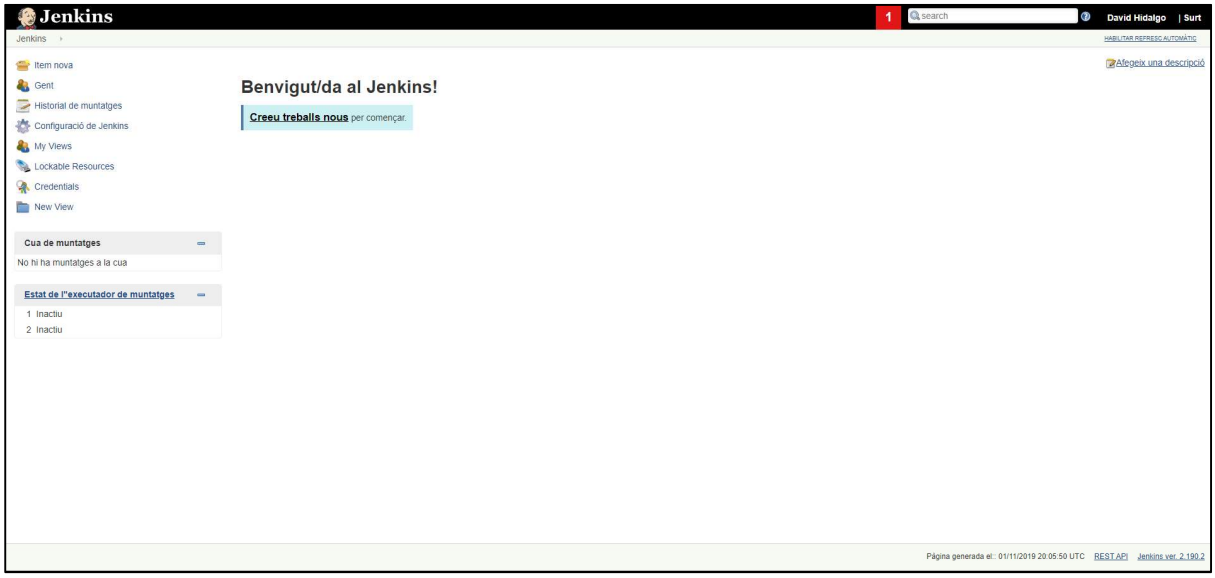


Figura 102: Instal·lació Jenkins, part 2

11.3. Instal·lació del servei GitLab CE

Creem una nova instancia EC2, seguint els passos anteriors.

Connectats per SSH a la instancia, executem les següents comandes:

```
sudo yum install curl policycoreutils-python openssh-server
```

Afegim el repositori:

```
curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash
```

Instal·lem GitLab Community Edition:

```
sudo yum install -y gitlab-ce
```

Realitzem algunes modificacions en les configuracions de GitLab:

No permetem que es registrin automàticament:

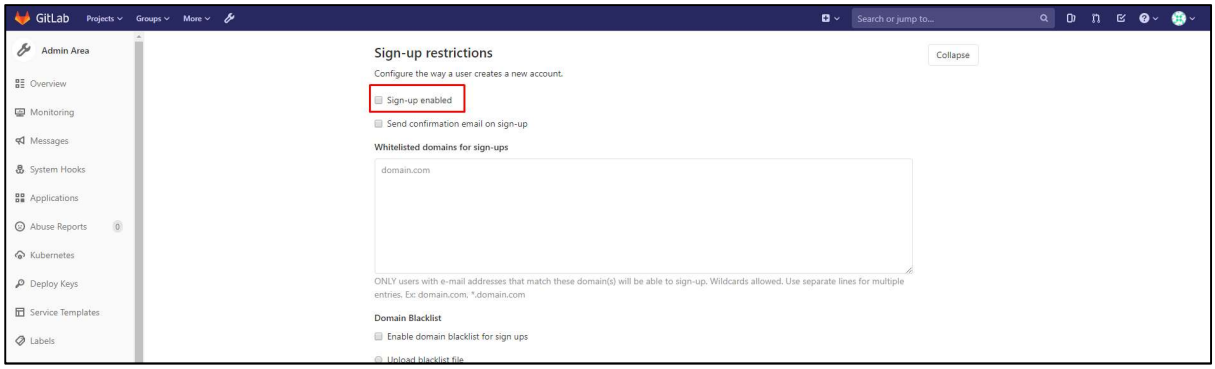


Figura 103: Instal·lació GitLab CE, part 1

Activem *two-factor authentication*:

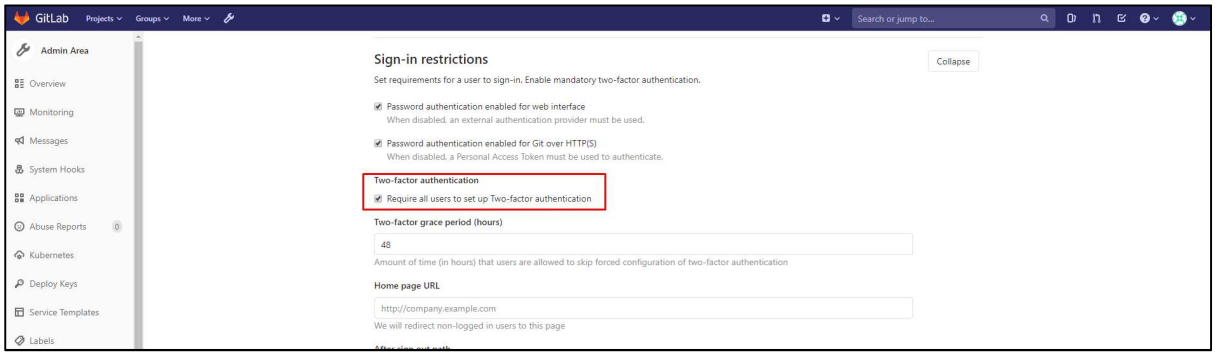


Figura 104: Instal·lació GitLab CE, part 2

Consultar l'*Annex 8. Pujar codi font al nostre GitLab* per veure les accions recomanades.

11.4. Instal·lació del servei SonarQube

Creem una nova instancia EC2, seguint els passos anteriors.

Comencem instal·lant el JDK de Java, descarreguem el Package:

```
wget --no-cookies --no-check-certificate --header "Cookie:oraclelicense=accept-securebackup-cookie" "http://download.oracle.com/otn-pub/java/jdk/8u131-b11/d54c1d3a095b4ff2b6607d096fa80163/jdk-8u131-linux-x64.rpm"
```

Instal·lem:

```
sudo yum -y localinstall jdk-8u131-linux-x64.rpm
```

Comprovem versió de Java instal·lat:

```
java -version
```

Consultem en la web de SonarQube l'ultima versió estable del software:

https://www.sonarqube.org/downloads/

Ara mateix és https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-8.0.zip

Connectats per SSH a la instancia, executem les següents comandes:

```
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.zip
```

Instal·lem el paquet *unzip* per a descomprimir *sonarqube-8.0.zip*:

```
sudo yum -y install unzip
```

y descomprimim el paquet:

```
sudo unzip /home/ec2-user/sonarqube-7.9.zip -d /opt
```

Canviem el nom al directori:

```
sudo mv /opt/sonarqube-8.0 /opt/sonarqube
```

Configurem el servei del sonar, editem l'arxiu sonar.service:

```
sudo vi /etc/systemd/system/sonar.service
```

Escrivim la següent configuració en l'arxiu:

```
[Unit]
Description=SonarQube service
After=syslog.target network.target

[Service]
Type=forking

ExecStart=/opt/sonarqube/bin/linux-x86-64/sonar.sh start
ExecStop=/opt/sonarqube/bin/linux-x86-64/sonar.sh stop

User=root
Group=root
Restart=always

[Install]
WantedBy=multi-user.target
```

Arranquem l'aplicació:

```
sudo systemctl start sonar
```

Per que cada vegada que arrenqui la instancia, arrenqui el servei de SonarQube:

```
sudo systemctl enable sonar
```

11.5. Subdominis

Només es posa d'exemple els passos a seguir per al servei de SonarQube.
Se han realitzat les mateixes passes per a GitLab, Jenkins i SonarQube.

Creem un nou *Target Group*, anomenat tg-sonar

Figura 105: Nou Target Group per a cada servei

S'ha de crear una nova *rule* en *Load Balancer*:

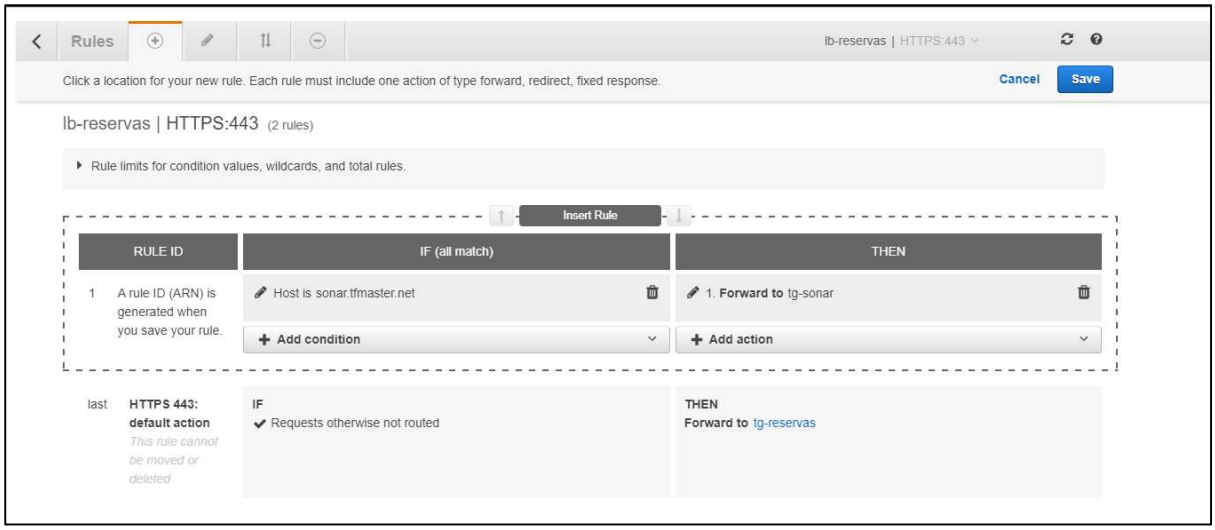


Figura 106: Nova rule de LB per a cada servei

Afegir un target en el *Target Group* en el port 9000 que es on escolta el servei de SonarQube:

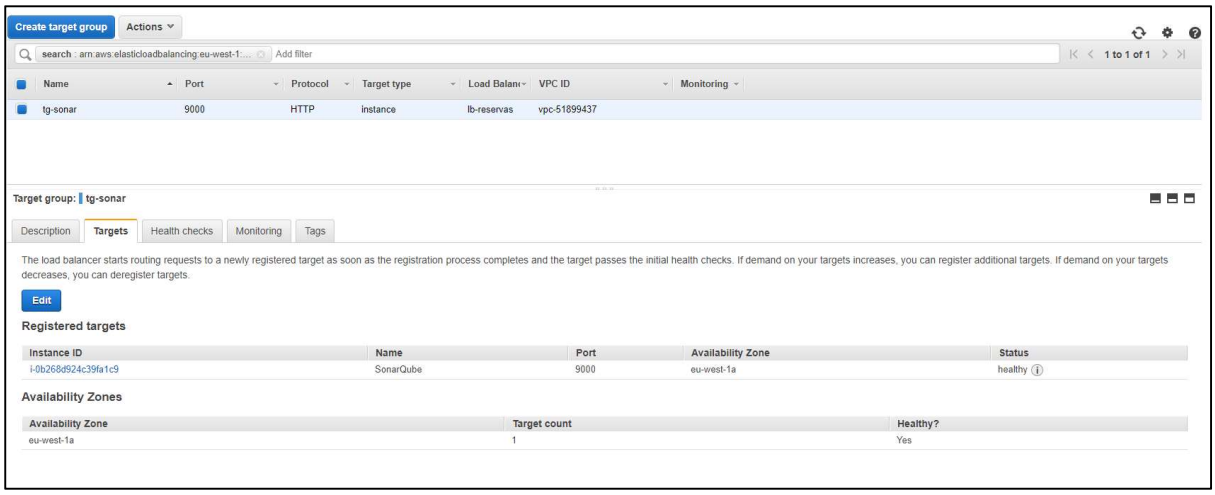


Figura 107: Creació de target en el Target Group per a cada servei

Creen un nou conjunt de registres en *Route 53* amb nom sonar.tfmaster.net i destí de l'àlies el *Load Balancer*:

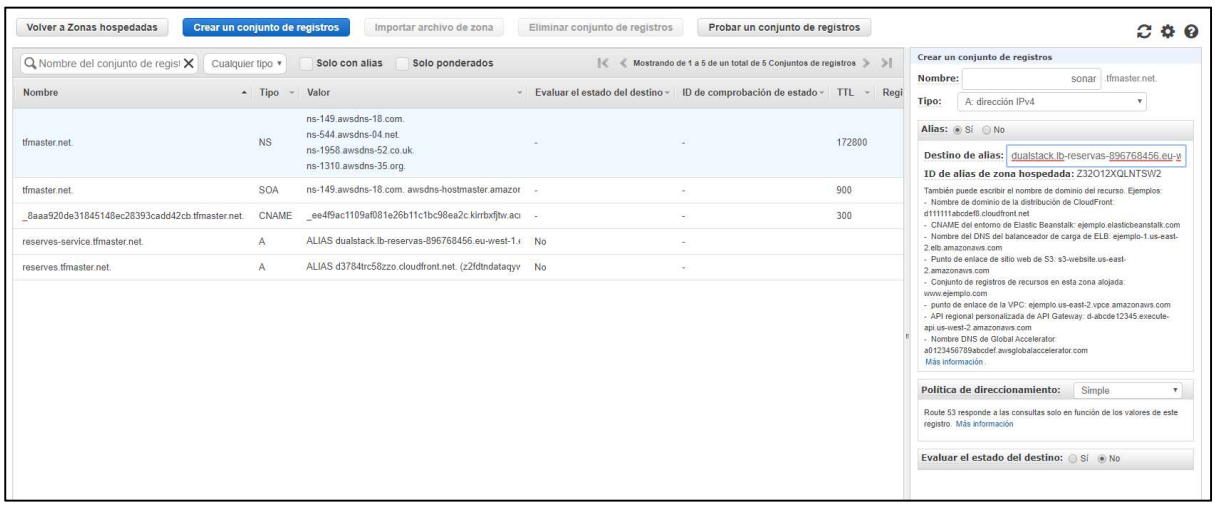


Figura 108: Creació nou conjunt de registres en Route 53 per a cada servei

https://sonar.tfmaster.net

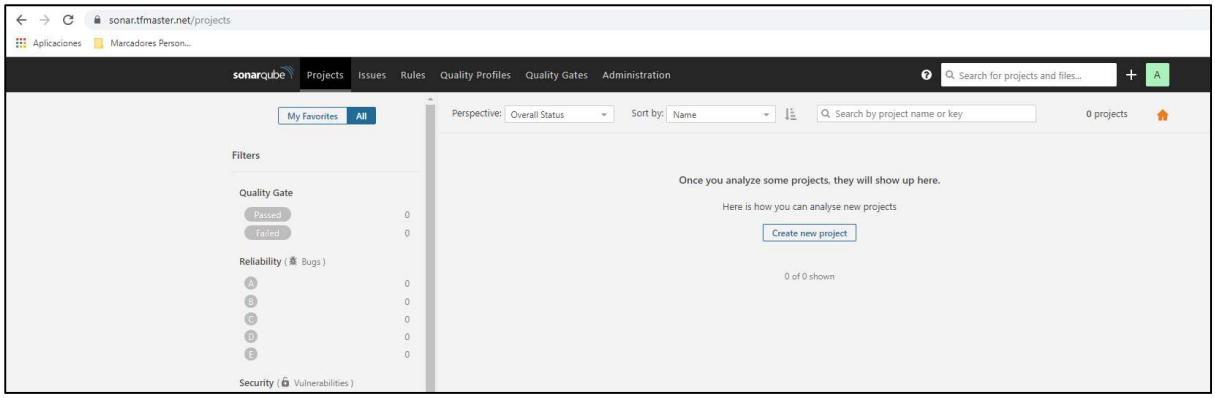


Figura 109: Resultant per a SonarQube

https://jenkins.tfmaster.net

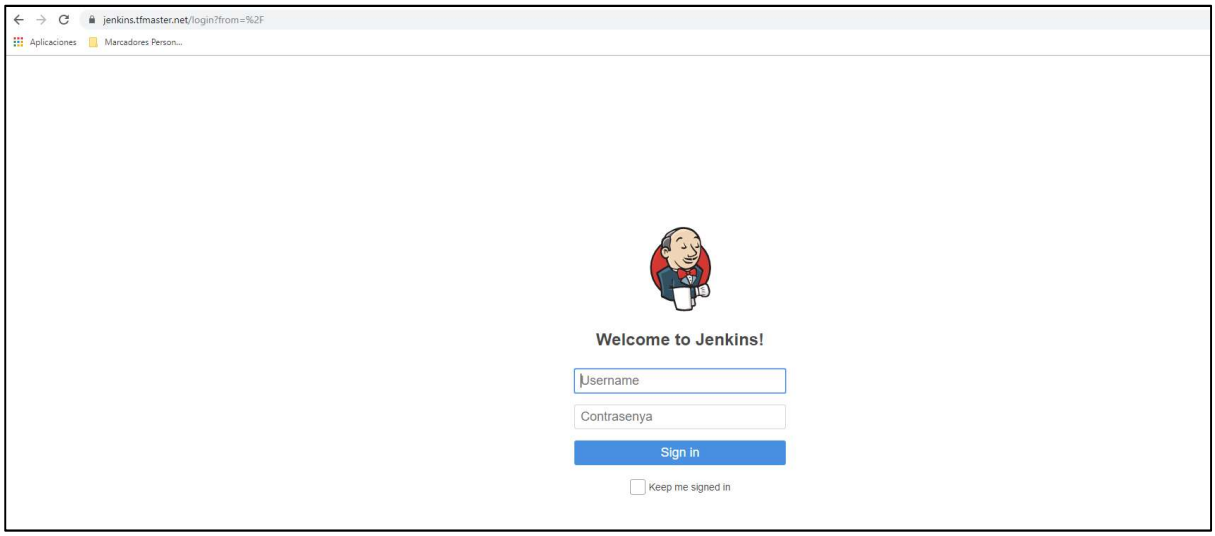


Figura 110: Resultant per a Jenkins

https://gitlab.tfmaster.net

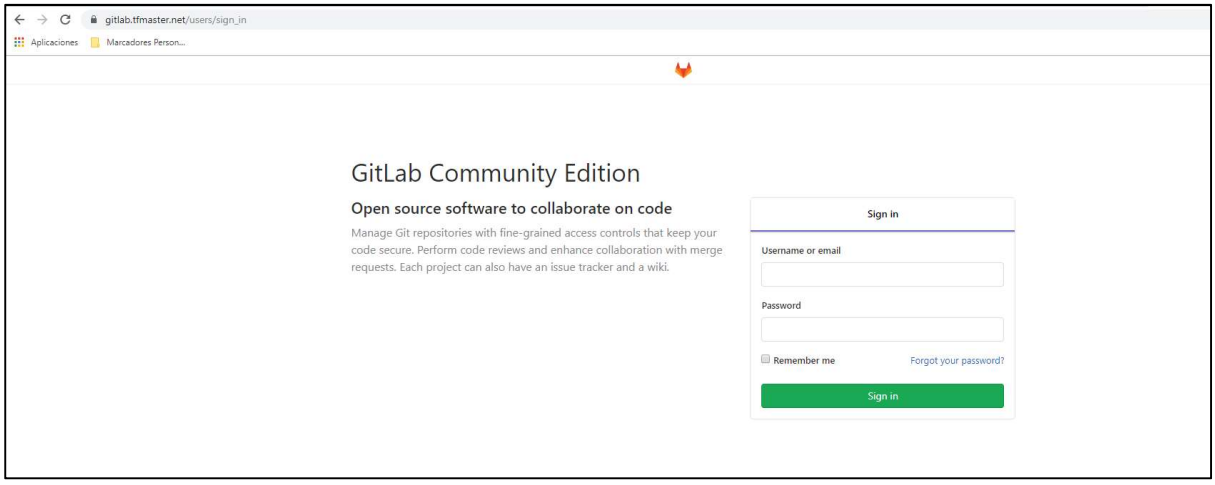


Figura 111: Resultant per a GitLab

11.6. Configurant SonarQube

Afegim un nou projecte a SonarQube, anomenat reserves-service de la següent manera:

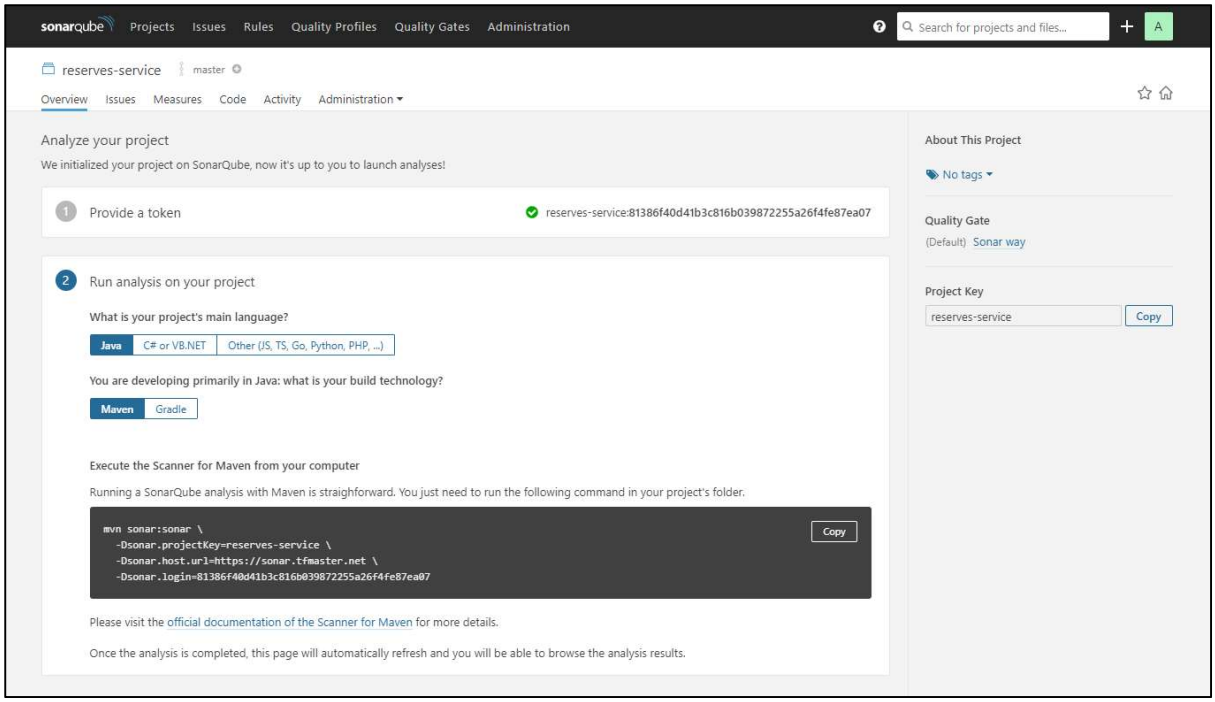


Figura 112: Afegint reserves-service a SonarQube

Afegim un altre projecte a SonarQube, anomenat reserves-frontend de la següent manera:

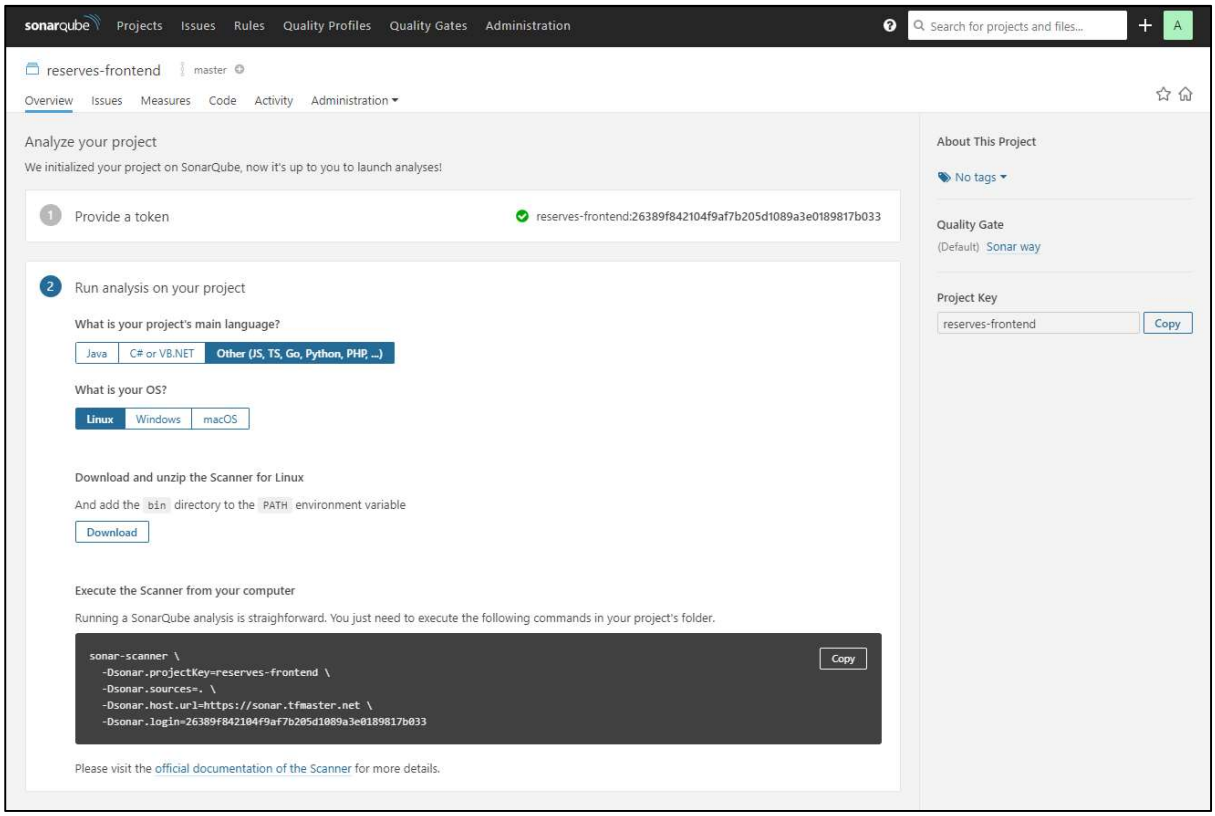


Figura 113: Afegint reserves-frontend a SonarQube

11.7. Preparacions prèvies en Jenkins

Abans de començar a treballar en el jobs pròpiament dits, hem de realitzar unes preparacions prèvies sobre el servidor de Jenkins.

Credencials per a GitLab

Volem agafar el codi font directament des de GitLab. Per a connectar-lo amb GitLab necessitem afegir a Jenkins un parell de *plugins* addicionals per a configurar la connexió i els credencials.

Figura 114: Plugins GitLab en Jenkins

Ara, volem configurar els credencials per a connectar-nos amb GitLab.
Creem un nou C*redential* d'username i password:

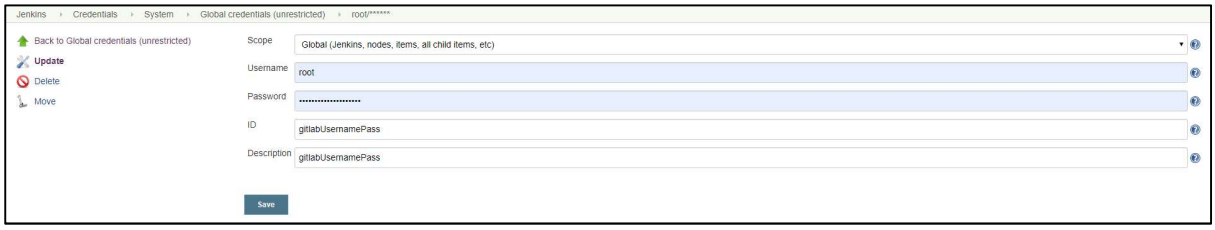


Figura 115: Nou *credential* per connexió a GitLab

Publicar artefacte en S3

Volem que una vegada empaquetat el codi font, pugi a un servidor S3 que és on el va a buscar la *Task Definition* que hem definit. Per això necessitem l'ajuda de plugins de Jenkins.
Hem d'instal·lar els plugins: *Flexible Publish* i *S3 Publisher*.

Després d'instal·lar els plugins, podem configurar un Amazon S3 profiles en la Configuració Global de Jenkins:

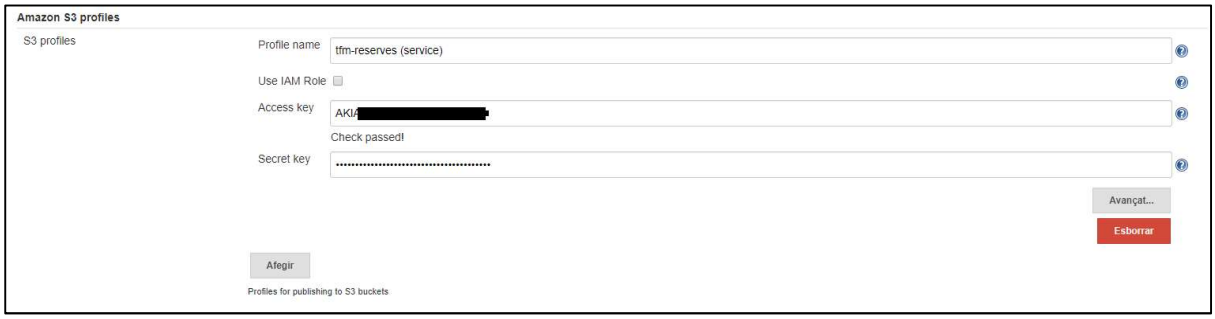


Figura 116: Amazon S3 profiles en Jenkins

AWS cli en Jenkins

Quan ja tinguem l'artefacte en el bucket d'S3, hem de manipular la Task Definition perquè torni a desplegar el nou paquet.

Necessitem tenir instal·lades les aws cli en el servidor de Jenkins:

```
sudo yum install epel-release
sudo yum update
sudo yum install python-pip
sudo yum install python34-pip
sudo pip install --upgrade pip
aws --version
```

JSON Query

JQ és un processador JSON de línia de comandes. Es pot utilitzar per manipular i llegir estructures JSON. Les instal·lem en el servidor Jenkins:

```
sudo yum install jq
```

Habilitant integració amb SonarQube

En la configuració de Jenkins, en *Global Tool Configuration*, s'ha d'afegir un *SonarQube Scanner*.

11.8. Jobs de Jenkins

Job reserves-service

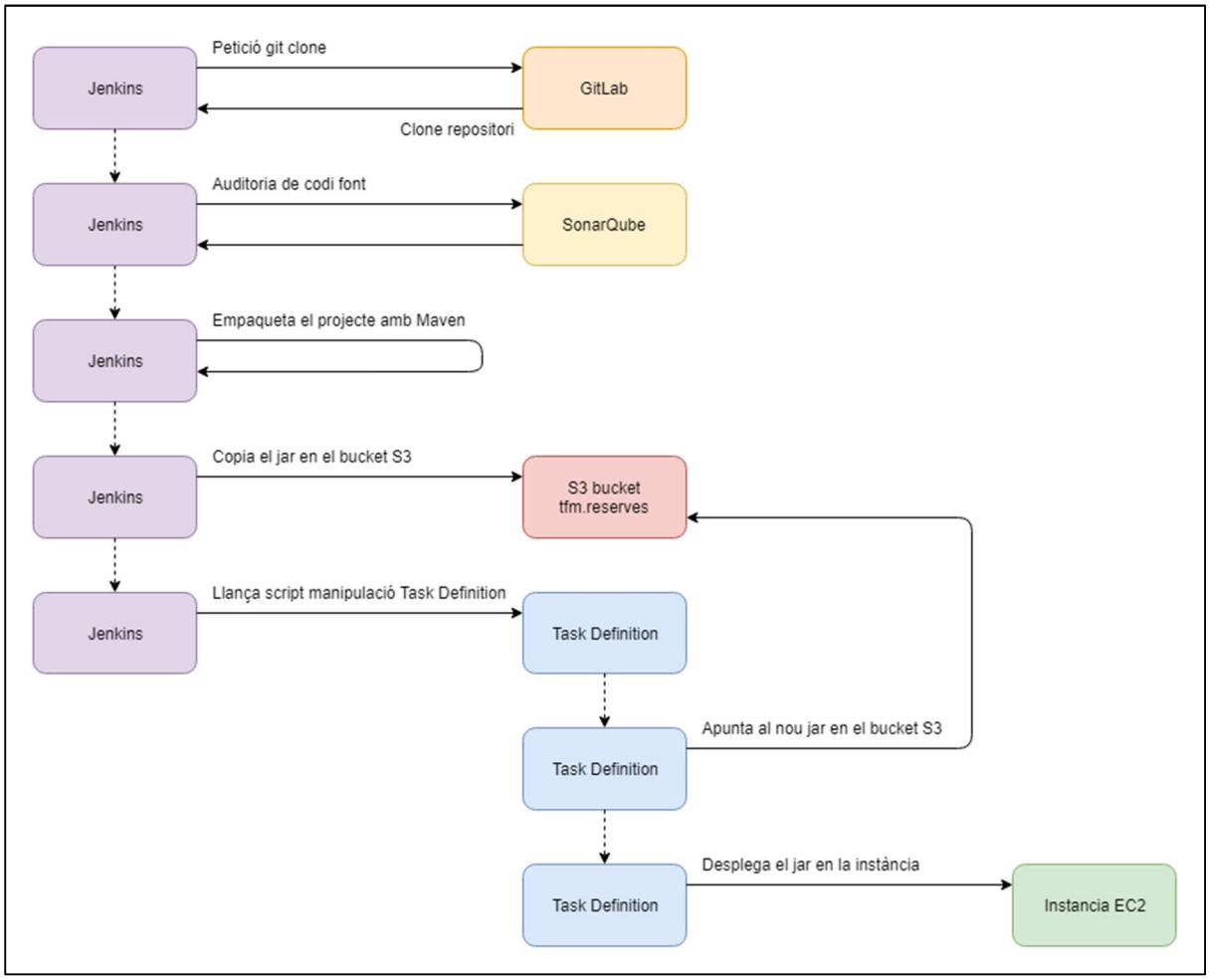


Figura 117: Esquema Job reserves-service

En el servidor Jenkins creem un nou Job anomenat reserves-service del tipus *Freestyle project*. Comencem per afegir l'origen de codi font.

Especifiquem l'URL del respository en GitLab i afegim els *Credentials* creats anteriorment. En branch deixem *master* ja que es des de la branca que volem ver les pujades:

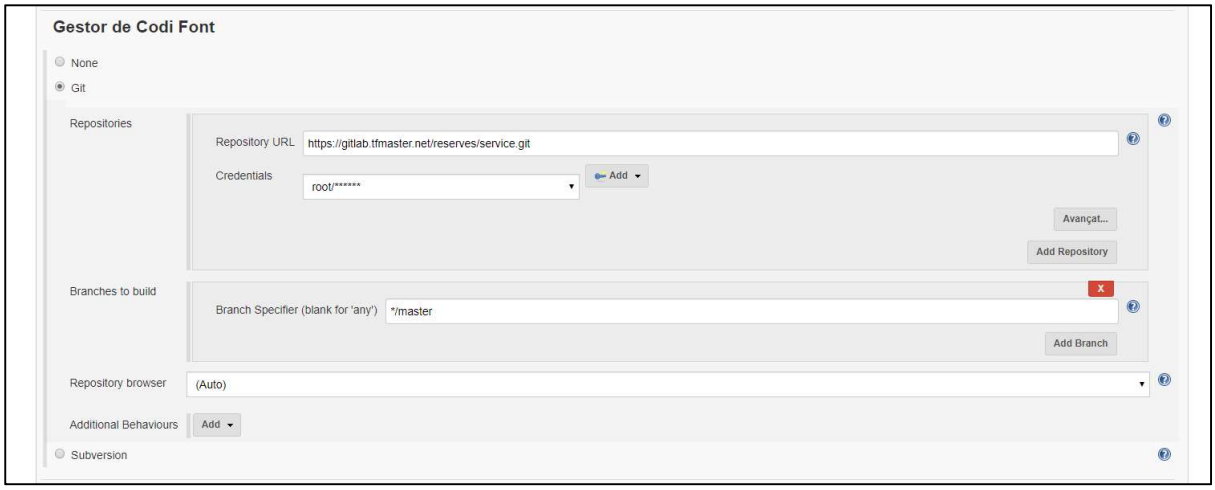


Figura 118: Job reserves-service, Gestor del Codi Font

En l'apartat Build, escollim versió de Maven i posem els goals de Maven necessaris per a realitzar el build. (els podem trobar en l'apartat *Instal·lació del microservei* en el punt *20. Instruccions d'instal·lació/implantació.*

Figura 119: Job reserves-service. Build

Per integrar amb el servidor de SonarQube, en *Build Enviroment*, marquem la *casella Prepare SonarQube Scanner enviroment*.

En l'apartat *Build*, afegir una altra tasca, aquesta vegada del tipus *Execute SonarQube Scanner* amb la següent configuració:

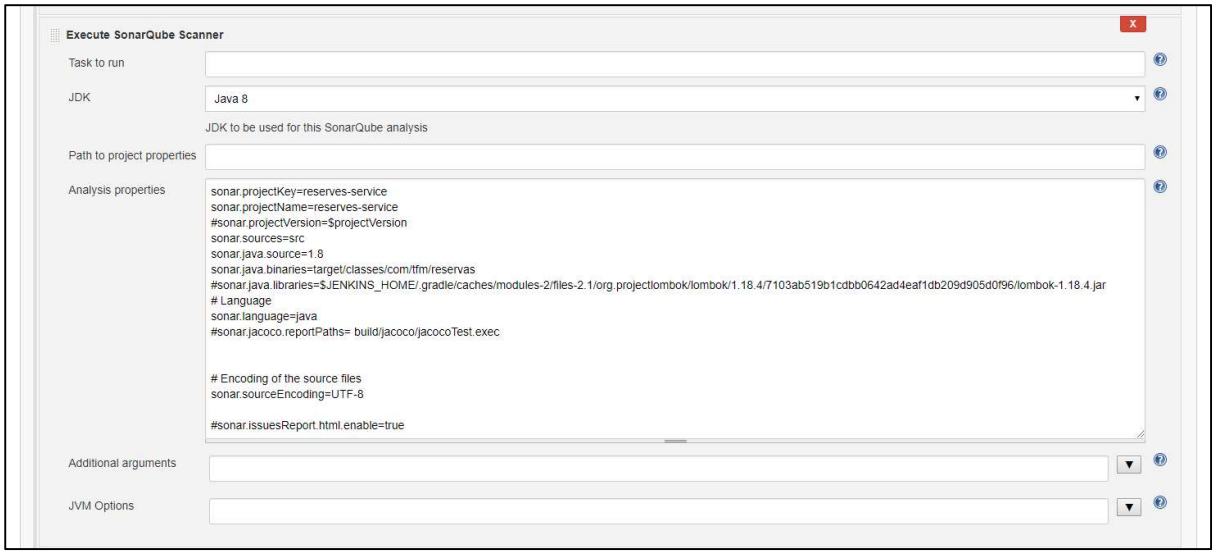


Figura 120: Job reserves-service. Execute SonarQube Scanner

Afegim una *Post-build Actions* del tipus *Publish artifacts to S3 Bucket*. Escollim en S3 profile el profile creat anteriorment amb els credencials del nostre bucket S3.
En *source* li diem on es troba el nostre jar compilat, i en *Destination bucket* el bucket on el copiarà.

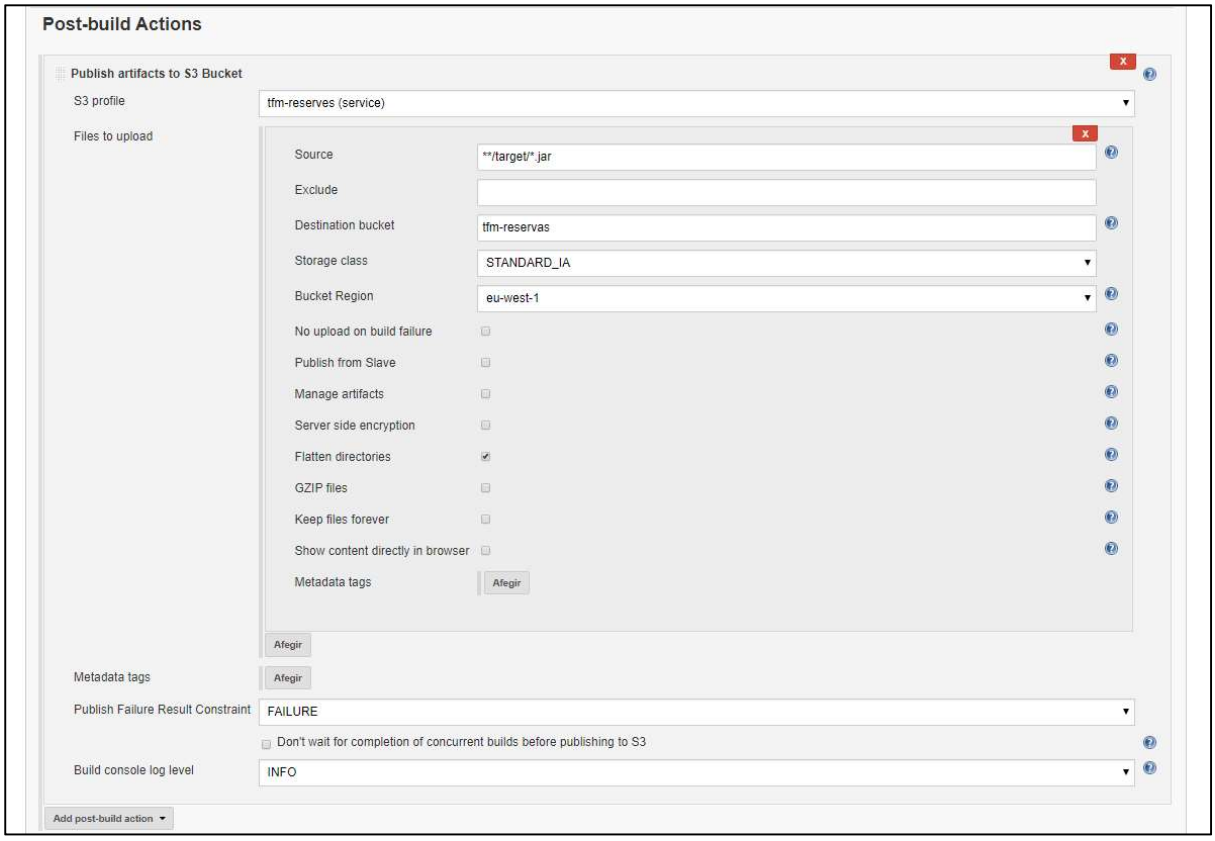


Figura 121: Job reserves-service. Publish artifacts to S3 Bucket

Ara necessitem que es desplegui aquest nou arxiu .jar amb la nova versió de la nostra aplicació. Per això, instal·lem un plugin nou en Jenkins que ens permet executar un Script en el pas de post-build, anomenat postbuild-script.

Afegim una nova post build action del tipus Execute Scripts. Ens permet posar un path a un script en el servidor.

Figura 122: Job reserves-service. Post build action, Execute Scripts

Creem l'script *ecs_deploy_task.sh* en la ruta */var/lib/jenkins/* del nostre servidor (veure Annex 2. Codi font (extractes) per a veure l'script).
Ens assegurem que sigui executable amb la següent comanda:

```
chmod a+x ecs_deploy_task.sh
```

En aquest script utilitzem les aws cli per detenir la tasca actual i aixecar una altra tasca nova. Per això necessitem també la definició de la tasca en format json per enviar-la.
Creem en el mateix directori l'arxiu *td-reservas.json* que conté la definició de la nova tasca a crear (veure Annex 2. Codi font (extractes) per a veure el contingut de l'arxiu).

Podem aconseguir un aproximat del contingut de l'arxiu td-reservas.json en la secció *ECS* dins del nostre clúster, dins la *Task Definition* polsar a *Create new Revision* i al final de tot, *Configure via JSON*.

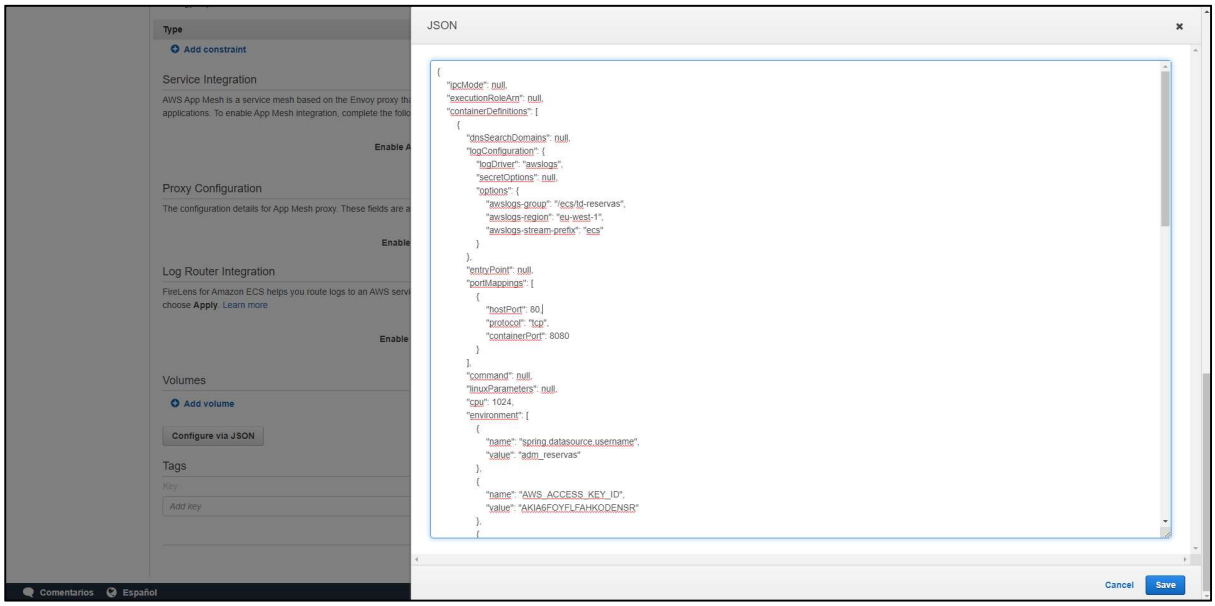


Figura 123: Job reserves-service.json task definition

NOTA: *Els valors a null donen error mitjançant crida en les aws cli, els podem eliminar en el nostre fitxer* td-reservas.json.

Job reserves-frontend

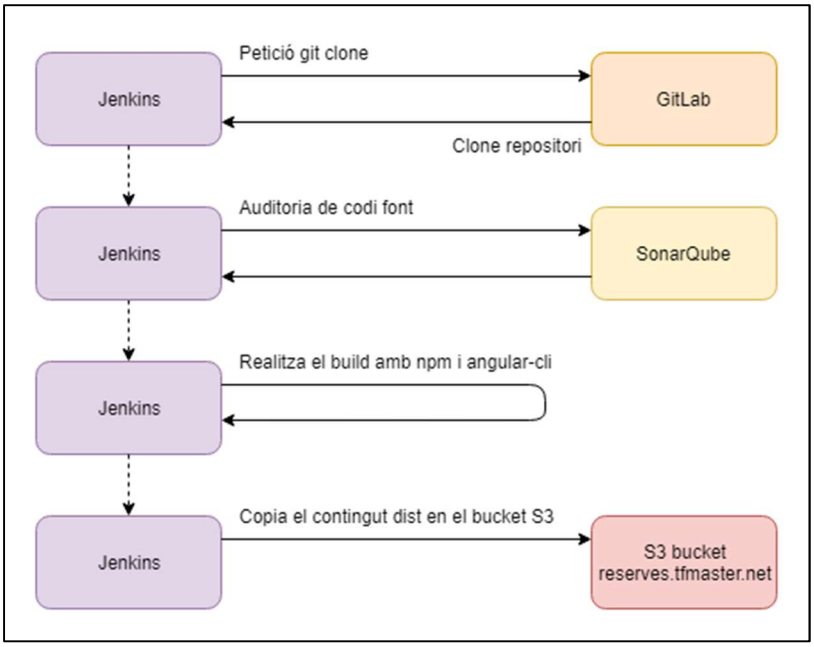


Figura 124: Esquema Job reserves-frontend

En el servidor Jenkins creem un nou Job anomenat reserves-frontend del tipus *Freestyle project*.

Comencem per afegir l'origen de codi font.

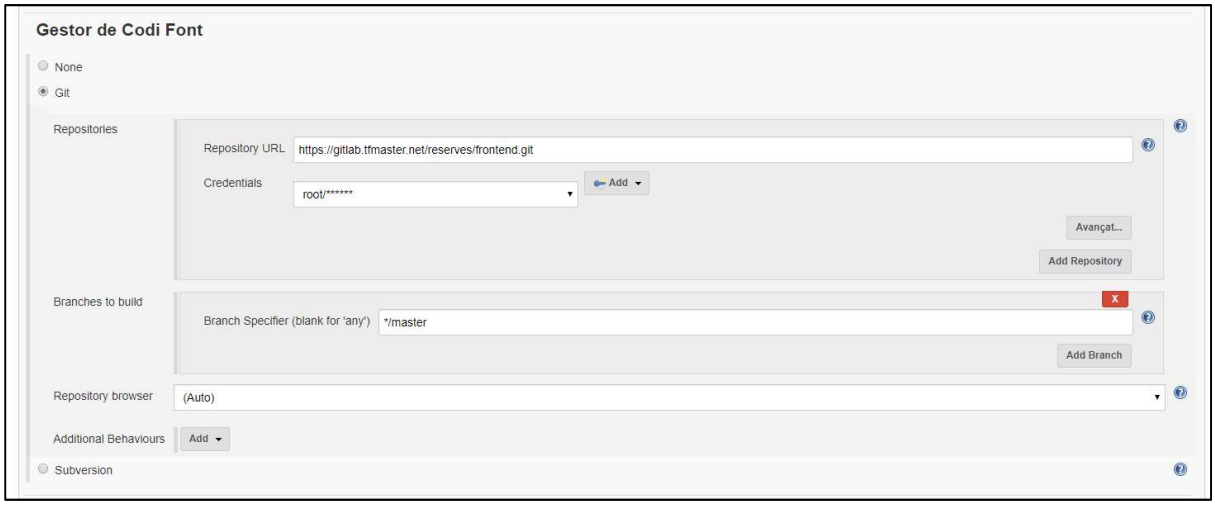


Figura 125: Job reserves-frontend, Gestor del Codi Font

En *Build*, escollim *Execute shell,* informem la comanda que volem executar per realitzar el build de la nostra aplicació, en el nostre cas l'script és el següent:

```
#!/bin/bash
npm i
ng build --prod
```

Per a realitzar aquesta acció, abans en el servidor Jenkins, mitjançant Shell, hem executat les següents comandes per instal·lar NodeJS i les Angular CLI:

```
curl -sL https://rpm.nodesource.com/setup_12.x | sudo -E bash -
sudo yum install nodejs
sudo npm install -g @angular/cli
```

En aquest moment ja construïm la *dist* en un directori dins del projecte dins del *workspace* de Jenkins.
Volem copiar els arxius de la *dist* al Bucket S3 on publiquem la nostra aplicació:

```
aws s3 sync dist/reservas-frontend/ s3://reserves.tfmaster.net --delete
```

Aquest nou comanda l'afegim al script del Execute shell. Amb el següent contingut:

```
#!/bin/bash
npm i
ng build -prod
aws s3 sync dist/reservas-frontend/ s3://reserves.tfmaster.net --delete
```

Figura 126: Job reserves-frontend, Buid, Execute shell

Per integrar amb el servidor de SonarQube, en *Build Enviroment*, marquem la *casella Prepare SonarQube Scanner enviroment*.
En l'apartat *Build*, afegir una altra tasca, aquesta vegada del tipus *Execute SonarQube Scanner* amb la següent configuració:

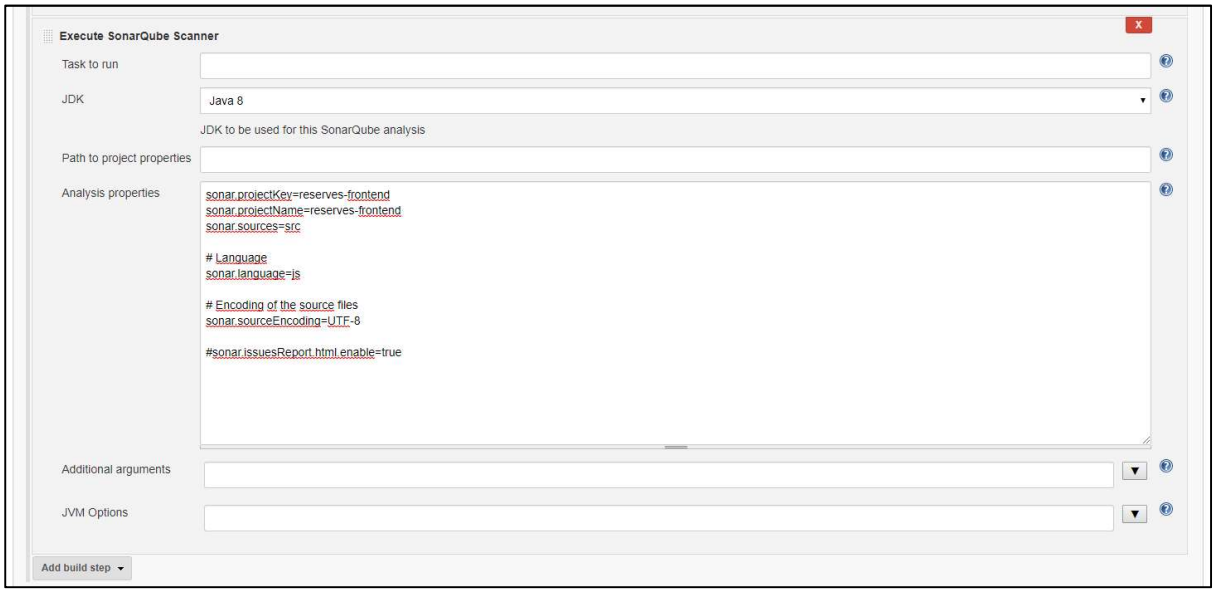


Figura 127: Job reserves-frontend. Execute SonarQube Scanner

11.9. AWS Lambdes per a realitzar snapshots

L'objectiu es tindre backups de les instancies EC2 que volem d'una manera completament automatitzada.
Per a realitzar això, modificarem un codi extern en Python i el publicarem en una AWS Lambda. Aquest codi font (explicat en el punt "APIs utilitzades") conté un README.md que hem de seguir per fer-ho funcionar. Expliquem les passes seguides que també contindrà el nostre README.md modificat.

Primer hem de crear una política IAM anomenada ***ebs-backup-worker*** seguint el document següent:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "ec2:CreateSnapshot",
        "ec2:CopySnapshot",
        "ec2:DeleteSnapshot",
        "ec2:CreateTags",
        "ec2:ModifySnapshotAttribute",
        "ec2:ResetSnapshotAttribute"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "*"
    }
  ]
}
```

Figura 128: Snapshots, nova política IAM

Continuem creant un role IAM anomenat ***ebs-backup-worker***, seleccionem *AWS Lambda* com a tipus de rol i li associem la política que acabem de crear.

Figura 129: Snapshots, nou rol IAM

AWS Lambda, no admeten valors separats per comes en les variables d'entorn, per això no podem afegir un llistat de regions que volem que passi l'script Python de la Lambda.
Per a solucionar això, codificarem el llistat de regions en Base64 i en el codi font, el descodifiquem i fem un Split per les comes, així aconseguim una llista d'elements.

Utilitzem Python per aconseguir el Base64 del llistat de les regions:

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> encoded = base64.b64encode(b'eu-west-1')
>>> encoded
b'ZXUtd2VzdC0x'
>>> data = base64.b64decode(encoded)
>>> data
b'eu-west-1'
>>>
```

En les variables d'entorn de la lambda hem de crear una variable anomenada ***aws_regions*** amb el valor `ZXUtd2VzdC0x` que ens ha donat com a resultat l'script de Python.

Hem de crear dues AWS Lambda. Una per fer funcionar l'script que realitza els snapshots i una altra és un script de neteja.

Escollim com a llenguatge Python 3.7 i com a rol d'execució el rol *ebs-backup-worker* que hem creat en aquest mateix punt.

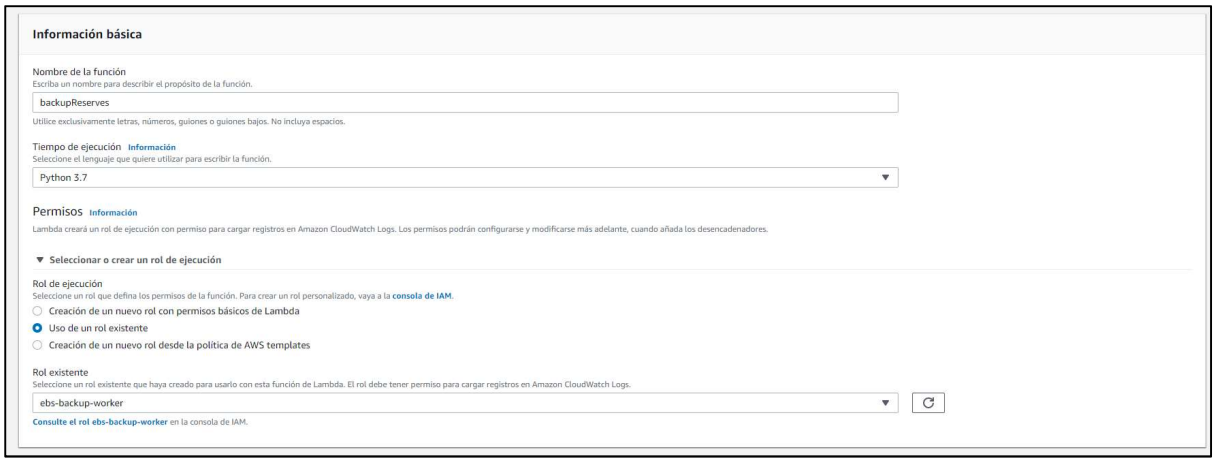


Figura 130: Snapshots, creació d'AWS Lambda, part 1

Creem un *trigger*, del tipus *CloudWatch Events* i amb una expressió de programació que és un cron. L'expressió serà cron(0 6 ? * MON *) és a dir, tots els dilluns a les 6:00h. s'executarà.

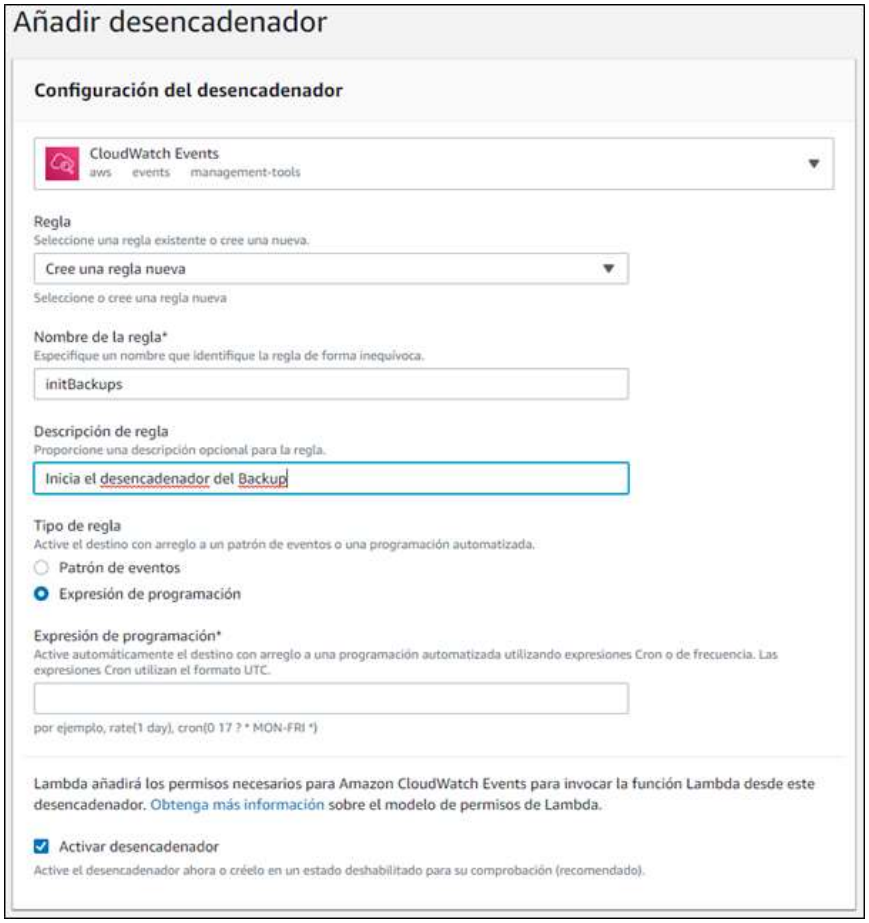


Figura 131: Snapshots, creació d'AWS Lambda, part 2

De la mateixa manera que s'ha descrit, hem de crear un altre AWS Lambda, aquesta anomenada backupCleanupReserves que contindrà el script de neteja. El cron en aquesta Lambda seria de posterior execució a l'anterior: cron(0 7 ? * MON *)

Ara hem de crear tags en les instancies EC2 sobre les que volem que actuï les Lambdes.
Les tags són les següents:

Tag Key	Tag Value	Notes
Backup		Value Not Needed
Retention	Number of Days to Retain Snapshot	Default is 7 Days
Skip_Backup_Volumes	volume id(s) in CSV string	List either a single volume-id, or multiple volumes-ids in a Comma Separated Value String

Taula 3: Tags EC2 per Lambda de backups

Una instancia EC2 amb la tag corresponent quedaria d'aquesta manera:

Figura 132: Snapshots, creació d'AWS Lambda, part 3

Volem realitzar snapshots de dues instancies, GitLab i Jenkins. Hem de posar la tag Backup (almenys) en ambdues.

Funcionament de Lambda realitzant snapshots de dues instancies:

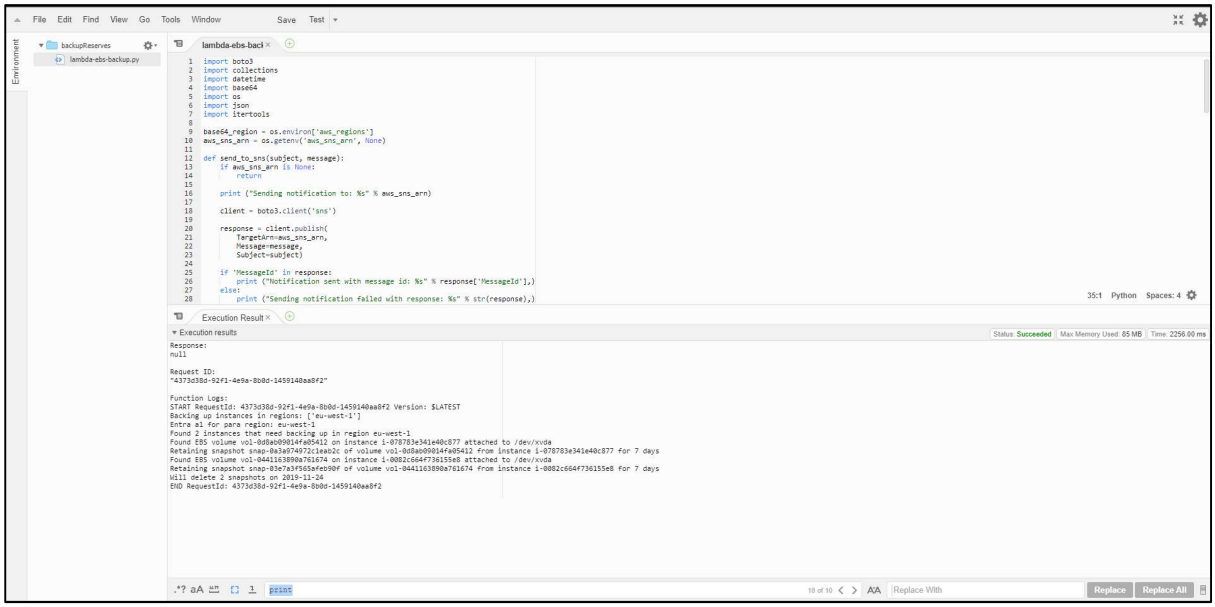


Figura 133: Snapshots, creació d'AWS Lambda, part 4

Funcionament de Lambda de neteja:

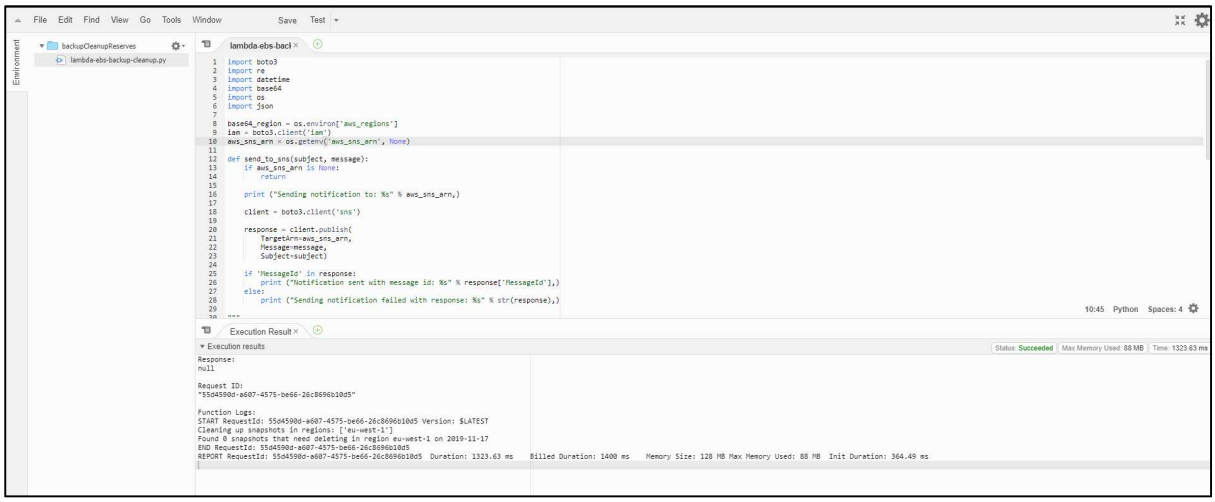


Figura 134: Snapshots, creació d'AWS Lambda, part 5

Les Lambdes s'executaran cada dilluns a les 6:00h del mati, i la de neteja a les 7:00h, però no volem tenir que estar pendents dels resultats, per això aprofitem una de les possibles sortides (outputs) de les lambdes amb Amazon SNS.

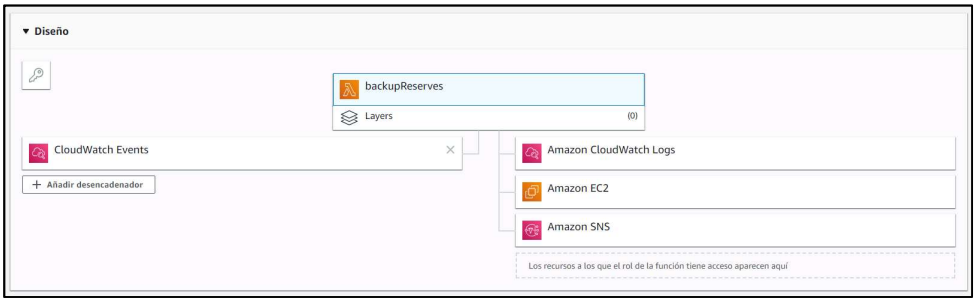


Figura 135: Snapshots, disseny d'AWS Lambda

En AWS SNS, creem un nou Tema que l'anomenem BackupsCI i una nova subscripció del tipus EMAIL, posem un e-mail vàlid i ens arribarà un correu per confirmar la subscripció.

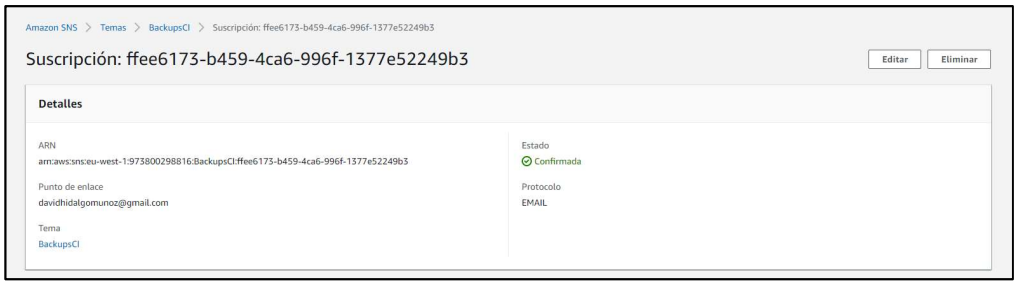


Figura 136: Creant una nova subscripció

Copiem l'ARN i configurem una nova variable d'entorn en la Lambda anomenada aws_sns_arn:

Figura 137: Snapshots, variable aws_sns_arn

El codi s'ha pujat al servidor GitLab a infraestructura/backups-lambda.

12. Diagrames UML

12.1. Diagrames de classes

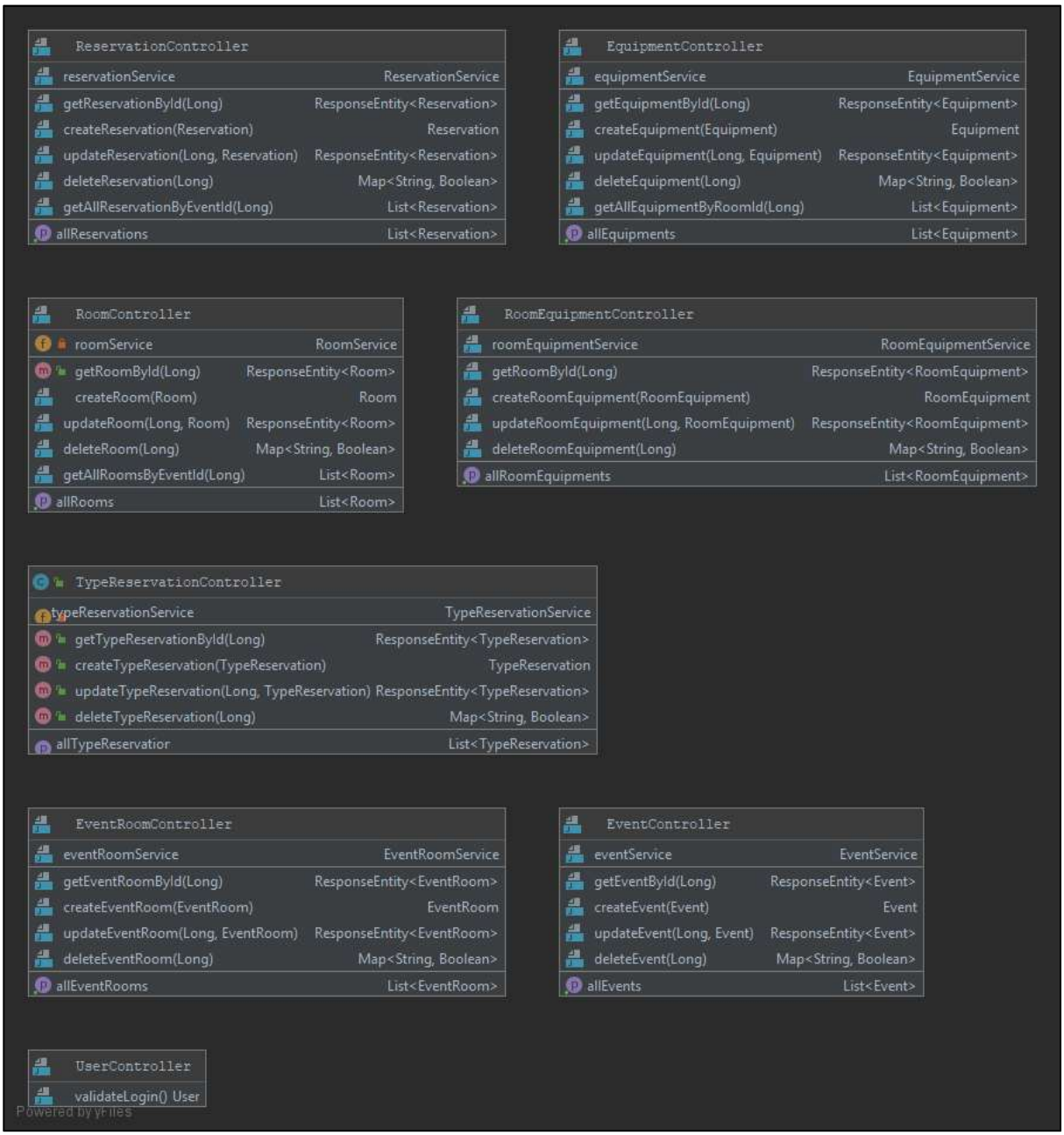


Figura 138: Diagrama de classes Package Controller generat per IntelliJ IDEA.

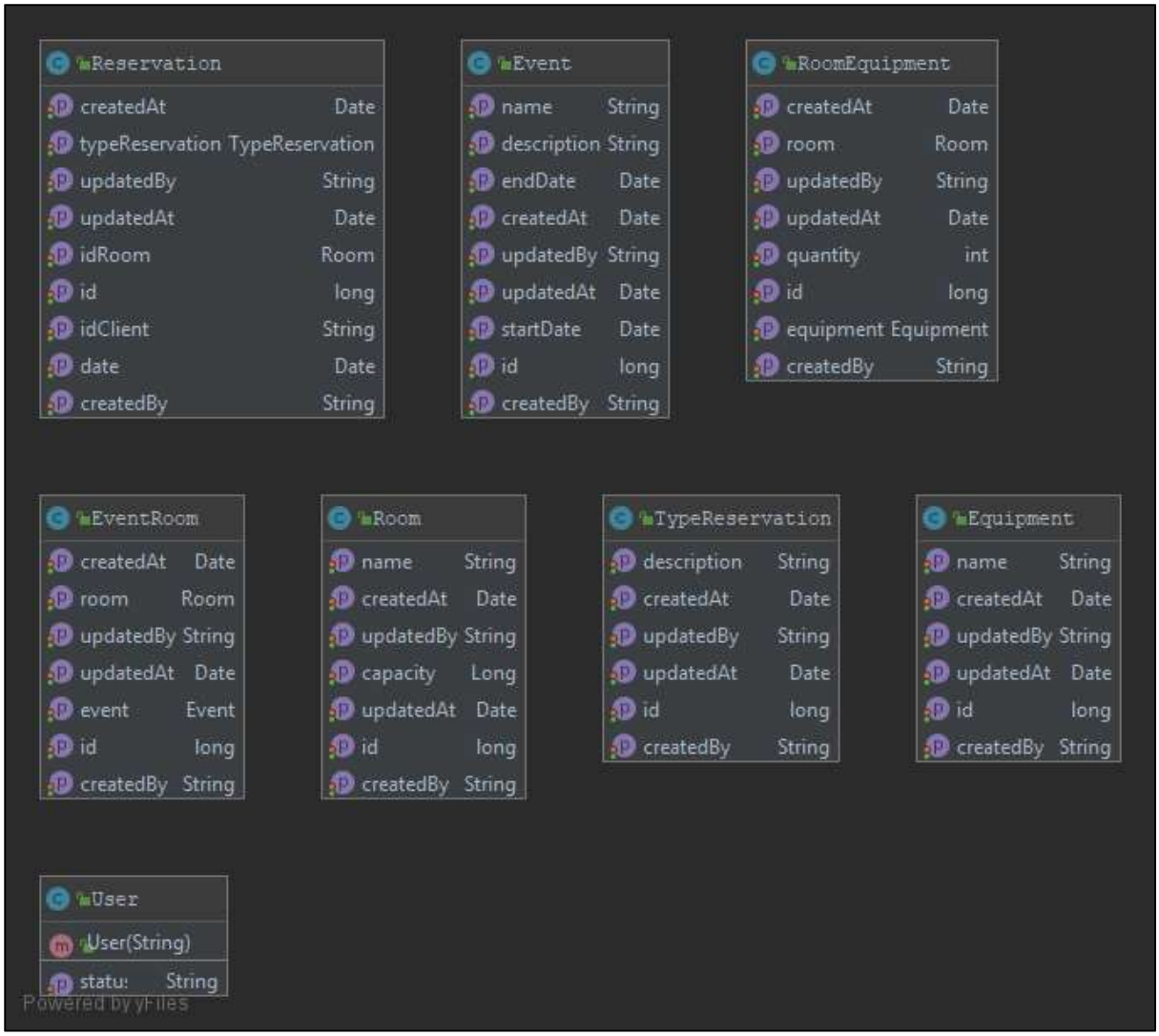


Figura 139: Diagrama de classes Package Model generat per IntelliJ IDEA.

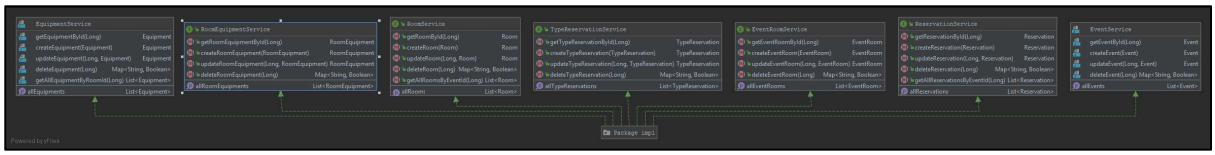


Figura 140: Diagrama de classes Package Service generat per IntelliJ IDEA.

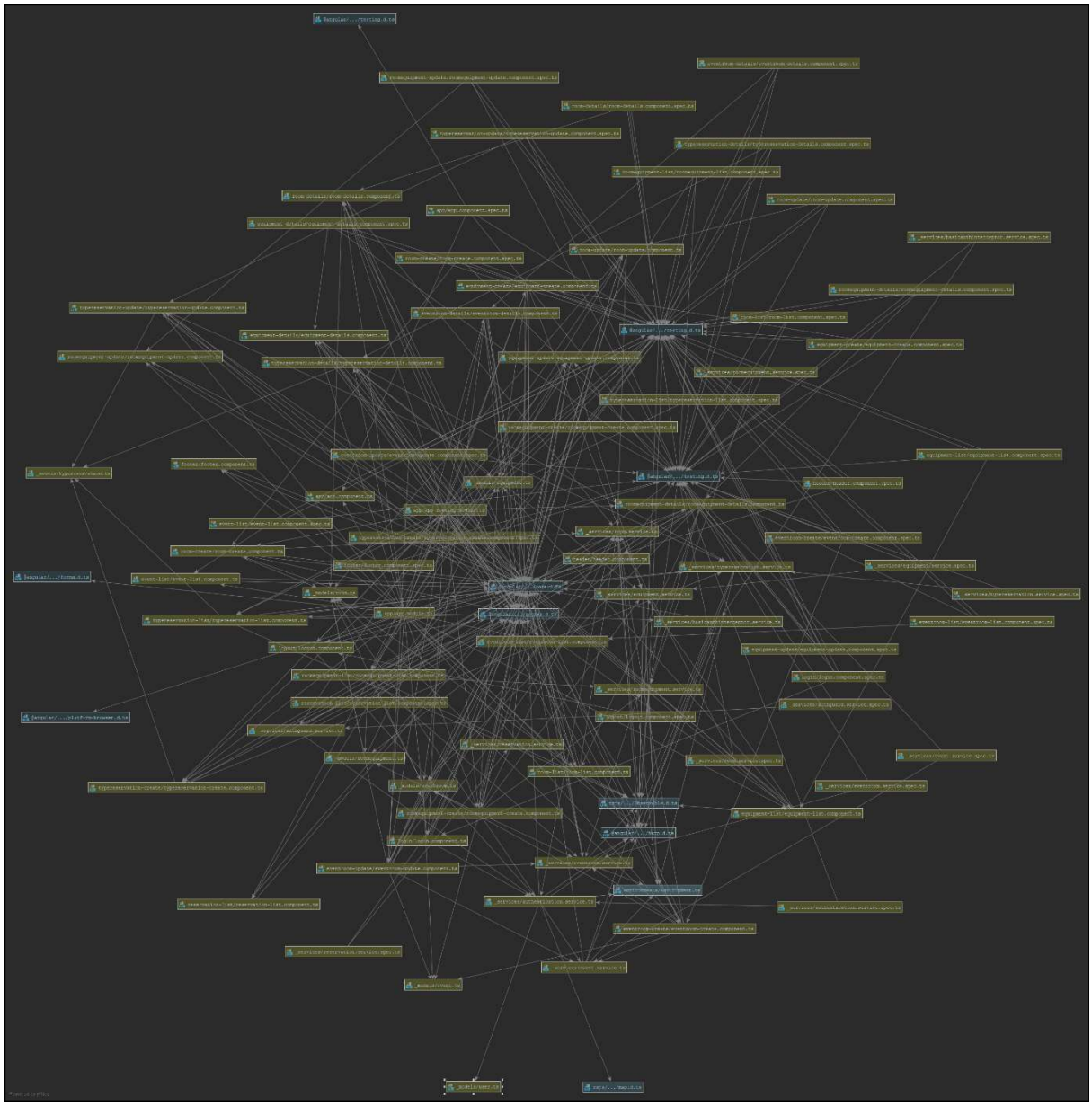


Figura 141: Diagrama de classes app Angular generat per IntelliJ IDEA.

S'inclouen els fitxers d'imatge en els lliurables per una major visualització.

12.2. Diagrama UML casos d'ús

S'ha realitzat un petit front-end per a realitzar els casos d'us de l'usuari *Editor*. Aquests casos d'ús estan marcats amb verd en el diagrama.
Els altres casos d'usos estan contemplats en el microservei i són totalment funcionals, disposats a ser cridats per qui correspongui segons els rols definits en el diagrama.

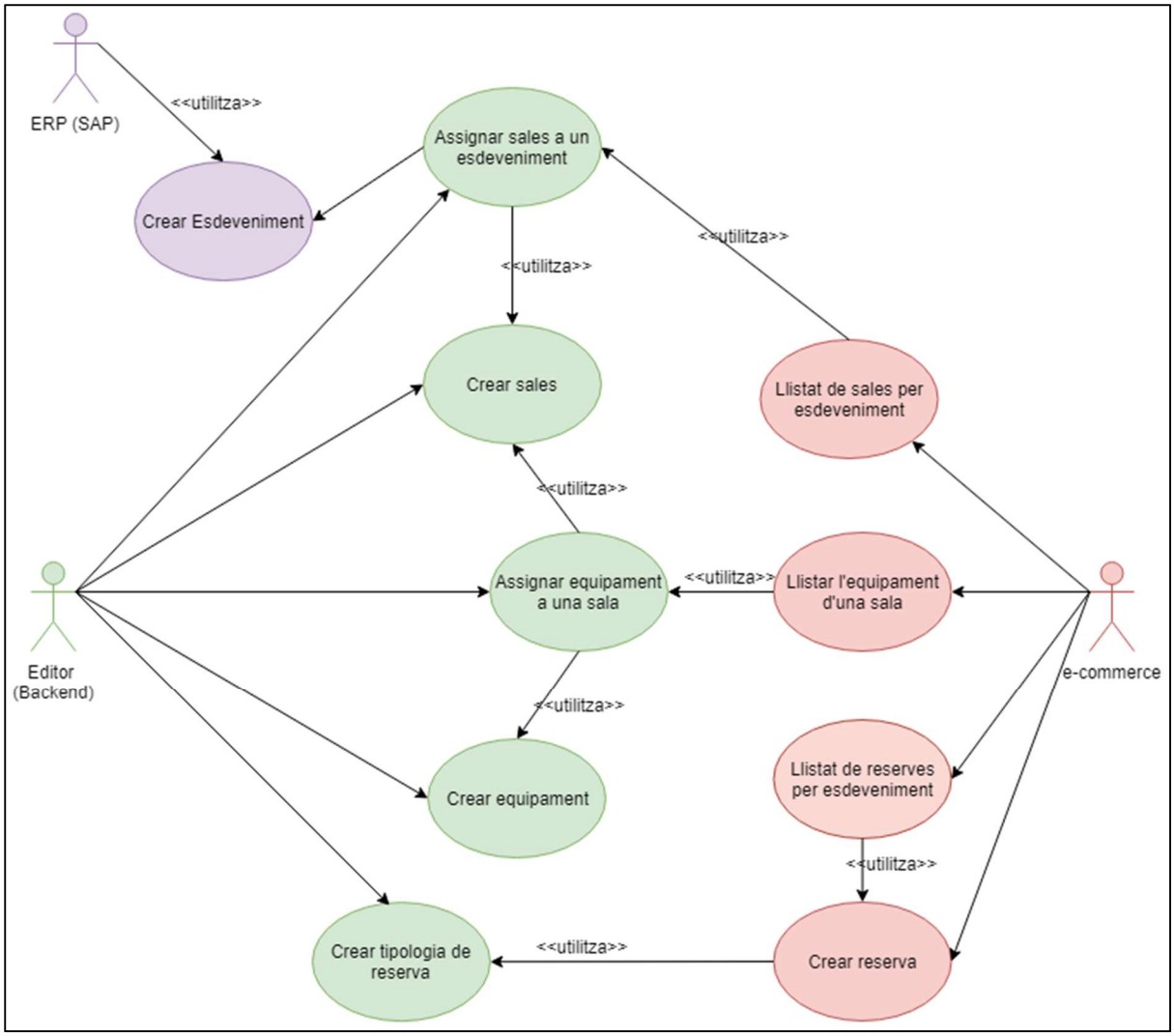


Figura 142: Diagrama UML casos d'ús

- L'Editor crearà Equipaments i Sales.
- L'Editor podrà realitzar l'acció d'assignar els equipaments a sales.
- L'Editor crearà Tipologies de Reserves.
- L'Usuari de SAP crearà Esdeveniments.
- L'Editor podrà realitzar l'acció d'assignar Tipologies de reserves a Sales.
- L'Editor podrà realitzar l'acció d'assignar Sales a Esdeveniments.
- L'e-commerce crearà una nova Reserva
- L'e-commerce podrà realitzar l'acció de llistar les Reserves per Esdeveniments.
- L'e-commerce podrà realitzar l'acció de llistar els Equipaments per Sala.
- L'e-commerce podrà realitzar l'acció de llistar les Sales per Esdeveniments.

12.3. Diagrama de components

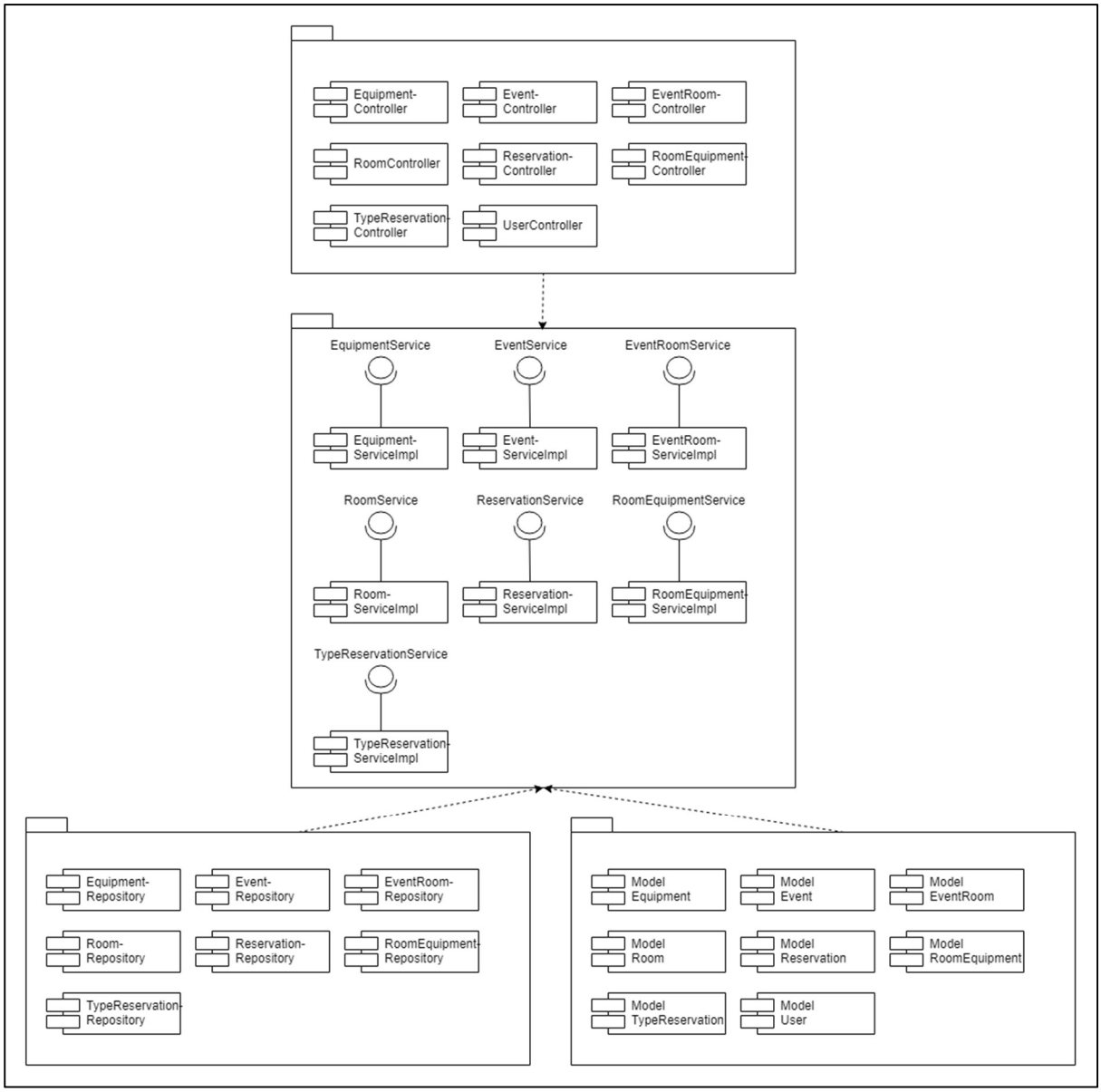


Figura 143: Diagrama de components del microservei Java

13. Perfils d'usuari

13.1. Perfils d'usuaris del servei Reserva de sales eventuals

El microservei serà cridat per 3 aplicacions diferents, cadascuna d'ella té diferents perfils d'usuaris amb els seus rols. Només nombraren els rols que afectin al nostre servei.

Usuaris de backoffice (part front-end) amb rol *editor*. Són els usuaris de backoffice que tenen drets ascendits, poden configurar l'eina de backoffice i entre altes coses, administrar les dades mestres.
Aquests usuaris realitzaran els casos d'us de creació de sales, creació d'equipament, assignació d'equipament a una sala, assignació d'una sala a un esdeveniment, creació d'un tipus de reserva de sala i assignació del tipus de reserva a una sala.

Usuaris de SAP amb rol *esdeveniment*. Són els usuaris de SAP que donen d'alta tot el necessari per a un nou esdeveniment en l'ERP i gracies a les seves integracions, a moltes més aplicacions. Aquests usuaris faran us de la creació d'un nou esdeveniment, integrat en el procés actual que fan servir.

Usuaris d'e-commerce amb rol *expositor* i rol *agencia*. Són els usuaris de l'e-commerce de venda de serveis per a esdeveniments amb rol expositor, es a dir que són expositors de l'esdeveniment. Altres rols són muntador, proveïdor, etc.
Integrat amb el formulari de reserves de sales eventuals, quan es reservi per part d'un usuari, estarà integrat amb la creació d'una nova reserva.

13.2. Perfils d'usuaris de la infraestructura CI/CD

Depenent del servei, hi hauran diferents perfils d'usuaris:

Usuaris de GitLab

Usuari de GitLab per a integrar amb Jenkins. És un usuari que el fa servir el servei Jenkins per a tenir accés al codi font allotjat en GitLab. Aquest usuari es configura en Jenkins i ha de tenir accés al menys de reporter en GitLab per a poder llegir els projectes creats en GitLab. Aquest usuari no ha de realitzar login a la web de GitLab, tota la utilització es batch.

Usuaris de GitLab per a versionat de codi. Són els usuaris de GitLab i corresponen a desenvolupadors generalment. En aquest grup d'usuaris podem diferenciar usuaris amb accés de Maintener que poden fer push a les branques master dels projectes, i els usuaris amb accés de Developer que poden crear noves branques i fer push a branques que no siguin la master.

Usuari de GitLab administrador del servei. És l'usuari amb drets d'administració del servei de GitLab. Aquest usuari crearà nous usuaris amb diferents rols, els assignarà a projectes, administrarà grups i projectes i diferents configuracions generals del servei.

Usuaris de Jenkins

Usuaris de Jenkins per a executar els jobs. Són els usuaris de Jenkins amb drets únicament per a executar els jobs de desplegaments. No podran configurar o crear cap job en Jenkins, ni podran accedir a la configuració general del servei.

Usuari de Jenkins administrador del servei. És l'usuari de Jenkins amb drets totals sobre el servei de Jenkins. Serà l'usuari que administri el servei, crearà nous jobs i donarà accés a aquests als altres usuaris.

Usuaris de SonarQube

Usuaris de SonarQube d'àmbit general. Farem públic l'accés a SonarQube a la xarxa privada. Tothom en la xarxa privada pot ser un usuari d'aquest tipus. Aquests usuaris podran consultar els resultats de l'auditoria de codi realitzat per SonarQube.

Usuaris de SonarQube per analitzar el codi. Són els usuaris de SonarQube. Aquests usuaris podran usar eines externes com per exemple el plugin per a IntelliJ IDEA SonarLint.

Usuari de SonarQube administrador del servei. És l'usuari de SonarQube amb drets totals sobre el servei de SonarQube. Serà l'usuari que administri el servei, crearà nous projectes, administrarà les regles de codi, crearà usuaris i donarà accés als usuaris als projectes creats.

Usuaris de la consola d'AWS

Usuari d'AWS administrador. Per ara no es contempla crear nous usuaris per a l'administració de la consola d'AWS. Aquest usuari serà l'encarregat d'administrar tota la infraestructura Cloud. Serà el receptor dels correus de les Lambda. Serà l'encarregat de millorar el servei, monitoritzar i donar suport en els nous projectes de software per al desplegament en AWS.

14. Usabilitat/UX

La usabilitat és un part molt important en el desenvolupament d'aplicacions.
La usabilitat es refereix a la facilitat amb la que les persones poden utilitzar una eina particular o qualsevol altre element fabricat per essers humans amb la fi d'assolir un fi concret.
El grau d'usabilitat es pot mesurar a partir de proves empíriques i relatives:

- **Empírica** perquè no es basa en opinions o sensacions, sinó en proves d'usabilitat realitzades en laboratoris o observades en treball de camp.
- **Relativa** perquè el resultat no es ni bo ni dolent, sinó que depèn de les metes plantejades o en comparació amb sistemes similars.

Si ens centrem en el nostre desenvolupament, un desenvolupament web, existeixen 5 regles principals que adaptades a la web, es pot considerar una web "usable".

- **Ràpid**. las pagines s'han de carregar en una mitja de 4 segons.
- **Simple**. Mantenir una navegació constant. Els usuaris no han d'aprendre diversos camins o esquemes per a la navegació.
- **Investigable**. Els motors de cerca busquen el text real, ni el codi de programació ni els gràfics. S'ha d'evitar aquestes situacions.
- **Per a la majoria**. Utilitzable per tots els navegadors i sistemes operatius. Utilitzar el codi més compatible possible.
- **Mantenir actualitzada**. Evitar informació antiquada.

El desenvolupament del front-end d'aquest treball no s'ha orientat cap a un desenvolupament centrat en l'usuari (DCU), ja que la part gràfica **no es objectiu d'aquest treball**, encara i això, si s'ha tingut en compte algunes mesures per satisfer objectius d'usabilitat per millorar l'experiència de l'usuari.

Formes d'interacció

Les dimensions del disseny d'interacció:

- **Dimensió Paraules**: les paraules han de ser senzilles d'entendre i permetre la interacció.
- **Dimensió Representacions visuals**: gràfics, il·lustració, diagrama, icona, etc. han de tenir un ús fonamentat.
- **Dimensió Espai**: amb el que s'interactua en el mon real, mouse, teclat etc, són utilitzats com a eines de comandament.
- **Dimensió Temps**: durada que l'usuari passa interactuant amb les paraules, representacions visuals i espai.
- **Dimensió Comportament**: emocions i reaccions que te l'usuari a l'operar, utilitzar o realitzar una acció en el sistema.

Es poden seguir unes pautes a seguir per a treballat orientat al Disseny de la interacció.

1. **Defineix com s'interactua**. En el nostre cas, la interactuació serà amb un mouse. Al ser un desenvolupament basat en web, els navegadors solucionen aquesta pauta per nosaltres.
2. **Donar pistes del comportament**. Els botons tenen forma de botó, donen pistes d'interactuacció a més, tenen colors diferents, vermell esborrar, blaus llistar, verds creació, etc. amb text clar de l'acció que passarà si s'oprimeix el botó.

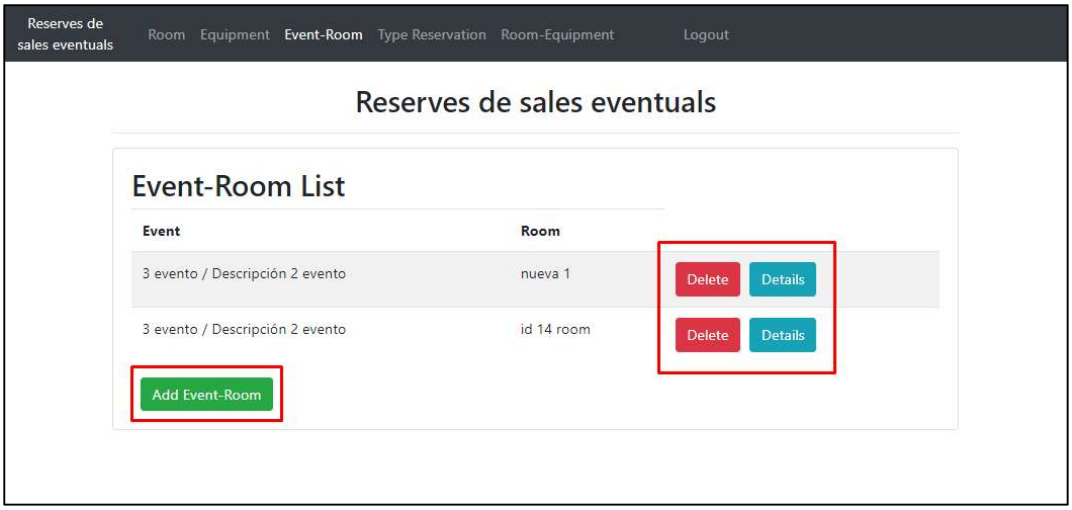


Figura 144: Botons diferenciats amb diferents colors

3. **Anticipació a errors potencials**. Claredat en els formularis. Tipus de dada, marcat de dades opcionals, etc.

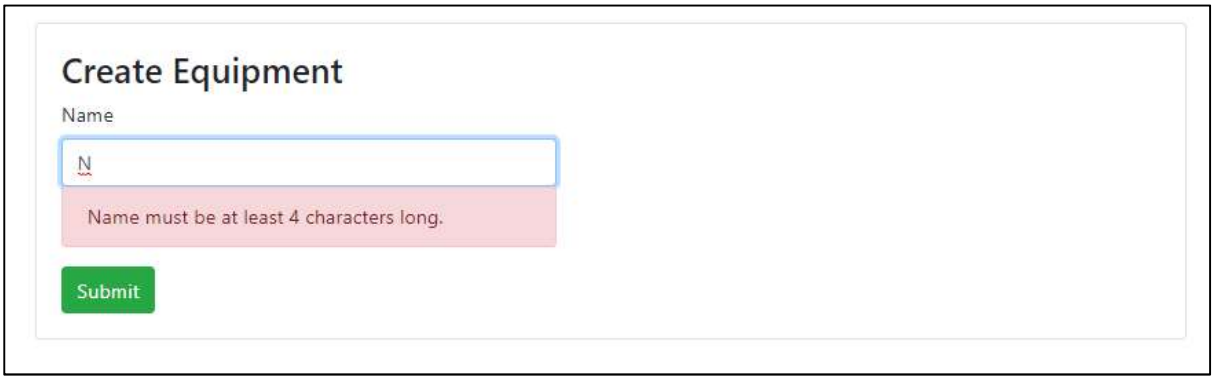


Figura 145: Validacions formularis, mida minima

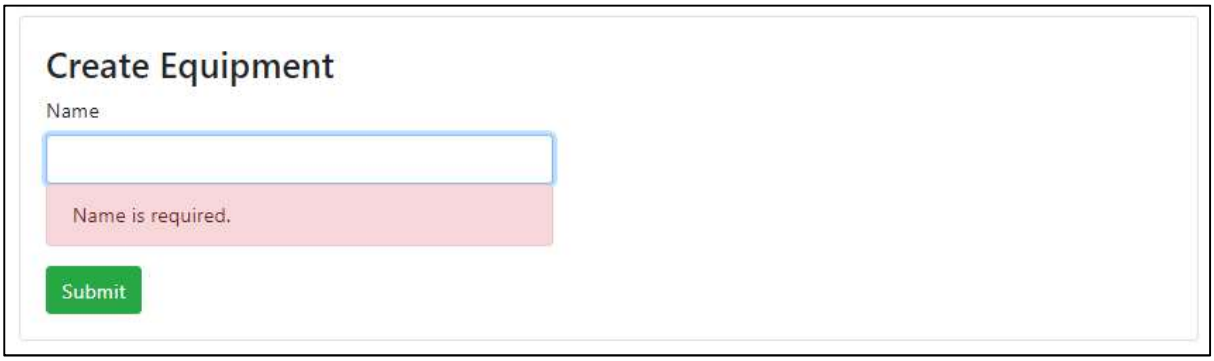


Figura 146: Validacions formularis, requerit

Figura 147: Validacions formularis, només permet numero

4. **Retroalimentació oportuna**. Quan l'usuari executa una acció, el sistema respon si va ser efectiva o no.

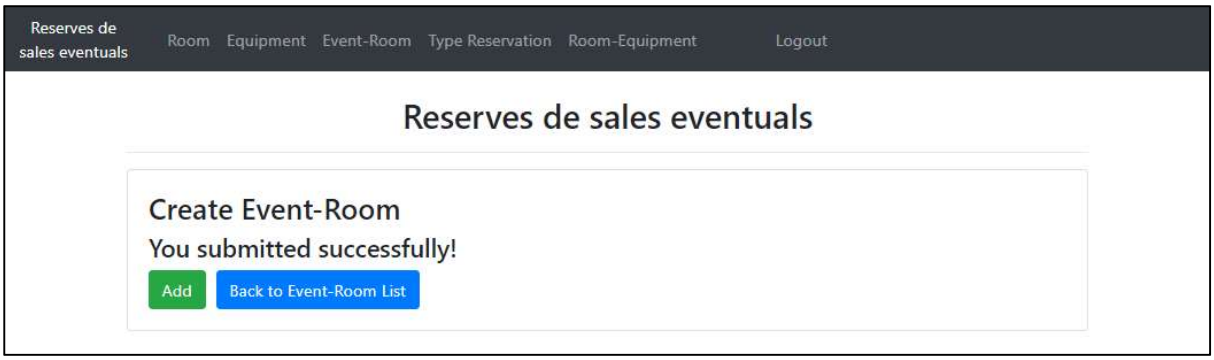


Figura 148: Retroalimentació oportuna

5. **Dissenyant pensant de manera estratègica**. S'ha dissenyat per fer ús en un navegador web, amb una mida de text i objectes com botons adequats. A més és responsive per mida de navegadors. S'ha destacat sobre tot el web el menú amb les diferents opcions de navegació.
6. **Simplificació**. Es una web senzilla, amb poques opcions, clares i ordenades.

Navegació

La navegació és sovint el que s'interposa entre l'usuari i el que es desitja que faci l'usuari en la web.

La navegació hauria de fer:

- Permetre als usuaris escollir entre una petita selecció de pagines a visitar.
- Proveir etiquetes clares per a les pagines en les pestanyes de navegació.
- Adaptar la pagina web per a que coincideixi amb les necessitats de l'usuari.
- Mostrar a l'usuari on es troba en tot moment i permetre tornar.
- Proporcionar una funció de cerca.

Amb la fi de poder assolir aquests objectius, s'ha dissenyant un menú superior horitzontal a l'aplicació.

Figura 149: Header de l'aplicació front-end

Al ser una aplicació petita de backoffice per a demostració, no s'ha realitzar la funció de cerca ja que tot el que es pot realitzar en la web es troba en el header.
Amb aquest menú complim els altres punts, permet escollir entre pagines, proveeix d'etiquetes clares, coincideix amb les necessitats de l'usuari amb rol Editor i mostra a l'usuari on es troba en tot moment.

Sitemap

Es presenta un diagrama del *sitemap* complet del nostre web:

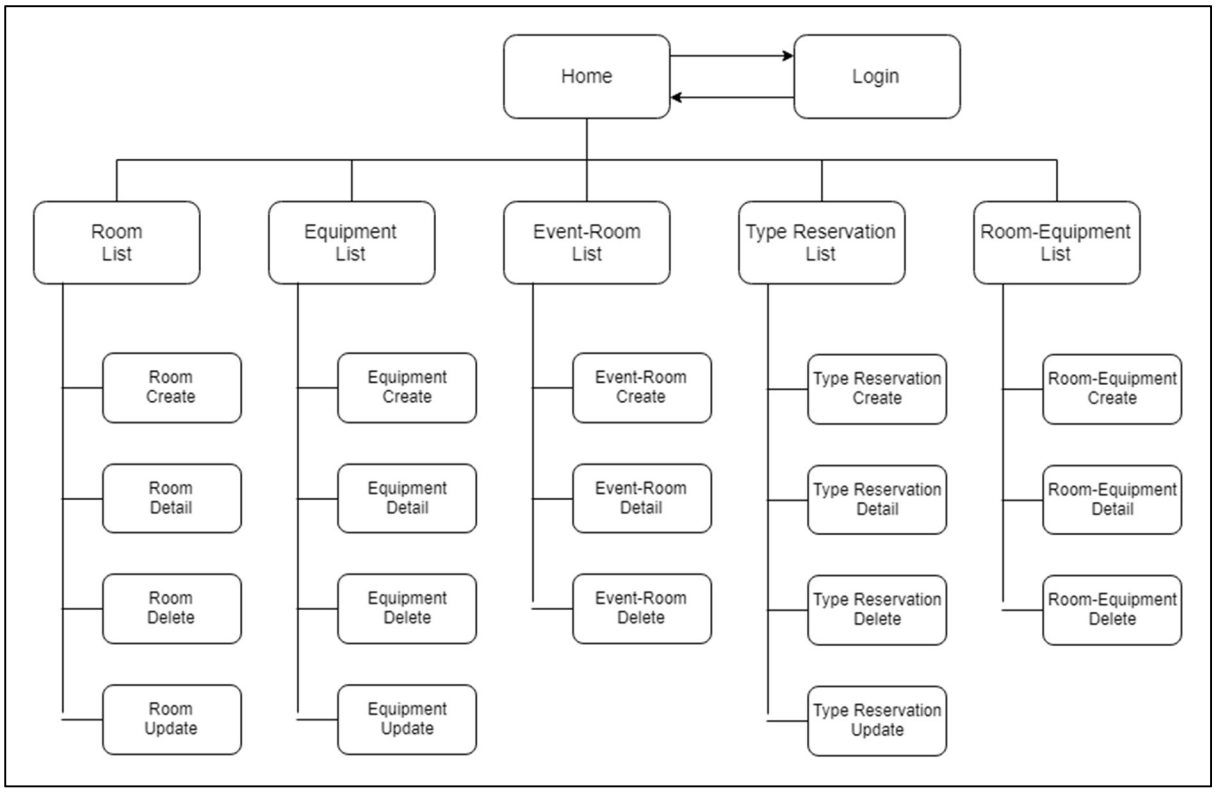


Figura 150: Sitemap

15. Seguretat

15.1. Mesures de seguretat global en tot el treball

Certificats SSL

S'ha aplicat certificats SSL al domini tfmaster.net i subdominis *.tfmaster.net, que són els utilitzats en tot el treball, tant en l'url de l'aplicació com en les urls de les eines de la infraestructura de CI/CD.
El SSL (secure socket layer), és un protocol de seguretat que garanteix que la transmissió de dades entre un usuari i el servidor i entre el servidor i un usuari per internet sigui completament segura.

Amazon Shield

AWS Shield Standard ofereix automàticament protecció davant els atacs DDoS més comuns que normalment ocorren en la capa de xarxa i transport dirigits a les aplicacions. A més, amb CloudFront i Route 53, també disposa de disponibilitat integral contra tots els atacs coneguts a la infraestructura (capa 3 i 4).

Features	AWS Shield Standard	AWS Shield Advanced
Active monitoring		
Network flow monitoring	✔	✔
Automated application (layer 7) traffic monitoring	-	✔
DDoS mitigations		
Helps protect from common DDoS attacks, such as SYN floods and UDP reflection attacks	✔	✔
Access to additional DDoS mitigation capacity	-	✔
Visibility and reporting		
Layer 3/4 attack notification and attack forensic reports	-	✔
Layer 3/4/7 attack historical report	-	✔
DDoS response team support		
Incident management during high severity events	-	✔
Custom mitigations during attacks	-	✔
Post-attack analysis	-	✔
Cost protection		
Reimburse related Route 53, CloudFront, and ELB DDoS charges	-	✔
Status	Activated	Not activated

Figura 151: Amazon Shield

15.2. Mesures de seguretat en l'aplicació Java i Angular

Spring Security

Amb la utilització del framework d'Spring Security en el desenvolupament de l'API, aconseguim protecció contra atacs del tipus:

- Session fixation. És un atac que permet a un atacant segrestar una sessió d'usuari vàlida. L'atac explora una limitació en la manera en que l'aplicació web gestiona l'ID de sessió.

L'atac consisteix a obtenir un ID de sessió vàlid, introduint a un usuari a identificar-se amb aquest ID de sessió i, després, segrestant la sessió validada per l'usuari.
En el nostre cas, no tenim ID de sessió ja que s'ha construït l'aplicació sobre controladors *sessionless*.

- Clickjacking. És una tècnica maliciosa que enganya als usuaris de web i que permet revelar informació confidencial d'aquests o prendre en control dels seus ordinadors, simplement fent clicks en pagines que semblen inofensives.
- Cross Site Request Forgery. (CSRF) és un ataca que obliga a un usuari final a executar accions no desitjades en una aplicació web en la qual actualment està autenticat. Els atacs CSRF tenen com a objectiu específic les sol·licituds de canvi d'estat, no el robatori de dades, ja que el atacant no té cap forma de veure la resposta a la sol·licitud falsificada.

Basic Auth

En el context d'una transacció HTTP, l'autenticació bàsica és un mètode dissenyat per a permetre un navegador web proveir de credencials en la forma d'usuari i password quan es sol·licita una pagina a un servidor.
Les credencials s'envien codificades en Base64 i per tant és reversible. Es un sistema fàcil d'implementar però no es recomanable utilitzar sobre línies de comunicacions públiques.
Tal i com està muntada la nostra infraestructura per al desplegament, entre el Cloud Front (front-end) i el Load Balancer (punt d'entrada a l'API), és una línia de comunicació pública. Per tant, no és un mètode aconsellar en aquest cas.

CloudFront

CloudFront és la xarxa de lliurament de contingut que utilitzem per al nostre front-end. CloudFront funciona de forma fluida amb serveis com AWS Shield per a mitigar atacs DDoS.

15.3. Mesures de seguretat en la infraestructura CI/CD

S'ha treballat tot en AWS amb security group. En aquests s'ha definit que ports resten oberts i a quines IPs se li donen accés.
Els ports 443 estan oberts per a tothom ja que són els ports https.
El port 22 de totes les instancies EC2, només resta obert per a la meva IP i per a la connexió és necessari l'arxiu .pem.
El port 22 del servidor de GitLab és més especial ja que s'utilitza per a realitzar els pull i push de git i ha d'estar obert per molta gent diferent i per al servidor Jenkins. En aquesta cas, s'utilitzen claus privades generades i públiques en el servidor de GitLab per als accessos.
La base de dades només es te accés des de les instàncies de l'aplicació i per IPs permeses per a realitzar gestió (la meva IP).

16. Tests

16.1. Tests d'integració en l'API

S'han desenvolupat tests d'integració amb spring-boot i TestRestTemplate sobre l'aplicació, posem una classe de test com a mode d'exemple:

```
package com.tfm.reservas;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotNull;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.web.client.HttpClientErrorException;

import com.tfm.reservas.model.Event;


@RunWith(SpringRunner.class)
@SpringBootTest(classes = ReservasApplication.class, webEnvironment =
SpringBootTest.WebEnvironment.RANDOM_PORT)
public class EventTests {

    @Autowired
    private TestRestTemplate restTemplate = new TestRestTemplate("dhidalgo","dhidalgo");

    @LocalServerPort
    private int port;

    @Value("${host}")
    private String host;

    private String getRootUrl() {
        return host + ":" + port;
    }

    @Test
    public void contextLoads() {
    }

    @Test
    public void testGetAllEvents() {
        HttpHeaders headers = new HttpHeaders();
        HttpEntity<String> entity = new HttpEntity<String>(null, headers);

        ResponseEntity<String> response = restTemplate.exchange(getRootUrl() + "/events",
                HttpMethod.GET, entity, String.class);

        assertNotNull(response.getBody());
    }

    @Test
    public void testGetEventById() {
        Event event = restTemplate.getForObject(getRootUrl() + "/events/1", Event.class);
        System.out.println(event.getName());
        assertNotNull(event);
    }

    @Test
    public void testCreateEvent() {
        Event event = new Event();
        event.setName("Evento Demo");
        event.setDescription("Descripción Evento Demo");
        event.setCreatedBy("admin_demo");
        event.setUpdatedBy("admin_demo");
```

```
        ResponseEntity<Event> postResponse = restTemplate.postForEntity(getRootUrl() +
"/events", event, Event.class);
        assertNotNull(postResponse);
        assertNotNull(postResponse.getBody());
    }

    @Test
    public void testUpdatePost() {
        int id = 1;
        Event event = restTemplate.getForObject(getRootUrl() + "/events/" + id, Event.class);
        event.setName("Evento Demo 2");
        event.setDescription("Descripción Evento Demo 2");

        restTemplate.put(getRootUrl() + "/events/" + id, event);

        Event updatedEvent = restTemplate.getForObject(getRootUrl() + "/events/" + id,
Event.class);
        assertNotNull(updatedEvent);
    }

    @Test
    public void testDeleteEvent() {
        int id = 2;
        Event event = restTemplate.getForObject(getRootUrl() + "/events/" + id, Event.class);
        assertNotNull(event);

        restTemplate.delete(getRootUrl() + "/events/" + id);

        try {
            event = restTemplate.getForObject(getRootUrl() + "/events/" + id, Event.class);
        } catch (final HttpClientErrorException e) {
            assertEquals(e.getStatusCode(), HttpStatus.NOT_FOUND);
        }
    }

}
```

16.2. Tests d'estrès

Amb un tests d'estrès volem realitzar proves de quantes peticions aguanta el nostre servei en un determinat nombre de temps.
Per a realitzar aquestes proves ens ajudarem d'una eina anomenada jMeter, en la qual configurarem un pla de proves i validarem la salut del servei amb gràfiques proporcionades per Java Melody i la mateixa consola d'AWS.

Configurem un nou pla de proves en jMeter. Podem començar a un pla de proves fàcil, només amb reads a la base de dades:

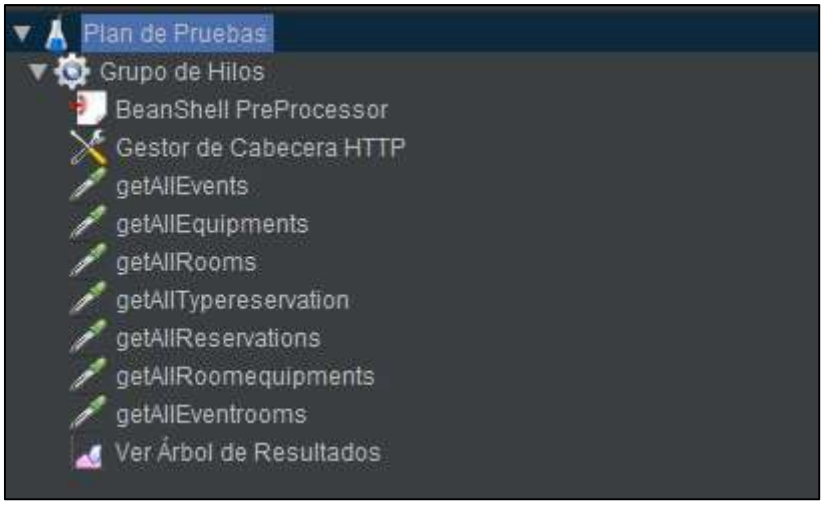


Figura 152: Configuració d'un grup de fills en jMeter

Creem un Grup de Fills, on configurem quants fills volem i quantes vegades volem que s'executi aquest grup de fills (en bucle).

Sota aquest grup de fills, afegim:

- 1 BeanShell PreProcessor
- 1 Gestor de Capçalera HTTP.
- Totes les crides que volem (en el exemple es veuen crides de read a la bbdd)
- 1 Arbre de resultats.

En el BeanShell PreProcessor codifiquem en Base64 usuari:password i creem una nova variable per al pla de proves anomenada base64HeaderValue.

```
import org.apache.commons.codec.binary.Base64;
byte[] encodedUsernamePassword =
Base64.encodeBase64("dhidalgo:xxxxxxxxxxxxxxx".getBytes());
vars.put("base64HeaderValue",new String(encodedUsernamePassword));
```

Aquesta variable l'utilitzem en el Gestor de Capçalera HTTP ja que les nostres criden han de portar aquesta variable ja que són autenticades:

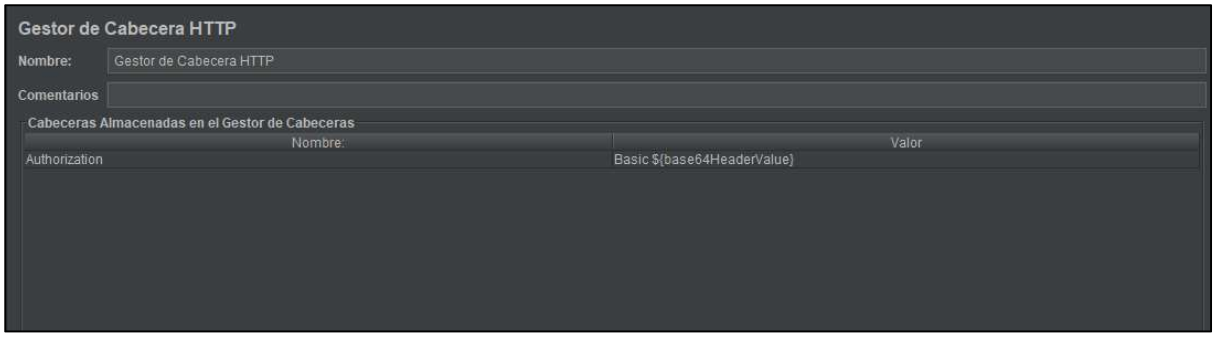


Figura 153: Capçalera HTTP en jMeter

Creem totes les peticions HTTP que volem testejar.

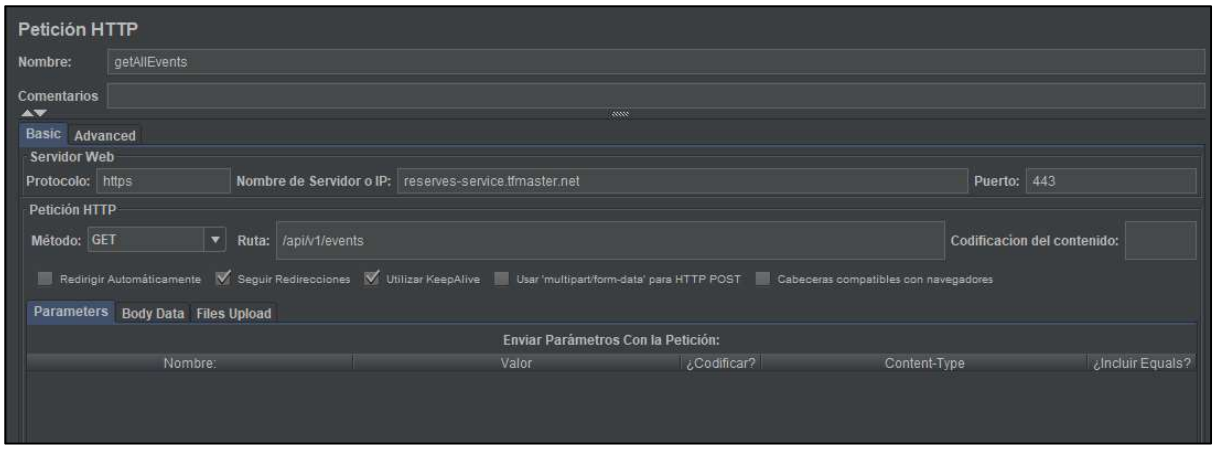


Figura 154: Capçalera HTTP en jMeter

I finalment podem afegir un Arbre de Resultats que veurem els resultats i informació de totes les peticions HTTP llançades en el bucle configurat en el grup de fills.

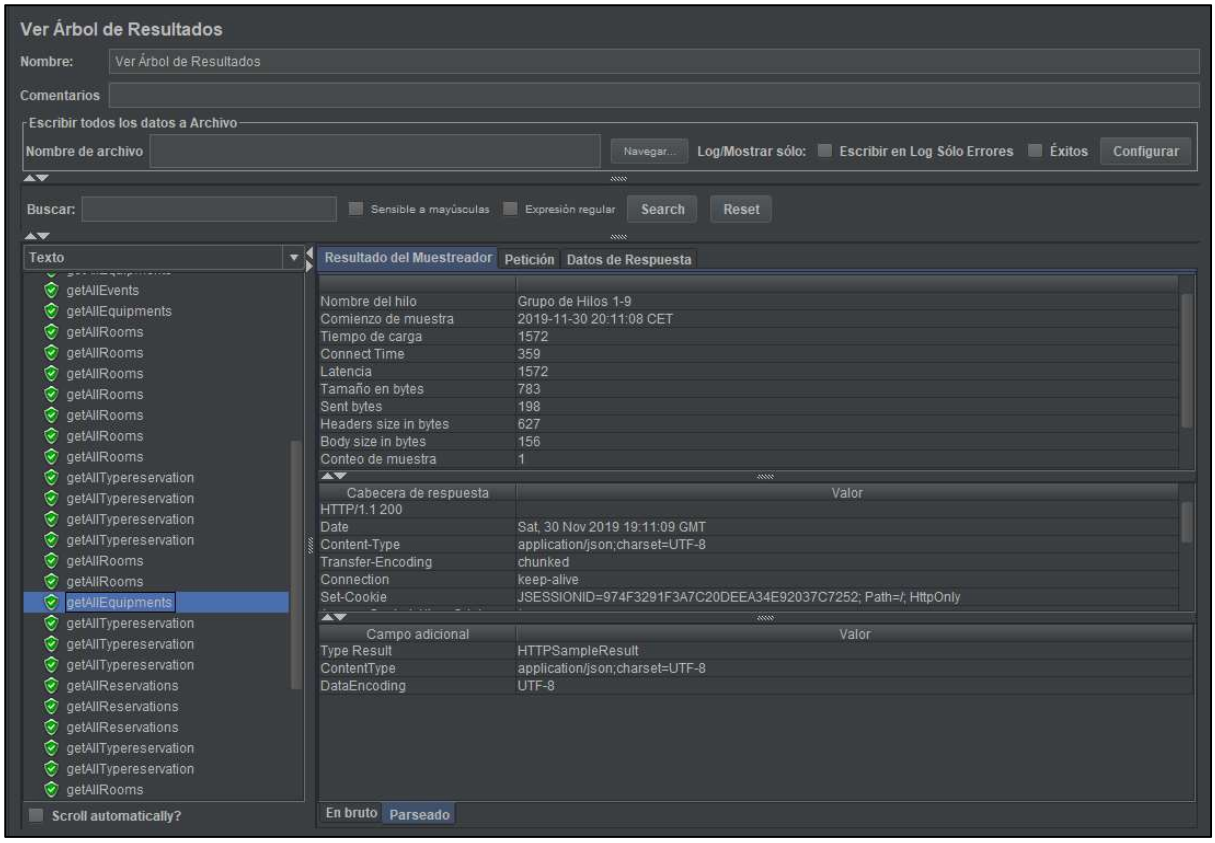


Figura 155: Arbre de resultats en jMeter

Per a realitzar la prova, s'ha configurat 30 threats en un bucle infinit.

Si consultem Java Melody, veiem clarament pics en CPU, http sessions, active threats, etc.

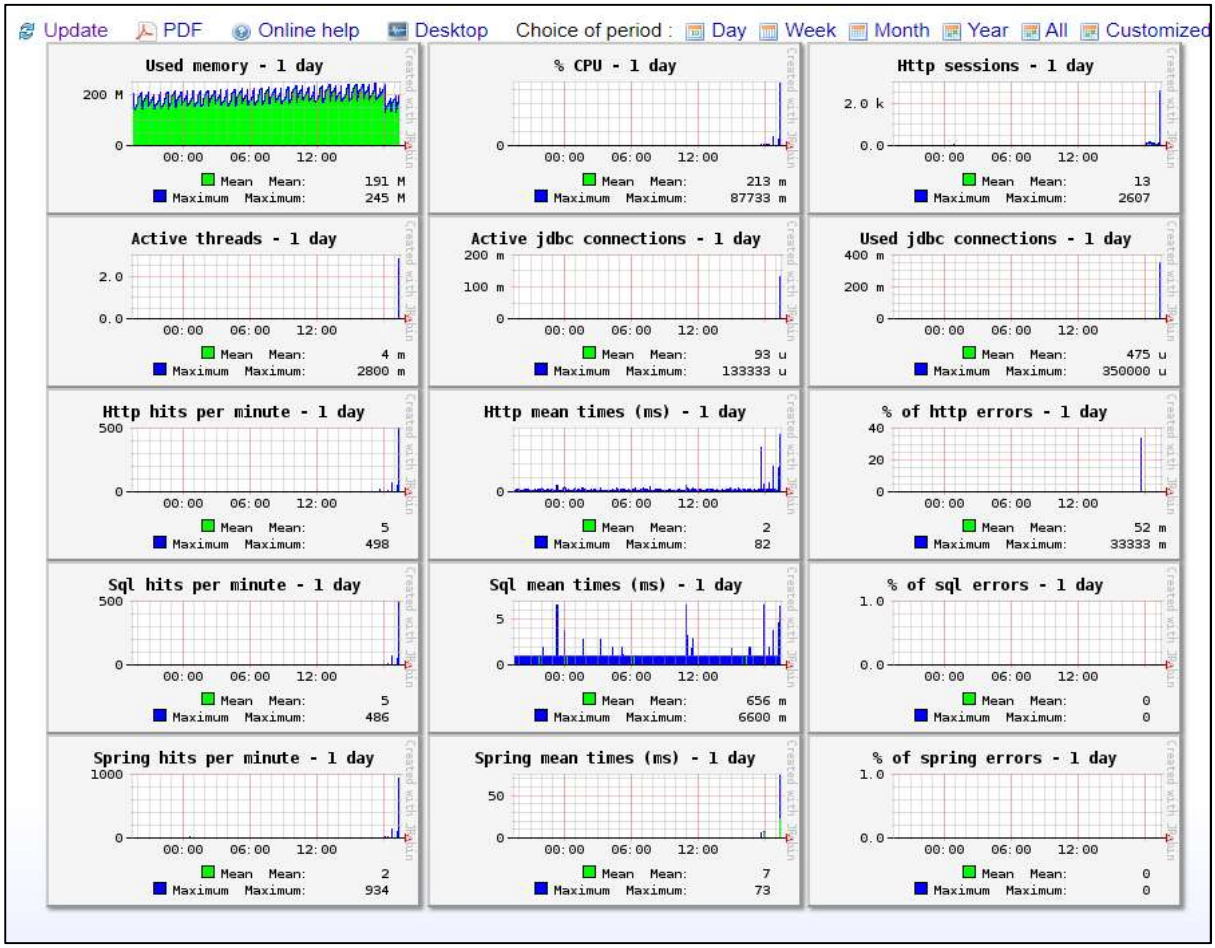


Figura 156: Resultats estrès en Java Melody

En la consola d'AWS, visualitzem les mètriques del clúster i ens donen un resultat molt semblant.

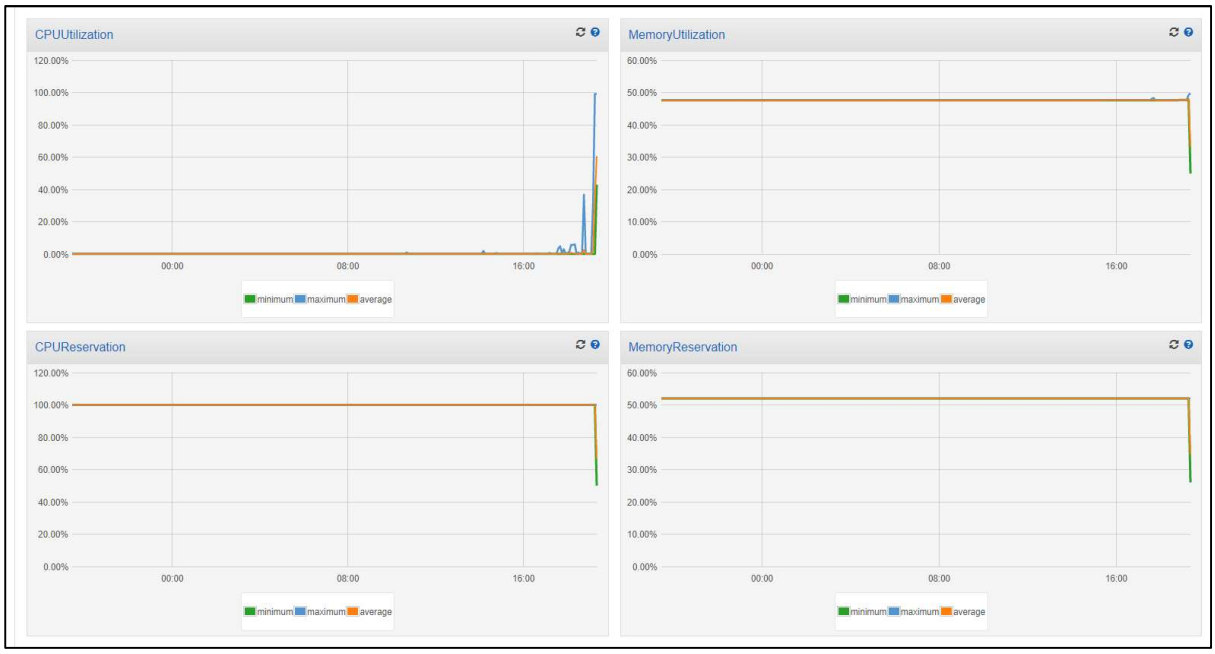


Figura 157: Resultats estrès en Clúster d'AWS

A més podem visualitzar com s'ha aixecat una nova instancia per a poder fer-se càrrec del alt nombre de peticions ja que la mitja ha sigut uns 500 hits HTTP per minut segons les gràfiques.

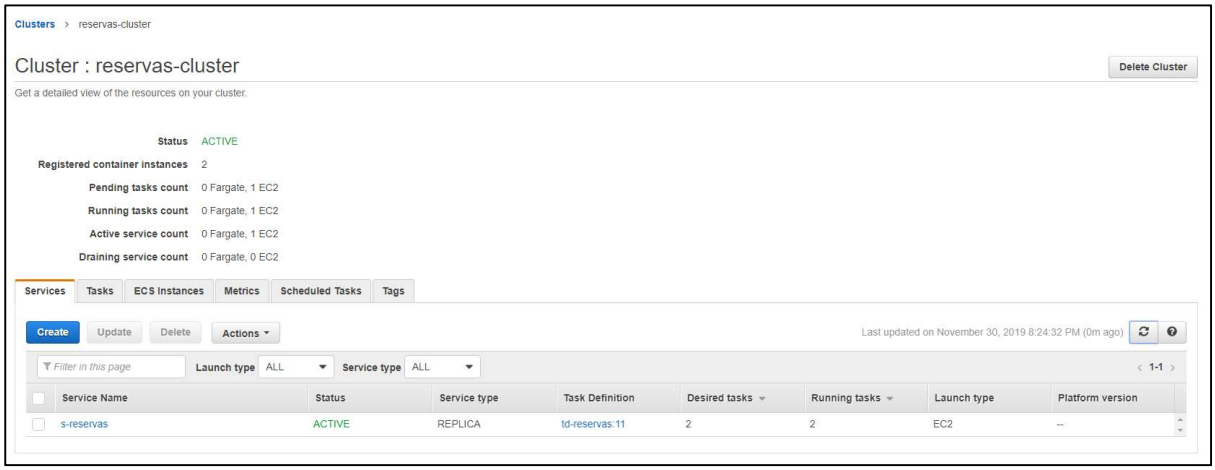


Figura 158: Resultats estrès, nova instancia

16.3. Tests de seguretat

OWASP Zed Attack Proxy

Es fa servir per als tests de seguretat l'*OWASP Zed Attack Proxy* (ZAP). És un software mantingut per cents de voluntaris i ens pot ajudar a trobar vulnerabilitats de seguretat en la nostra aplicació web.

Es llança contra l'URL publicada (https://reserves.tfmaster.net)

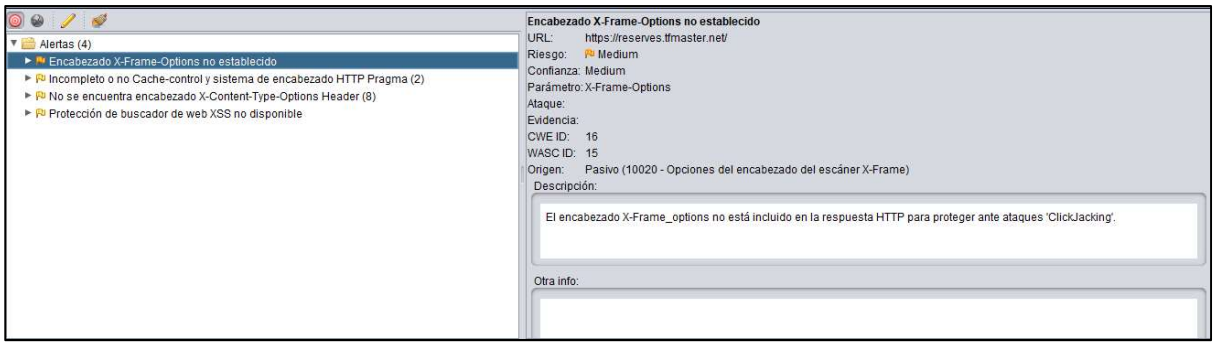


Figura 159: Resultats ZAP

Nessus

És un programa d'escaneig de vulnerabilitats. Comença escanejant els ports i després intenta explotar les vulnerabilitats.

Es registren dos hosts. Un contra el front-end i un altre contra l'api, ja que encara que tingui autenticació, es una api pública a la qual podem arribar a partir d'una url.

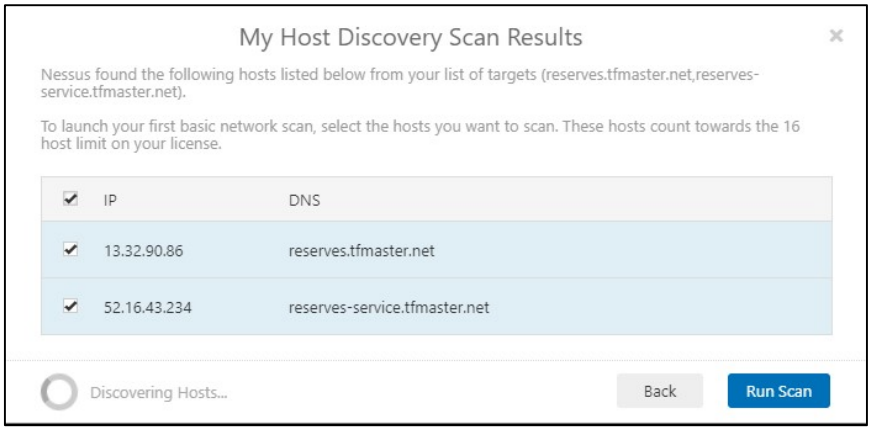


Figura 160: Host Discovery de Nessus

Ens mostra el resultat a mode de gràfic. El resultat exportat s'ha inclòs en l'*Annex 5*.

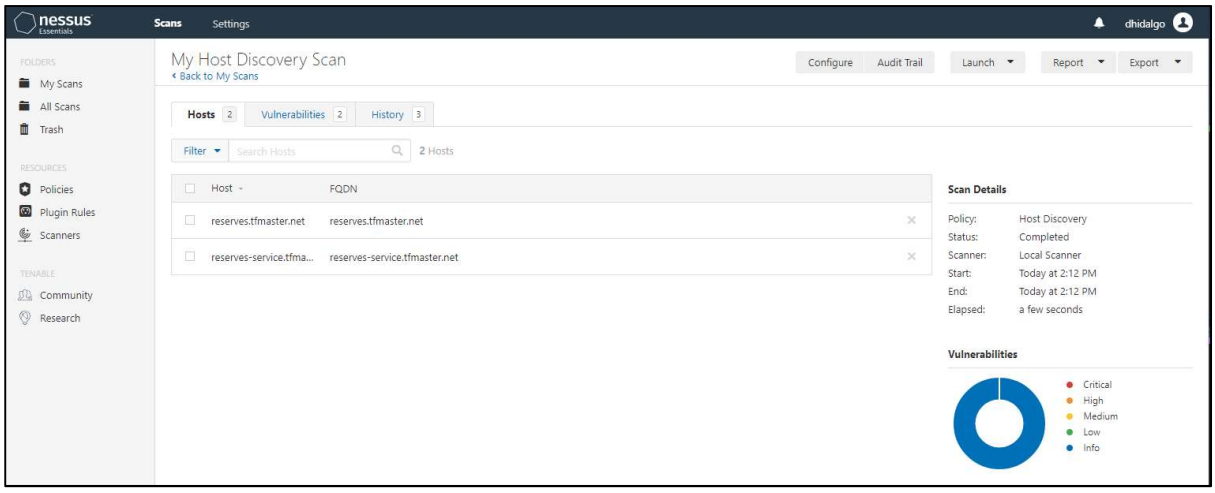


Figura 161: Resultats de Nessus

17. Requisits d'instal·lació/implantació/ús

17.1. Requisits d'instal·lació i d'implantació de l'aplicació reserva de sales eventuals

Requisits de hardware

L'aplicació funciona en un clúster amb una instancia de mida t2.micro* en AWS. Aquest clúster està configurat per l'auto escalat horitzontal, es a dir, si la maquina arriba, en el nostre cas, a un 60% de CPU, instantàniament aixeca una altre instancia de la mateixa mida amb l'aplicació fins a un màxim configurat de 4 instancies.

És necessari un servidor de base de dades. És una instancia RDS d'AWS de MySQL de mida db.t2.micro*.

**t2.micro. 1 vCPU, 1gb de memòria RAM.*
**db.t2.micro. 1 vCPU, 1gb de memòria RAM.*

Per al front-end es necessari un bucket d'S3 i un servei de lliurament de contingut (CDN) que en el nostre cas, fem ús del servei d'AWS CloudFront.

Requisits de software

Es necessària una màquina amb un sistema operatiu, i el JRE de Java versió 1.8.
La base de dades necessita un MySQL Server versió 8.0.16.

17.2. Requisits d'instal·lació i d'implantació de CI/CD

Requisits de hardware

Com que és un projecte en Cloud públic d'AWS, disposem de tots els recursos de hardware que necessitem.
S'ha procurat utilitzar els recursos més ajustats possible.
Cadascun dels serveis s'han desplegats en servidors t2.medium, servidors que permeten treballar còmodament. Evidentment en un entorn productiu amb més usuaris, potser els servidors quedin una mica petits i s'haurien d'ajustar a les necessitats.

**t2.medium. 2 vCPU, 4gb de memòria RAM.*

Per a les Lambda, s'ha configurat el mínim de memòria, 128 mb.
Es realitzen snapshots dels servidors de Jenkins i GitLab però s'esborren snapshots antics.

Requisits de software

GitLab CE, versió 12.5.0
Jenkins, versió 2.190.3

SonarQube Community Edition, versió 8.0

Tot el software utilitzat és OpenSource i es poden trobar descarregar per a diferents SO, Docker, etc.

Formació

Formació per als desenvolupadors.

Els desenvolupadors haurien de formar-se en l'ús de les noves eines, git i GitLab i com treballar amb branques i realitzar push al servidor.

Una petita formació en Jenkins per a executar el jobs, ja que només tindran accés a aquesta opció.

SonarQube possiblement no es necessiti formació de l'eina concretament, sinó saber implementar els resultats de l'auditoria de codi que ens proporciona.

Formació per a administrador de sistemes.

Els administradors de sistemes necessiten una amplia formació en l'ús d'AWS.

Formació en l'administració de Jenkins i de com realitzar nous jobs.

Formació en l'administració de GitLab.

Formació en l'administració de SonarQube.

18. Instruccions d'instal·lació/implantació

18.1. Instal·lació del microservei

S'han creat dos profiles en l'aplicació, *development* i *production*.
L'entorn de desenvolupament (profile *development*) és en local i l'entorn productiu és el desplegament en AWS.
L'aplicació necessita connectar-se a una base de dades quan s'executa, per canviar les dades referents a la base de dades s'han de modificar els següents fitxers:

- application-development.properties
- application-production.properties

L'esquema de base de dades, el nom de l'esquema, ha d'estar creat, i s'ha d'anomenar "*reservas*".

A part de la base de dades, en desenvolupament no necessitem cap mena d'infraestructura, ja que aprofitem Spring Boot per executar-la amb el pluging *spring-boot* i el goal *run*, marcant el profile *development*.

Per a empaquetar l'aplicació amb el profile de *production*, hem d'executar la comanda:
`mvn clean install -P production`
Aquesta comanda te com a resultar un arxiu .jar amb els compilats i les llibreries en el directori intern /lib.
El podem llançar per executar-lo amb una comanda de Java:
`java -jar nom_de_la_app.jar`

S'inclou a l'arrel del projecte l'arxiu README.md amb les instruccions d'execució i instal·lació.

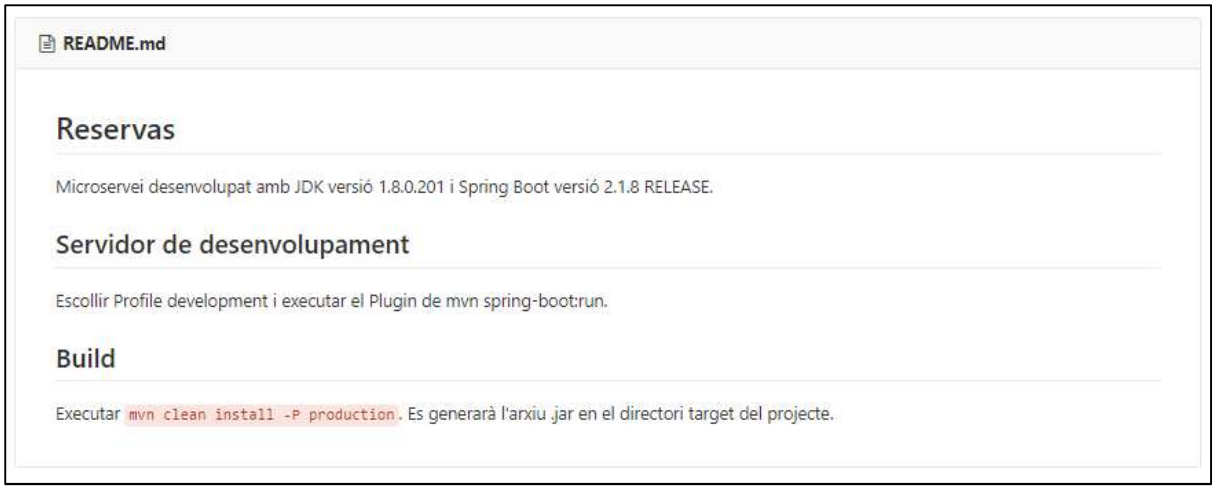


Figura 162: README.md de l'API

18.2. Instal·lació del front-end

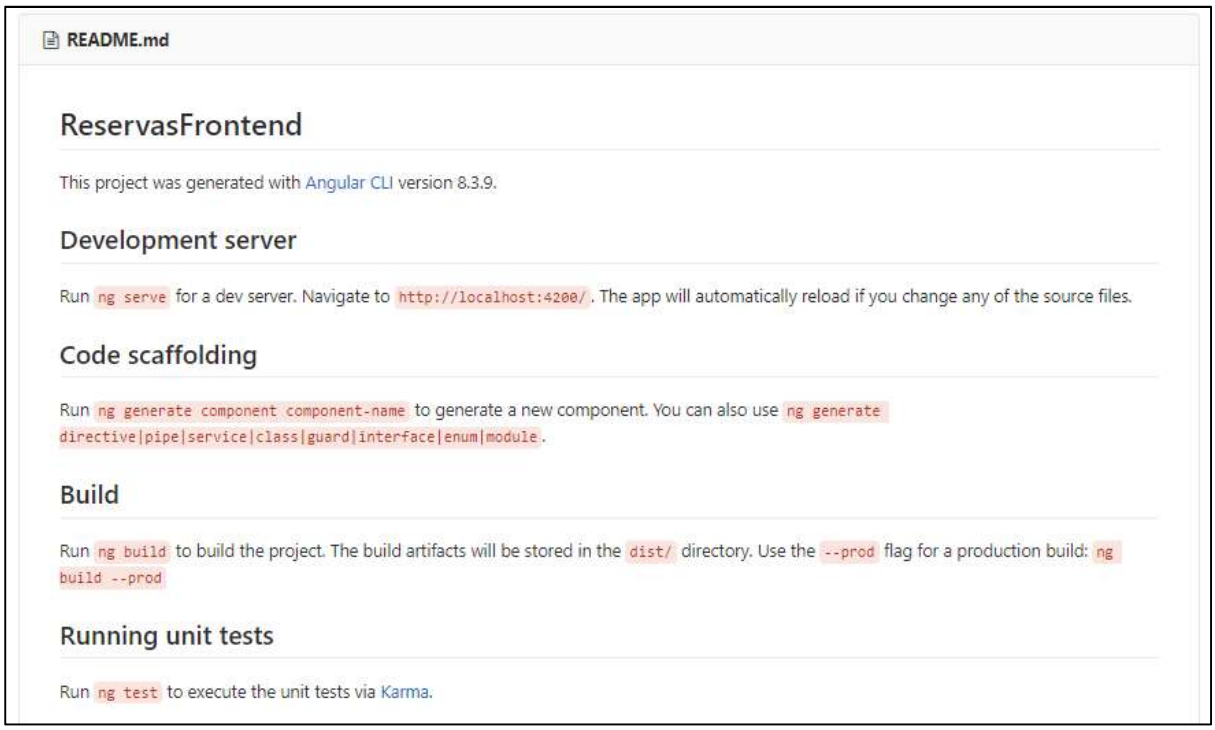
S'han utilitzat també en aquesta ocasió dos profiles en aquesta aplicació.
L'entorn de desenvolupament (arxiu *enviroment.ts*) és en local i conté la variable *production:false* i l'entorn productiu està configurat en l'arxiu *enviroment.prod.ts* que conté la variable *production:true*.
També es defineix l'url de l'api on ha de connectar.

Per a crear els arxius distribuïbles de l'aplicació s'ha de llançar la comanda:

```
ng build --prod
```
Si la volem portar a l'entorn productiu.

Per executar l'aplicació, la comanda és:
```
ng serve --prod
```

S'inclou a l'arrel del projecte l'arxiu README.md amb les instruccions d'execució i instal·lació.



README.md

ReservasFrontend

This project was generated with Angular CLI version 8.3.9.

Development server

Run `ng serve` for a dev server. Navigate to `http://localhost:4200/`. The app will automatically reload if you change any of the source files.

Code scaffolding

Run `ng generate component component-name` to generate a new component. You can also use `ng generate directive|pipe|service|class|guard|interface|enum|module`.

Build

Run `ng build` to build the project. The build artifacts will be stored in the `dist/` directory. Use the `--prod` flag for a production build: `ng build --prod`

Running unit tests

Run `ng test` to execute the unit tests via Karma.

Figura 163: README.md del front-end

18.3. Instal·lació Lambda backup i neteja

En l'apartat *11.10. AWS Lambdes per a realitzar snapshots* s'explica com s'ha treballat per a la instal·lació de les dues lambda amb detall.

Es pot consultar aquesta informació també en l'arxiu README.md inclòs en l'arrel del projecte de les lambda.

README.md

backups-lambda

Python scripts per a executar-se com a servei d'AWS Lambda per a crear i esborrar Snapshots de Volums EBS.

[Modificat des del codi de github: https://github.com/cmachler/aws-lambda-ebs-backups]

Configurant IAM Permissions

Primer hem de crear una política IAM anomenada ebs-backup-worker seguint el document següent:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:*"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:CreateSnapshot",
                "ec2:CopySnapshot",
                "ec2:DeleteSnapshot",
                "ec2:CreateTags",
                "ec2:ModifySnapshotAttribute",
                "ec2:ResetSnapshotAttribute"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "*"
        }
    ]
}
```

Continuem creant un role IAM anomenat ebs-backup-worker, seleccionem AWS Lambda com a tipus de rol i li associem la política que acabem de crear.

Es pot comprovar la relació en el rol en "Edit Trust Relationship" deuria semblar a això:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Afegint AWS Regions a la variable "aws_regions".

AWS Lambda, no admeten valors separats per comes en les variables d'entorn, per això no podem afegir un llistat de regions que volem que passi l'script Python de la Lambda. Per a solucionar això, codificarem el llistat de regions en Base64 i en el codi font, el descodifiquem i fem un Split per les comes, així aconseguim una llista d'elements

Utilitzem Python per aconseguir el Base64 del llistat de les regions:

```
Python 3.8.0 (tags/v3.8.0:fa919fd, Oct 14 2019, 19:21:23) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import base64
>>> encoded = base64.b64encode(b'eu-west-1')
>>> encoded
b'ZXUtd2VzdC0x'
>>> data = base64.b64decode(encoded)
>>> data
b'eu-west-1'
>>>
```

En les variables d'entorn de la lambda hem de crear una variable anomenada aws_regions amb el valor ZXUtd2VzdC0x que ens ha donat com a resultat l'script de Python.

Si es vol publicar mitjançant SNS, afegir la variable "aws_sns_arn"

ÉS una variable d'entorn opcional. Si es vol publicar en SNS, s'ha d'afegir la variable amb valor l'arn de la publicació.

Creant les funcions Lambda

Hem de crear dues AWS Lambda. Una per fer funcionar l'script que realitza els snapshots i una altra és un script de neteja. Escollim com a llenguatge Python 3.7 i com a rol d'execució el rol ebs-backup-worker que hem creat en aquest mateix punt. Creem un trigger, del tipus CloudWatch Events i amb una expressió de programació que és un cron. L'expressió serà cron(0 6 ? * MON *) és a dir, tots els dilluns a les 6:00h. s'executarà.

De la mateixa manera que s'ha descrit, hem de crear un altre AWS Lambda, aquesta anomenada backupCleanupReserves que contindrà el script de neteja. El cron en aquesta Lambda seria de posterior execució a l'anterior: cron(0 7 ? * MON *)

Tagging your EC2 instances to backup

Ara hem de crear tags en les instancies EC2 sobre les que volem que actuï les Lambdes.

Tag Key	Tag Value	Notes
Backup		Value Not Needed
Retention	*Number of Days to Retain Snapshot*	Default is 7 Days
Skip_Backup_Volumes	*volume id(s) in CSV string*	List either a single volume-id, or multiple volumes-ids in a Comma Separated Value String

Figura 164: README.md del projecte de Lambda backup

19. Instruccions d'ús

19.1. Microservei i front-end

L'aplicació conté les llibreries de swagger. Swagger és una especificació oberta per a definir API Rest. El document swagger especifica el llistat de recursos disponibles en l'API del nostre microservei.
Es pot consultar les figures 21 i 22 i l'apartat *10.9. Documentació d'API* per a més informació.
Es pot accedir en la següent url: https://reserves-api.tfmaster.net/swagger-ui.html#/

El projecte service i el projecte front-end compten amb un arxiu Readme.md amb instruccions. Veure apartat *18. Instruccions d'instal·lació/implantació* per a més referencia.

19.2. Serveis infraestructura CI/CD.

No es pot abastar un manual d'us per a tota la infraestructura d'Amazon Web Services. En aquesta memòria s'ha intentat que fos també un manual de referencia.

Referent a l'ús dels serveis per a rols d'usuaris i no d'operaris que han de mantenir la infraestructura, es pot consultar l'*Annex 5. Guia d'usuari*.

20. Projecció a futur

Referent a l'arquitectura, el primer que es tindria que realitzar seria un entorn d'staging. Separat de l'entorn de producció i on es realitzarien els tests i s'empaquetaria per a la pujada a producció.
Per tant, el servi de GitLab i SonarQube es quedarien en entorn d'staging, S'hauria d'instalar un servidor per a artefactes, tipus Nexus o guardaria els paquets realitzats en staging.
Es tindria que repensar els jobs de Jenkins o realitzar de nous per a agafar el paquet compilat en staging i desplegar en producció. Es pot valorar també realitzar-ho en subcomptes d'AWS per un major ordre.

Per a fer un entorn més complet, es podria instal·lar un servidor de Test-Link per a integrar els casos d'ús amb un Jira o un software que s'utilitzi.

Infraestructura com a codi va ser en un principi un objectiu que va caure per temps. Tenir tota la infraestructura dels desplegaments com a codi, versionat en GitLab.

Crear alertes en els servidors i connectar-les amb un servidor Nagios.

Per als serveis, plantejaria la creació d'un nou microservei que fes ús del servei AWS Cognito per al login d'usuaris i APIs en els serveis desplegats (o potser connectar-lo a un LDAP) però centralitzant els usuaris de totes les aplicacions i autoritzant amb rols d'usuaris que podrien ser grups d'LDAP per exemple.

Amb la versió de pagament de GitLab, es pot integrar que al realitzar un push a una branca cridi a job de Jenkins. No va donar temps ja que la versió instal·lada era la CE i s'havia d'instal·lar la versió EE (i pagar).

21. Pressupost

Equip de desenvolupament per al servei de reserves.

Back-end: 1 desenvolupador back-end 5 jornades.

Front-end: 1 desenvolupador front-end 4 jornades.

Tests de seguretat: 1 administrador de seguretat 1 jornada.

Equip per a la infraestructura per al desplegament del servei de reserves.

1 administrador de sistemes cloud 2 jornades.

Equip per a la construcció de la infraestructura de CI/CD.

Creació infraestructura i instal·lació de serveis: 1 administrador de sistemes 12 jornades.

Creació jobs Jenkins: 1 administrador de sistemes 4 jornades.

Formació.

A l'equip de desenvolupadors: 1 jornada.

A l'equip d'operacions: 3 jornades.

Compra domini web: 11$

Infraestructura amb ús moderat:

La consola de facturació d'AWS detalla el cost fins a la data i el cost mensual per servei.

S'ha calculat que aproximadament serien un 30€ mensuals amb les màquines especificades en el punt *19. Requisits d'instal·lació/implementació*.

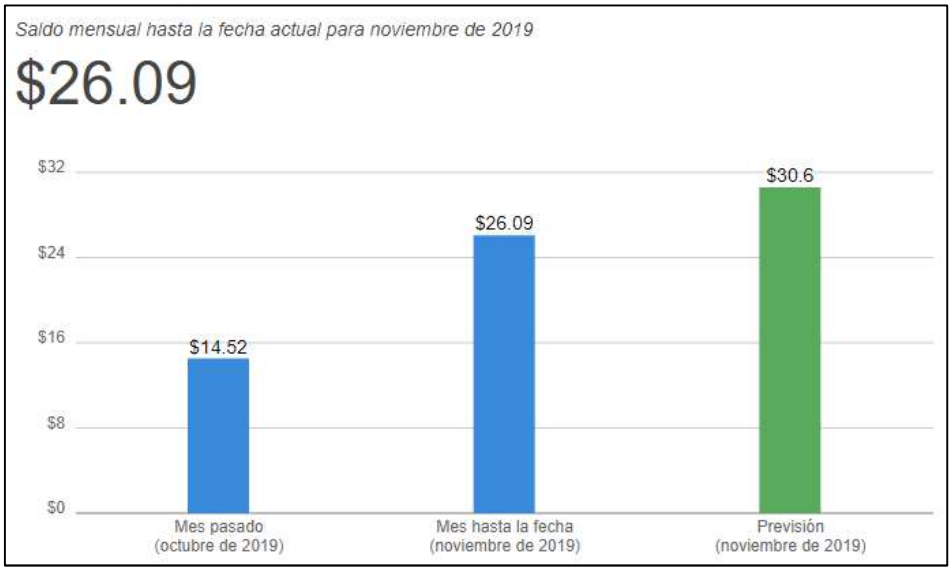


Figura 165: Cost mensual serveis AWS

Figura 166: Cost mensual fins a la data per serveis

22. Beneficis de la implantació de pràctiques DevOps en una organització

Quins beneficis aporta la implementació de pràctiques de DevOps?

No és d'estranyar, les organitzacions que han implementat pràctiques de DevOps tenen fins a cinc vegades més probabilitats de tenir un alt rendiment que les que no ho han estat. De fet, com més llargues les organitzacions utilitzen pràctiques de DevOps, millor serà el seu rendiment.

Ara que ja sabem quin és el rendiment d'alt rendiment, com s'aconsegueix realment? Les organitzacions d'alt rendiment comparteixen dues pràctiques comunes:

- El 89% utilitza sistemes de control de versions per a la gestió d'infraestructures.
- El 82% ha automatitzat els seus desplegaments de codi.

Utilitzant el control de versions com a font única de veritat, podreu identificar la causa dels falles, tornar-vos a enviar a un estat conegut i desplegar ràpidament noves instàncies de servei. Tot el necessari per executar una aplicació, incloent-hi la infraestructura i el codi de configuració, ha de tenir el control de versions.

L'automatització de desplegaments de codi proporciona diversos avantatges que contribueixen directament a un alt rendiment. Primer, automatitzar la configuració de les màquines de desenvolupament, prova i producció, elimina la deriva de configuració entre ambients, un punt comú de fracàs en el procés de desplegament. En segon lloc, l'automatització redueix significativament el temps de canvi canviant substituint els treballs manuals per un procés constant i repetible. En tercer lloc, les eines d'automatització donen visibilitat a l'impacte dels canvis abans que es promoguin a la producció, reduint el risc.

DevOps augmenta l'agilitat i la fiabilitat. Organitzacions d'alt rendiment

Les organitzacions d'alt rendiment:

- **Despleguen codi 200 vegades més freqüentment** i **2555 vegades més ràpid** que la resta d'organitzacions, desplegant-se varies vegades al dia, enfront d'una mitjana d'un cop al mes. Els desplegaments freqüents i els temps d'administració de canvis més ràpids, permeten una agilitat operativa.
- Tenen **3 vegades menys de fallides**. Aquestes organitzacions d'alt rendiment, tenen el doble de taxa d'èxit en un canvi i **restableixen el servei 24 vegades més ràpidament** que altres organitzacions. Menys fallides i recuperació més rapida suposen menys risc per al negoci quan es despleguen nous canvis.

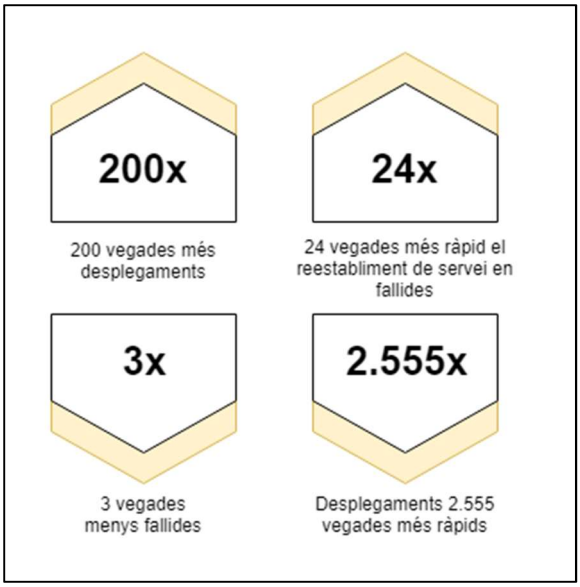


Figura 167: Organitzacions d'alt rendiment

- Millorar la qualitat és feina de tots. Les organitzacions d'alt rendiment dediquen un **21% menys de temps a treballar i no tornar a planificar-les**. Com a resultat, poden dedicar un 49% més de temps a treballs nous, com ara noves funcions o codi. Poden fer-ho perquè incorporen qualitat en cada etapa del procés de desenvolupament mitjançant l'ús de pràctiques de lliurament continuades, en lloc d'adaptar la qualitat al final d'un cicle de desenvolupament.

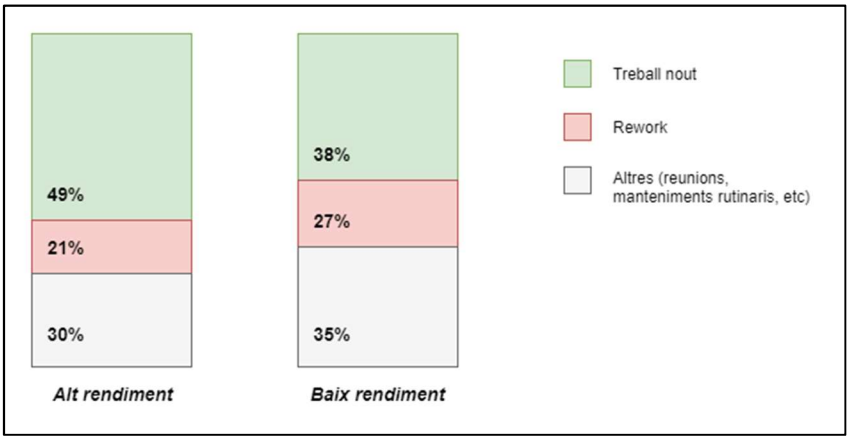


Figura 168: Treball nou VS. rework

- Les organitzacions d'alt rendiment dediquen un **50% menys de temps a eliminar problemes de seguretat** que els que presenten nivells més baixos. En integrar millor els objectius de seguretat de la informació en el treball diari, els equips aconsegueixen nivells més alts de rendiment informàtic i construeixen **sistemes més segurs**.
- L'aprofitament experimental del desenvolupament de productes pot **millorar el vostre rendiment informàtic i organitzatiu**. El cicle de desenvolupament del producte s'inicia molt abans que un desenvolupador comenci a codificar. La capacitat del vostre equip de producte per descompondre productes i funcions en lots petits; proporcionar visibilitat al flux de treball des de l'idea a la producció.
- Desenvolupant una transformació tecnològica pot **produir un estalvi de costos important** per a qualsevol organització. Tots els líders tecnològics volen saber què esperar per invertir en una transformació tecnològica.

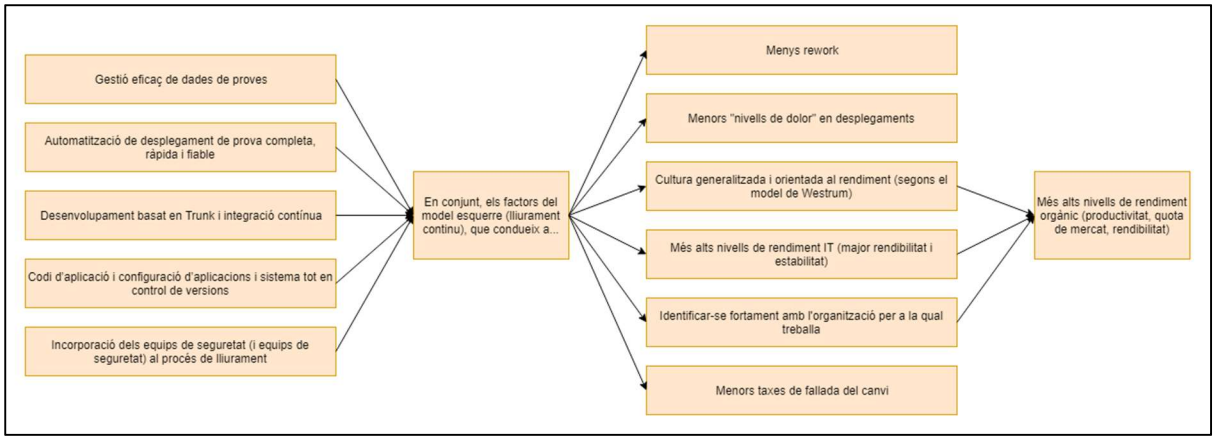


Figura 169: Reducció de la integració continua

Identificació de les barreres per a l'adopció de DevOps

Els processos i les eines DevOps contribueixen a un alt rendiment, però aquestes pràctiques per si soles no són suficients per assolir l'èxit organitzatiu. Les barreres més comunes a l'adopció de DevOps són les culturals: la manca de directius o la manca d'equip o el valor de DevOps no s'entén fora d'un grup específic.

- La manca de gestor com a bloqueig, és la causa més freqüent per a no implementación DevOps amb un 49%.
- La manca d'equip és la segona causa amb un 38%.
- Un 48% de les empreses que no han implementat processos DevOps, pensen que no s'entenen aquests processos fora del seu grup (TIC)

Quines habilitats són les més desitjades en un equip DevOps?

- Codificació / script per a automatitzar tasques. (84%)
- Habilitats socials. Comunicatiu, cooperatiu, són claus per a l'èxit DevOps. (60%)
- Experiencia amb eines no és una prioritat. (19%)

Recomanacions per a la implementació

Automatitzar. L'automatització és el motor per al alt rendiment, augmentant la qualitat i la velocitat de les implementacions de codi. Fins i tot les organitzacions informàtiques establertes poden fer millores incrementals mitjançant l'automatització. Es pot començar amb una petita automatització, que demostri el valor i utilitzar la visibilitat que suposa l'èxit per fer front a projectes més grans.
Trencar barreres culturals. DevOps no requereix la compra per part de tota l'empresa. Construir relacions "mitjançant reunions del passadís" permetrà que tothom entengui els problemes que enfronta les diferents parts de l'organització, cosa que suposa un llarg camí per aconseguir que tothom treballi cap als mateixos objectius.
Aprendre les eines. Compartir una cadena d'eines comuna pot ajudar a fomentar la comunicació entre equips i a difondre l'empatia sobre els reptes que es tenen davant.
Fomentar les habilitats de DevOps en els equips actuals. Amb seguretat membres dels equips ja tenen habilitats de DevOps. S'ha de fomentar i donar suport. No es tracta de contractar un nou equip

i formar un altra sitja. Es tracta de crear un equip transversal responsable d'un servei o d'un producte en concret.

Elaborar i utilitzar mètriques. Les mètriques són crucials per explicar la història de l'èxit. Ajuden a entendre el rendiment i l'equip, a més d'altres que demostren per què val la pena la inversió de DevOps. Mètriques com ara la taxa de desplegament, el temps de variació, la taxa de fallada de canvi i el temps mitjà per recuperar-se, poden mostrar el valor empresarial. Utilitzar mètriques funcionals com el temps de cicle de prova, el temps de desplegament, la taxa d'erros en la producció, per demostrar l'èxit.

Fomentar la comunicació lateral. Afavorir una cultura de comunicació directa entre iguals, en lloc d'utilitzar l'enfocament de dalt a baix. Sovint, les millors idees s'enrotllaran des de baix: Com més gent col·labori, més dinàmic serà l'intercanvi d'idees.

Beneficis de les organitzacions que ja han implementat DevOps

En una enquesta entre 4000 professionals d'IT s'han estret les següents dades:

- Millora de la qualitat del desenvolupament del programari (63%)
- Més releases del programari (63%)
- Millora en la visibilitat en el procés i requisits (61%)
- Col·laboració / Cooperació. Canvis culturals (55%)
- Més responsabilitat davant les necessitats empresarials (55%)
- Desenvolupament més àgil (51%)
- Procés de gestió de canvis més àgil (45%)
- Millora en la qualitat del codi (38%)

Calculant el ROI

Es pot calcular el ROI per temps no dedicat al rework:

Cost de rework = Total de persones tècniques implicades en desenvolupament o desplegaments * salari mig * percentatge de temps del personal tècnic gasta en rework.

Es pot calcular el ROI pels costos no generats de temps morts:

Cost de temps morts = Freqüència de desplegament * % de taxa de fallida en els canvis * Temps mitjà per recuperar-se * Cost horari d'aturada

23. Conclusió/-ns

Al principi del treball hem vaig marcar una sèrie d'objectius que vaig tenir present durant tot moment en el desenvolupament d'aquest.
El primer objectiu va ser el de conèixer i operar amb garanties en un Cloud públic. Des d'un primer moment tenia quasi segur que aquest Cloud públic seria Amazon Web Service. Perquè és el Cloud més extens ara mateix, perquè ofereix serveis molt competitius i per el seu grau de maduresa. Encara això, durant les primeres fases vaig realitzar un petit anàlisis sobre Azure i Google Cloud. Vaig dubtar molt per Google Cloud pel servei que ofereix amb Kubernetes en comptes del servei Amazon EKS (Kubernetes administrat). Finalment AWS va ser l'opció i penso que ha sigut un encert per varis motius, per la gran quantitat d'informació a la xarxa, etc.
Tornant al primer objectiu principal, conèixer i operar amb garanties en un cloud públic va ser un objectiu molt avariciós. AWS és gegant, molt més del que pensava en un moment inicial. Hem vaig centrar en els serveis més populars i vaig informar-me de quasi tots els serveis que ofereix. Vaig aconseguir tota la documentació per al curs AWS Fundamentals i me'l vaig preparar. Ara mateix penso que AWS es un monstre que creix cada dia i he de continuar coneixent.
El segon objectiu era obvi. Creació d'un microservei. Si volia realitzar una infraestructura completa en AWS, necessitava aplicacions per al seu desplegament. He sigut programador i cap tècnic desenvolupador molts anys, una aplicació web per a mi no era un repte. Així que hem vaig marcar reptes en el seu desenvolupament. Tenia que ser tot un exemple de bones pràctiques. Fa uns anys vaig llegir el llibre de Codi net i altres llibres de bones practiques. Estava preparat per a fer-ho. Només vaig fer un recull de bones practiques, metodologies i les vaig plasmar en el microservei. Ser un microservei era un "deure", dins del mon DevOps.
A més volia que el microservei es desplegués en un contenidor Docker i a més que fos escalable horitzontalment de manera automàtica. Es va crear una imatge Docker, programar una dockerfile i es va pujar per a la seva administració al servei EKS d'AWS.
Fer que un clúster fos escalable va ser tot un descobriment. Es van utilitzar alarmes de CloudWatch. Configurar alarmes i aquestes disparen esdeveniments. Per exemple, si la CPU puja del 40% prepara una nova instancia i si puja del 60%, posa-la en actiu. I a l'inrevés, si les CPUs baixen, fes fora una instancia i després destrueix-la. Aquí va ser molta prova-error. Tenia que donar temps per crear-la abans de fer que fos activa. També s'ha de protegir un mínim d'una instancia perquè sinó sense activitat destrueix totes i es queda a cero, etc.

Parlem de l'objectiu *extra*. En un principi no vaig pensar de realitzar un front-end, però finalment es va realitzar un front-end. Finalment estic content d'haver-ho realitzat ja que hem va permetre fer el desplegament continu de la part front-end d'una aplicació. Per a realitzar el front-end vaig dubtar entre Angular i React. No tenia quasi experiència en cap dels dos perquè en l'època que van prendre força ja no desenvolupava. Finalment vaig optar per Angular, però be, això va ser una decisió personal, era més repte, contínuament llegia que la corba d'aprenentatge d'Angular era major, però que després era més potent. Penso que qualsevol de les dues opcions podria haver sigut correcta i satisfactòria.

Volia realitzar tot el necessari i amb un nivell molt professional per a un desplegament continu. Vaig trobar-me que n'hi ha multituds d'opcions. Per delimitar les opcions, vaig pensar que realitzar amb un

software opensource seria bona idea. Necessitava un software pel versionat de codi. GitLab ofereix l'opció Community Edition que la pots instal·lar en un servidor propi. Necessitava una auditoria de codi, si les pujades a producció han de ser automàtiques, ens hem d'assegurar que el codi que pugi sigui fiable i comprovar que realment te una cobertura alta de codi amb tests unitaris. Per això SonarQube és l'eina més estesa. Algú ha de coordinar l'empaquetat, el desplegament, controlar que SonarQube doni ok en els resultats. Sense dubte, el més estens per aquests Jobs és Jenkins. Aquesta arquitectura es podria ampliar amb un servei per als tests case, com podria ser TestLink per exemple. Seria un bon afegit i molt desitjable.

No només volia realitzar tota aquesta infraestructura, sinó volia que fos segura, robusta, fiable i mantenible. Es va treballar molt intensament amb les eines d'AWS d'usuaris, rols i security groups. Addicionalment en temes de seguretat s'han instal·lat certificats SSL i s'han realitzat tests de seguretat amb dues eines diferents, tant a l'aplicació com a la infraestructura.
Volia com no podia ser d'altre manera, tenir backups dels serveis, potser no de tots en aquest entorn didàctic, però si aconseguir una manera fàcil per afegir els serveis amb molt poc esforç. Semblava que no tindria molta dificultat però no va ser així. Sobretot va ser un treball molt de recerca i proves de diferents opcions. Finalment vaig adaptar un desenvolupament en Python i vaig crear Lambda (serverless) per a realitzar snapshots i esborrar els snapshots amb 15 dies d'antiguitat. Ara mateix, afegir un servei seria tan fàcil com posar una label en la instancia EC2 que es volgués. Menys d'un minut afegir o eliminar un servei de backup. Estic molt content amb aquesta solució.

També volia realitzar un marc teòric sobre Agile, DevOps, CI/CD, microserveis, etc. Volia contextualitzar el treball en un marc, que no semblés que sorgia del no res. Que algú que agafes la memòria i comences la lectura en el marc teòric quedi convençut que necessita fer que la seva empresa adopti aquestes practiques i que ell, com a TI/TIC, ha d'impulsar aquest canvi.

Una dificultat que hem vaig trobar des d'un primer moment va ser Linux i sobretot els scripts sh. No vaig comptar amb això. Quan vaig decidir que sistema operatiu tindrien els servidors, sense dubtar va ser: -Es clar que Linux! Dubtava quina distribució de Linux, però no si serien Windows. El problema va ser quan vaig començar a treballar amb ells. Malacostumat a Windows, la creació d'usuaris per als serveis instal·lats, donar-li els permisos d'execució, fer que els serveis s'aixequessin quan arranqués el servidor i un llarg etcètera van fer perillar tant la programació de tot el treball com els objectius d'aquest. Així que vaig decidir fer una aturada, anava bé de temps segons la programació. Vaig instal·lar CenOs als dos ordinadors que normalment utilitzo. Vaig documentar-me, vaig preguntar i vaig llegir molt i molt. Finalment m'he sortit amb totes les dificultats, el que més puc retreure es que no vaig tenir en compte aquest punt en la planificació ni en els riscos del projecte.

En resum estic satisfet amb el resultat final del projecte. Per a mi ha sigut un gran repte ja que tot el tema d'infraestructura feia més de 15 anys que no m'aproximava.
Volia aconseguir un resultat el més professional possible, no em valia desplegar en un servidor un Docker, sinó que volia auto escalat, alta disponibilitat, fer servir els serveis Cloud, automatitzar tot el procés, etc.

En un principi no s'havia pensat realitzar front-end però finalment es va acabar realitzant un petit front-end que si be es cert, ens ha servit en gran mesura per realitzar tota la resta del treball i penso que finalment ha quedat més complet.

El tema principal del treball és treballar en un desplegament continu i fer-ho de forma teòrica i pràctica, treballar de forma professional el software, amb bones pràctiques, seguretat, monitorització i tests.
Els temes tocats en aquest treball són temes molt actuals i espero que algun altre tregui profit de tot el treballat aquí, jo al menys em porto molts coneixements i ganes d'implementar coses noves professionalment i de continuar avançant en aquest mon que sembla infinit quan et submergeixes.
N'hi ha un munt de possibilitats a l'hora de realitzat tot el plantejament, jo he intentat utilitzar els recursos més coneguts i sobretot OpenSource. També hi ha un munt de possibilitats a l'hora de plantejar la integració i el desplegament continu, en aquest treball només s'explora un dels múltiples camins per a arribar a una finalitat.

Finalment, donar les gracies a totes les persones que han arribat fins aquí, i desitjo que us sigui tan profitós com ha sigut per a mi.

Annex 1. Lliurables del projecte

Lliurable	Descripció	Ubicació / Nom
Memòria	On es descriu amb exhaustivitat tot el treball realitzat.	PAC_FINAL_mem_HidalgoMuñoz_David.pdf
Vídeo demostratiu	Mostra el funcionament del treball realitzat.	Plataforma "Presenta" / PAC_FINAL_vid_HidalgoMuñoz_David.wmv
Presentació del projecte en format lliure	Presentació per a públic en general.	PAC_FINAL_prs_HidalgoMuñoz_David.pptx
Codi font del microservei Java	Conté tot el codi font i l'arxiu README.md amb informació sobre el funcionament.	codi_font/reserves-service-master.zip
Codi font del front-end	Conté tot el codi font i l'arxiu README.md amb informació sobre el funcionament.	codi_font/reserves-frontend-master.zip
Fitxer amb els curls de les crides postman	Es poden importar a un PostMan i fer crides a l'API.	codi_font/Reserves.postman_collection.json
Fitxer .jmx	Amb la configuració de les proves d'estrès passades.	codi_font/reserves.jmx
Fitxers Dockerfile i entrypoint.sh	Per realitzar la imatge de Docker per a desplegar l'aplicació.	codi_font/dockerfiles-master.zip
Fitxer d'ajuda en la construcció de la imatge Docker	És un fitxer amb comandes útils, no necessari, però és molt es de gran ajuda.	codi_font/ajuda_docker.txt
Projecte Lambda backups	Projecte que inclou les dues lambda i una tercera que no fem servir. A més, conté l'arxiu README.md amb tota la informació per a fer-les funcionar.	codi_font/backups-lambda-master.zip
Fitxer manipulació task definition (script .sh)	Necessari per el job de desplegament del servei. És un script sh amb comandes de les aws cli.	codi_font/ecs_taskdef.zip

Taula 4: Lliurables del projecte

Annex 2. Codi font (extractes)

Afegim en aquesta secció el codi font aliè al servei Java o front-end però necessari per al desenvolupament del treball. El codi font de l'aplicació ja s'ha destacat i comentant en els diferents apartats anteriors.

Dockerfile per al desplegament del servei reserves

Dockerfile

```
FROM openjdk:8-jre
RUN apt-get update && apt-get install -y gettext-base python3
RUN curl -O https://bootstrap.pypa.io/get-pip.py && python3 get-pip.py && pip install awscli --upgrade
COPY entrypoint.sh /entrypoint.sh
WORKDIR /
RUN chmod +x /entrypoint.sh
ENTRYPOINT [ "/entrypoint.sh" ]
```

entrypoint.sh

```
#!/bin/bash
if [ -z "$S3_URL" ]; then
    echo "S3_URL is empty"
else
    echo "S3_URL is $S3_URL"
    if [ -n "$AWS_ACCESS_KEY_ID" ]; then
      export AWS_ACCESS_KEY_ID=${AWS_ACCESS_KEY_ID}
      export AWS_SECRET_ACCESS_KEY=${AWS_SECRET_ACCESS_KEY}
    fi
    export AWS_DEFAULT_REGION=${AWS_DEFAULT_REGION}
    aws s3 cp $S3_URL app.jar

    export JAVA_MAX_MEM_RATIO=100

    echo "Using JAVA_OPTS = $JAVA_OPTS"

    java $JAVA_OPTIONS $JAVA_OPTS -jar app.jar
fi
```

Comandes útils per Docker:

```
# Arrancar Docker en local:
docker run -it --rm \
-e S3_URL=s3://tfm-reservas/reservas-0.0.1-SNAPSHOT.jar \
-e AWS_ACCESS_KEY_ID=AKIA6FOYFLFAHKODENSR \
-e AWS_SECRET_ACCESS_KEY=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 \
-e AWS_DEFAULT_REGION=eu-west-1 \
-e JAVA_OPTS="-Dspring.profiles.active=production" \
-p 80:8080 openjdk

#Per fer build:
docker build --tag="973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas" .

#Login
Invoke-Expression -Command (Get-ECRLoginCommand -Region eu-west-1).Command

#Per fer push:
docker push 973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas:latest
```

Lambda

lambda-ebs-backup

```
import boto3
import collections
import datetime
import base64
import os
import json
import itertools

base64_region = os.environ['aws_regions']
```

```
aws_sns_arn = os.getenv('aws_sns_arn', None)

def send_to_sns(subject, message):
    if aws_sns_arn is None:
        return

    print ("Sending notification to: %s" % aws_sns_arn,)

    client = boto3.client('sns')

    response = client.publish(
        TargetArn=aws_sns_arn,
        Message=message,
        Subject=subject)

    if 'MessageId' in response:
        print ("Notification sent with message id: %s" % response['MessageId'])
    else:
        print ("Sending notification failed with response: %s" % str(response))

def lambda_handler(event, context):
    decoded_regions = base64.b64decode(base64_region)
    regions = ['eu-west-1']

    print ("Backing up instances in regions: %s" % regions,)

    for region in regions:
        print ("Entra al for para region: %s" % region,)
        ec = boto3.client('ec2', region_name=region)
        reservations = ec.describe_instances(
            Filters=[
                {'Name': 'tag-key', 'Values': ['backup', 'Backup']},
            ]
        ).get(
            'Reservations', []
        )

        instances = sum(
            [
                [i for i in r['Instances']]
                for r in reservations
            ], [])

        print ("Found %d instances that need backing up in region %s" % (len(instances), region),)

        to_tag_retention = collections.defaultdict(list)
        to_tag_mount_point = collections.defaultdict(list)

        for instance in instances:
            try:
                retention_days = [
                    int(t.get('Value')) for t in instance['Tags']
                    if t['Key'] == 'Retention'][0]
            except IndexError:
                retention_days = 7

            try:
                skip_volumes = [
                    str(t.get('Value')).split(',') for t in instance['Tags']
                    if t['Key'] == 'Skip_Backup_Volumes']
            except Exception:
                pass

            from itertools import chain
            skip_volumes_list = list(chain.from_iterable(skip_volumes))

            for dev in instance['BlockDeviceMappings']:
                if dev.get('Ebs', None) is None:
                    continue
                vol_id = dev['Ebs']['VolumeId']
                if vol_id in skip_volumes_list:
                    print ("Volume %s is set to be skipped, not backing up" % (vol_id),)
                    continue
                dev_attachment = dev['DeviceName']
                print ("Found EBS volume %s on instance %s attached to %s" % (
                    vol_id, instance['InstanceId'], dev_attachment),)

                instance_name = ''
                try:
                    instance_name = [ x['Value'] for x in instance['Tags'] if x['Key'] == 'Name' ][0]
                except IndexError:
                    pass
```

```
                snap = ec.create_snapshot(
                    VolumeId=vol_id,
                    Description='{} {}'.format(instance_name, instance['InstanceId'])
                )

                to_tag_retention[retention_days].append(snap['SnapshotId'])
                to_tag_mount_point[vol_id].append(snap['SnapshotId'])


                print ("Retaining snapshot %s of volume %s from instance %s for %d days" % (
                    snap['SnapshotId'],
                    vol_id,
                    instance['InstanceId'],
                    retention_days,
                ),)

                ec.create_tags(
                    Resources=to_tag_mount_point[vol_id],
                    Tags=[
                        {'Key': 'Name', 'Value': dev_attachment},
                    ]
                )

        for retention_days in to_tag_retention.keys():
            delete_date = datetime.date.today() + datetime.timedelta(days=retention_days)
            delete_fmt = delete_date.strftime('%Y-%m-%d')
            print ("Will delete %d snapshots on %s" % (len(to_tag_retention[retention_days]),
delete_fmt),)
            ec.create_tags(
                Resources=to_tag_retention[retention_days],
                Tags=[
                    {'Key': 'DeleteOn', 'Value': delete_fmt},
                ]
            )
        message = "{} instances have been backed up in region {}".format(len(instances), region)
        send_to_sns('EBS Backups', message)
```

lambda-ebs-backup-cleanup

```
import boto3
import re
import datetime
import base64
import os
import json

base64_region = os.environ['aws_regions']
iam = boto3.client('iam')
aws_sns_arn = os.getenv('aws_sns_arn', None)

def send_to_sns(subject, message):
    if aws_sns_arn is None:
        return

    print ("Sending notification to: %s" % aws_sns_arn,)

    client = boto3.client('sns')

    response = client.publish(
        TargetArn=aws_sns_arn,
        Message=message,
        Subject=subject)

    if 'MessageId' in response:
        print ("Notification sent with message id: %s" % response['MessageId'])
    else:
        print ("Sending notification failed with response: %s" % str(response))

"""
This function looks at *all* snapshots that have a "DeleteOn" tag containing
the current day formatted as YYYY-MM-DD. This function should be run at least
daily.
"""

def lambda_handler(event, context):
    decoded_regions = base64.b64decode(base64_region)
    regions = ['eu-west-1']

    print ("Cleaning up snapshots in regions: %s" % regions,)

    for region in regions:
```

```
        ec = boto3.client('ec2', region_name=region)
        account_ids = list()
        try:
            """
            You can replace this try/except by filling in `account_ids` yourself.
            Get your account ID with:
            > import boto3
            > iam = boto3.client('iam')
            > print iam.get_user()['User']['Arn'].split(':')[4]
            """
            iam.get_user()
        except Exception as e:
            # use the exception message to get the account ID the function executes under
            account_ids.append(re.search(r'(arn:aws:sts::)([0-9]+)', str(e)).groups()[1])


        delete_on = datetime.date.today().strftime('%Y-%m-%d')
        filters = [
            {'Name': 'tag-key', 'Values': ['DeleteOn']},
            {'Name': 'tag-value', 'Values': [delete_on]},
        ]
        snapshot_response = ec.describe_snapshots(OwnerIds=account_ids, Filters=filters)

        print ("Found %d snapshots that need deleting in region %s on %s" % (
            len(snapshot_response['Snapshots']),
            region,
            delete_on),)

        for snap in snapshot_response['Snapshots']:
            print ("Deleting snapshot %s" % snap['SnapshotId'],)
            ec.delete_snapshot(SnapshotId=snap['SnapshotId'])

        message = "{} snapshots have been cleaned up in region
{}".format(len(snapshot_response['Snapshots']), region)
        send_to_sns('EBS Backups', message)
```

Script .sh manipulació Task Definition i JSON Task Definition

ecs_deploy_task.sh

```
#!/bin/bash
#10-11-2019.
#Script to deploy new version of app on ECS Cluster.

REGION="eu-west-1"
S3_URL="s3://tfm-reservas/"
DESIRED_COUNT="0"
NAME="td-reservas"
CLUSTER="reservas-cluster"
SERVICE_NAME="s-reservas"
APPVERSION="reservas.jar"
FAMILY="td-reservas"


#Get task id
TASK_ID=`aws ecs list-tasks --cluster ${CLUSTER} --family ${FAMILY} --region ${REGION}  | jq
.taskArns[] | awk -F/ '{print $2}' | cut -d "\"" -f1 | tail -1`

REVISION=`aws ecs describe-task-definition --task-definition ${NAME} --region ${REGION}  | jq
.taskDefinition.revision`

#Update service to desired-count=0
aws ecs update-service --cluster ${CLUSTER} --region ${REGION} --service ${SERVICE_NAME} --
task-definition ${FAMILY}:${REVISION} --desired-count ${DESIRED_COUNT} 1>/dev/null

#Stop task if TASK_ID is not null
```

```
if [ -n $TASK_ID ]; then
  aws ecs stop-task --cluster ${CLUSTER} --task ${TASK_ID} --region ${REGION}
  #Wait until task is Stopped
  aws ecs wait tasks-stopped --cluster ${CLUSTER} --tasks ${TASK_ID} --region ${REGION}
fi

#Change war o jar to deploy
#APPTOCHANGE=$(cat ${NAME}.json | grep "s3://" | grep deploy | awk -F/ '{print $4}' | cut -d
"\"" -f1)
#sed -e "s;${APPTOCHANGE};${APPVERSION};g" ${NAME}.json > ${NAME}-tmp.json

#Create new task revison
aws ecs register-task-definition --family ${FAMILY} --region ${REGION} --cli-input-json
file:///var/lib/jenkins/${NAME}.json --region ${REGION} 1>/dev/null
REVISION=`aws ecs describe-task-definition --task-definition ${NAME} --region ${REGION}  | jq
.taskDefinition.revision`

#Update service
DESIRED_COUNT="1"
aws ecs update-service --cluster ${CLUSTER} --region ${REGION}  --service ${SERVICE_NAME} --
task-definition ${FAMILY}:${REVISION} --desired-count ${DESIRED_COUNT} --output table

STATUS=`aws ecs describe-services --cluster ${CLUSTER} --service ${SERVICE_NAME} --region
${REGION}  | jq .services[].events[].message | head -1 | grep steady`
echo -n
echo "${STATUS}"

#Wait 1 minute before new task is staring
echo "Waiting to start new task with app "${APPVERSION}""
sleep 60
STATUS=`aws ecs describe-services --cluster ${CLUSTER} --service ${SERVICE_NAME} --region
${REGION}  | jq .services[].events[].message | head -1`
echo "${STATUS}"

#Check last event and steady state
STATUS=`aws ecs describe-services --cluster ${CLUSTER} --service ${SERVICE_NAME} --region
${REGION}  | jq .services[].events[].message | head -1 | grep steady`

while [ -z "${STATUS}" ]
  do
        sleep 15
        STATUS=`aws ecs describe-services --cluster ${CLUSTER} --service ${SERVICE_NAME} --
region ${REGION}  | jq .services[].events[].message | head -1 | grep steady`
    LASTSTATUS=`aws ecs describe-services --cluster ${CLUSTER} --service ${SERVICE_NAME} --
region ${REGION}  | jq .services[].events[].message | head -1`
        if [ -n "${STATUS}" ]; then
                echo "Deploy has finished OK. "${STATUS}""
                exit 0
        #else
                #echo ""${LASTSTATUS}". Checking again in 15 seconds..."
        fi
  done
```

td-reservas.json

```
{
    "containerDefinitions": [
```

```
        {
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/td-reservas",
                    "awslogs-region": "eu-west-1",
                    "awslogs-stream-prefix": "ecs"
                }
            },
            "portMappings": [
                {
                    "hostPort": 80,
                    "protocol": "tcp",
                    "containerPort": 8080
                }
            ],
            "cpu": 1024,
            "environment": [
                {
                    "name": "spring.datasource.username",
                    "value": "adm_reservas"
                },
                {
                    "name": "AWS_ACCESS_KEY_ID",
                    "value": "AKIA6FOYFLFAHKODENSR"
                },
                {
                    "name": "spring.datasource.url",
                    "value": "jdbc:mysql://db-comun.cqkijhpkvrhi.eu-west-
1.rds.amazonaws.com:3306/reservas?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacy
DatetimeCode=false&serverTimezone=UTC"
                },
                {
                    "name": "AWS_SECRET_ACCESS_KEY",
                    "value": "0zqc9PUYXjUYSidlUukUYBybhqPtXZsOKTiTrxrk"
                },
                {
                    "name": "AWS_DEFAULT_REGION",
                    "value": "eu-west-1"
                },
                {
                    "name": "S3_URL",
                    "value": "s3://tfm-reservas/reservas-0.0.1-SNAPSHOT.jar"
                },
                {
                    "name": "spring.datasource.password",
                    "value": "TFMDavid2019"
                }
            ],
            "mountPoints": [],
            "memoryReservation": 512,
            "volumesFrom": [],
            "image": "973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas:latest",
            "essential": true,
            "name": "reservas-container"
        }
```

```
    ],
    "family": "td-reservas",
    "requiresCompatibilities": [
        "EC2"
    ],
    "volumes": [],
    "placementConstraints": []
}
```

Annex 3. Llibreries/Codi extern utilitzat

AWS cli

La interfase de línia de comandes d'AWS és una eina unificada per administrar els productes d'AWS. Utilitzada en aquest treball per pujar les imatges dels Dockers a AWS.

Exemple de comandes, comandes utilitzades per a realitzar login:

```
#Login AWS cli
Invoke-Expression -Command (Get-ECRLoginCommand -Region eu-west-1).Command
```

Comanda utilitzada per fer push d'un Docker:

```
#Per a realitzar push:
docker push 973800298816.dkr.ecr.eu-west-1.amazonaws.com/reservas:latest
```

Comanda utilitzada per manipular arxius en un bucket S3:

```
aws s3 sync dist/reservas-frontend/ s3://reserves.tfmaster.net --delete
```

Boto / Boto3

Boto és una llibreria de codi obert per a Python (i també PHP encara que en aquest treball utilitzem la llibreria per a Python) que proporciona una interfase als serveis d'AWS.
Podem consultar tots els serveis suportats i les serves referencies en aquesta web:
http://boto.cloudhackers.com/en/latest/

Backup i BackupCleanup

Per a realitzar els serveis de backup i cleanup realitzat per les lambdes, m'he basat en un projecte i l'he reescrit per tal d'acomodar-lo al meu treball. El codi font s'ha inclòs en l'*Annex 2. Codi font* sota l'apartat *Lambda* i en l'apartat *19. Instruccions d'instal·lació/implantació*, s'explica la manera de com es pot instal·lar i implementar.

Annex 4. Resultats scan Nessus

Resultats de l'scanner de Nessus contra els dominis reserves.tfmaster.net i reserves-service.tfmaster.net

```
Plugin ID,CVE,CVSS,Risk,Host,Protocol,Port,Name,Synopsis,Description,Solution,See Also,Plugin
Output
10180,"","","None","reserves-service.tfmaster.net","tcp","0","Ping the remote host","It was
possible to identify the status of the remote host (alive or
dead).","Nessus was able to determine if the remote host is alive using one or
more of the following ping types :

  - An ARP ping, provided the host is on the local subnet
    and Nessus is running over Ethernet.

  - An ICMP ping.

  - A TCP ping, in which the plugin sends to the remote host
    a packet with the flag SYN, and the host will reply with
    a RST or a SYN/ACK.

  - A UDP ping (e.g., DNS, RPC, and NTP).","n/a","","The remote host is up
The remote host replied to a TCP SYN packet sent to port 80 with a SYN,ACK packet"
19506,"","","None","reserves-service.tfmaster.net","tcp","0","Nessus Scan Information","This
plugin displays information about the Nessus scan.","This plugin displays, for each tested
host, information about the
scan itself :

  - The version of the plugin set.
  - The type of scanner (Nessus or Nessus Home).
  - The version of the Nessus Engine.
  - The port scanner(s) used.
  - The port range scanned.
  - Whether credentialed or third-party patch management
    checks are possible.
  - The date of the scan.
  - The duration of the scan.
  - The number of hosts scanned in parallel.
  - The number of checks done in parallel.","n/a","","Information about this scan :

Nessus version : 8.7.2
Plugin feed version : 201910260240
Scanner edition used : Nessus Home
Scan type : Normal
Scan policy used : Host Discovery
Scanner IP : 192.168.0.164

WARNING : No port scanner was enabled during the scan. This may
lead to incomplete results.

Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report verbosity : 1
Safe checks : yes
Optimize the test : yes
Credentialed checks : no
Patch management checks : None
CGI scanning : disabled
Web application tests : disabled
Max hosts : 256
Max checks : 5
Recv timeout : 5
Backports : None
Allow post-scan editing: Yes
Scan Start Date : 2019/10/27 14:12
Scan duration : 28 sec
"
10180"","""","""","""None"","""reserves.tfmaster.net"","""tcp"","""0"","""Ping the remote
host"","""It was possible to identify the status of the remote host (alive or"
dead).","Nessus was able to determine if the remote host is alive using one or
more of the following ping types :

  - An ARP ping, provided the host is on the local subnet
    and Nessus is running over Ethernet.

  - An ICMP ping.

  - A TCP ping, in which the plugin sends to the remote host
```

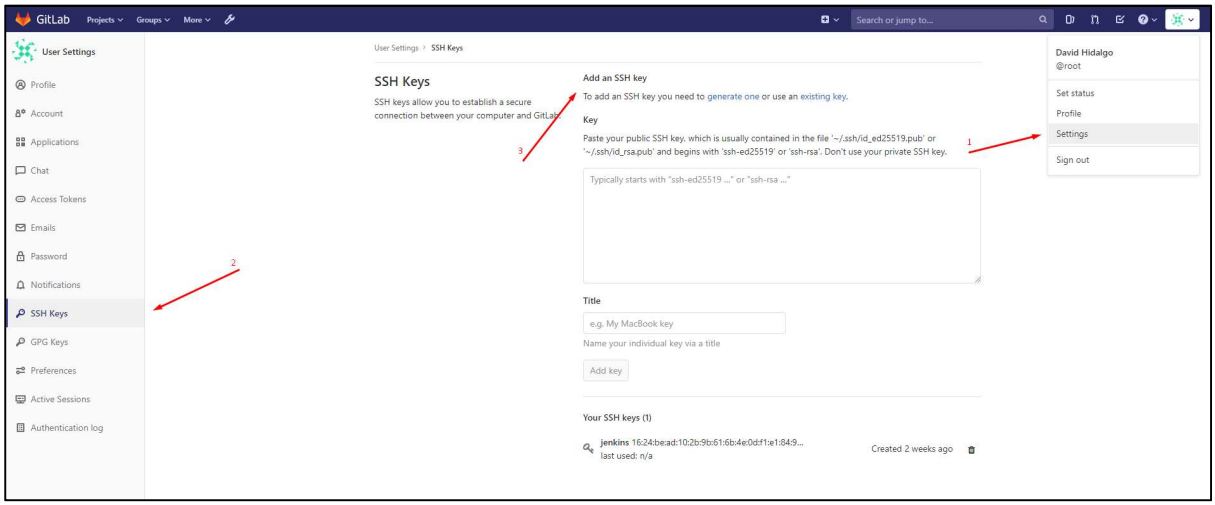
```
    a packet with the flag SYN, and the host will reply with
    a RST or a SYN/ACK.

  - A UDP ping (e.g., DNS, RPC, and NTP).","n/a","","The remote host is up
The remote host replied to an ICMP echo packet"
19506,"","","None","reserves.tfmaster.net","tcp","0","Nessus Scan Information","This plugin
displays information about the Nessus scan.","This plugin displays, for each tested host,
information about the
scan itself :

  - The version of the plugin set.
  - The type of scanner (Nessus or Nessus Home).
  - The version of the Nessus Engine.
  - The port scanner(s) used.
  - The port range scanned.
  - Whether credentialed or third-party patch management
    checks are possible.
  - The date of the scan.
  - The duration of the scan.
  - The number of hosts scanned in parallel.
  - The number of checks done in parallel.","n/a","","Information about this scan :

Nessus version : 8.7.2
Plugin feed version : 201910260240
Scanner edition used : Nessus Home
Scan type : Normal
Scan policy used : Host Discovery
Scanner IP : 192.168.0.164

WARNING : No port scanner was enabled during the scan. This may
lead to incomplete results.

Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report verbosity : 1
Safe checks : yes
Optimize the test : yes
Credentialed checks : no
Patch management checks : None
CGI scanning : disabled
Web application tests : disabled
Max hosts : 256
Max checks : 5
Recv timeout : 5
Backports : None
Allow post-scan editing: Yes
Scan Start Date : 2019/10/27 14:12
Scan duration : 27 sec
```

Annex 5. Guia d'usuari

Guia d'ús del Desplegament Continu

Versionat de codi font

Es pot accedir al servidor de GitLab mitjançant aquesta direcció web: https://gitlab.tfmaster.net.
Podeu accedir mitjançant l'usuari i password que disposeu. Després del login podeu accedir als projectes que teniu assignats i veure els canvis realitzats en aquests.



Es necessari accedir a les *Settings* (1) del vostre usuari i anar a SSH Keys (2). Podem seguir els passos descrits en *Add an SSH key*.
D'aquesta manera podeu configurar els vostres IDEs per a realitzar pull i push amb aquesta clau.
No funciona els push/pull amb port web, es necessari realitzar aquest pas.

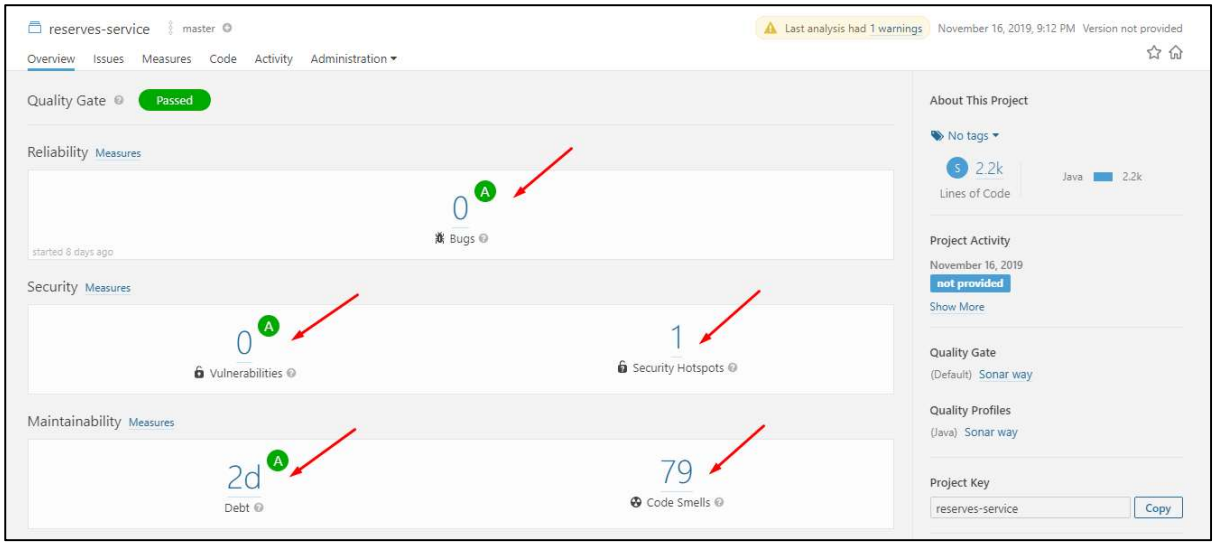
Els usuaris amb rol de Maintenance són els únics que poden fer push a la branca master del seu projecte, els rol Developer poden crear noves branques i fer push de les mateixes.

El codi font que es desplegarà sempre serà de la branca master.

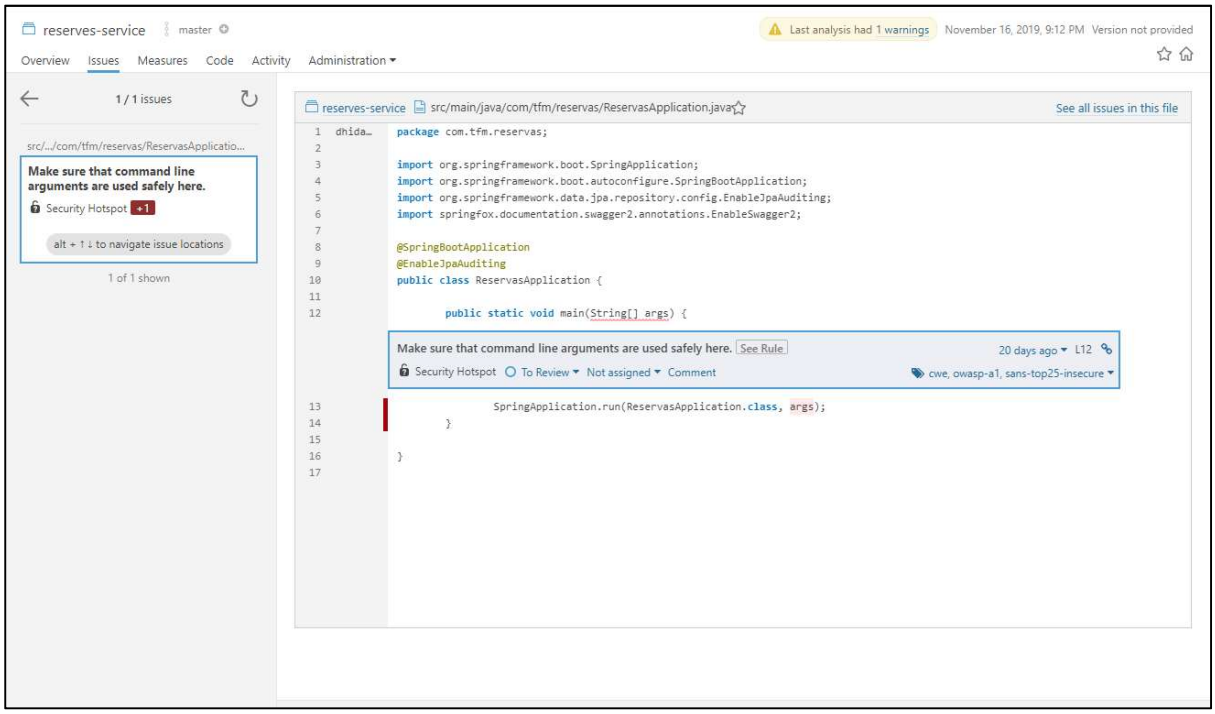
Auditoria de codi font

En el moment de realitzar un desplegament, el Job connecta amb el servidor de SonarQube i es realitza una auditoria de codi font abans de desplegar. Si l'auditoria no té èxit, segons els paràmetres configurats, el desplegament no es realitzarà.
Per accedir al servidor de SonarQube es pot realitzar mitjançant aquesta url: https://sonar.tfmaster.net.

Podeu accedir als vostres projectes en SonarQube mitjançant l'usuari i password que disposeu.
En cadascun dels vostres projectes es mostren uns paràmetres que estan en vermell faran que el projecte no es desplegui.



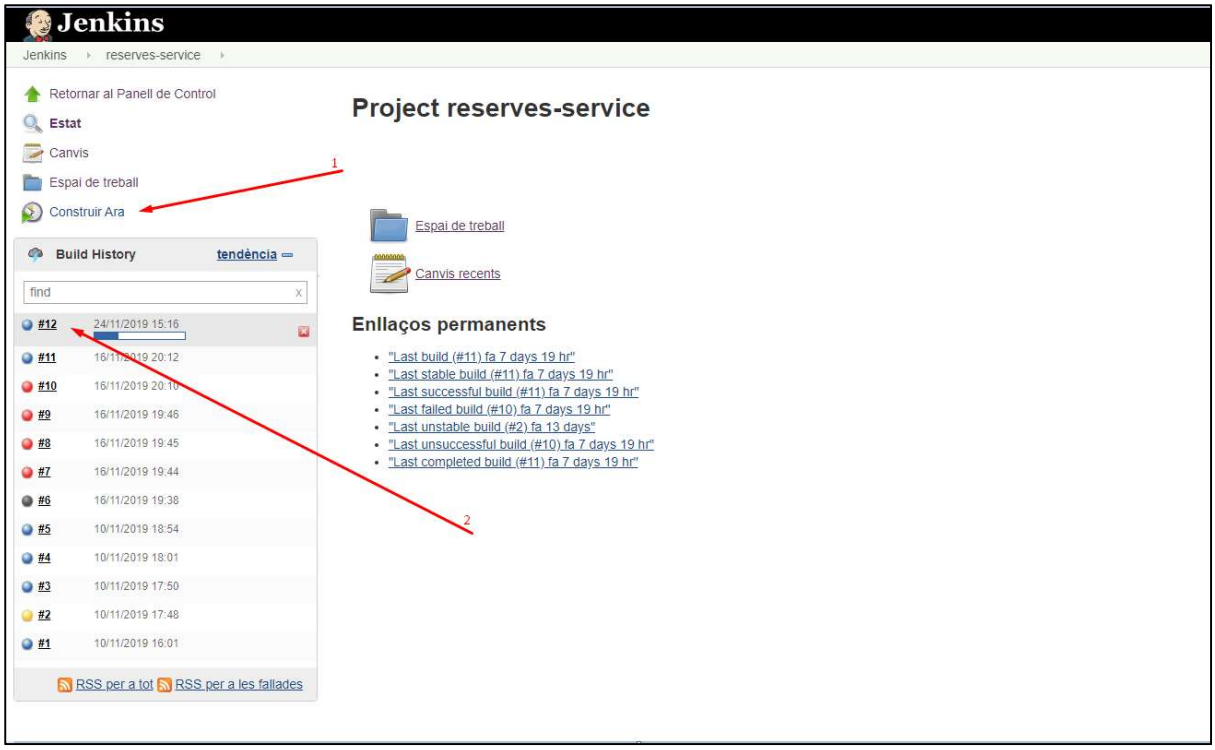
Es pot explorar les mètriques i SonarQube ofereix informació més detallada i informació sobre l'issue.



Desplegament automàtic

Per a realitzar la integració continua i el desplegament, s'ha instal·lat un servidor Jenkins.
A partir d'ara, podeu accedir a aquesta servei mitjançant l'URL https://jenkins.tfmaster.net.

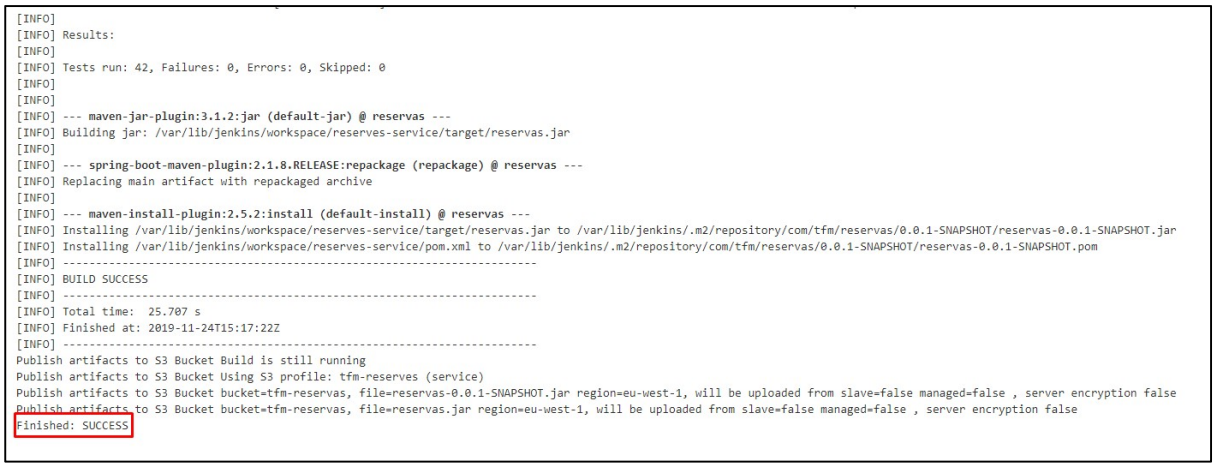
Utilitzeu el vostre usuari i password per a realitzar login en el servei.
Veureu un llistat de jobs disponibles segons projectes assignats. Accedint al job seleccionat, podeu realitzar el desplegament del codi font de la branca master seleccionant l'opció *Construir Ara* (1).



Podeu consultar que es va executant i veure si hi ha algun error en el deploy, per això, podeu anar al job executant-se (2) i accedireu a la pantalla del job:

En aquesta pantalla polsar Console Output per accedir a tota la informació de l'execució del job.

És important que al finalitzar el job sigui un SUCCESS:



Annex 6. Baixant els serveis per estalviar costos

Estem aprofitant la free tier que ofereix AWS. Això te limitacions.

Servei	Límit d'ús de capa gratuïta
Amazon EC2 Container Registry (ECR)	500 MB-month of Amazon EC2 Container Registry storage for new customers
Amazon Simple Storage Service	20,000 Get Requests of Amazon S3
Amazon Simple Storage Service	2,000 Put, Copy, Post or List Requests of Amazon S3
Amazon Relational Database Service	750 hours of Amazon RDS Single-AZ db.t2.micro Instances
AWS Key Management Service	20,000 free requests per month for AWS Key Management Service

Taula 5: Límit d'ús de capa gratuïta

Es realitzen diferents accions després de cada sessió d'ús per no sobrepassar, si es possible, els límits de la capa gratuïta.

1. Servei

Es posen a 0 *number of task* en el *Service*.

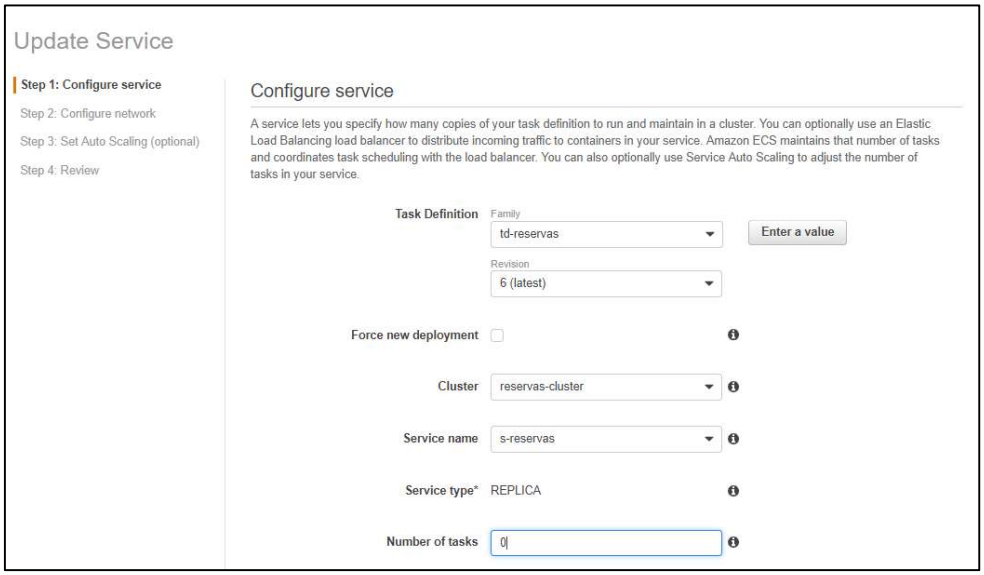


Figura 170: Baixant els serveis, number of tasks en Service

2. RDS

En l'instancia de la nostra base de dades, la marquen i fem *Stop*.

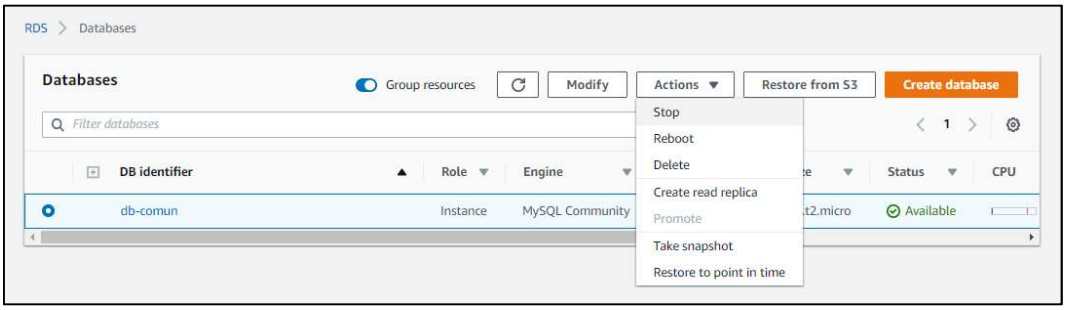


Figura 171: Baixant els serveis, aturant la BBDD

Annex 7. Connectar amb una instancia d'AWS

Per connectar-nos per SSH a la instancia que acabem de crear, ens hem de connectar mitjançant ssh i les .pem que hem creat i com a direcció l'arn que ens proporciona AWS, per exemple:

```
ssh -i servidors_ci.pem ec2-user@ec2-34-244-47-67.eu-west-1.compute.amazonaws.com
```

Hem d'assegurar-nos que en el security group assignat (sg-reserves en el nostre cas), te activat el port 22 (per accedir per ssh) i el port 8080 per accedir per HTTP. Una recomanació es filtrar per IP l'accés al port 22.

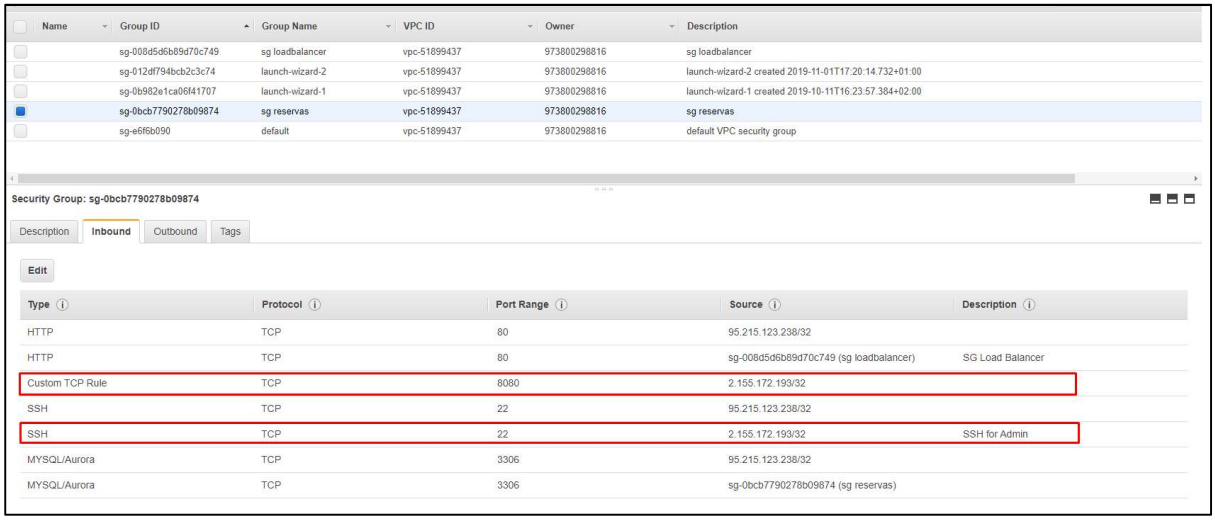


Figura 172: Configurant instancia per als servidors CI, part 8

Si s'usa Windows per a la connexió SSH, haurem d'executar les següents sentencies per donar accés R a les clau servidors_ci.pem.

```
# Reset to remove explict permissions
icacls.exe servidors_ci.pem /reset
# Find out your user name
whoami
# Give current user explicit read-permission
icacls.exe servidors_ci.pem /GRANT:R ntnet\dhidalgo:R
# Disable inheritance and remove inherited permissions
icacls.exe servidors_ci.pem /inheritance:r
```

En Linux s'ha d'executar:

```
chmod 400 servidors_ci.pem
```

Finalment podem connectar:

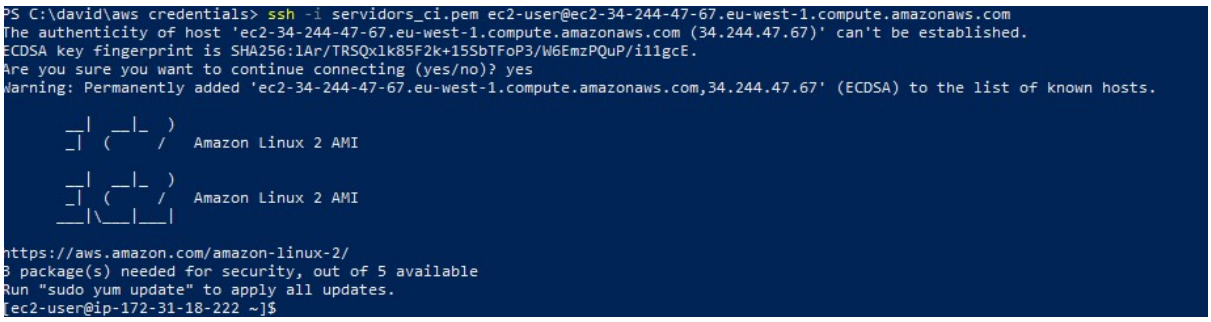


Figura 173: Configurant instancia per als servidors CI, part 9

Annex 8. Pujant codi font al nostre GitLab

Creem grup i projectes a GitLab:

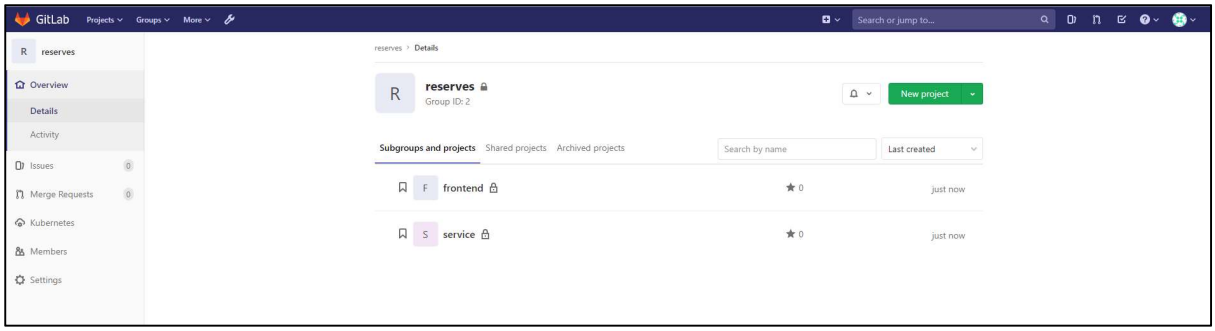


Figura 174: Projectes en GitLab

Creem les claus SSH amb Git Bash i les pugem a GitLab. Per a generar les claus la mateixa pàgina de GitLab ofereix un link que explica com realitzar-ho:

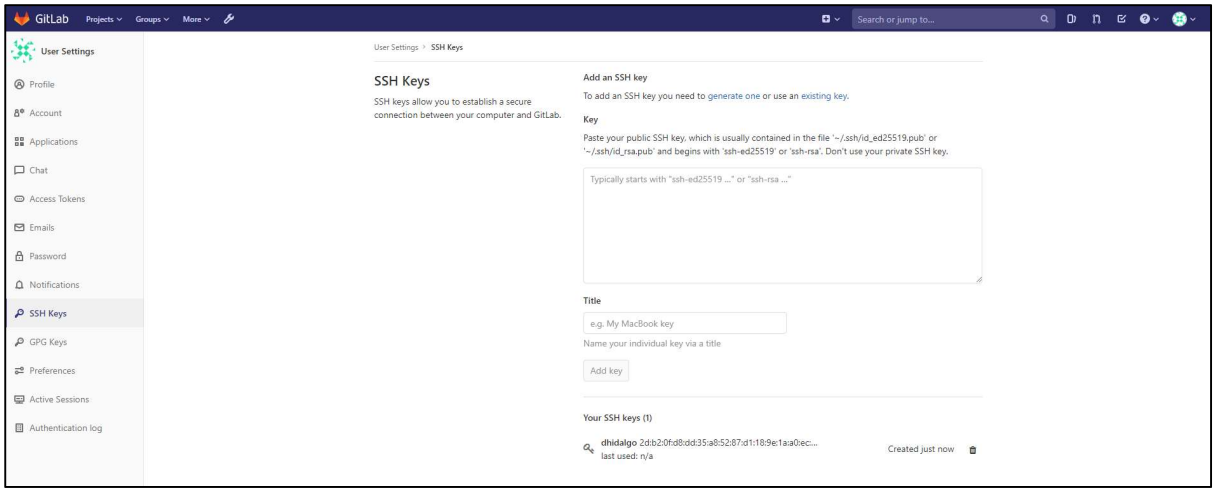


Figura 175: Claus SSH per a GitLab

El *Load Balancer* tal com està configurat, tot el que rep del host gitlab.tfmaster.net, el redirigeix pel port 80 a la instancia.

Per tant, si volem connectar pel port 22 (ssh), s'ha de crear un nou conjunt de registres en Route 53 per redirigir el tràfic:

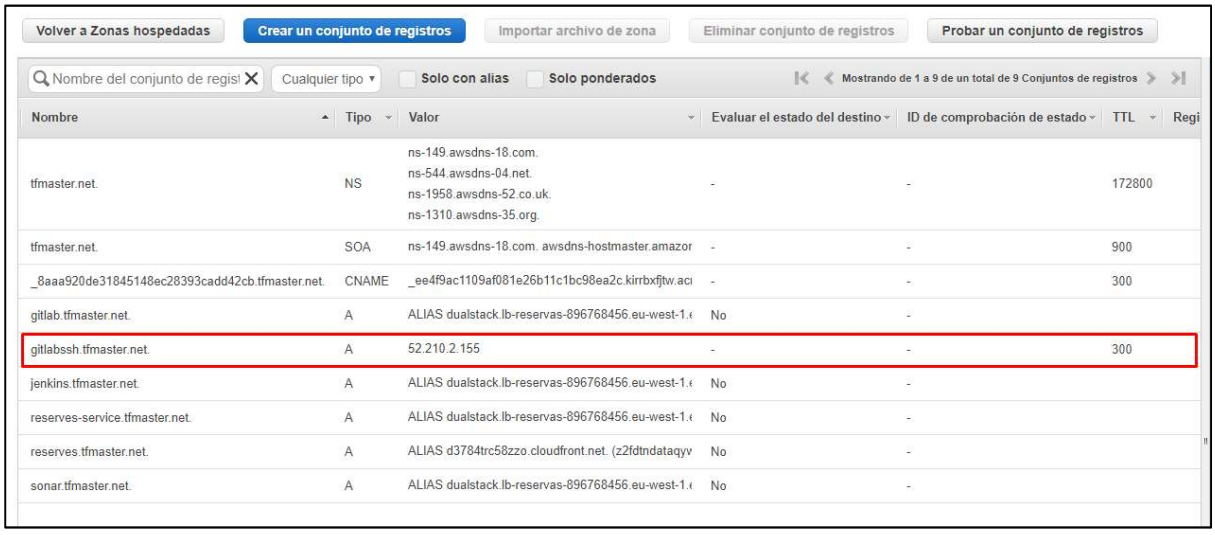


Figura 176: Nou conjunt de registres gitlabssh.tfmaster.net

S'ha de canviar el ssh_host en l'arxiu pel nou subdomini creat (gitlabssh.tfmaster.net):

```
sudo vi /opt/gitlab/embedded/service/gitlab-rails/config/gitlab.yml
```

Per a pujar el codi font, en la mateixa pàgina del projecte ens dona les comandes per pujar un "*existing folder*"

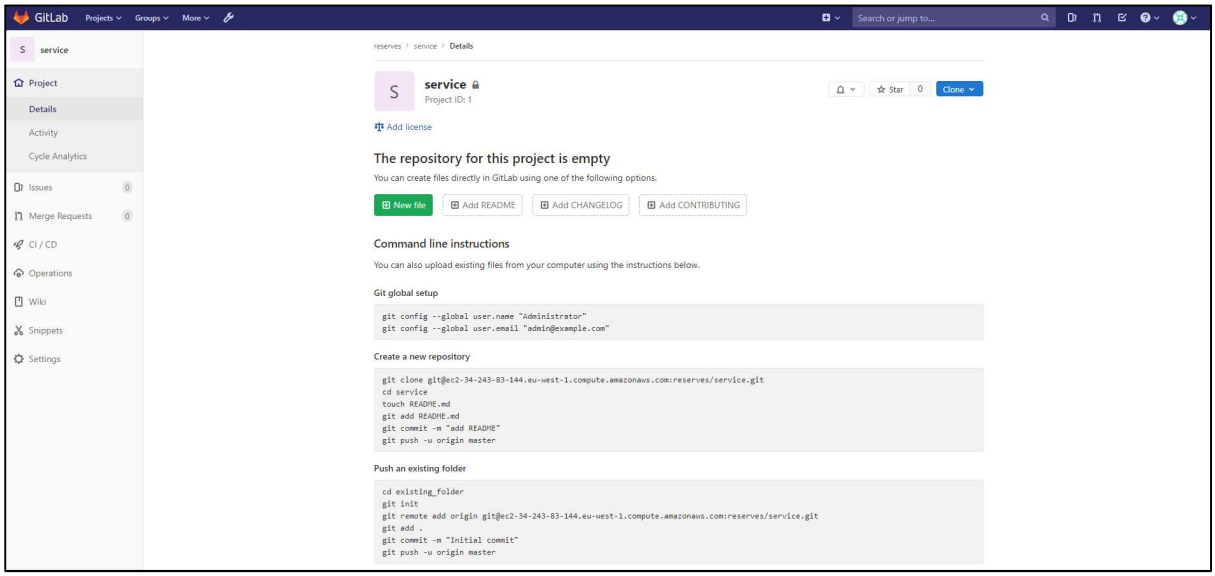


Figura 177: Pujant codi font

També s'aprofita i es crea un grup i un projecte per versionar els Dockerfiles:

Figura 178: Projectes a GitLab

Annex 9. Llibre d'estil

Per als estils gràfics, s'ha optat per Bootstrap. Bootstrap és un framework de JavaScript, HTML5 i CSS3 que permet la creació d'interfases web molt vistoses i amb disseny responsive.
Bootstrap ofereix una gama de possibilitats bastant amplia però destaca perquè els seus dissenys són simples, nets i intuïtius.
Bootstrap funciona com una grid de 12 columnes que es poden adaptar segons necessitats i en funció de quatre mides de dispositius.

Bootstrap proporciona un conjunt de fulles d'estil que proveeixen definicions bàsiques d'estil per a tots els components HTML. Això proporciona uniformitat a la navegació i a tota la web.
Els elements més importants utilitzats són els següents:

Titulars

Per als titulars s'utilitza l'etiqueta <*h2*>.

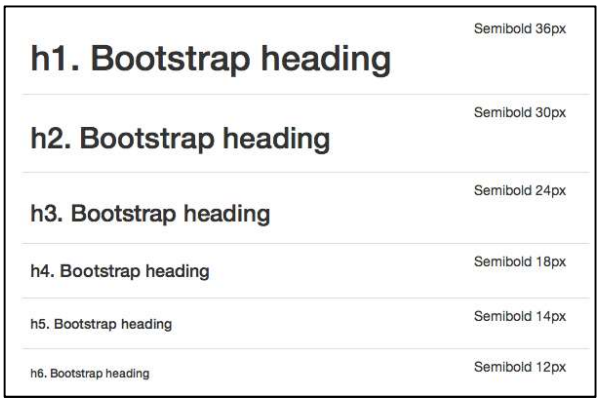


Figura 179: Llibre d'estils, titulars

Text

La mida de lletra es 14px i el interlineat és 1.428

Taules

S'han utilitzat taules cebreades amb la classe *table-striped*.

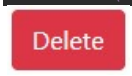
```
<table class="table table-striped">
```

Botons

S'utilitzen 3 estils diferents segons l'acció del botó.

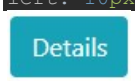
- Eliminar:

```
<button (click)="deleteEquipment(equipment.id)" class="btn btn-danger">Delete</button>
```



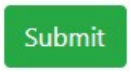
- Llistar, modificar, tornar:

```
<button (click)="equipmentDetails(equipment.id)" class="btn btn-info" style="margin-
left: 10px">Details</button>
```



- Crear, Submit:

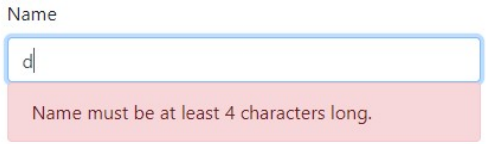
```
<button type="submit" class="btn btn-success">Submit</button>
```



Missatges d'alerta

Per als missatges d'alerta en la validació dels formularis, s'utilitza la classe *alert-danger*.

```
<div *ngIf="name.invalid && (name.dirty || name.touched)" class="alert alert-danger">
```



Annex 10. One-page business pla/Resum executiu

Bussiness Plan per a UOC en: 6 de gener de 2020

Telèfon:933 546 509 **E-mail**: davidhidalgomunoz@gmail.com

Propostes de valor

Automatitzar les tasques que no requereixin una presa de decisió.

Evitar processos redundants o sense valor afegit.

Millora en time to market i minimitzar els errors humans.

Objectivització de la qualitat del codi.

Automatització del testing i versionat.

Oportunitat

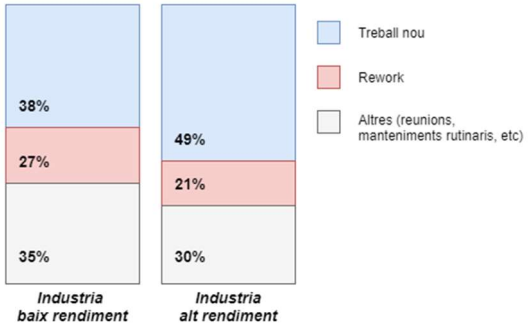
Estalvi de costos en personal d'operacions, minimització d'errors humans.

Pagament per us, evitant el sobre cost per cobrir l'alta activitat puntual.

Poca o nula inversió en Hardware.

Automatització dels desplegaments en tots els entorns.

Anàlisis de l'industria



Nosaltres

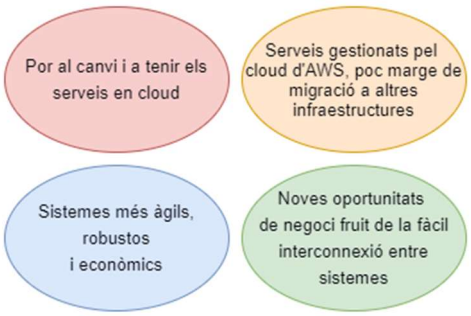
Partners d'Amazon Web Service.

Titulats en metodologies Agile.

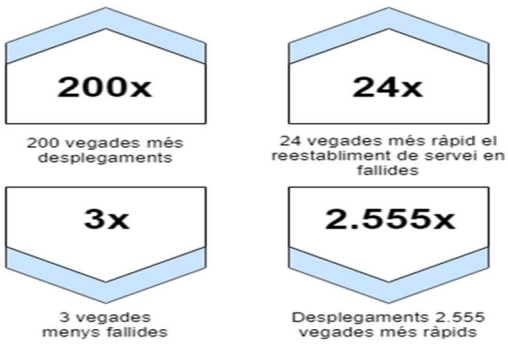
Experts en DevOps.

Coneixem els valors del negoci.

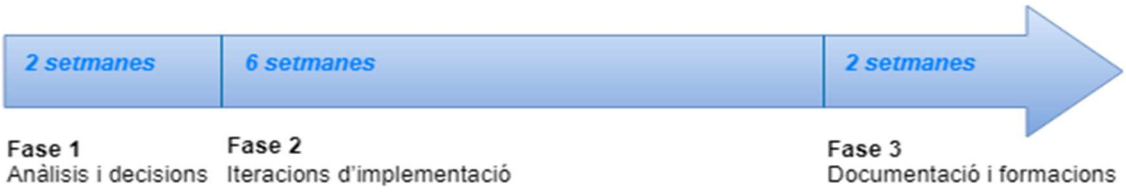
DAFO



Beneficis



Time-line



Resum financer

ROI:

3 operaris - 2 anys / 5 operaris - 1 any

Cost inicial: 60.000€

*millora d'un 16% efectivitat. Cost operari 70.000€ any

Expertise

Més de 15 anys en projectes tecnològics en tot tipus d'entorns.

Implementació i execució de més de 20 projectes Cloud en els darrers 3 anys.

Annex 11. Glossari

Agile: Agile és molt més que una metodologia per al desenvolupament de projectes que necessiten de rapidesa i flexibilitat, és una filosofia que suposa una forma diferent de treballar i d'organitzar-se. De tal manera que cada projecte es 'trosseja' en petites parts que han de completar-se i lliurar-se en poques setmanes. L'objectiu és desenvolupar productes i serveis de qualitat que responguin a les necessitats d'uns clients les prioritats canvien a una velocitat cada vegada més gran.
Angular: És un framework de javascript de codi obert, mantingut per google, que s'utilitza per crear i mantenir aplicacions web d'una sola pàgina.
API: Sigles d'Application Programming Interface. És un conjunt de regles (codi) i especificacions que les aplicacions poden seguir per comunicar-se entre elles: servint d'interfície entre programes diferents.
Auditoria de codi: És el procés d'analitzar el codi font d'una aplicació amb l'objectiu de descobrir vulnerabilitats de seguretat, problemes de disseny de seguretat, i potencials millores en les pràctiques de programació.
AWS: Sigles de 'Amazon Web Services'. És una plataforma de serveis de núvol que t'ofereix potència de còmput, emmagatzematge de bases de dades, lliurament de contingut i una altres funcionalitats; a més de ser molt més segura que un servidor físic.
AWS CLI: És una eina de codi obert que permet interactuar amb els serveis d'AWS mitjançant l'ús de comandaments en el shell.
Basic Auth: És un mètode dissenyat per permetre un navegador web, o altre programa client, proveir credencials en la forma d'usuari i contrasenya quan se us demana una pàgina a servidor.
Bucket: Els buckets són contenidors d'objectes. Per a cada bucket, podrà controlar l'accés (qui pot crear, eliminar i enumerar objectes del bucket), consultar els logs d'accés a l'bucket i els seus objectes i triar la regió geogràfica en què Amazon S3 emmagatzemarà el bucket i el seu contingut.
CI/CD: La CI/CD és un mètode per a distribuir aplicacions als clients amb freqüència mitjançant l'ús de l'automatització en les etapes de desenvolupament d'aplicacions. Els principals conceptes que s'atribueixen a la CI/CD són la integració contínua, la distribució contínua i el desplegament continu.
CloudWatch: CloudWatch ofereix dades i informació processable per monitoritzar les aplicacions, respondre a canvis de rendiment que afecten tot el sistema, optimitzar l'ús de recursos i aconseguir una vista unificada de l'estat de les operacions. CloudWatch recopila dades de monitorització i operacions en format de registres, mètriques i esdeveniments, la qual cosa ofereix una vista unificada dels recursos, les aplicacions i els serveis d'AWS que s'executen en servidors locals i d'AWS.
Contenidor: La virtualització a nivell de sistema operatiu és un mètode de virtualització de servidor en el qual el nucli d'un sistema operatiu permet que hi hagi múltiples instàncies aïllades d'espais d'usuari, en lloc de només un. Tals instàncies son anomenades contenidors.
Desplegament continu: Veure *CI/CD*.
DevOps: Devops és un acrònim anglès de development (desenvolupament) i operations (operacions), que es refereix a una metodologia de desenvolupament de programari que se centra en la comunicació, col·laboració i integració entre desenvolupadors de programari i els professionals de sistemes en les tecnologies de la informació (IT).
Docker: Docker és un projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors de programari, proporcionant una capa addicional d'abstracció i automatització de virtualització d'aplicacions en múltiples sistemes operatius.

EC2: Amazon Elastic Compute Cloud (Amazon EC2) és un servei web que proporciona capacitat informàtica en el núvol segura i de mida modificable. Està dissenyat per facilitar als desenvolupadors l'ús de la informàtica en el núvol a escala de la web.
ECS: Amazon Elastic Container Service (Amazon ECS) és un servei d'administració de contenidors altament escalable i ràpid que facilita la tasca d'executar, aturar i administrar contenidors de Docker en un clúster.
EKS: Amazon Elastic Kubernetes Service (Amazon EKS) és un servei administrat que li permet executar fàcilment Kubernetes en AWS sense necessitat de crear ni mantenir el seu propi pla de control de Kubernetes. Kubernetes és un sistema de codi obert per automatitzar la implementació, escalat i administració de les aplicacions en contenidors.
Git: És un programari de control de versions, pensant en l'eficiència i la fiabilitat del manteniment de versions d'aplicacions quan aquestes tenen un gran nombre d'arxius de codi font. El seu propòsit és portar registre dels canvis en arxius d'ordinador i coordinar el treball que diverses persones realitzen sobre arxius compartits.
GitLab: Gitlab és un servei web de control de versions i desenvolupament de programari col·laboratiu basat en Git. A més de gestor de repositoris, el servei ofereix també allotjament de wikis i un sistema de seguiment d'errors, tot això publicat sota una Llicència de codi obert.
Integració continua: Veure *CI/CD*.
Java: Java és un llenguatge de programació orientat a objectes que es va incorporar a l'àmbit de la informàtica en els anys noranta. L'idea de Java és que pugui realitzar-programes amb la possibilitat d'executar en qualsevol context, en qualsevol ambient, sent així la seva portabilitat un dels seus principals èxits.
Javascript: JavaScript (abreujat comunament JS) és un llenguatge de programació interpretat, dialecte de l'estàndard ECMAScript. Es defineix com orientat a objectes, basat en prototips, imperatiu, dèbilment tipat i dinàmic. S'utilitza principalment en la seva forma de la banda del client (client-side), implementat com a part d'un navegador web permetent millores en la interfície d'usuari i pàgines web dinàmiques
Jenkins: Jenkins és un servidor d'automatització de codi obert escrit en Java. Jenkins ajuda en l'automatització de part de el procés de desenvolupament de programari mitjançant integració contínua i facilita certs aspectes del lliurament contínua. Admet eines de control de versions i pot executar projectes basats en Apache Ant i Apache Maven, així com seqüències d'ordres de consola i programes per lots de Windows.
Lambda: AWS Lambda és un servei informàtic que permet executar codi sense aprovisionar ni administrar servidors. AWS Lambda executa el codi només quan cal, i es escala de manera automàtica, passant de poques sol·licituds a el dia a milers per segon.
Load Balance: El balanceig de càrrega (Load Balance) es refereix a la distribució del tràfic de xarxa entrant a través d'un grup de servidors backend, també conegut com Server Farm (conjunt de servidor) o Server Pool (conjunt de servidors).
Microservei: Són unitats funcionals concretes i independents, que treballen juntes per oferir la funcionalitat general d'una aplicació. Cada microservei pot ser actualitzat o escalat sense que això afecti la disponibilitat dels altres unitats i de l'aplicació en el seu conjunt. Els microserveis són els principals components d'una Arquitectura de Microserveis.
MySQL: MySQL és un sistema de gestió de base de dades relacional (RDBMS) de codi obert, basat en llenguatge de consulta estructurat (SQL).

Python: Python és un llenguatge de programació interpretat la filosofia posa l'accent en la llegibilitat del seu codi. Es tracta d'un llenguatge de programació multiparadigma, ja que suporta orientació a objectes, programació imperativa i, en menor mesura, programació funcional. És un llenguatge interpretat, dinàmic i multiplataforma.

RDS: Amazon Relational Database Service (Amazon RDS) és un servei administrat que facilita les tasques de configuració, operació i escalat d'una base de dades relacional en el núvol. Proporciona capacitat rendible i de mida modificable i, a el mateix temps, administra les àrdues tasques d'administració de les bases de dades.

S3: Amazon S3 és un servei d'emmagatzematge d'objectes creat per emmagatzemar i recuperar qualsevol volum de dades des de qualsevol ubicació d'Internet. És un servei d'emmagatzematge simple que ofereix una infraestructura per emmagatzemar dades amb un nivell extremadament alt de durabilitat, disponibilitat i escalabilitat a un cost molt baix.

Snapshot: Volum lògic que permet realitzar una còpia de seguretat coherent d'un altre volum lògic del mateix grup de volums. La creació de l'snapshot consisteix a prendre una «fotografia», una instantània d'un volum lògic concret i, a partir d'ella, guardar les modificacions realitzades en aquest volum.

SNS: Amazon Simple Notification Service (SNS) és un servei de missatgeria de publicació / subscripció completament administrada, d'alta disponibilitat, segur i amb durabilitat que permet desacoblar microserveis, sistemes distribuïts i aplicacions sense servidor.

SonarQube: És una plataforma desenvolupada en Java que ens permet realitzar anàlisis de codi amb diferents eines de forma automatitzada.

SQL: És un llenguatge de domini específic utilitzat en programació, dissenyat per a administrar, i recuperar informació de sistemes de gestió de bases de dades relacionals. Una de les seves principals característiques és el maneig de l'àlgebra i el càlcul relacional per efectuar consultes amb la finalitat de recuperar , de forma senzilla, informació de bases de dades, així com realitzar canvis en elles.

Task Definition: Descriu les definicions de contenidor i volum d'una tasca d'Amazon Elastic Container Service (Amazon ECS). Podeu especificar quines imatges de Docker utilitzar, els recursos necessaris i altres configuracions relacionades amb el llançament de la definició de tasca a través d'un servei o tasca d'Amazon ECS.

Annex 12. Bibliografia

12.1. Apartats introductoris

- Jeff Humble i David Farley (First Printing August 2010) Continuous Delivery - Reliable Software Releases Through Build, Test And Deployment Automation
- Gene Kim, Jez Humble, Patrick Debois i John Willis () The DevOps Handbook.
- https://es.wikipedia.org/wiki/Desarrollo_en_cascada
- https://www.slideshare.net/jallspaw/10-deploys-per-day-dev-and-ops-cooperation-at-flickr
- https://es.wikipedia.org/wiki/DevOps
- The rise of DevOps, Dmitriy Samovskiy, http://www.somic.org/2010/03/02/the-rise-of-devops/
- DevOps Culture, John Willis, 2012, http://itrevolution.com/devops-culture-part-1/
- DevOps mixing dev, ops, agile, cloud, open source and business, Jay Lyman, 2010, https://web.archive.org/web/20150914010853/https://blogs.the451group.com/opensource/2010/03/03/devops-mixing-dev-ops-agile-cloud-open-source-and-business/
- 10 Deep DevOps Thoughts From Chef's Jez Humble, https://blog.newrelic.com/technology/devops-jez-humble/
- 2013 State of DevOps Report. Puppet labs & IT revolution press.
- 2016 State of DevOps Report. Puppet labs + DORA.

12.2. Desenvolupament del software

- https://spring.io/ (Spring boot)
- https://www.callicoder.com/spring-boot-actuator/ (Actuator/Health)
- https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/ (Java 8)
- https://medium.com/the-resonant-web/spring-boot-2-0-project-structure-and-best-practices-part-2-7137bdcba7d3 (Spring boot)
- https://vladmihalcea.com/whats-new-in-jpa-2-2-java-8-date-and-time-types/ (Hibernate/jpa)
- https://thoughts-on-java.org/best-practices-many-one-one-many-associations-mappings/ (Hibernate/jpa)
- https://thepracticaldeveloper.com/2017/07/31/guide-spring-boot-controller-tests/#Unit_and_Integration_Tests_for_RestControllers_in_Spring_Boot (Test)
- https://www.carlosble.com/downloads/disenoAgilConTdd_ebook.pdf (TDD)
- https://github.com/javamelody/javamelody/wiki (JavaMelody)
- https://apiumhub.com/es/tech-blog-barcelona/beneficios-de-tdd/ (TDD/Tests unitaris)
- https://www.redhat.com/es/topics/microservices (microserveis)
- https://www.oscarblancarteblog.com/tutoriales/java-persistence-api-jpa/ (JPA)
- https://www.javaguides.net/2019/06/spring-boot-angular-8-crud-part-2-create-angular-8-app.html (Angular 8)
- https://es.wikipedia.org/wiki/Bootstrap_(framework) (css)
- https://getbootstrap.com/ (css)

12.3. Arquitectura AWS

- https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
- https://docs.gitlab.com/omnibus/docker/
- https://www.sololinux.es/instalar-y-configurar-gitlab-en-centos-7/

Annex 13. Vita

Data de naixement 15/09/1980 en Esplugues de Llobregat, Barcelona. Enginyer tècnic en informàtica de gestió per l'UOC.
Visc a l'Hospitalet de Llobregat, tinc un fill de 5 anys anomenat Edgar.
Tinc tants hobbies que no acabaria... el que em falta és temps... Soc consumidor incansable de llibres, (ara digitals), m'agraden els videojocs (físics), música (en vinil), lego (bricks), i sobretot m'agrada les tecnologies, la robòtica, L'IA, etc.

El primer treball en informàtica va ser al 2002 com a tècnic de sistemes junior. Al 2005 vaig voler donar una volta i vaig fer el que m'agradava realment, la programació.
Vaig programar en Java i altres llenguatges des del 2005; començant com a junior en un parell d'anys ja liderava un grup de 3 persones. Vam créixer fins a 9, creant així un departament web que vaig estar al capdavant gracies a les grans persones que el composaven.
Tot arriba a un fi, i altre cop vaig donar un cop de volta a la vida i vaig buscar altre cop el programar per sobre de gestionar. Vaig liderar un grup inoblidable de gent del quals vaig aprendre molt, sobretot com ser grans companys i persones.

Ara mateix treballo per a la Fira de Barcelona com a PMO i cap de l'oficina tècnica. Trec temps per continuar programant una miqueta i desitjo que així sigui durant molt de temps.
Ara tinc dos passions, DevOps, moviments Agile, etc i tot el mon Cloud i els seus serveis i al acabar el màster m'agradaria continuar els meus estudis en aquests dos fronts.