Orquestració de microserveis amb una estratègia Enterprise Service Bus (ESB)

Treball de final de grau

Autor: **Sergi Rusiñol Vellvehi**
Director: **Iván Jesús Correa Negrín**
Ponent: **Rubèn Tous Liesa**

Grau en Enginyeria Informàtica
Tecnologies de la Informació

FACULTAT D'INFORMÀTICA DE BARCELONA (FIB)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

everis
an NTT DATA Company

AGRAÏMENTS

Primerament, agrair al meu tutor Iván i a l'empresa Everis, per haver-me donat l'oportunitat de dur a terme aquest projecte. M'han facilitat ajuda en tot moment que ho he necessitat, i sense ells aquest projecte no s'hagués pogut dur a terme.

Agrair també al meu ponent Rubèn, que m'ha donat els millors consells que se li pot donar a un alumne, i m'ha guiat en tot moment durant el transcurs d'aquest projecte. M'ha donat ànims per poder realitzar aquest treball i ha confiat en mi en tot moment, per això li estic molt agraït.

Vull agrair també al meu tutor de GEP, Francesc Xavier Llinas Audet, per donar-me la clau per poder realitzar un bon projecte. No solament em va ajudar a aprendre a redactar un projecte en condicions, sinó que també he après a fer presentacions formals i a tenir molt en compte tots els aspectes estètics, que poden fer d'un bon treball, un treball excel·lent.

Finalment, agrair a tots els meus amics i a la família, per poder compartir amb ells tots els bons moments i els mals moments durant la realització d'aquest treball. Ells han sigut el meu principal punt de suport durant tot aquest temps i els hi estic agraït plenament.

Moltes gràcies a tots!

ÍNDEX

ÍNDEX D'IL·LUSTRACIONS

ÍNDEX DE TAULES

1. Introducció

1.1. Context

Des del seu origen, la informàtica s'ha trobat en constant desenvolupament. Al llarg d'aquest temps han anat apareixent tota mena de nous dispositius i programes per tal de satisfer les necessitats de les persones i empreses. Però aquesta millora a gran escala no sempre és beneficiosa.

Actualment un dels àmbits més perjudicats per aquest alt desenvolupament, és el de les tecnologies de la informació, i en concret les seves arquitectures. Per posar un exemple, la informació generada per una empresa augmenta exponencialment amb el temps, cosa que fa que es necessiti implementar noves mesures per poder gestionar aquesta informació. Això provoca que una empresa amb molta antiguitat es trobi ja amb una arquitectura interna definida però obsoleta. En aquest cas l'empresa té dues opcions, o bé redefinir la seva arquitectura a partir de les noves tecnologies que tinguin a l'abast en aquell moment, o bé afegir les noves tecnologies i estructures a l'arquitectura actual, per tal que compleixi el seu propòsit.

Aquesta serà la temàtica d'aquest projecte: desenvolupar una arquitectura basada en tecnologies recents per poder aplicar-la en futurs sistemes de la informació.

1.2. Actors implicats

En aquest projecte s'hi troben implicades diferents persones i l'empresa on s'ha desenvolupat el treball. A continuació es detallarà la funció que tindrà cadascú.

1.2.1. Desenvolupador

L'autor serà el mateix desenvolupador del projecte. No només s'encarregarà de crear i desenvolupar la nova arquitectura, sinó que també haurà de documentar tota la part practica i contrastar-la amb fonts d'informació fiables.

Estarà influenciat pel director i ponent, per tal de seguir unes pautes específiques a l'hora d'elaborar el projecte, ja sigui en termes de planificació, com per l'aconsellament de bones pràctiques, per tal d'obtenir un resultat satisfactori.

1.2.2. Director i ponent

Tant el director com el ponent s'encarregaran de supervisar, aconsellar i corregir al desenvolupador.

El director del projecte és l'Iván Jesús Correa Negrín, i és l'expert en l'àmbit de l'arquitectura de les tecnologies de la informació. Fora d'aquest treball, és el cap de projectes en el sector de serveis de l'empresa Everis, on s'ha desenvolupat aquest projecte. S'encarregarà de donar suport al desenvolupador en la part pràctica i supervisar-lo en termes de planificació.

El ponent és en Rubèn Tous Liesa, professor de l'UPC en la branca de les tecnologies de la informació. Dins del treball té la funció de guiar al desenvolupador i aconsellar-lo en la part de documentació.

1.2.3. Principals beneficiaris

Aquest treball s'ha desenvolupat dins l'empresa Everis, i serà la principal beneficiària del projecte. L'empresa està interessada a desenvolupar una arquitectura basada en microserveis, per tal de poder oferir al client funcionalitats que no estan disponibles en aquests moments. Això permetrà grans millores en l'entorn de treball i, per tant, millorarà l'eficiència de l'empresa en el sector de serveis.

2. Formulació del problema

2.1. Problema

Tal com s'ha esmentat a l'apartat anterior, moltes empreses i altres entitats es troben actualment amb problemes d'arquitectura, més concretament amb problemes d'escalabilitat i de compatibilitat amb noves tecnologies. Aquestes empreses afectades solen ser les que tenen una antiguitat considerable. Un exemple clar, són les entitats

bancàries, que han desenvolupat estructures amb tecnologies antigues, amb les que tenen problemes per donar suport a noves tecnologies i eines per millorar els serveis que ofereixen als seus clients i així obtenir més beneficis.

El que busca aquest projecte, és solucionar un problema similar que presenta l'empresa Everis. L'empresa té una sèrie de microserveis que necessiten que s'hi presenti una interoperabilitat entre ells. Ara per ara, amb l'arquitectura que s'està fent ús, no s'ha trobat cap manera de fer possible aquesta comunicació entre aquests serveis. Per això, l'empresa es va plantejar redissenyar una arquitectura basada en microserveis, per mirar de recuperar funcionalitats perdudes fins al moment.

Una possible solució que es va estudiar, és la utilització d'un ESB dins l'arquitectura de microserveis. Un ESB és un bus de comunicació entre serveis, el qual permetria orquestrar-los. El que falta per determinar, és si a partir de l'ESB i aquest tipus d'arquitectura, seria possible orquestrar els microserveis, ja que és una eina enfocada al trànsit d'informació entre serveis.

2.2. Objectius

Aquest projecte té com a objectiu principal, crear aquest tipus d'arquitectura i demostrar que és possible orquestrar microserveis a partir d'un ESB, dirigits a oferir aplicacions i complementar informació amb un servei superior.

Per tal de poder aconseguir-ho, caldrà desplegar tota una sèrie de microserveis i crear les seves connexions pertinents. Les connexions més importants, per determinar si es pot arribar a obtenir aquest tipus de sistema, seran les connexions entre els microserveis i l'ESB. Caldrà estudiar el seu comportament.

A més a més, s'haurà de crear prèviament un esquema de fluxos de l'ESB, per tal que quan aquest estigui desplegat, sigui conscient de quines connexions entrants tindrà, quines transformacions de la informació entrant haurà de realitzar i quin serà el destinatari.

Finalment, un cop determinat si és, o no, possible aquest tipus d'arquitectura, serà interessant estudiar el rendiment que té i comparar-la amb el que podria ser un altre tipus d'arquitectura amb una estructura i nombre de tecnologies i connexions similar. D'aquesta manera es podrà veure si és un sistema interessant a establir en qualsevol empresa o projecte, sempre que sigui viable aquest tipus d'infraestructura.

3. Estat de l'art

Les arquitectures basades tant en microserveis, com en ESBs, estan despuntant molt avui en dia en l'àmbit de les tecnologies de la informació, i això és degut principalment per l'alta escalabilitat que presenten.

A continuació es presenten amb més detall ambdues arquitectures.

3.2. Microserveis

Els microserveis[1] són serveis modulars i autònoms que treballen amb un objectiu concret. Cada micro-servei és independent dels altres serveis o microserveis. Per tal d'assolir aquesta independència és necessari establir una comunicació via xarxa, sigui LAN o per Internet, per VPN o servidor web.

Moltes empreses han pogut comprovar que, utilitzant arquitectures basades en microserveis, poden desenvolupar i distribuir software ràpidament i establir noves tecnologies dins d'aquesta arquitectura. Els microserveis ens aporten llibertat a l'hora de reaccionar i prendre diferents decisions als canvis inevitables que puguin sorgir amb el pas del temps.

Hi ha una gran diversitat de mètodes de desenvolupament d'aquests microserveis, però en el cas d'aquest projecte s'ha decidit utilitzar Node.js. Aquesta decisió s'ha pres, a causa de les facilitats que ofereix, amb eines com Express.js, que ens permet desplegar servidors HTTP per tal de deixar visibles les nostres aplicacions a altres serveis, o Body-parser, que permet especificar les capçaleres de les transmissions HTTP.

3.2. ESB

Dins del projecte ens trobem amb una altra arquitectura diferent, l'arquitectura basada amb una estratègia ESB.

Un ESB[10] (Enterprise Service Bus) és una arquitectura de missatgeria. Aquesta està composta per una sèrie de normes i principis per tal d'integrar diferents aplicacions a través del seu bus de missatgeria. S'utilitza per a poder comunicar aplicacions i serveis de punt a punt, permetent que, enmig d'aquesta comunicació, es pugui alterar el contingut o format del missatge, per tal que l'aplicació o servei destinatari pugui entendre el missatge resultant, per complementar les dades del missatge, etc.

Actualment els ESBs estan en auge d'ús en les infraestructures de TI de diferents empreses i projectes. Està demostrat que els ESBs augmenten l'agilitat a l'hora d'organitzar les intercomunicacions i redueix el temps de llançament de noves iniciatives al mercat. A més a més, ofereixen una alta escalabilitat, ja que és un element que està en constant creixement. Cada cop ofereix més aplicacions de comunicació i transformació, per tal que si l'empresa decideix afegir dins l'arquitectura noves tecnologies, aquestes es puguin comunicar sense problemes amb altres tecnologies i/o serveis.

En aquest projecte s'ha decidit utilitzar, com a plataforma de desenvolupament de sistemes de comunicació ESB, l'aplicació Mule ESB[2].

3.2.1 Mule ESB

Mule ESB no és més que un sistema dins d'una arquitectura ESB, que permet als desenvolupadors connectar diferents aplicacions de manera ràpida i senzilla. Per tal d'aconseguir-ho, Mule aporta una sèrie de funcionalitats[8]:

- **Creació i allotjament de serveis:** permet la creació de serveis petits i els allotja dins l'ESB, en contenidors, com en Docker.

- **Mediació entre serveis:** permet a serveis externs poder comunicar-se amb els serveis interns i del canal d'entrada de l'ESB, però oferint protecció als serveis locals

dels formats de missatges i els protocols utilitzats. A més a més, per temes de seguretat, la lògica d'empresa es manté separada del servei de missatgeria.

- **Enrutament de la missatgeria:** permet recórrer, filtrar, agregar i reordenar els missatges segons els seus paràmetres i regles.

- **Transformació de les dades:** per poder establir una correcta comunicació entre serveis, permet que es modifiqui el contingut i capçaleres del missatge dins de l'esquema de flux, per poder enviar un missatge amb el format adient pel servei destinatari.

3.3. Arquitectura de microserveis VS. Arquitectura ESB

Actualment, hi ha diverses discussions presents sobre les arquitectures que es troben presents en el projecte. La comparativa entre ambdues arquitectures és inevitable, ja que tenen un propòsit comú, oferir portabilitat a noves i velles tecnologies i millorar eficiència de transmissió de dades entre serveis.

De les dues, l'arquitectura de microserveis[9] és la més recent i la que més possibilitats d'estructuració presenta. De fet, és considerada com una versió evolutiva de l'arquitectura ESB.

No és d'estranyar, que molts experts arquitectes de TI donin a conèixer el seu punt de vista sobre aquesta temàtica. El cas és, que molts d'ells discuteixen sobre l'obsolescència de l'arquitectura ESB com a arquitectura SOA. Ara per ara, la gran majoria considera que l'aparició dels microserveis no serà el final de les arquitectures ESB, sinó que es seguiran utilitzant per diversos motius. El més destacat d'aquests motius, és el fet que, tenint en compte l'objectiu que té un ESB (integració, coordinació, enrutament i monitoratge de les activitats de l'empresa)[7], es pot mirar d'utilitzar l'ESB com a eina per satisfer les necessitats de l'empresa i la dels seus clients. Una funcionalitat present en l'ESB, de la qual en manca a les arquitectures de microserveis, és l'orquestració de serveis, el qual permet la interoperabilitat d'aquests serveis per una millora significativa en eficiència.

4. Abast

Fins al moment, s'ha parlat dels objectius, els problemes i la temàtica del projecte. En aquest apartat ens centrarem en els aspectes particulars del projecte: programari utilitzat, metes finals previstes i els obstacles que han succeït durant la seva realització.

4.1. Programari i eines de treball

El treball s'ha realitzat a partir d'un portàtil amb sistema operatiu Windows 10. Dins d'aquest s'hi van instal·lar diferents programes: Vagrant[3], Docker[4], Anypoint Studio (plataforma de desenvolupament de diagrames de flux del Mule ESB) i Node.js. D'entre aquests programes, els únics que han estat instal·lats directament dins el sistema operatiu són Vagrant i Anypoint Studio. Tan Docker com Node.js s'han instal·lat dins el sistema Vagrant.

4.2. Metes finals

Els següents punts que es comenten a continuació, formen part de les metes finals del projecte, previstes dins el temps límit de realització d'aquest:

- Obtenir un entorn preconfigurat, intuïtiu i portable a qualsevol sistema.
- Determinar que és possible una orquestració de microserveis a partir d'un ESB.
- Crear, configurar i establir una arquitectura formada per microserveis, un ESB i altres serveis.
- Determinar que aquesta arquitectura és més eficient que l'arquitectura actual de l'empresa.

4.3. Obstacles

Cal tenir en compte tots els contratemps i obstacles que poden sorgir durant la realització del projecte. S'ha fet una previsió de tots ells, per tal de poder solucionar-los de la manera més rapida possible. Els obstacles previstos són els següents:

- **Problemes de connexió:** el projecte, com ja s'ha mencionat anteriorment, es desenvoluparà dins de l'empresa Everis. Com és normal a la majoria d'empreses, hi ha instaurades una sèrie de tecnologies y programari a la xarxa per tal d'oferir seguretat. Però aquesta seguretat també és contraproduent en part pels treballadors interns, ja que estan lligats a unes restriccions que complica el fet d'establir connexions amb servidors i webs d'Internet.

- **Incompatibilitat de Software:** molt del Software que s'utilitzarà, operarà simultàniament amb altres programes i tecnologies, i això pot comportar incompatibilitat. Ara per ara s'ha comprovat que el Software sigui compatible per tal de poder crear l'arquitectura desitjada, però amb el temps el Software cal actualitzar-se i es podria donar el cas que algun d'ells ja no sigui compatible per desenvolupar l'arquitectura.

- **Temps limitat:** el treball es realitza dins d'un període de temps establert i es pot donar el cas que, encara que la planificació estigui establerta, no hi hagi prou temps per poder abastar-ho tot i s'hagi de descartar algunes tasques secundàries per prioritzar les principals.

5. Metodologia i rigor

5.1. Metodologia SCRUM

Per aquest projecte s'ha escollit utilitzar una metodologia àgil, en concret SCRUM. Aquesta decisió es va prendre conjuntament amb el meu tutor a l'empresa.

La metodologia SCRUM està pensada per ser utilitzada en grups de treball petits, on es divideix la durada del projecte en etapes, anomenades Sprints, i el treball a realitzar en tasques. Així doncs, per cada Sprint s'estableixen un cert nombre de tasques per a cada desenvolupador, tenint en compte el pes en dificultat, temps requerit i coneixements previs, que es necessitarà per a realitzar la tasca.

Així doncs, vam trobar que aquesta metodologia era la que encaixava més al tipus de planificació del treball.

5.2. Eines de seguiment

Pel fet de ser un projecte realitzat únicament per l'autor, no s'ha trobat necessari utilitzar plataformes com Jira o Trello. En comptes d'optar per una eina de seguiment empresarial, es tenien les reunions d'Sprint amb el tutor i es validava que el treball realitzat durant aquell període de temps fóra correcte. Per tenir un recordatori de les tasques a realitzar, s'ha utilitzat l'aplicatiu web GanttPRO[13], amb el qual visualitzavem el diagrama de Gantt amb les tasques a realitzar i les ja realitzades i validades.

5.3. Eines de control de versions

Per tal de tenir un control de versions i evitar pèrdues en el treball, s'ha decidit utilitzar el repositori github. Amb Github podrem tenir un control general del projecte, detallant els canvis de cada versió, editant el projecte fitxer a fitxer, tornant a versions anteriors a causa del mal funcionament de l'arquitectura més recent, i moltes més funcionalitats.

6. Planificació temporal

6.1. Dates provisionals

Es té previst que el projecte es finalitzi en un període de temps de 4 mesos. La data d'inici del projecte es troba a l'11 de setembre i la data final s'estima que serà al 12 de gener de 2018.

Durant aquest període de temps es dedicaran unes 25 hores setmanals a realitzar el projecte. Tenint en compte que el període de dates establert consta de 113 dies, es dedicaran un total de 404 hores de treball aproximadament.

6.2. Fases del projecte

El projecte constarà de quatre fases ben definides. Aquestes fases són: la fase d'aprenentatge, la de creació de l'arquitectura, la de desenvolupament de l'arquitectura i la de monitoratge i control d'errors del sistema.

Cadascuna de les fases té una sèrie de tasques a realitzar. Aquestes tasques es troben el diagrama de Gantt.

6.3. Diagrama de Gantt

En el diagrama es mostra el títol descriptiu de les tasques previstes, juntament de la seva data d'inici, la data de fi i la seva durada estimada en dies. A continuació es descriurà cada tasca, explicant el seu contingut previst, la seva durada, el seu nivell de risc a patir contratemps i els recursos que es necessitaran per poder-les dur a terme.

6.3.1 Cerca d'informació

La primera tasca consisteix a obtenir informació sobre el concepte principal i els diferents elements que formaran part del projecte, per poder tenir una idea de què s'haurà de fer i què s'haurà d'utilitzar per poder elaborar el projecte.

Aquesta tasca es farà conjuntament amb la fase d'aprenentatge i la tasca dels lliurables de GEP. La seva durada estimada serà d'uns 15 dies.
El risc de la tasca és mínim, ja que no es preveu cap inconvenient que pugui alterar la duració estimada.

Els recursos a utilitzar serà únicament un ordinador amb connexió a Internet.

6.3.2. Entregables de Gestió de Projectes (GEP)

Durant el transcurs del projecte, es realitzarà l'assignatura de GEP. Aquesta assignatura constarà 5 setmanes del projecte, i es basa en elaborar la fita inicial del treball de fi de grau. Durant les primeres 4 setmanes s'entregarà un informe setmanal sobre diferents aspectes inicials del projecte: abast i contextualització, planificació temporal, gestió econòmica i sostenibilitat, i una presentació preliminar. L'última setmana es dedicarà a arreglar els errors que puguin haver-hi en informes anteriors i a recopilar aquests lliurables, per fer l'entrega final de la fita inicial. A més a més, s'haurà de fer una presentació de la fita inicial i

un lliurable sobre el plec de condicions de l'especialitat, que en aquest cas es tracta de les tecnologies de la informació.

No es preveu cap risc durant la realització de l'assignatura, ja que les dates d'entrega estan establertes i no hi pot haver contratemps. La duració d'aquesta tasca es preveu que serà d'uns 34 dies.

Es necessitarà un ordinador amb Internet i programes d'ofimàtica.

6.3.3. Instal·lació de programari

Serà la primera tasca de la fase d'aprenentatge. Com bé indica el seu nom, aquesta tasca es basarà en instal·lar tots els programes pertinents a l'arquitectura.

Aquesta tasca té un risc baix, degut que l'únic inconvenient que hi hagi es trobi en poder obtenir els programes i tecnologies necessaris.

Es té previst que la instal·lació duri un dia i tots els programes s'instal·laran dins del mateix ordinador. Per tant, es necessitarà únicament un ordinador amb Internet.

6.3.4. Aprenentatge Mule ESB

El primer programa en iniciar l'aprenentatge serà el de Mule ESB. S'aprendrà a utilitzar l'Anypoint Studio, un programa propi de Mule per poder desenvolupar l'ESB.

L'aprenentatge durarà aproximadament 7 dies i serà necessari disposar d'un ordinador, connexió a Internet i el programa Anypoint Studio.

Aquesta tasca suposa un risc baix, degut que els inconvenients no ens centrarem en el desenvolupament de l'arquitectura, sinó en posar en pràctica el programa.

6.3.5. Aprenentatge Vagrant

El pròxim programa a estudi serà Vagrant. Amb Vagrant crearem més endavant l'entorn de treball amb totes les seves configuracions necessàries.

Aquesta tasca té un temps estimat de 2 dies i es necessitarà un ordinador amb Internet i el programa Vagrant.

Igual que amb l'aprenentatge de Mule ESB, el seu risc és mínim.

6.3.6. Aprenentatge Docker

Properament, iniciarem l'aprenentatge amb Docker. A partir de Docker es crearan les màquines amb els microserveis i altres tecnologies: ESB, bases de dades, servei web...

S'ha previst un temps de tasca de 5 dies i com a recursos es necessitarà un ordinador amb Internet i el software de Docker.

Pel fet que només es necessitarà practicar amb el programa, el seu risc de produir-se algun contratemps també és baix.

6.3.7. Aprenentatge d'eines de monitoratge

Amb aquesta tasca es finalitza la fase d'aprenentatge. Les eines de monitoratge que es té previst utilitzar seran ElasticSearch[12] i Kibana[11]. Amb aquestes eines controlarem el sistema a partir d'una API i visualitzant-ho en gràfiques.

Aquesta tasca tindrà una durada d'uns 7 dies. Per a dur-la a terme es necessitarà un ordinador amb Internet i dels programes de monitoratge esmentats.

El risc d'aquesta tasca ja és una mica més elevat que les darreres tasques d'aprenentatge, però això és degut que aquestes eines necessiten ser provades dins de màquines, com per exemple màquines de Docker i s'haurà d'avaluar el seu comportament conjunt.

6.3.8. Creació de l'entorn

La primera tasca de la fase de creació serà la tasca de creació de l'entorn amb Vagrant i els diferents elements que compondran el sistema: sistema operatiu de l'entorn Vagrant, editors de text, comandes necessàries com wget, netstat, curl...

La tasca tindrà una duració prevista de 10 dies i es necessitarà un ordinador amb Internet i Vagrant instal·lat.

Té un risc moderat, ja que serà el primer cop que es provaran els diferents programes dins de l'entorn virtual Vagrant i es pot donar el cas que sorgeixin problemes de Software.

6.3.9. Creació i administració de l'arquitectura

En aquesta tasca es crearà el primer esquema de l'arquitectura, on es crearan i configuraran una serie de maquines amb Docker[5] dins de l'entorn Vagrant, per poder, en les tasques següents, desenvolupar l'arquitectura.

Aquesta tasca durarà aproximadament 15 dies i es necessitarà un ordinador amb Vagrant i, dins de Vagrant, Docker instal·lat.

Es té previst un risc bastant elevat, pel fet que poden sorgir problemes de connexió i incompatibilitats entre màquines i Software.

6.3.10. Establiment de comunicacions entre màquines

Amb aquesta tasca s'inicia la fase de desenvolupament. La tasca es basarà en configurar les connexions de les màquines Docker[6], creades a la tasca anterior. Aquestes comunicacions seran principalment entre màquines, però és probable que en algun cas es necessiti configurar connexió a Internet a partir de la màquina. Per tal d'aconseguir-ho caldrà comprovar primer que l'entorn Vagrant té connexió a Internet i posteriorment, que la maquina pot accedir al canal de comunicació creat entre Vagrant i l'entorn principal de l'ordinador.

Aquesta tasca es té previst que duri uns 10 dies, amb molt risc que es produeixin contratemps, principalment perquè a l'empresa on es desenvoluparà hi ha present un Proxy corporatiu que pot complicar aquesta interconnexió.

Els recursos necessaris seran un ordinador amb Internet i amb Vagrant i Docker instal·lat.

6.3.11. Desenvolupament de l'arquitectura

Aquesta tasca comportarà la major part pràctica del projecte. Es desenvoluparan els microserveis, el diagrama de flux de la màquina amb l'ESB, els serveis addicionals, les màquines amb les bases de dades...

S'estima que la tasca tingui una durada de 25 dies, amb un risc molt important per possibles incompatibilitats i problemes de connexió que es puguin presentar.

Es necessitarà un ordinador amb Internet i els programes utilitzats en les tasques anteriors.

6.3.12. Monitoratge de les màquines

En aquesta tasca es tractarà d'establir noves tecnologies a l'arquitectura per poder avaluar el rendiment del sistema. És la primera tasca de la fase de monitoratge i control d'errors.

Es té previst una durada de 8 dies per completar aquesta tasca i per desenvolupar-la es necessitarà un ordinador amb Internet, i els programes ElasticSearch i Kibana.

El risc de la tasca es força alt, ja que s'haurà d'instaurar nou programari que pot entrar en conflicte amb altres programes i tecnologies del sistema.

6.3.13. Control d'errors

Partint de la informació obtinguda per part de les eines de monitoratge, podrem detectar on hi ha, o i pot haver, més fallades del sistema. Afegirem tractament d'errors dins de cada micro-servei, a l'esquema de flux de Mule, i a totes les màquines que es trobin al sistema. És l'última tasca de la fase de monitoratge i control d'errors.

El risc d'aquesta tasca és baix, ja que la dificultat que implica no és gaire elevada. S'estima que es tardi un mínim de 5 dies en finalitzar la tasca.

Es necessitarà un ordinador amb Internet i l'arquitectura establerta amb totes les tecnologies esmentades.

6.3.14. Test i correcció de possibles errors

Aquesta tasca se centrarà en provar que l'entorn i l'arquitectura funciona per a qualsevol sistema correctament. En el cas que no sigui possible el seu bon funcionament en algun cas, s'estudiarà la causa i en cas de tenir solució es procurarà de posar-la en pràctica.

Aquesta tasca té un alt risc, ja que la incompatibilitat d'alguns elements del sistema vers els dispositius on es provi, pot no tenir solució. Val a dir que és una tasca secundària, i que la finalitat del projecte és el funcionament correcte de l'arquitectura, no pas que funcioni en qualsevol sistema operatiu. Així doncs, si la tasca comporta més dies dels esperats, la descartarem i continuarem amb la següent.

La tasca es té previst que es desenvolupi en 5 dies i es necessitarà provar l'arquitectura almenys en un ordinador amb Windows, un MacOS i alguna distribució Linux.

6.3.15. Temps extra per contratemps

En el cas que sorgeixin alguns inconvenients en el correcte desenvolupament de la part pràctica del projecte, s'ha previst que es deixi un marge de 15 dies per poder finalitzar-la correctament. En cas de no ser així, com s'ha esmentat anteriorment, s'estudiaria descartar algunes tasques per poder aconseguir una arquitectura funcional.

6.3.16. Documentació de la memòria

Durant tot el temps invertit en el desenvolupament de l'arquitectura, també s'invertirà part d'aquest temps acabant de documentar tota la seva implementació i altres aspectes referents a la part practica.

Aquesta tasca durarà uns 50 dies i es necessitarà un ordinador amb Internet i programari d'ofimàtica.

El seu risc depèn de com s'hagi anat desenvolupant el treball, així que es té previst un risc moderat per aquesta tasca.

6.3.17. Presentació final del projecte

Es té previst que la presentació final del projecte es faci entre el 23 i 27 de gener de 2018. Per tant, caldrà fer i practicar la presentació final de la defensa del projecte.

S'estima que es tardi uns 5 dies en realitzar aquesta última tasca. Es necessitarà un ordinador amb l'arquitectura desenvolupada funcionant correctament i programari d'ofimàtica pertinent per desenvolupar la presentació amb diapositives.

El seu risc és mínim, ja que el projecte ja estarà desenvolupat i només es necessitarà recopilar dades obtingudes i descriure els diferents elements que s'han utilitzat i documentat en el projecte.

6.1. Pla d'acció i valoració d'alternatives

Com hem pogut observar, cada tasca pot comportar certs riscos que cal preveure amb antelació per tal que el projecte es pugui finalitzar dins el temps establert.

Per tal de tenir en compte aquests retards, s'ha afegit la tasca de "Temps extra per contratemps" que ja hem esmentat a l'apartat anterior. Però es pot donar el cas que aquest període de dies no sigui suficient per abastar tot el retard acumulat per altres tasques. Si es donés aquest cas, cal que tinguem un en compte un pla d'acció.
La primera etapa d'aquest pla, consisteix en determinar si una tasca ha necessitat més o menys temps del previst. En el cas que la tasca s'hagi finalitzat abans de temps, s'afegiran els seus dies restants a la tasca de temps extra. En cas contrari, cal tenir en compte si queden prou dies de marge per a finalitzar la tasca. En cas de no poder finalitzar-la a temps,

analitzarem les tasques pendents, fixant-nos amb les de menys prioritat. Un cop tinguem aquestes tasques detectades, retallarem els dies necessaris per poder acabar la tasca pendent. Es pot donar el cas que arribem a descartar aquestes tasques secundàries per falta de temps, ja que el que volem prioritzar són les tasques que ens permeten complir els objectius esmentats en apartats anteriors.

7. Identificació dels costos

Per tal de realitzar l'arquitectura d'aquest projecte, cal tenir en compte els costos que comportarà el procés per obtenir-la. Primer de tot, es necessitaran recursos humans, la mà d'obra per configurar i desenvolupar l'entorn. També caldrà tenir en compte els recursos materials, que en aquest cas es limiten bàsicament al Software i al Hardware. A més a més hi ha altres recursos que es necessitaran per obtenir aquesta arquitectura, els quals comentarem més endavant.

7.1. Costos directes

Els recursos humans representen els costos directes del projecte. Com que aquest projecte s'ha realitzat només amb un treballador, caldrà tenir en compte totes les tasques realitzades pel treballador, amb el seu cost estimat. A continuació es mostren les taules de costos per cada rol realitzat pel treballador, segons les tasques del projecte, seguint el diagrama de Gantt explicat anteriorment.

Fases i tasques	Rol	Hores estimades
Cerca d'informació	Cap de projecte	54h
Lliurables de GEP	Cap de projecte	40h
Test i correcció de possibles errors	Tester	15h
	Desenvolupador	8h
Documentació de la memòria	Cap de projecte	35h
Presentació final del projecte	Cap de projecte	10h
Aprenentatge		
Instal·lació de programari	Desenvolupador	8h
Aprenentatge Mule ESB	Desenvolupador	25h

Aprenentatge Vagrant	Desenvolupador	10h
Aprenentatge Docker	Desenvolupador	20h
Aprenentatge d'eines de monitoratge	Desenvolupador	15h
	Tester	10h
Creació		
Creació de l'entorn	Analista	16h
	Desenvolupador	24h
Creació i administració de l'arquitectura	Analista	12h
	Desenvolupador	35h
	Tester	10h
Desenvolupament		
Establiment de comunicacions entre màquines Docker	Desenvolupador	50h
Desenvolupament de l'arquitectura	Desenvolupador	110h
Monitoratge i control d'errors		
Monitoratge de les màquines	Desenvolupador	30h
Control d'errors	Tester	25h
Total d'hores estimades		**562h**

Taula 1: taula d'hores estimades

Rol	Cost per hora	Total d'hores	Cost estimat
Cap de projecte	45€/h	139h	6,255€
Analista	35€/h	28h	980€
Desenvolupador	25€/h	335h	8,375€
Tester	20€/h	60h	1,200€
Cost total estimat			16,810€

Taula 2: taula de costos estimats

7.2. Costos indirectes

El cost del Software, Hardware i altres despeses que es produiran, representen els costos indirectes, ja que no fan referència al projecte en si, sinó que fan referència a un propòsit més general de desenvolupament de qualsevol projecte.

En el cas del Software, no s'utilitzarà cap programari amb llicència pel desenvolupament del projecte, ja que tots ells són programes Open Source. L'únic programa amb llicència que s'utilitzarà, serà el sistema operatiu Windows 10, però aquest cost es té en compte en la part de Hardware, ja que ve inclòs en la compra del portàtil. Per tant, el cost en Software per aquest projecte ha estat nul.

Pel que fa al Hardware, com hem esmentat, s'utilitzarà un portàtil Dell Latitude e5570[16] pel desenvolupament del projecte, juntament amb un monitor Acer V223HQV[17]. A més a més, per millores de comoditat, s'ha comprat també un teclat Microsoft Wired Keyboard 200[15] amb un ratolí Targus AMU30EUZ[14]. A continuació es mostra una taula amb el cost total del hardware.

Hardware	Cost
Portatil Dell Latitude e5570	835€
Monitor Acer V223HQV	85€
Teclat Microsoft Wired Keyboard 200	11€
Ratolí Targus AMU30EUZ	9.50€
Cost total	**941€**

Taula 3: taula de despeses en Hardware

A més a més, en aquest projecte s'han tingut en compte altres costos necessaris: la llum, l'establiment des d'on s'ha realitzat el projecte, els viatges en tren i l'Internet. No obstant això, per determinar els costos indirectes globals del projecte, caldrà tenir en compte la seva dedicació dins del projecte.

Despeses	Ús habitual (projecte)	Màxim ús	Dedicació
Llum	4h/dia	24h/dia	16.67%
Internet	4h/dia	24h/dia	16.67%
Desplaçament amb tren	2 dies/setmana	7dies/setmana	28.57%
Lloguer establiment	5 dies/setmana	7dies/setmana	71.43%
Amortització Hardware	4 mesos	5 anys (vida útil)	6.67%

Taula 4: taula de dedicació aplicada per despesa

Costos indirectes	Preu	Dedicació	Cost
Llum	62€/mes	16.67%	41.33€
Internet	35€/mes	16.67%	23.33€
Desplaçament amb tren	40€/mes	28.57%	45.71€
Lloguer establiment	300€/mes	71.43%	857.14€
Amortització Hardware	941€	6.67%	62.73€
Cost total estimat			**1,030.26€**

Taula 5: taula de costos indirectes

El cost amortitzat s'ha obtingut tenint en compte que el projecte es realitzarà durant 4 mesos, tal com s'ha mencionat a la planificació i es veu reflectit en el Gantt.

7.3. Contingència

Per tal de tenir suficient pressupost, en cas de produir-se algun problema, es va establir un fons extra de contingència. Es va decidir afegir un 15% del cost total estimat, tant per costos directes com per costos indirectes, per evitar futurs problemes.

Tipus de cost	Preu	Cost
Costos directes	25.425€	3,814€
Costos indirectes	1.441€	216€
Cost total de contingència		**4,030€**

Taula 6: taula de costos de contingencia

7.4. Imprevistos

A més a més del pla de contingència, es va deixar un marge de diners en cas de patir imprevistos. Aquests imprevistos ja poden ser a causa d'obsolescència de programari o d'avaria de l'ordinador des d'on s'ha treballat, o per retards inesperats de les tasques, etc. Per tant, hem distingit els imprevistos en retard de tasques i en avaries de material, que en el cas d'aquest projecte seran principalment avaries de Hardware.

Imprevist	Probabilitat	Unitats	Preu estimat	Cost
Retard en les tasques	20%	40h	30€	240€
Averia de Hardware	5%	1	800€	40€
Cost total d'imprevistos				**280€**

Taula 7: taula de costos per imprevistos

7.5. Pressupost final

Finalment, tenint en compte tots els costos esmentats en els apartats anteriors, vam obtenir el pressupost destinat a aquest projecte, per tal de poder-lo dur a terme. A continuació es mostra la taula amb el pressupost final.

Tipus de costos	Cost
Costos directes	16,810€
Costos indirectes	1,030€
Contingència	4,030€
Imprevistos	280€
Pressupost total	**22,150€**

Taula 8: taula de cost total del projecte

8. Control de gestió

Com ja s'ha explicat a l'apartat de "*Metodologia i rigor*", per tal de tenir un control de les hores realitzades en el treball, s'ha utilitzat l'aplicació web GanttPRO. La previsió i estimació de les hores per cada tasca es realitzaven a les reunions de la planificació Sprint establerta, la qual constava d'una reunió setmanal amb el tutor del projecte per tal de supervisar el desenvolupament correcte del treball.

En el projecte va haver-hi una estimació final de 562 hores. Per tal d'obtenir una desviació del cost del projecte s'ha utilitzat la següent fórmula:

Desviació del projecte = (hores totals estimades - hores totals realitzades) * cost per hora

Fent ús d'aquesta fórmula, obtindrem el tipus de desviació. Si és una desviació positiva, significarà que el projecte ha finalitzat dins el pressupost esperat i, fins i tot, han sobrat diners que podríem invertir per a futurs millores del projecte. Si la desviació és negativa, significarà que el cost del projecte ha estat superior a l'esperat i que s'haurà d'invertir més diners en la seva finalització.

9. Sostenibilitat

9.1. Sostenibilitat Econòmica

En aquest projecte s'han tingut en compte els costos tant de recursos humans, com de béns materials. A més a més, s'ha fet una estimació dels marges extres afegits, tenint en compte els diferents problemes que es podien produir en el transcurs del desenvolupament del projecte.

Tenint en compte el pressupost final vist en apartats anteriors i el fet que tot el programari utilitzat era Open Source, dóna a entreveure que aquest projecte podria ser bastant viable per empreses dedicades o amb sistemes de TI, ja que el cost final és assequible per i els beneficis seran molt significants, degut a la millora en eficiència del sistema.

Tal com s'ha esmentat anteriorment, aquest treball només ha estat fet per un treballador, fet que comporta, que amb un equip de treball d'entre 3 i 6 persones, es pugui acabar fent aquest mateix projecte en bastant menys temps, i per tant, menys cost, ja que la major part del cost del pressupost final radica en els recursos humans. A més a més, hi ha tasques que han comportat un temps de finalització molt elevat, que es podria haver disminuït en el cas d'haver utilitzat una arquitectura prèvia d'algun projecte de l'empresa

Everis.

Com ja s'ha esmentat en apartats anteriors, aquest projecte està associat a l'empresa Everis, i és un cas d'estudi per veure si seria factible el seu ús dins de l'empresa. Per tant aquest projecte és independent a qualsevol altre projecte dins l'empresa.

9.2. Sostenibilitat Ambiental

Anteriorment, s'ha especificat quins recursos seran necessaris per desenvolupar el projecte. D'aquests, els únics que tindran algun impacte relacionat directament amb el medi ambient seran els recursos de Hardware. Els dispositius implicats seran l'ordinador portàtil i el monitor. Ambdós sumen un consum de 0.136Wh[18], que expressat en emissions CO^2 són 0.088kg/h[19].

Al final del desenvolupament del projecte, tenint en compte que es té previst finalitzar-lo en 562 dies i sabent que es treballarà una mitja de 25 hores setmanals, el nostre projecte generarà un total de 176.62Kg de CO^2.

Si volem preveure les emissions que es produiran, un cop el projecte en marxa, caldrà tenir en compte la vida útil del projecte. Com ja s'ha mencionat, es tracta d'un projecte totalment portable, per tant, qualsevol ordinador podrà executar el programari per necessari per desplegar l'arquitectura. Això ens ofereix la possibilitat d'aprofitar dispositius que funcionin, per poder mantenir el sistema en funcionament. Per tant, no podem estimar les emissions de CO^2 a partir de la vida útil dels dispositius, ja que amb el deteriorament del Hardware, no es compromet el deteriorament del projecte.

L'altre problema medi ambiental que suposarà el projecte, és la contaminació per emissió de residus. Com qualsevol projecte informàtic, les tecnologies es van deteriorant amb el pas del temps i un cop deixen de funcionar, es canvien per altres de noves. En el cas d'aquest projecte s'ha estimat que tant l'ordinador, com el monitor amb els que s'ha desenvolupat, tenen una vida útil de 5 anys. Així doncs, passat aquest període de temps, es generaran residus materials.

Val a dir, que tant per emissió de CO^2, com per generació de residus electrònics, el nostre projecte te un impacte ambiental negatiu mínim.

9.3. Sostenibilitat Social

L'empresa on s'ha realitzat el projecte es localitza a Barcelona. Actualment dins l'empresa s'estan desenvolupant força canvis d'estructuració i d'innovació, fet que comporta que aquest projecte pugui ser molt útil per Everis. El projecte podria ser necessari per modificar l'esquema intern de l'arquitectura de l'empresa. Això simplificaria l'ús i desenvolupament dels serveis i això afavoriria tant a empleats, com a clients, ja que permetrà l'ús dels diferents microserveis que havien deixat de funcionar o ser compatibles, utilitzant aquesta nova implementació estudiada. Això comportarà un gast en temps i diners per part de l'empresa, perquè establir un nou sistema d'arquitectura dins d'un implementat des de fa temps no és pas senzill. Ara bé, això permetrà a l'empresa reutilitzar eines i millorar la seva eficiència de treball, i per tant, recuperar aquesta inversió amb beneficis bastant destacables.

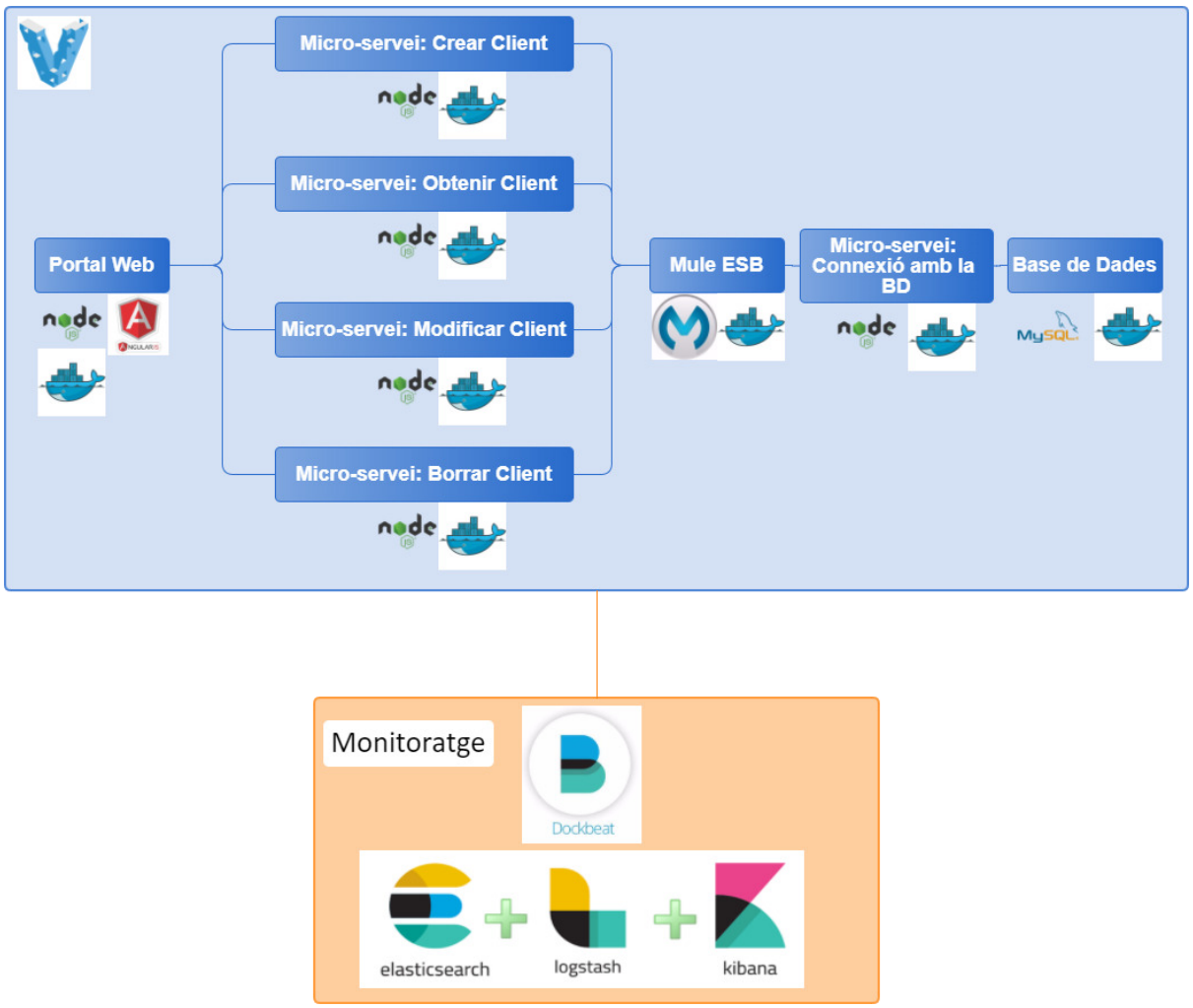
9.4. Matriu de sostenibilitat

	PPP	Vida útil	Riscs
Ambiental	Consum del disseny	Petjada ecologica	Riscs ambientals
	7	**18**	**-2**
Econòmic	Factura	Pla de viabilitat	Riscs econòmics
	5	**15**	**-8**
Social	Impacte personal	Impacte social	Riscs socials
	7	**20**	**-12**
Rang de sostenibilitat	**19**	**53**	**-22**
	50		

Taula 9: matriu de sostenibilitat

10. Disseny

Després de realitzar diverses versions de l'esquema de l'arquitectura, hem acabat amb el disseny següent:



Il·lustració 1: Esquema de l'arquitectura

Tal com es pot apreciar, s'han creat els quatre microserveis. El conjunt d'aquests microserveis formen un entorn CRUD (Create , Read, Update i Destroy), amb el qual l'empresa The Box que hem simulat, és capaç d'administrar els seus usuaris. Per part de l'usuari, aquest també serà capaç d'afegir un nou compte, consultar-lo, modificar-lo o donar-se de baixa d'aquest, a través d'un portal web.

Com s'ha esmentat en apartats anteriors, hem afegit una nova eina que forma part normalment de l'arquitectura ESB, el Mule ESB. Aquest s'encarregarà d'orquestrar els microserveis per tal que la informació d'aquests arribi a la base de dades i que aquesta sigui capaç d'entendre-la.

Per últim s'ha afegit un entorn de monitoratge format per màquines ELK (ElasticSearch, Logstash i Kibana) a més a més de DockBeat, un software bastant recent.

A continuació a l'apartat d'implementació explicarem amb més detall cada element que conforma el disseny de l'arquitectura.

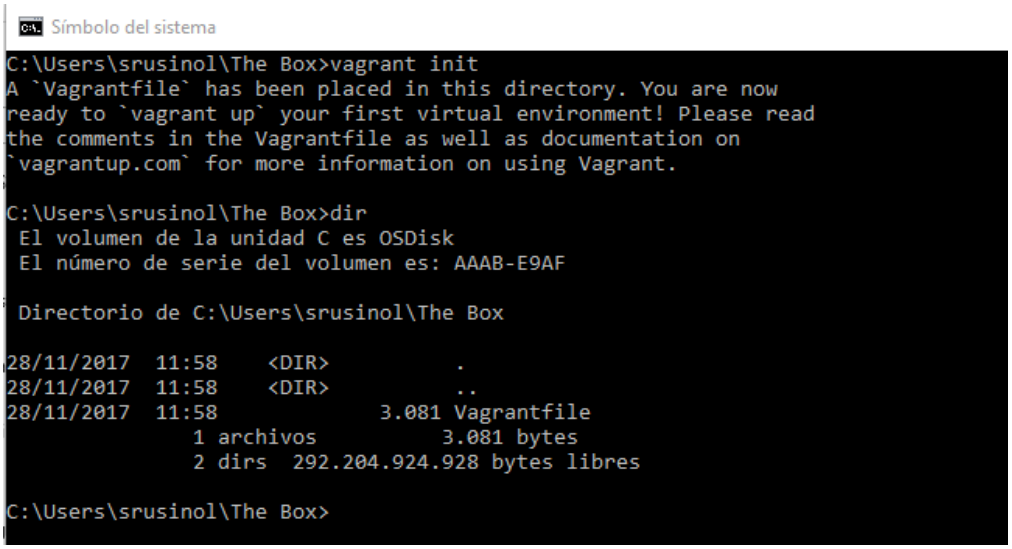
11. Implementació

Partint del diagrama de Gantt, i tenint en compte els diferents elements i objectius del nostre projecte, procedim a implementar el nostre projecte.

Desglossarem el projecte partint dels diferents programaris i eines utilitzats i en l'ordre que s'han inserit al projecte.

11.1. Vagrant

Vagrant ens permetrà tenir un entorn virtual amb un sistema operatiu aliè al del sistema on estigui instal·lat. Per tal de configurar Vagrant, cal crear un fitxer anomenat Vagrantfile a la carpeta on es desitgi crear l'entorn. Aquest fitxer està programat amb Ruby, però no cal tenir amplis coneixements del llenguatge per configurar l'entorn. La comanda de terminal *vagrant init* ens permet crear un fitxer Vagrantfile inicial. Altrament, també podem crear el fitxer manualment, seguint els passos i instruccions pertinents a Vagrant que podem trobar a la seva web oficial (www.vagrantup.com).



Il·lustració 2: vagrant init

En el cas del nostre projecte, hem decidit utilitzar com a sistema operatiu l'Ubuntu 14.04, ja que després de realitzar diverses proves era l'entorn més estable per dur a terme l'arquitectura. Aquest entorn tindrà una memòria RAM de 4GB i 2 CPUs de processament, per la raó de que les eines de monitoratge, que són ElasticSearch, Kibana i Logstash consumeixen força recursos del sistema. Per exemple, amb 2GB de RAM i 1 CPU no era possible desplegar l'entorn de monitoratge.

Tenint en compte que posteriorment utilitzarem Docker per poder crear l'arquitectura, hem afegit al Vagrantfile l'opció *provision,* amb la qual indiquem que, quan el sistema es configuri, inclogui instal·lat Docker.

A més a més, en el fitxer hem declarat que volem que el sistema amfitrió (*host*) i el sistema virtual Vagrant (*guest*) es comuniquin a través de l'adreça IP 127.0.0.1 o localhost. El *host* té configurat com a port d'enllaç el 8081 i el *guest* el port 80. D'aquesta manera, accedint al 127.0.0.1:8081 des del navegador del nostre sistema local, podem accedir al portal web que es trobarà dins la nostra arquitectura Vagrant.

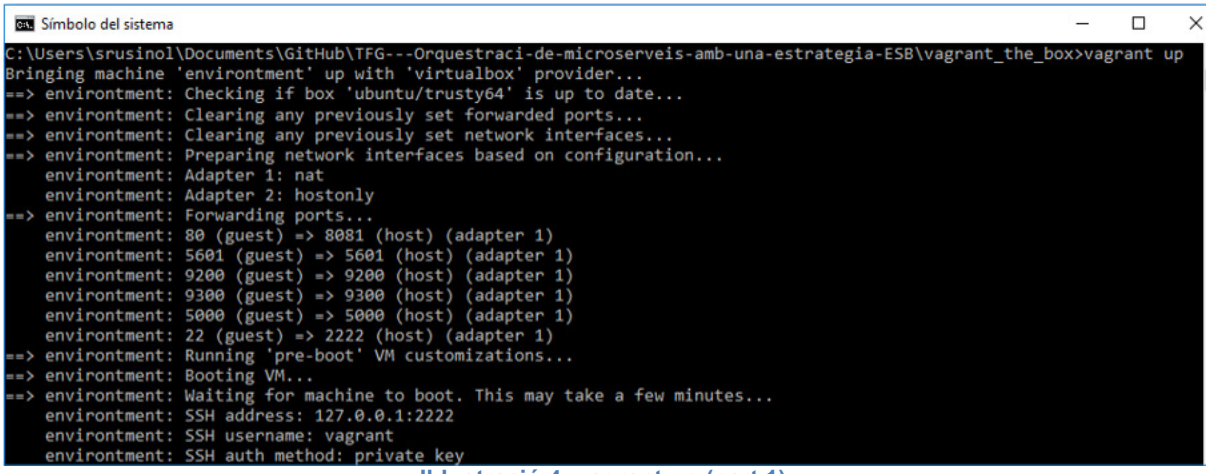
Finalment s'ha afegit la configuració de l'entorn de monitoratge. Li hem assignat la IP privada 10.110.0.210 amb els ports següents establerts per a cada component de l'entorn: el 5601 per Kibana, el 9200 i el 9300 per ElasticSearch i el 5000 per Logstash. Els ports esmentats són simètrics, es a dir, tant per a l'entorn global com per l'entorn Vagrant, els

ports seran els mateixos. D'aquesta manera, per tal d'accedir a Kibana, cal que naveguem a la IP 10.110.0.210:5601.

```ruby
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/trusty64"
  config.vm.hostname = "docker-host"
  config.vm.provider "virtualbox" do |v|
    v.memory = 4096
        v.cpus = 2
  end

  config.proxy.http     = "http://10.110.8.42:8080"
  config.proxy.https    = "http://10.110.8.42:8080"
  config.proxy.no_proxy = "localhost,127.0.0.1"

  config.vm.provision "docker"
  config.vm.network :forwarded_port, guest: 80, host: 8081, host_ip: "127.0.0.1"
  config.vm.define "environtment" do |server|
    server.vm.hostname = "monitoring"
    server.vm.network "private_network", ip: "10.110.0.210"
    server.vm.network "forwarded_port", guest: 5601, host: 5601     #KIBANA
    server.vm.network "forwarded_port", guest: 9200, host: 9200     #ELASTICSEARCH
    server.vm.network "forwarded_port", guest: 9300, host: 9300     #ELASTICSEARCH
    server.vm.network "forwarded_port", guest: 5000, host: 5000     #LOGSTASH
  end


end
```

Il·lustració 3: Vagrantfile

Un cop hem acabat de crear el nostre fitxer Vagrantfile, ja només queda desplegar l'entorn virtual Vagrant i accedir-hi. Per tal de fer-ho, Vagrant ens proporciona dues noves instruccions: vagrant up i vagrant ssh. Amb vagrant up es crea i configura l'entorn a partir del Vagrantfile i amb vagrant ssh s'accedeix a l'entorn Vagrant.



Il·lustració 4: vagrant up (part 1)

```
==> environment: Machine booted and ready!
==> environment: Checking for guest additions in VM...
    environment: The guest additions on this VM do not match the installed version of
    environment: VirtualBox! In most cases this is fine, but in rare cases it can
    environment: prevent things such as shared folders from working properly. If you see
    environment: shared folder errors, please make sure the guest additions within the
    environment: virtual machine match the version of VirtualBox you have installed on
    environment: your host and reload your VM.
    environment:
    environment: Guest Additions Version: 4.3.36
    environment: VirtualBox Version: 5.1
==> environment: Setting hostname...
==> environment: Configuring and enabling network interfaces...
==> environment: Configuring proxy for Apt...
==> environment: Configuring proxy for Docker...
==> environment: Configuring proxy environment variables...
==> environment: Mounting shared folders...
    environment: /vagrant => C:/Users/srusinol/Documents/GitHub/TFG---Orquestraci-de-microserveis-amb-una-estrategia-ESB/vagrant_the_box
==> environment: Machine already provisioned. Run `vagrant provision` or use the `--provision`
==> environment: flag to force provisioning. Provisioners marked to run always will still run.
```

Il·lustració 5: vagrant up (part 2)

```
C:\Users\srusinol\Documents\GitHub\TFG---Orquestraci-de-microserveis-amb-una-estrategia-ESB\vagrant_the_box>vagrant ssh
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 3.13.0-119-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Wed Nov 29 09:09:33 UTC 2017

  System load:                    0.96
  Usage of /:                     17.7% of 39.34GB
  Memory usage:                   3%
  Swap usage:                     0%
  Processes:                      99
  Users logged in:                0
  IP address for eth0:            10.0.2.15
  IP address for eth1:            10.110.0.210
  IP address for docker0:         172.17.0.1
  IP address for br-3c0954f4eb67: 172.16.0.1
  IP address for br-a7f4452daf4d: 172.15.0.1
  IP address for br-f261555ceec4: 172.18.0.1

  Graph this data and manage this system at:
    https://landscape.canonical.com/

  Get cloud support with Ubuntu Advantage Cloud Guest:
    http://www.ubuntu.com/business/services/cloud

New release '16.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.
```

Il·lustració 6: vagrant ssh

11.2. Docker

Finalitzat el procés de creació i configuració de l'entorn virtual, ara toca la creació i configuració de l'arquitectura. Per fer-ho, tal com hem pogut apreciar al disseny de l'arquitectura, caldrà desplegar tota una sèrie de màquines Docker.

Les primeres màquines a desplegar seran les dels microserveis, la de Mule i la de la base de dades, per tal de comprovar que la infraestructura principal del projecte funciona correctament i que es pot establir una comunicació.

Igual que amb Vagrant, per desplegar certes màquines Docker, també necessita un fitxer de configuració anomenat Dockerfile. En el nostre cas voldrem vincular una sèrie de fitxers dins del directori de cada micro-servei creat a una màquina Docker. Per tant, és necessari

crear un fitxer Dockerfile per configurar cada màquina.

```
# Use Node v4 as the base image.
FROM node:4

# Add everything in the current directory to our image, in the 'app' folder.
ADD . /app

# Install dependencies
#RUN cd /app; \
#    npm install --production

# Expose our server port.
EXPOSE 8087

# Run our app.
CMD ["node", "/app/index.js"]

```

Il·lustració 7: Dockerfile

Tots els Dockerfile tenen la mateixa estructura que l'exemple de la imatge anterior. A continuació explicarem les comandes usades en aquests fitxers:

- **FROM:** amb aquesta instrucció li proporcionem a Docker la imatge que aquell container utilitzarà. Aquesta imatge s'haurà de trobar dins dels repositoris Docker, anomenats Docker Hub. En cas de ser un repositori privat creat pel propi desenvolupador, caldrà fer *log in* a Docker Hub abans de compilar el Dockerfile.

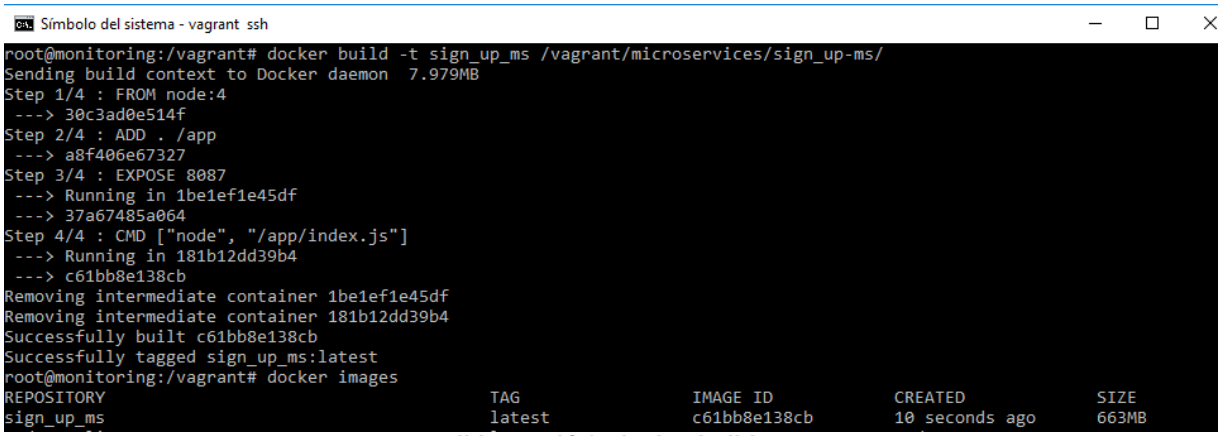
- **ADD:** permet copiar fitxers, carpetes i fins i tot arxius remots a partir d'una URL, afegint-les a un directori dins de la imatge base del container.

- **RUN:** executa comandes dins de la màquina. S'utilitza per establir una configuració inicial cada cop que es desplegui el container.

- **EXPOSE:** per tal que la màquina pugui connectar-se amb altres màquines i entorns, cal determinar quin port estarà exposat a l'exterior per tal que es puguin realitzar enllaços, transmissió de dades i missatges.

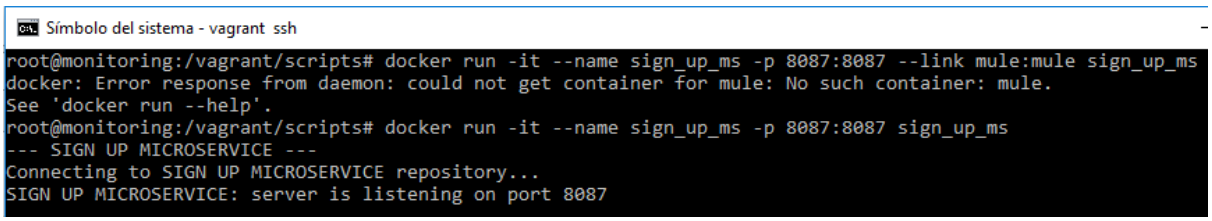
- **CMD:** proporciona valors per defecte d'un container. Només pot haver-hi una sola comanda CMD al Dockerfile. En cas d'haver-hi més d'una, sols s'executarà l'últim CMD del fitxer.

Una vegada ja hem creat el Dockerfile desitjat, cal executar les comandes de compilació i execució de Docker: *docker build* i *docker run*. Amb *docker build*, especifiquem quin serà el nom de la nova màquina i la ruta on es troba el Dockerfile. Un cop executada la comanda, ja tindrem la nova imatge de la màquina compilada al Docker.



Il·lustració 8: docker build

Quan ja tinguem la imatge creada, per tal de desplegar-la utilitzarem la comanda *docker run*. Amb aquesta comanda hem d'especificar, com a mínim, quina imatge volem arrancar. A més a més podem afegir-hi diferents paràmetres addicionals com: quins ports volem que la màquina estigui escoltant, quins directoris locals volem vincular amb la màquina, amb quines màquines volem que es creí un enllaç directe, quines variables d'entorn volem que tingui ja establertes, amb quin nom volem que es despleguin, etc.

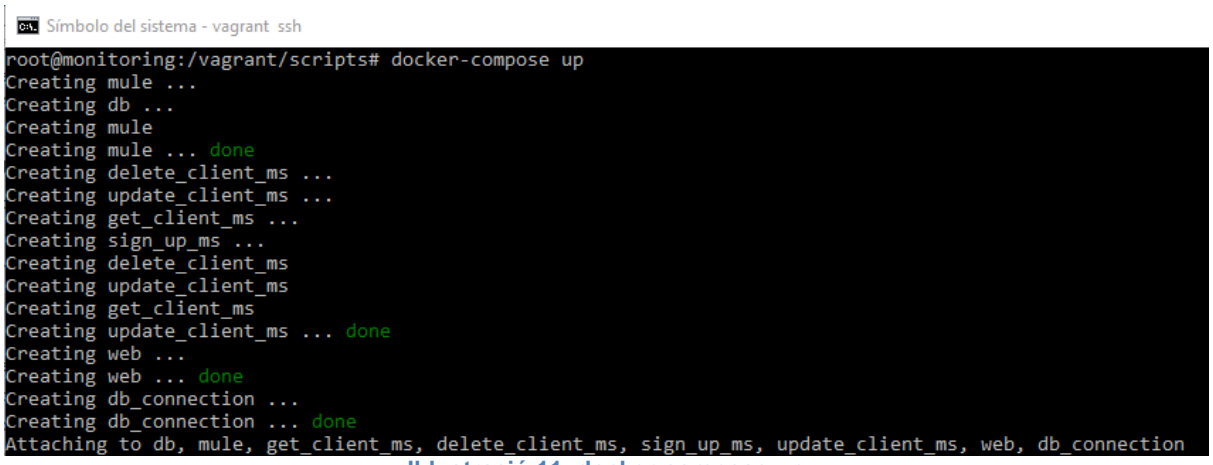


Il·lustració 9: docker run

Inicialment, desplegàvem totes les màquines manualment amb aquestes comandes. A mesura que anava creixent el projecte, aquest procés es va anar fent costos i es va procedir a utilitzar una altra eina de Docker anomenada *docker-compose*. Aquesta eina ofereix la possibilitat de desplegar tots els containers amb una sola comanda: *docker-compose up*. Abans d'utilitzar la comanda hem de crear un altre fitxer de configuració que s'anomenarà *docker-compose.yml*. En aquest fitxer especifiquem totes les màquines que volem desplegar amb els seus paràmetres adients.

```yaml
version: '2'
services:

  sign_up_ms:
    container_name: sign_up_ms
    ports:
      - 8087:8087
    links:
      - mule:mule
    restart: always
    image: sign_up_ms
    networks:
      the_box:
        ipv4_address: 172.16.0.9
    logging:
      driver: "syslog"
      options:
        syslog-address: "tcp://127.0.0.1:5000"

```

Il·lustració 10: fracció de codi del fitxer docker-compose.yml



Il·lustració 11: docker-compose up

Pel que fa a la part de desenvolupament de cada màquina, cal tenir en compte el seu propòsit. Com ja hem esmentat anteriorment, tots els microserveis estan implementats amb Node.js.

Node.js està orientat a desenvolupar aplicacions i serveis web. Utilitza Javascript, cosa que facilita el desenvolupament d'aquestes aplicacions i serveis, ja que és un llenguatge molt utilitzat avui en dia. És per aquest motiu que s'adequa perfectament al desenvolupament dels microserveis del projecte. A continuació mostrem un exemple d'implementació d'un micro-servei de l'arquitectura amb Node.js.

```
var express = require('express');
var http = require('http');
var app = express();
var bodyParser = require('body-parser');

var port = 8087;

console.log("--- SIGN UP MICROSERVICE ---");
console.log("Connecting to SIGN UP MICROSERVICE repository...");

//  Log unhandled exceptions.
process.on('uncaughtException', function(err) {
  console.error('SIGN UP MICROSERVICE: Unhandled Exception', err);
});
process.on('unhandledRejection', function(err, promise){
  console.error('SIGN UP MICROSERVICE: Unhandled Rejection', err);
});

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.post('/sign_up_client', function(req, res) {

  console.log("SIGN UP MICROSERVICE -> Client to sign up: " + JSON.stringify(req.query));

  var requestOptions = {
    hostname: 'mule',
    port: 8083,
    path: "/sign_up_client",
    method: "POST"
  }
```

Il·lustració 12: Implementació del micro-servei *sign up microservice (part 1)*

```

  var externalRequest = http.request(requestOptions, (externalResponse) => {
    externalResponse.on('data', function(data) {
      console.log("SIGN UP MICROSERVICE -> External response: " + data);
    })
    externalResponse.on('end', () => {
      res.end('Data has been sent to destination');
    })
  })
  var data = JSON.stringify(req.query);
  externalRequest.end(data);
})

app.listen(port, (err) => {
  if (err) {
    return console.error('SIGN UP MICROSERVICE: An error ocurred on microservice!: ', err)
  }

  console.log('SIGN UP MICROSERVICE: server is listening on port 8087')
})
```

Il·lustració 13: Implementació del micro-servei *sign up microservice* (part 2)

Aquest codi representa el micro-servei de creació d'usuari. Bàsicament, el micro-servei escolta pel port 8087 i espera que li arribi una petició POST per aquest port. Després de rebre aquesta petició amb els paràmetres de l'usuari, li envia a Mule totes les dades pel port 8083 amb una altra petició POST.

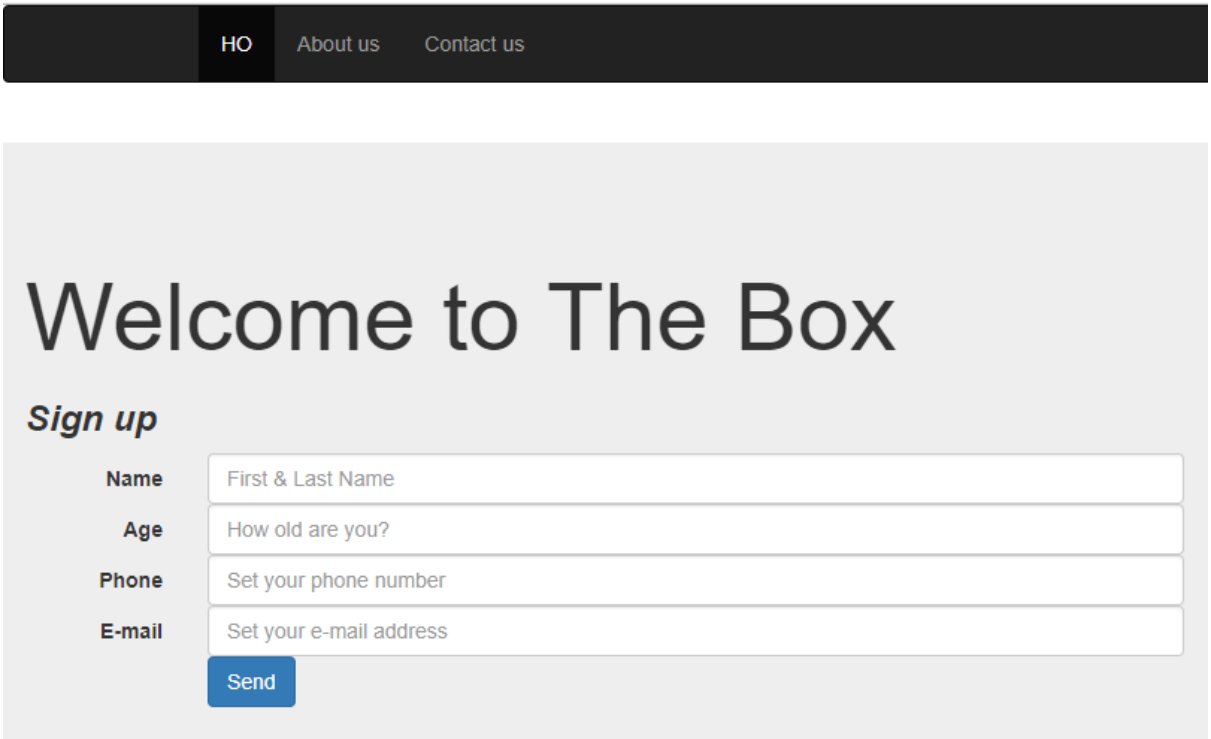
Les altres màquines Docker de microserveis tenen una estructura similar a l'anterior codi. A part d'aquestes màquines, també hi ha un portal web desenvolupat amb Node.js i Angular, que facilita les peticions als diferents microserveis. També cal tenir en compte les màquines Mule, desenvolupat amb esquemes XML i la base de dades amb MySQL. A més a més, els dispositius encarregats del monitoratge, tan Logstash, com Kibana, com ElasticSearch, estan desplegats en maquines Docker. L'únic element de monitoratge que no està en un container Docker és Dockbeat.

11.2.1. Portal web

El portal web consta de quatre formularis, amb relació directa amb els serveis oferts pels microserveis, amb la finalitat de simular un entorn pels usuaris o administradors de l'arquitectura.

Tal com s'ha explicat anteriorment, el portal web s'ha desenvolupat dins d'un container Docker amb Node.js. Per la part de presentació s'ha programat el portal amb Angular.js, ja que és el *framework* més utilitzat avui en dia pel desenvolupament *front-end* de llocs web i ofereix facilitats, com la posibilitat d'afegir estils de forma eficient, o poder utilitzar elements creats a partir del seu llenguatge base (HTML, Javascript).

Aquest portal ens simplifica les peticions a cada micro-servei i ens permet veure que el sistema està funcionant correctament. A cada petició el sistema ens respon amb un missatge d'alerta segons si s'ha pogut realitzar l'acció, si no s'ha pogut realitzar, o bé si s'ha realitzat però no s'ha obtingut el resultat esperat.

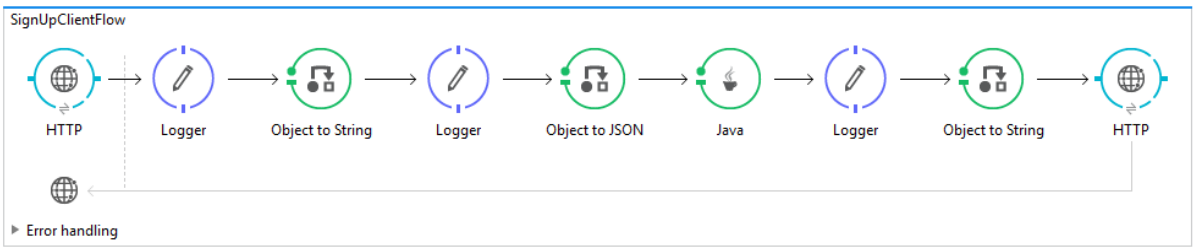


Il·lustració 14: vista previa del portal web

11.3. Mule

Amb Mule, tal com s'ha esmentat anteriorment, ens hem encarregat d'orquestrar els microserveis i de transformar els missatges obtinguts per aquests, per tal que la base de dades els pugui entendre correctament.

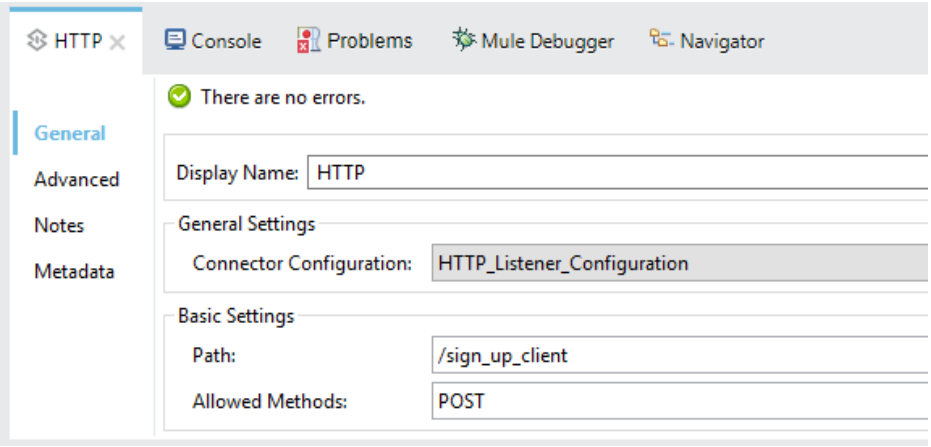
Per tal de poder aconseguir aquest funcionament amb Mule, hem utilitzat una plataforma de desenvolupament pròpia de Mule, anomenada Anypoint Studio. Amb aquest programa podem desenvolupar esquemes XML de forma intuïtiva, a través d'una interfície gràfica amb elements anomenats connectors.



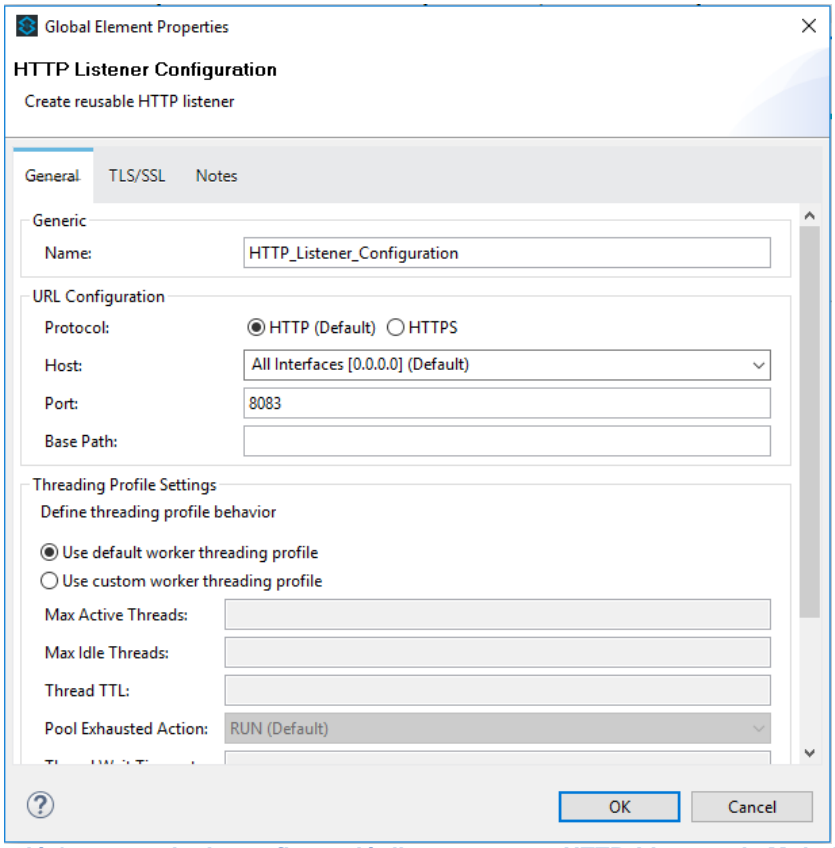
Il·lustració 15: exemple d'un esquema de flux de Mule

Aquí tenim l'exemple del flux del micro-servei *sign_up_ms*. Hi ha un flux per cada micro-servei dins de l'esquema. A continuació explicarem l'ús de cada connector que es mostren en l'exemple anterior:

- **HTTP:** depenent de la posició on es trobi el connector, funcionarà com a *listener*, o com a *request*. El primer connector HTTP és el *listener,* que s'encarrega d'escoltar una IP amb port especificats, per tal d'acceptar o denegar l'entrada de missatges al flux.



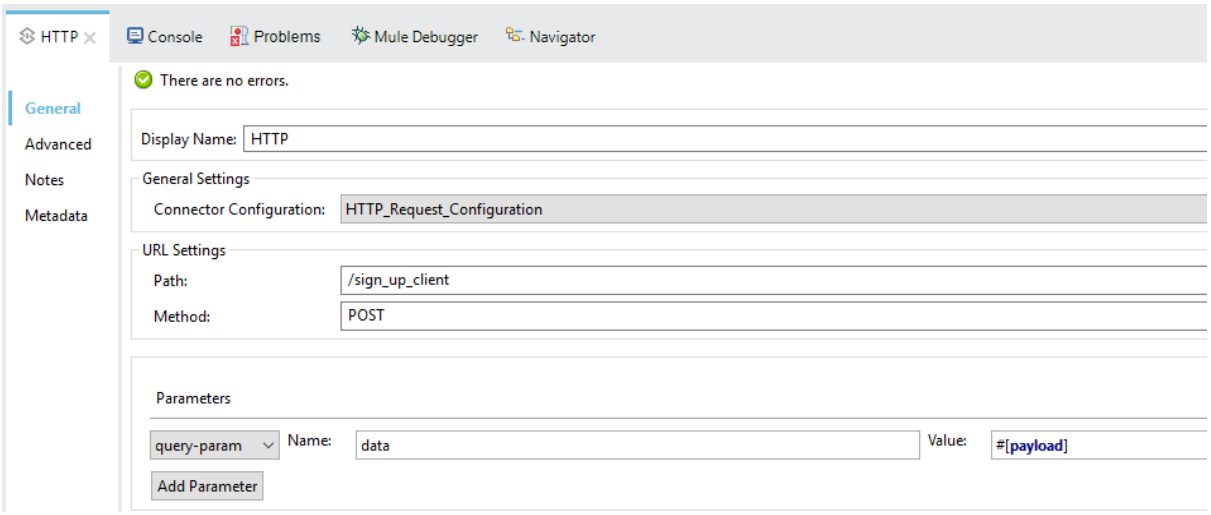
Il·lustració 16: exemple de configuració d'un connector HTTP *Listener* de Mule (part 1)



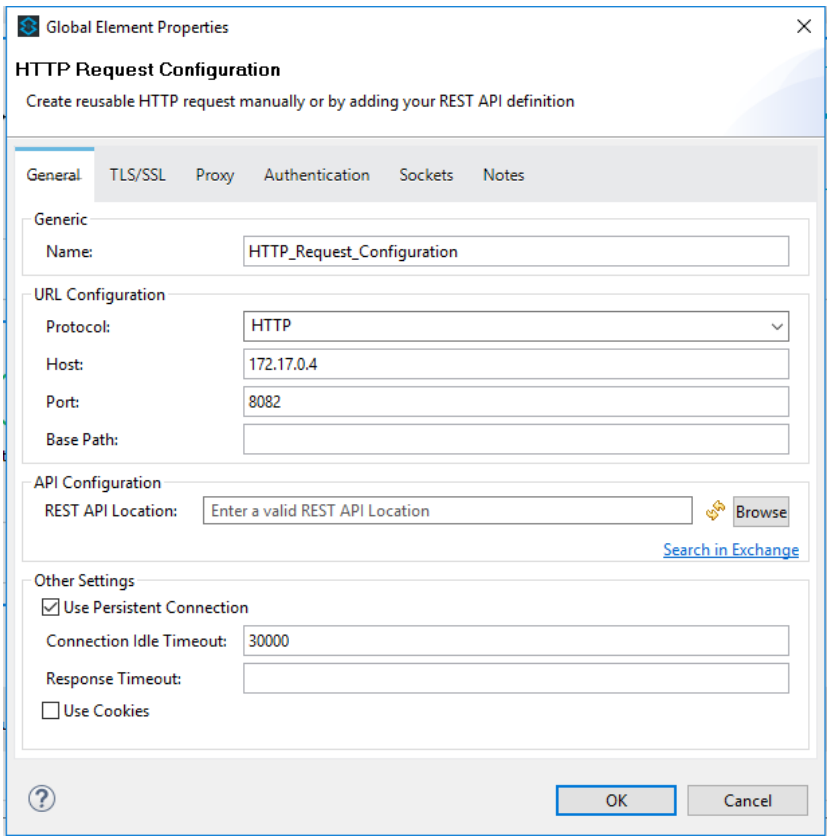
Il·lustració 17: exemple de configuració d'un connector HTTP *Listener* de Mule (part 2)

Si el connector es troba al final del flux, funcionarà com a *request*. S'encarregarà

d'enviar peticions a una IP i port determinant, adjuntant informació al missatge, en el cas que sigui adient. En el cas del micro-servei *sign_up_ms*, s'enviaran les dades de l'usuari que es vol donar d'alta a l'empresa.

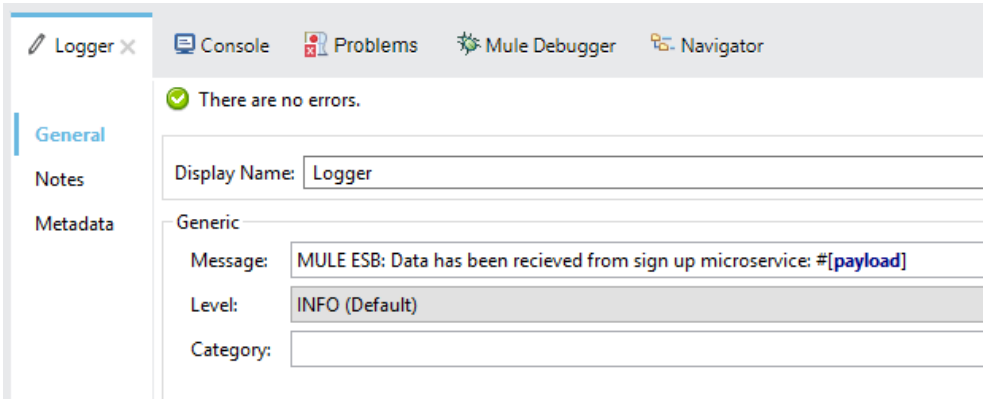


Il·lustració 18: exemple de configuració d'un connector HTTP *Request* de Mule (part 1)



Il·lustració 19: exemple de configuració d'un connector HTTP *Request* de Mule (part 2)

- **Logger:** genera registres sobre qualsevol cosa que escrivim en el connector. Aquests registres seran utilitzats més tard per les màquines de monitoratge, ja que aquests registres informen sobre l'estat del flux.



Il·lustració 20: exemple de configuració d'un connector *Logger* de Mule (part 1)

- **Object to [...]:** canvia el format de les dades del missatge d'entrada, independentment del seu format original. Els formats als quals pot ser transformat són: XML, Byte Array, JMSMessage, JSON, MIME, Message i String. En aquest cas hem utilitzat únicament les traduccions a JSON i String.

- **Java:** permet afegir un arxiu programat amb Java per tal de poder editar el missatge i transformar les seves dades de la manera que el desenvolupador desitgi. En el nostre cas, es transformaran les dades d'entrada per tal de formar una comanda SQL perquè la base de dades la reconegui i l'executi.

Un cop creat l'esquema de flux, falta comprovar que les comunicacions entre microserveis i Mule s'està efectuant correctament, i que les dades arriben i s'emmagatzemen dins la base de dades.

11.4. Monitoratge

Pel que fa al monitoratge, hem utilitzat programari de la companyia Elastic. Aquestes eines utilitzades són: ElasticSearch, Logstash, Kibana i Dockbeat. A continuació explicarem l'ús i implementació de cadascun d'aquests dins de l'arquitectura.

11.4.1. Logstash

Logstash s'utilitzarà per recollir tots els registres de les diferents màquines Docker. Aquests registres són els *console.log(...)* dels diferents microserveis i els Loggers de Mule.

Com ja hem dit, totes les eines de monitoratge estaran desplegades amb màquines Docker, amb l'excepció de Dockbeat. Per tal d'aconseguir aquestes màquines, s'han descarregat les imatges Docker del repositori oficial de la companyia Elastic. L'únic que ha calgut tenir en compte a l'hora de procedir amb el desplegament de les màquines han estat els paràmetres addicionals necessaris. En el cas de Logstash, hem hagut de vincular uns fitxers a directoris interns de la imatge Logstash, per tal d'obtenir el funcionament adient. Aquests fitxers són el *logstash.conf* i el *logstash.yml*.

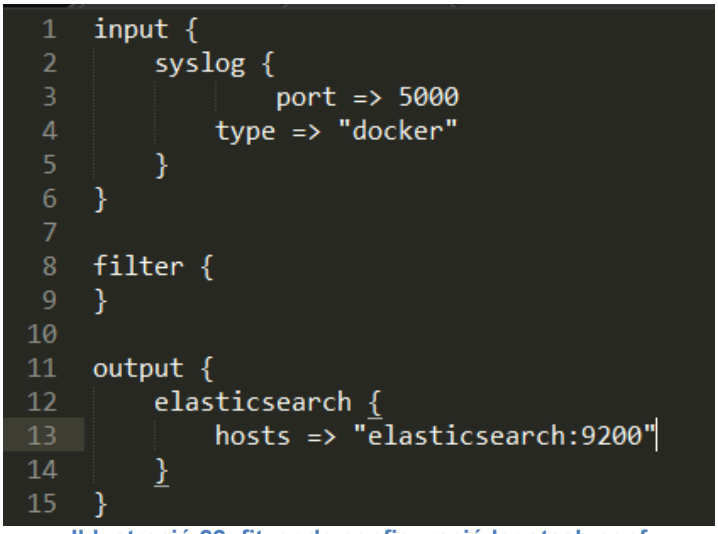
En el fitxer *logstash.yml*, s'hi troben els paràmetres necessaris per a poder fer funcionar Logstash. En aquest cas només hem hagut de configurar tres paràmetres bàsics:

```
1  http.host: "0.0.0.0"
2  path.config: /usr/share/logstash/pipeline
3
4  xpack.monitoring.enabled: false
```

Il·lustració 21: fitxer de configuració logstash.yml

El *http.host* especifica una IP del host on s'enllaçarà Logstash, el *path.config* especifica on es trobarà l'arxiu de configuració logstash.conf i *xpack.monitoring.enabled* és un paràmetre que ens permet determinar si s'utilitzarà o no X-Pack, que és una extensió força recent d'Elastic, que engloba totes les seves tecnologies en un sol paquet d'instal·lació. En aquest cas, l'hem desactivat, ja que no utilitzarem X-Pack en el projecte.

En el cas de logstash.conf, cal configurar també tres paràmetres, però aquests amb més detall. Aquests paràmetres són: *input*, *filter*, *output*.



Il·lustració 22: fitxer de configuració logstash.conf

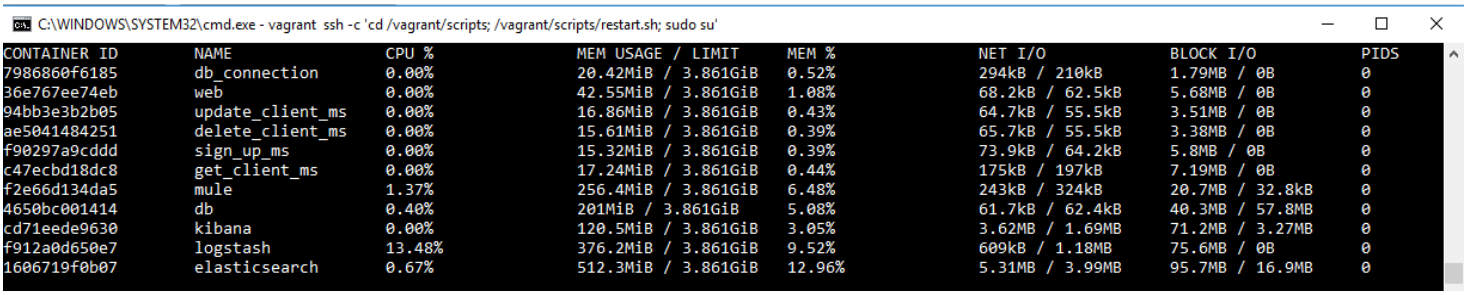
A l'*input* determinem quin serà el canal per on obtindrem els registres. En aquest cas, Docker ens facilita el treball, ja que té un *driver* que s'anomena *syslog* que permet a les màquines dirigir els seus registres cap aquest *driver* per després enviar-los a Logstash. Per aquest motiu, afegim el port 5000, que serà el port d'entrada de Logstash, per on el *syslog* enviarà totes les dades recollides, i com a tipus Docker, per determinar que seran màquines Docker.

Logstash ens permet filtrar aquests registres abans d'enviar-los a tercers. Aquest filtre s'aplica a l'apartat *filter*, però en el nostre cas no s'ha utilitzat, ja que els registres són prou específics i només filtrarem a Kibana, que explicarem en apartats posteriors.

Finalment, a *output*, s'especifica quin serà el destí d'aquests registres que hem recollit. En el nostre cas, només cal enviar totes les dades a ElasticSearch, ja que no fa falta difondre'l a cap altra màquina.

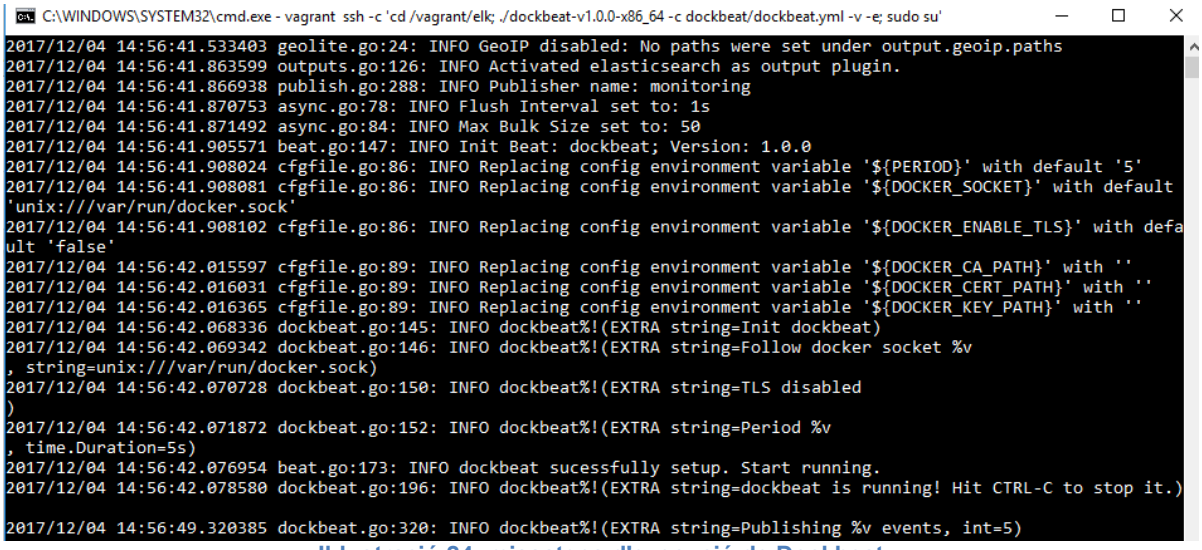
11.4.2. Dockbeat

Aquesta eina és una de les més recents de la companyia Elastic. Ens permet obtenir dades de rendiment i estat de les màquines Docker. Aquests estats els extreu de la comanda Docker: *docker stats*.



Il·lustració 23: docker stats

Per poder desplegar Dockbeat, només cal executar un fitxer anomenat *dockbeat-v1.0.0-x86_64* des d'un terminal. Un cop executat s'espera una sèrie de missatges com els següents:



Il·lustració 24: missatges d'execució de Dockbeat

Després ja només queda desplegar les màquines de monitoratge restants.

11.4.3. ElasticSearch

ElasticSearch, tal com indica el seu nom, ens permet efectuar qualsevol tipus de cerca dins els registres aportats per Dockbeat i Logstash. Per tal d'obtenir aquestes cerques, cal realitzar peticions a l'adreça d'ElasticSearch. Per mostrar un exemple, utilitzarem la comanda *curl*:

```
root@monitoring:/vagrant/scripts# curl -XPOST 'localhost:9200/logstash*/_search?pretty' -d '
{
"query": { "match": { "message": "[ERROR]" } }
}'
```

Il·lustració 25: exemple de cerca a ElasticSearch

En aquest cas, volem buscar totes les entrades de registres que tenen continguda dins el missatge la paraula *ERROR* i ElasticSearch ens retorna un llistat de registres amb error:

```
{
  "took" : 10,
  "timed_out" : false,
  "_shards" : {
    "total" : 10,
    "successful" : 10,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : 348,
    "max_score" : 5.655677,
    "hits" : [
      {
        "_index" : "logstash-2017.12.01",
        "_type" : "docker",
        "_id" : "AWARizzGAbsSmDeHqmB0",
        "_score" : 5.655677,
        "_source" : {
          "severity" : 0,
          "@timestamp" : "2017-12-01T10:07:05.899Z",
          "@version" : "1",
          "host" : "172.15.0.1",
          "message" : "<27>Dec  1 10:07:05 fae2ddf22807[918]: Error: DB CONNECTION MICROSERVICE: An error occured sig
ning up user: Error: ER_DUP_ENTRY: Duplicate entry 'user10' for key 'PRIMARY'\n",
          "type" : "docker",
          "priority" : 0,
          "facility" : 0,
          "severity_label" : "Emergency",
          "tags" : [
            "_grokparsefailure_sysloginput"
          ],
          "facility_label" : "kernel"
        }
      },
```


Il·lustració 26: exemple de resposta d'ElasticSearch

Aquests missatges d'error, són deguts a un script d'automatització de peticions, per aportar càrrega de treball dins l'arquitectura. Aquests errors són provocats voluntariament, per veure que l'arquitectura funciona com és degut. Com es pot veure a la imatge anterior, els errors indiquen que s'ha intentat donar d'alta un usuari anomenat *user10*, però aquest ja es trobava dins la base de dades. En el cas que no sortís cap error, significaria que la base de dades no està configurada correctament.

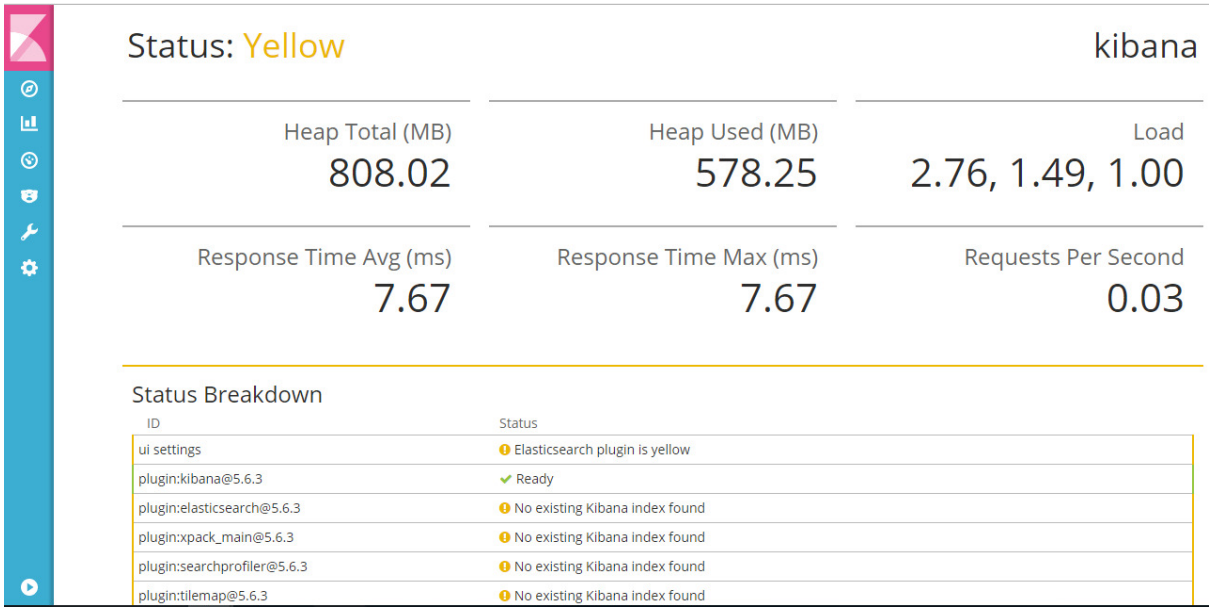
ElasticSearch és el nucli principal del sistema de monitoratge, però amb Kibana podem complementar aquesta potent tecnologia, per tal de fer més visible les nostres dades i poder avaluar el rendiment i funcionament de l'arquitectura, amb més facilitat.

11.4.4. Kibana

Amb Kibana podem representar tots els missatges i dades de cada container, a partir de gràfiques de diferents tipus, amb les quals es pot realitzar un estudi sobre l'estat de l'arquitectura.

A continuació explicarem en detall el seu funcionament i algunes de les particularitats de l'eina que hem utilitzat per desenvolupar el projecte.

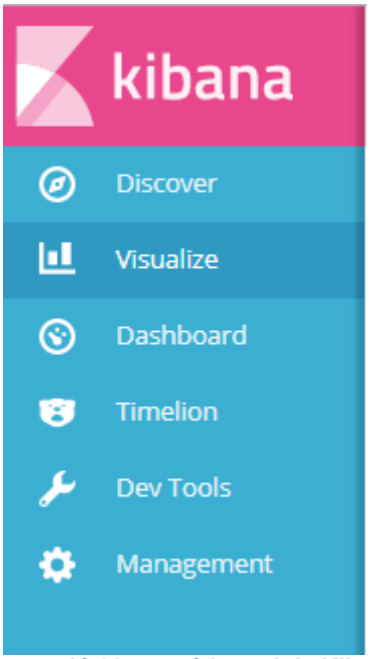
Primerament, un cop desplegat Kibana, juntament amb ElasticSearch i Logstash, accedim a l'adreça IP assignada en el Dockerfile.



Il·lustració 27: portal informatiu de Kibana

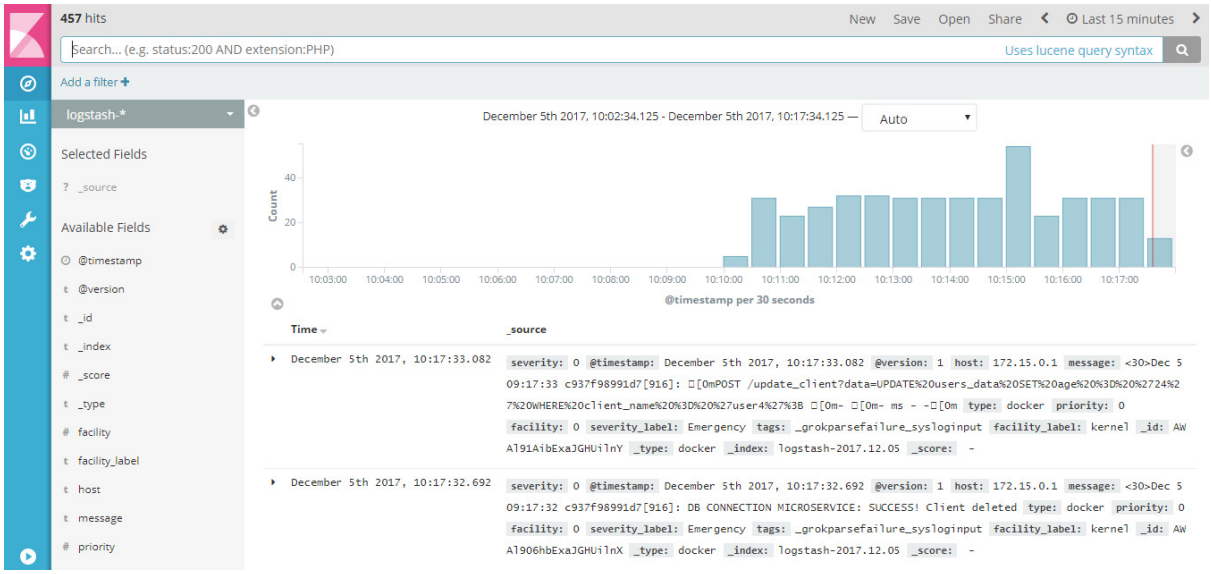
Un cop hem accedit a la IP, se'ns mostrarà una pàgina com la imatge anterior. En aquesta pàgina se'ns mostra l'estat actual de Kibana. Hi ha tres tipus d'estat: *Green, Yellow i Red*. *Green* indica que el sistema funciona correctament, *Yellow* indica que el sistema funciona correctament, però només té un sol node de processament, i *Red* significa que el sistema no funciona. En el nostre cas, ja va bé que estigui en estat *Yellow*, ja que el volum de treball és força relatiu i no és necessari tenir més d'un node de CPU.

A la imatge també podem veure un menú de navegació lateral amb diferents opcions:



Il·lustració 28: menú lateral de Kibana

- **Discover:** es mostra una gràfica amb el nombre de registres obtinguts durant un cert transcurs de temps. Sota d'aquesta gràfica es mostren els últims registres amb els seus camps pertinents. Al lateral esquerre es mostren els camps dels registres i ens permet seleccionar els camps desitjats per tal de filtrar registres.



Il·lustració 29: discover de Kibana

- **Visualize:** ens permet crear gràfiques a partir d'acotar aquestes amb els camps dels registres que es vulguin comparar.

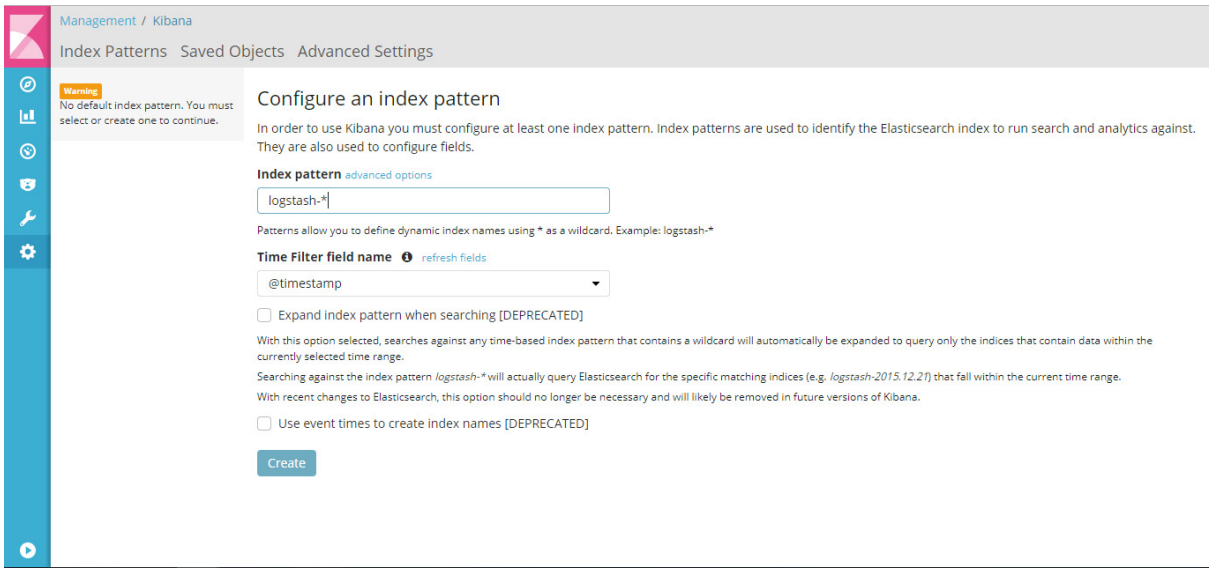
- **Dashboard:** se'ns mostra una nova vista on podem afegir els diferents elements creats a *Visualize*, per tal de tenir el tipus de monitoratge desitjat.

- **Timelion:** Es mostra la mateixa gràfica que a *Discover*, però amb les dades globals sense acotar.

- **Dev Tools:** permet a desenvolupadors modificar paràmetres del propi Kibana.

- **Management:** ens permet comprovar i modificar els ajustos del sistema.

El primer que cal fer per iniciar el correcte funcionament de Kibana, cal vincular els registres de Logstash i de Dockbeat al Kibana. Per fer-ho, cal que ens dirigim a la secció de *Management* (l'engranatge del menú de navegació lateral), i un cop allà seleccionem la secció *Index Pattern* i al botó *create new index*.

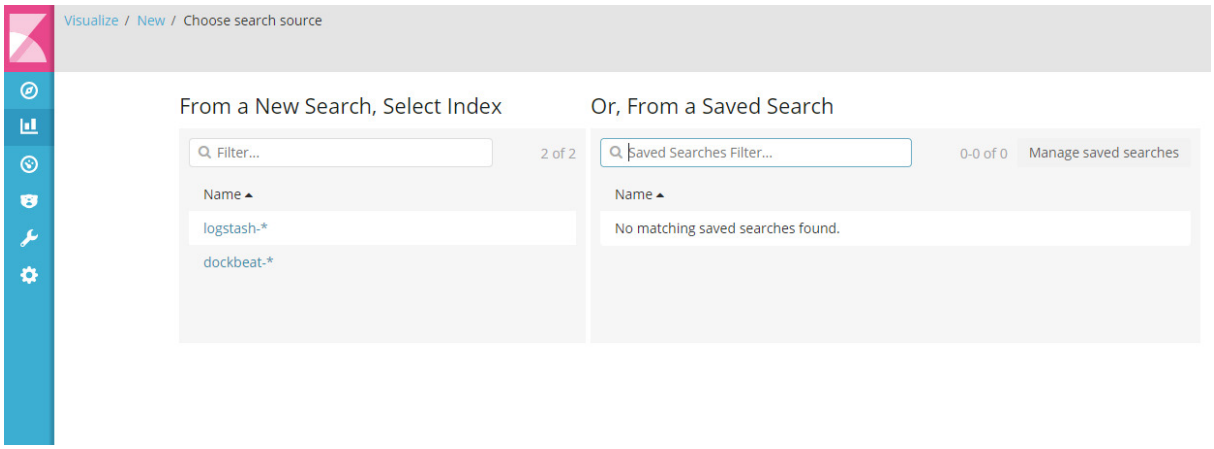


Il·lustració 30: configuració de l'*index pattern*

Per tal d'afegir els registres de les màquines al Kibana, cal afegir els noms de les màquines acompanyats de "-*". És a dir, cal afegir logstash-* i dockbeat-*. Un cop creats els dos índexs, el Kibana ja pot mostrar i diferenciar els missatges dels registres a les diferents pestanyes esmentades anteriorment: Discover, Visualize, Dashboard, Timelion.

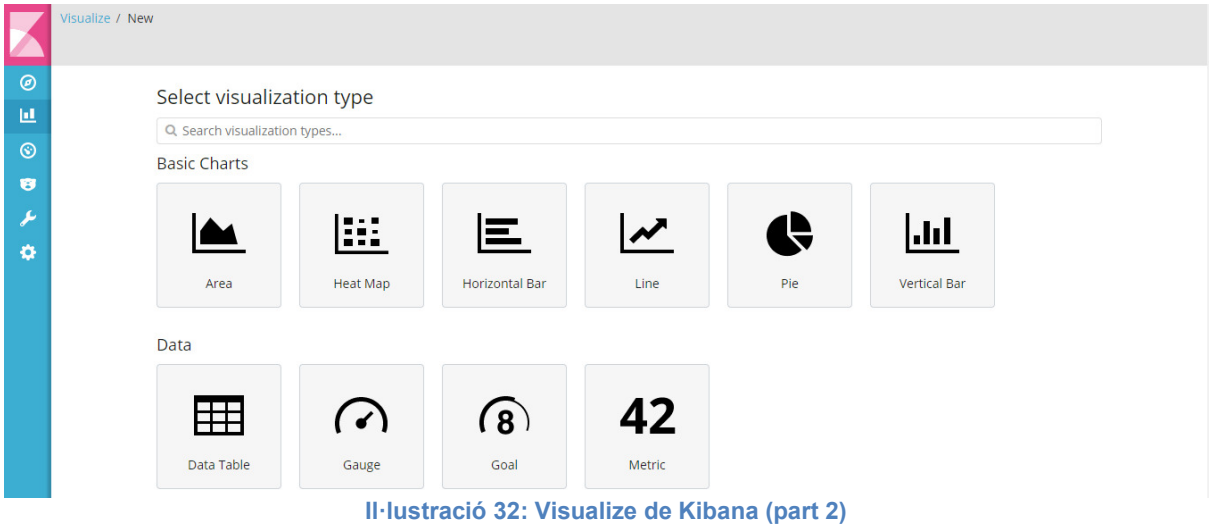
En el nostre projecte el que volem és avaluar l'estat en el qual es troba cada màquina i els missatges que transmeten sobre el funcionament del sistema. Per aconseguir-ho, hem creat diferents gràfiques i components que després hem afegit al *Dashboard*. A continuació, mostrarem i explicarem un exemple del procés realitzat per crear aquests elements.

Una de les gràfiques realitzades és la de control de memòria de disc. Aquesta gràfica l'hem creat a partir d'un gràfic de barres vertical. Abans d'escollir el tipus de gràfic, Kibana necessita saber a partir de quin índex es començarà a treballar.



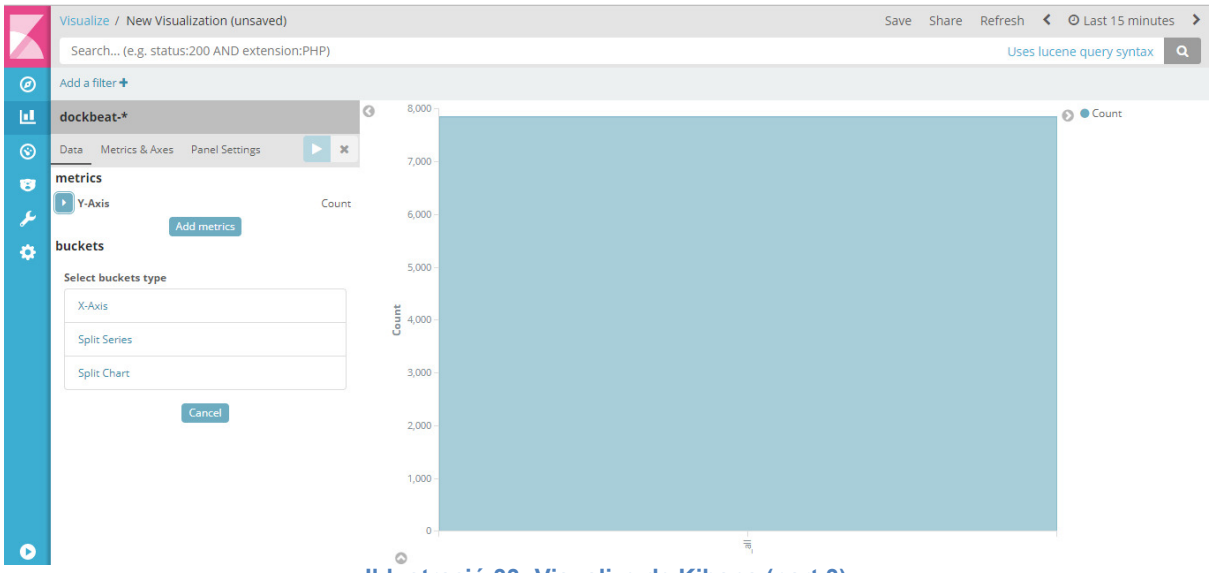
Il·lustració 31: Visualize de Kibana

Degut que les dades que volem mostrar, són dades de sistema, escollirem dockbeat-* i se'ns mostrarà la finestra de selecció de tipus de gràfic que mostrem a continuació:



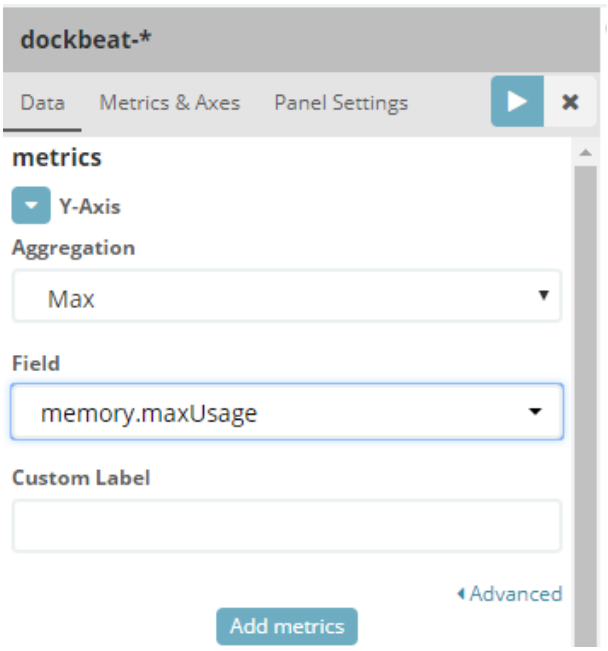
Il·lustració 32: Visualize de Kibana (part 2)

En aquest cas, com a exemple, voldrem realitzar un gràfic sobre l'espai ocupat de memòria de cada màquina. Per poder tenir un gràfic amb el qual les dades es reflecteixin apropiadament, hem escollit el gràfic de barres verticals (*Vertical Bar*).

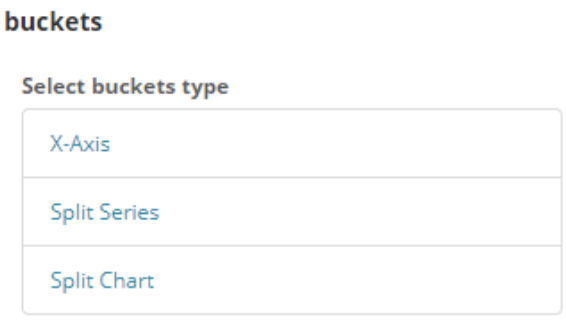


Il·lustració 33: Visualize de Kibana (part 3)

El primer que se'ns mostra a l'hora d'escollir el gràfic, és una sola barra vertical, en la que s'indica quants missatges de registre ha obtingut Kibana de Dockbeat, des que s'ha desplegat la màquina. Per tal de poder obtenir les dades pertinents, cal modificar els paràmetres de *Data* en el menú lateral.



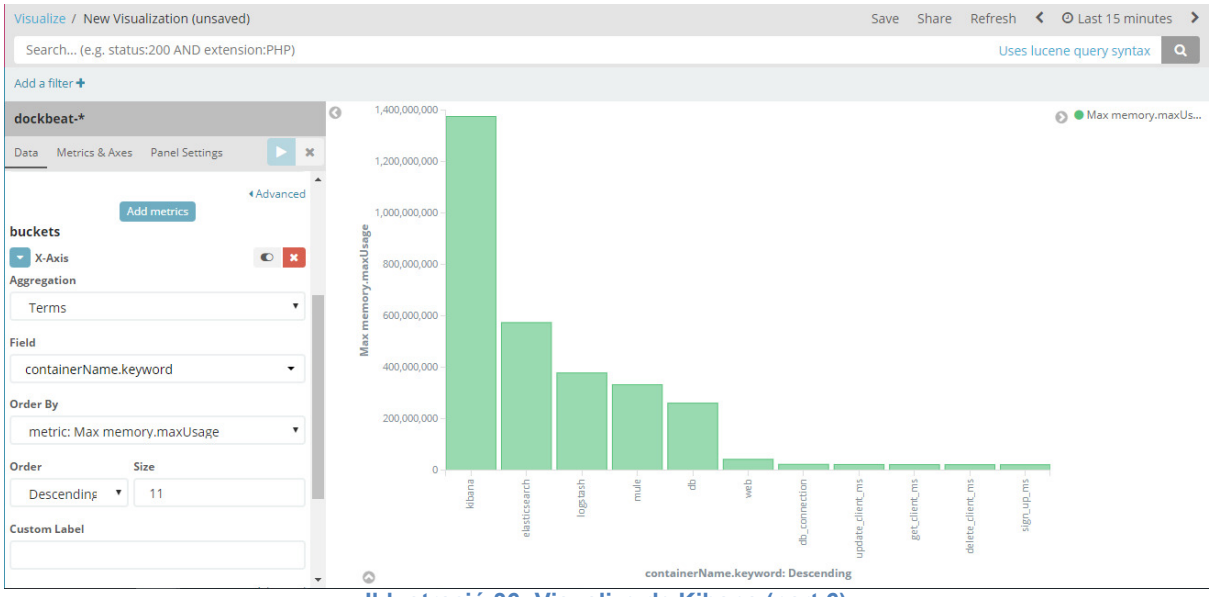
Il·lustració 34: Visualize de Kibana (part 4)



Il·lustració 35: Visualize de Kibana (part 5)

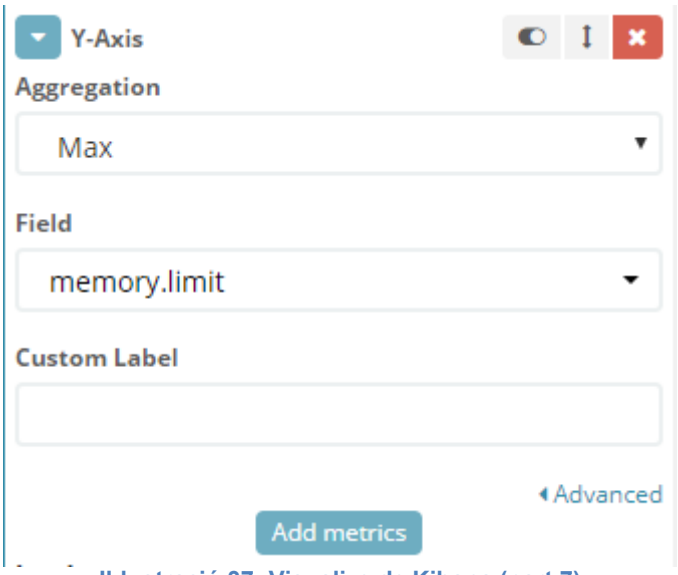
Primerament, cal modificar la mostra representada per l'eix Y. En comptes de tenir el comptador (*Count*), el que volem és el volum de dades de memòria interna, que en aquest cas DockBeat ens proporciona el paràmetre *memory.maxUsage*. Com a mètode d'agregació, hem escollit *Max*, ja que el que volem comprovar amb aquest gràfic és que les màquines no estiguin saturades per insuficiència d'espai de memòria.

A més a més, cal afegir una mostra a l'eix X, ja que actualment no s'està representant res. Per fer-ho, ens hem de dirigir a la secció inferior *buckets* i seleccionar *X-Axis*. Un cop seleccionat, ens apareixerà una nova secció idèntica a la de l'eix Y, però en aquest cas, el que desitgem és diferenciar els volums de dades de cada màquina. DockBeat ens proporciona el nom de cada màquina, i això ens permet obtenir aquesta diferència, afegint aquests noms a l'eix X. Així doncs, seleccionem com a mètode d'agregació *Terms* per indicar que les dades no seran dades numèriques, sinó els noms que volem mostrar, que es troben a containerName.keyword, proporcionat per DockBeat.



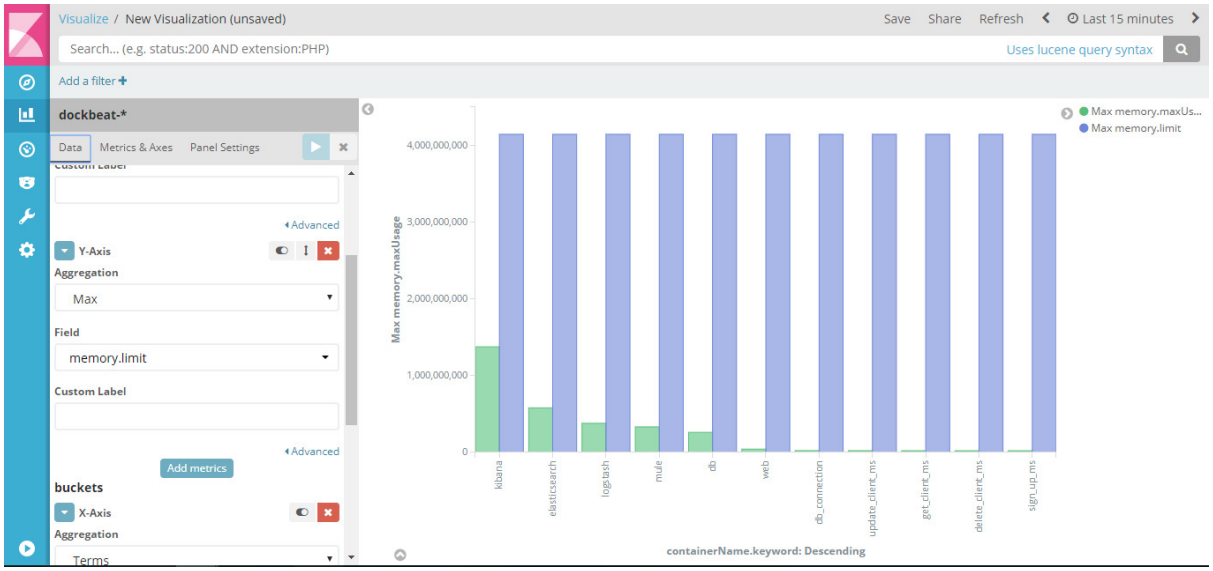
Il·lustració 36: Visualize de Kibana (part 6)

D'aquesta manera obtenim el màxim volum de dades ocupat per cada màquina durant el temps transcorregut des que Kibana s'ha desplegat. Però, falta saber quina és la màxima capacitat de cada màquina.



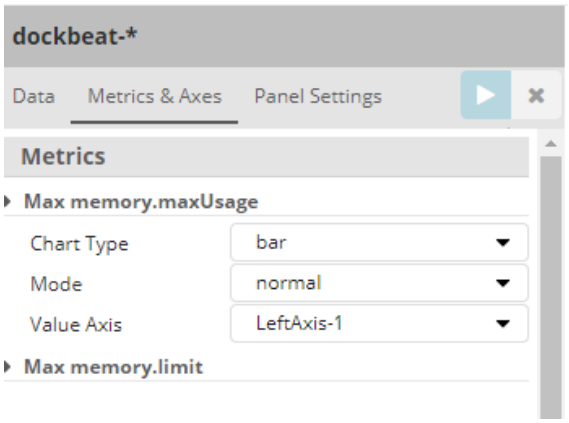
Il·lustració 37: Visualize de Kibana (part 7)

Així doncs, cal afegir una altra mostra en l'eix Y. Per afegir-la, cal seleccionar el botó que hi ha sota *Y-Axis*: *Add metrics*. Es mostrarà una altra secció idèntica a la del primer eix Y. En aquesta secció afegirem el paràmetre *memory.limit* de DockBeat, amb el mateix mètode d'agregació *Max*.

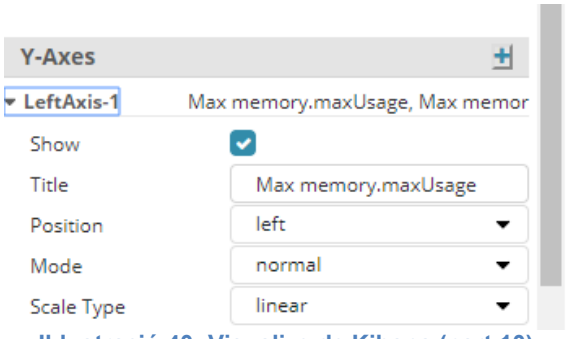


Il·lustració 38: Visualize de Kibana (part 8)

Ara ja tenim constància de la memòria ocupada en cada màquina. Però si es desitja, a l'apartat *Metrics & Axes* es poden realitzar retocs que ens permetin millorar les dades representades, en l'aspecte visual.



Il·lustració 39: Visualize de Kibana (part 9)

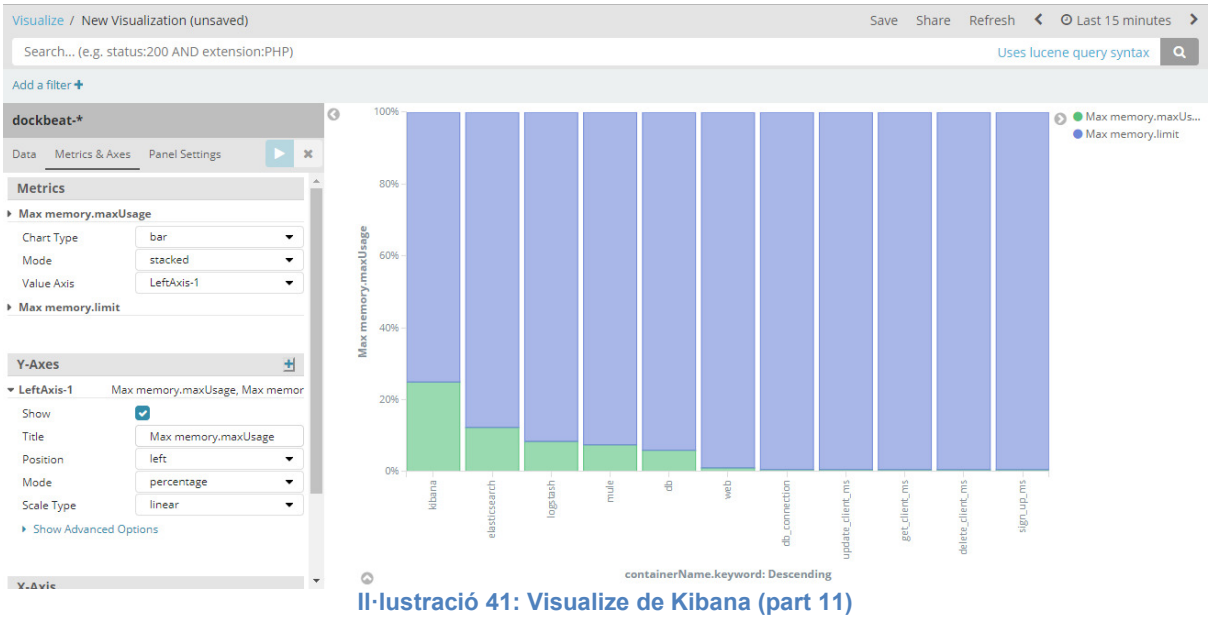


Il·lustració 40: Visualize de Kibana (part 10)

En aquest cas, en estar representant dades d'emmagatzematge, seria interessant canviar el mode de representació per valor, a un mode de representació per percentatge. Per aconseguir-ho, cal modificar el paràmatre *Mode* del *Y-Axes*, de *normal* a *percentage*.

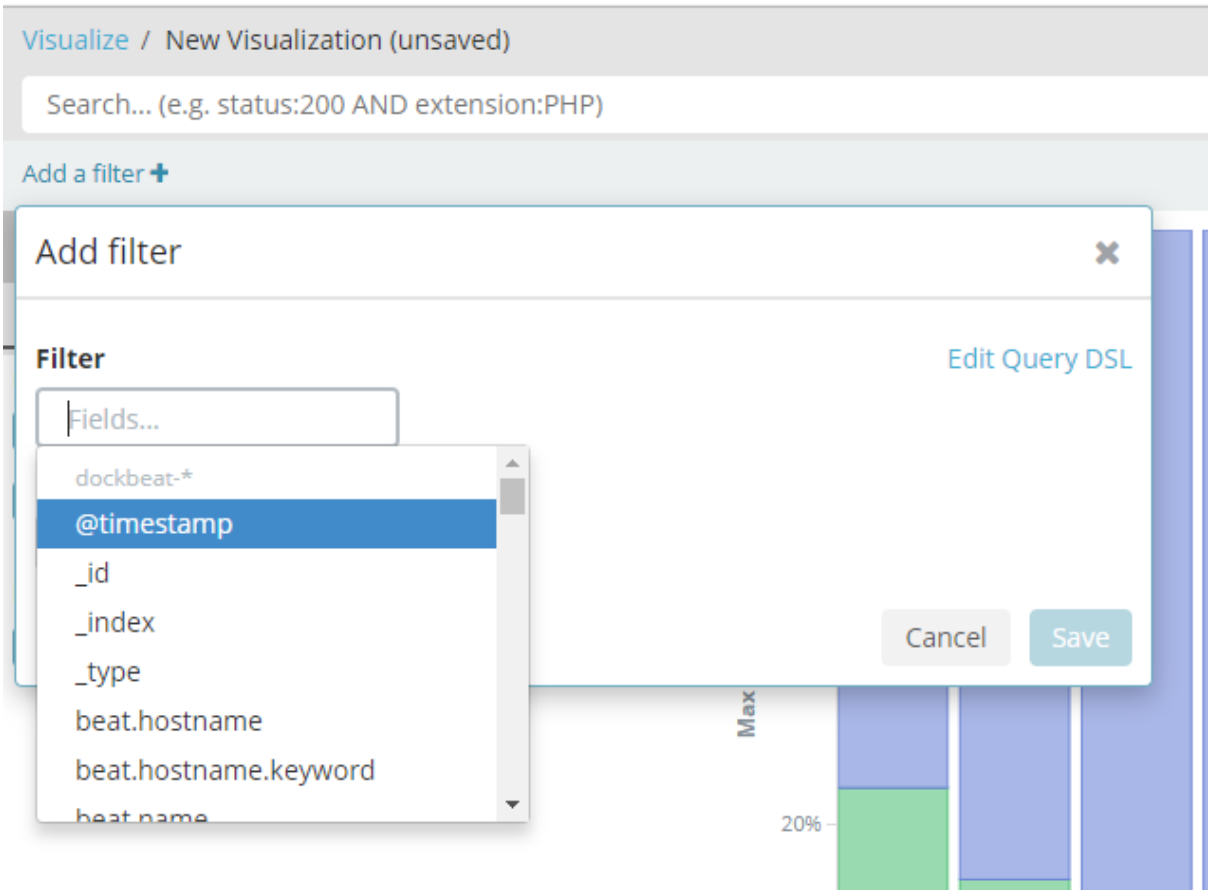
Un altre canvi que milloraria l'enteniment de les dades, seria tenir les columnes d'ambdues mostres de l'eix Y, juntes en una sola columna. Per obtenir aquesta representació, cal que modifiquem el camp *Mode* de *Metrics*, canviant el valor *normal* per *stacked*.

D'aquesta manera, obtenim el gràfic següent:



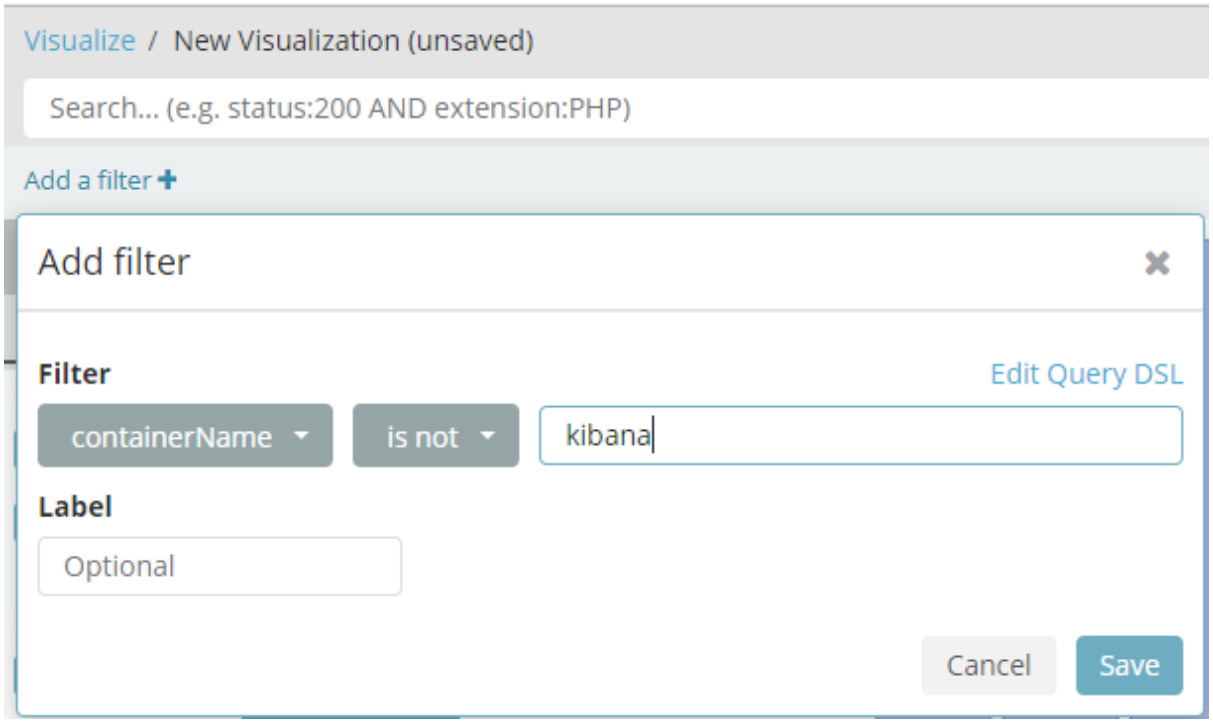
Il·lustració 41: Visualize de Kibana (part 11)

Per últim, vam aplicar filtres al gràfic pel fet que hi ha tres màquines representades que no ens interessa mostrar, ja que no formen part directa de l'arquitectura. Aquestes màquines són les pròpies del monitoratge: Kibana, ElasticSearch i Logstash.



Il·lustració 42: Visualize de Kibana (part 12)

Per aplicar el filtre, cal que ens dirigim a la barra superior de cerca, i just a sota, seleccionar el camp *Add a filter*. Se'ns desplegarà una nova finestra amb un camp *Filter*, on haurem d'escollir entre un llistat de paràmetres pertinents de Dockbeat.



Il·lustració 43: Visualize de Kibana (part 13)

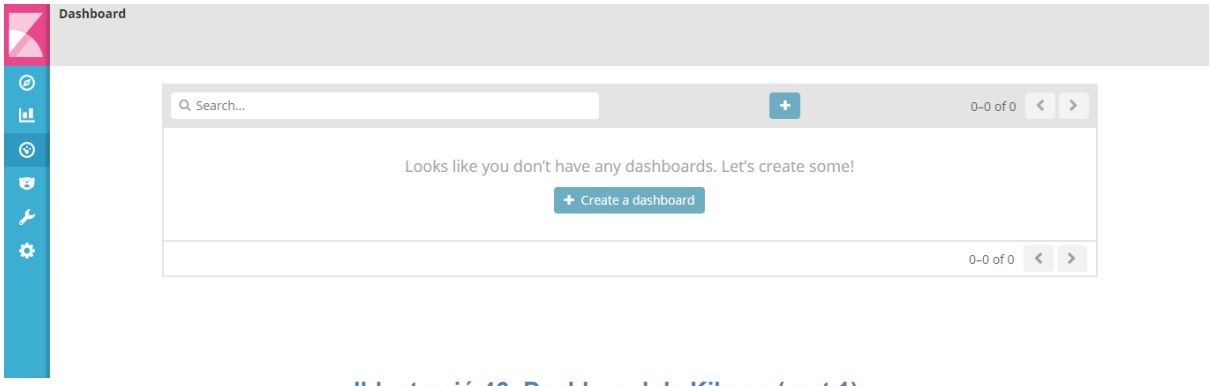
Per tal d'eliminar del gràfic les màquines de monitoratge hem filtrat per nom les màquines. El filtre afegit indica que les màquines que s'anomenin Kibana, ElasticSearch i Logstash, no estiguin representades en el gràfic. D'aquesta manera obtenim el gràfic final següent:

Il·lustració 44: Visualize de Kibana (part 14)

Ja només queda guardar el gràfic per poder afegir-lo al *dashboard*. Per guardar el gràfic hem de seleccionar l'opció que es troba a dalt de tot a la dreta *Save*, i guardar el gràfic amb el nom que es desitgi.

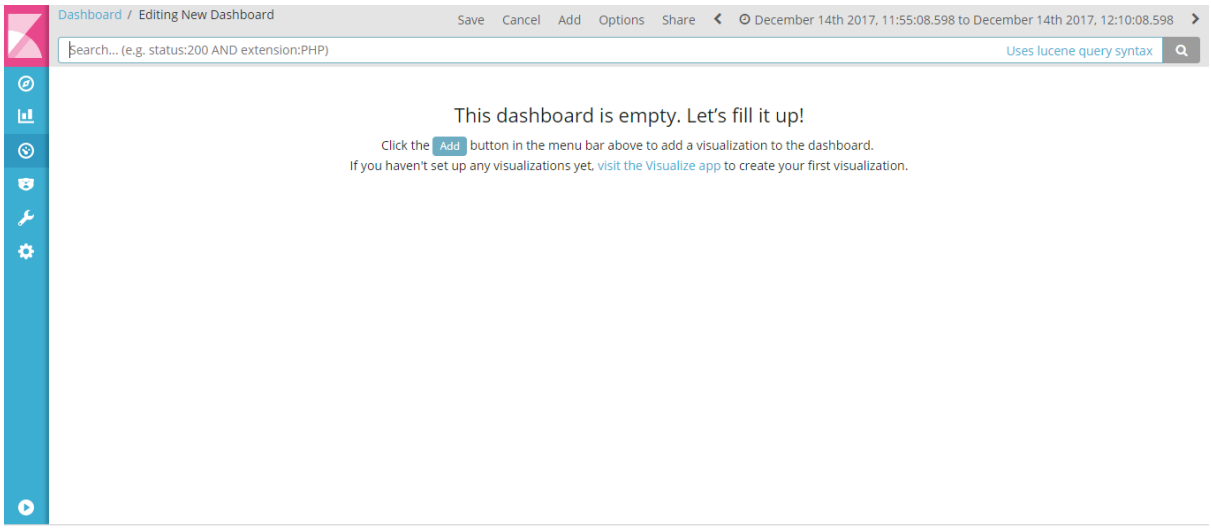
Il·lustració 45: Visualize de Kibana (part 15)

Un cop realitzats els gràfics, ja només queda crear un taulell on representar tots els gràfics que desitgem. Per a fer-ho haurem de dirigir-nos a la secció *Dashboard*.



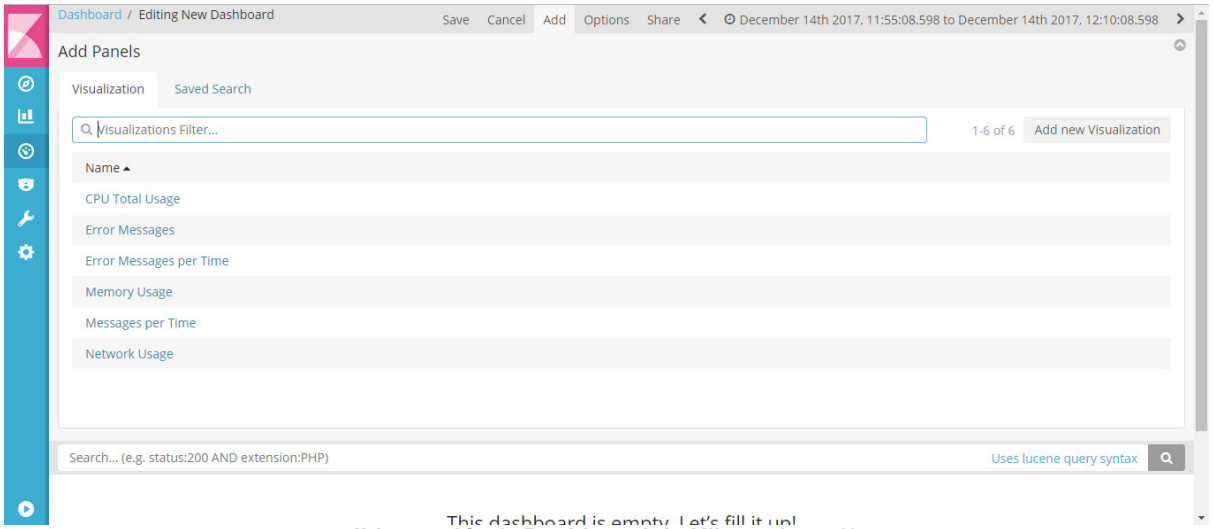
Il·lustració 46: Dashboard de Kibana (part 1)

En aquesta secció, seleccionem *Create a dashboard*, o bé seleccionem el botó "+" que hi ha a la dreta de la barra de cerca.



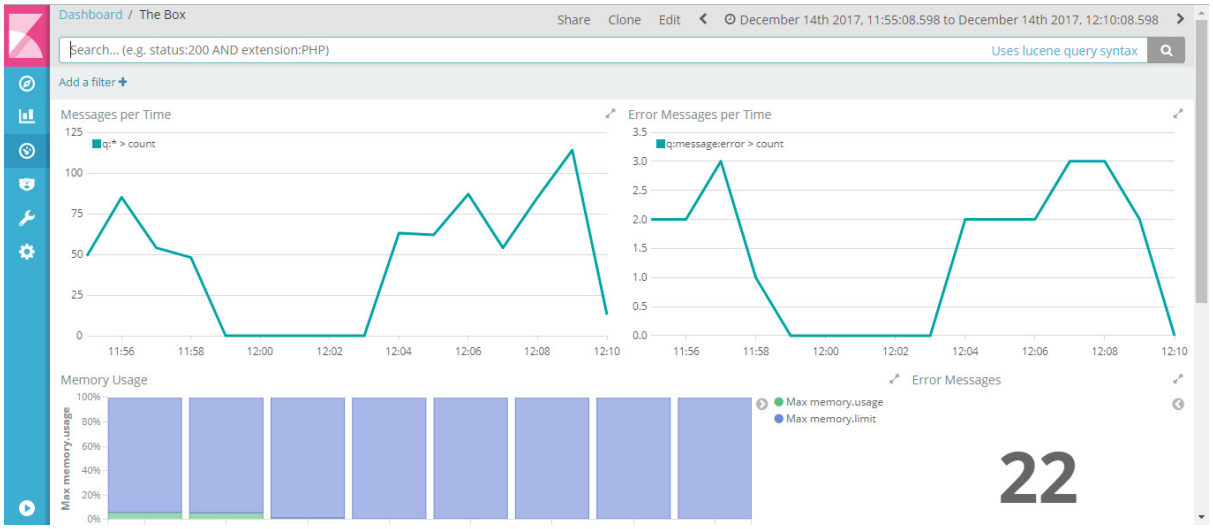
Il·lustració 47: Dashboard de Kibana (part 2)

Se'ns mostrarà una pàgina buida, on clicant el botó *Add* podrem començar a afegir les gràfiques que hàgim creat anteriorment.



Il·lustració 48: Dashboard de Kibana (part 3)

Escollim qualsevol gràfica del llistat i directament, l'aplicació la representa en el dashboard. Un cop inserida la gràfica, tenim total llibertat per moure-la pel taulell, canviar les seves dimensions i, fins i tot, editar la gràfica, encaminant-nos directament a la secció Visualize.



Il·lustració 49: Dashboard de Kibana (part 4)

12. Funcionament de l'arquitectura

En aquest apartat explicarem els passos a realitzar per tal de desplegar l'arquitectura. Primer de tot, cal que executem el sistema Vagrant. En el cas de no tenir Vagrant instal·lat en el sistema, cal descarregar l'executable d'instal·lació del seu lloc web: https://www.vagrantup.com/downloads.html. Un cop instal·lat, per facilitar el desplegament, s'ha creat un *script* per tal d'automatitzar l'execució del Vagrant. L'script, anomenat *init.bat*, executa la comanda *vagrant up*, per iniciar el sistema virtual amb totes

les característiques addicionals que s'han afegit al Vagrantfile. Un cop iniciat el sistema, es tanca la finestra de comandes des d'on s'ha iniciat Vagrant i s'obren 4 finestres noves de comandes, cadascuna amb un propòsit específic.

La primera finestra que s'executarà, entrarà dins l'entorn Vagrant i s'encarregarà de desplegar totes les màquines de l'entorn de monitoratge. Aquestes màquines han de ser les primeres a ser executades, ja que els *containers* que pertanyen a l'arquitectura depenen d'aquestes per poder desplegar-se.

A la següent finestra s'executaran les comandes pertinents per tal de poder desplegar les màquines de l'arquitectura de microserveis. Quan aquestes màquines s'hagin desplegat, el sistema estarà llest per fer les proves pertinents. A més a més cada màquina enviarà constantment registres de dades cap a les màquines de monitoratge.

La següent finestra s'hi executarà Dockbeat. A l'executar-se la comanda, el servidor començarà a recollir dades de rendiment de les màquines Docker i a enviar aquestes dades cap a la màquina d'ElasticSearch, per poder veure les dades plasmades en gràfiques a partir de Kibana.

L'última finestra es basa en l'execució d'un altre *script*, que es basa en un codi en bucle indefinit, el qual executa peticions consecutivament a cada micro-servei. D'aquesta manera podem avaluar la càrrega de treball de cada micro-servei i veurem si amb el pas del temps el sistema funciona correctament, o bé si es produeix alguna anomalia a l'arquitectura.

Finalment, un cop l'*script* s'ha acabat d'executar, el sistema està llest per realitzar-hi les proves pertinents. Accedint a la direcció del portal web (127.0.0.1:8081), podem realitzar les peticions que desitgem per tal de fer proves amb més comoditat. Si es desitja, també es poden fer peticions *curl* o programar un nou *script* per realitzar noves peticions. Per tal de poder veure les dades obtingudes de les peticions realitzades, només cal que accedim a la direcció del servidor Kibana (10.110.0.210:5601) i crear les visualitzacions i els *dashboards* desitjats.

13. Conclusió

Hem pogut comprovar que les arquitectures de les Tecnologies de la Informació no són immutables. Aquestes tendeixen a evolucionar a noves infraestructures, sigui afegint noves tecnologies, o millorant la seva topologia, o bé utilitzant elements de diferents arquitectures, per tal de resoldre les seves deficiències i obtenir un sistema més escalable i eficient.

En el cas d'aquest projecte, s'ha decidit afegir un element aliè a l'arquitectura de microserveis. Aquest element és l'ESB que, tal com ja s'ha esmentat, ens permetrà obtenir una orquestració dels microserveis.

Des de l'aparició del ESB, el seu ús ha estat orientat a l'intercanvi d'informació entre serveis, però no hi ha constància de poder oferir compatibilitat entre microserveis. Amb aquest projecte hem pogut determinar que és possible establir connexions entre microserveis i ESB amb el comportament esperat i amb les finalitats establertes. A partir de l'ESB, el nostre sistema és capaç d'assolir nous reptes que amb una arquitectura de microserveis seria inviable. La nostra arquitectura ens permet: orquestrar els fluxos d'informació, transformar el format i contingut dels missatges, control d'errors entre client i servidor i oferir un entorn molt més compatible amb les tecnologies aplicades a l'arquitectura.

Pel que fa a les metes establertes a l'inici d'aquest projecte, es donen totes per finalitzades.

La primera meta a assolir consistia a obtenir un sistema preconfigurat, intuïtiu i portable. Amb Vagrant s'ha pogut complir aquesta meta, degut que tota l'arquitectura es manté constant a l'entorn virtual de Vagrant. És un entorn intuïtiu i a la disposició de qualsevol usuari, només cal executar l'script *init.bat* per iniciar al sistema, i accedir al portal web, perquè qualsevol usuari pugui fer-ne ús. Per últim, és un entorn portable, ja que només cal carregar la carpeta del projecte a qualsevol ordinador.
La segona meta coincidia amb l'objectiu principal del projecte: determinar que és possible una orquestració dels microserveis. Com s'ha pogut veure en el projecte, aquesta fase s'ha complert amb satisfacció.

La tercera meta consistia a obtenir un sistema format per microserveis, un ESB i altres serveis. Tal com s'ha pogut veure a l'esquema de l'arquitectura, s'ha pogut aconseguir crear aquesta topologia, ja que en el nostre sistema consta de quatre microserveis, l'ESB, una base de dades, el portal web i tota la resta màquines encarregades del monitoratge.

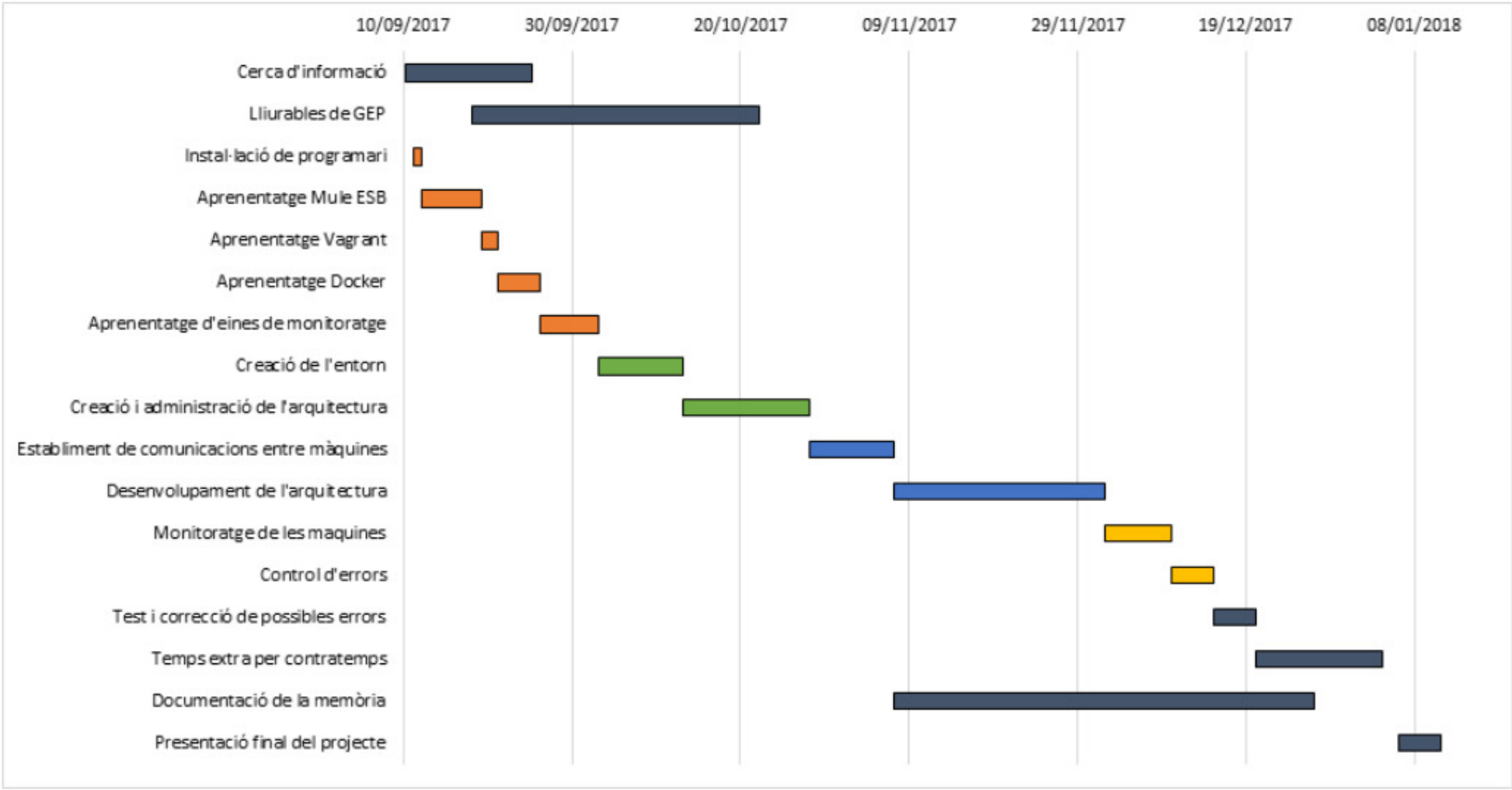
Ara bé, l'última meta no ens ha proporcionat els resultats esperats. Aquesta meta es basava a constatar que l'arquitectura d'aquest projecte era més eficient que una arquitectura de microserveis. Per poder fer la prova, es va crear la mateixa arquitectura que teníem però sense l'ESB. El cas és que sense l'ESB, l'arquitectura de microserveis era més eficient que no pas l'arquitectura que s'ha realitzat en el projecte, ja que no havia de passar tota la informació per l'ESB, que feia d'intermediari entre els microserveis i la base de dades, sinó que totes les dades es transmetien directament. Tot i això, la diferencia de rendiment i temps de finalització entre ambdues arquitectures era mínima, i la finalitat de la nostra arquitectura era tenir un sistema més escalable, que oferís noves característiques i possibilitats. En aquest sentit, la nostra arquitectura ofereix millores vers la seva implementació genèrica, i per tant l'eficiència passa a un segon pla.

Així doncs, havent complert l'objectiu i les metes proposades, aquí finalitza el nostre projecte.

14. Annex

Tasques			
Data d'inici	**Data final**	**Descripció**	**Durada (dies)**
11/09/2017	26/09/2017	Cerca d'informació	15
19/09/2017	23/10/2017	Lliurables de GEP	34
12/09/2017	13/09/2017	Instal·lació de programari	1
13/09/2017	20/09/2017	Aprenentatge Mule ESB	7
20/09/2017	22/09/2017	Aprenentatge Vagrant	2
22/09/2017	27/09/2017	Aprenentatge Docker	5
27/09/2017	04/10/2017	Aprenentatge d'eines de monitoratge	7
04/10/2017	14/10/2017	Creació de l'entorn	10
14/10/2017	29/10/2017	Creació i administració de l'arquitectura	15
29/10/2017	08/11/2017	Establiment de comunicacions entre màquines	10
08/11/2017	03/12/2017	Desenvolupament de l'arquitectura	25
03/12/2017	11/12/2017	Monitoratge de les maquines	8
11/12/2017	16/12/2017	Control d'errors	5
16/12/2017	21/12/2017	Test i correcció de possibles errors	5
21/12/2017	05/01/2018	Temps extra per contratemps	15
08/11/2017	28/12/2017	Documentació de la memòria	50
07/01/2018	12/01/2018	Presentació final del projecte	5

Taula 10: taula de tasques



Il·lustració 50: diagrama de Gantt

15. Referències

[1] Newman, Samuel. 2015. Building microservices: designing fine-grained microservices Sebastopol (CA): OReilly Media. ISBN 9781491950357.

[2] Pàgina oficial de Mule. [En línia]
http://www.mulesoft.com/ [Consulta: 25 de Setembre de 2017]

[3] Pàgina oficial de Vagrant. [En línia]
https://www.vagrantup.com/ [Consulta: 25 de Setembre de 2017]

[4] Pàgina oficial de Docker. [En línia]
https://www.docker.com/ [Consulta: 25 de Setembre de 2017]

[5] Using Containers to Build a Microservices Architecture. Medium. [En línia]
https://medium.com/aws-activate-startup-blog/using-containers-to-build-amicroservices-architecture-6e1b8bacb7d1 [Consulta: 25 de Setembre de 2017]

[6] Docker...Containers, Microservices and Orchestrating the Whole Symphony - DZone DevOps. [En línia]
https://dzone.com/articles/dockercontainers-microservices [Consulta: 25 de Setembre de 2017]

[7] Tien K. Microservices with Mule. MuleSoft Blog. [En línia]
https://blogs.mulesoft.com/dev/microservices-dev/microservices-with-mule/ [Consulta: 25 de Setembre de 2017]

[8] What is Mule ESB?. MuleSoft. [En línia]
https://www.mulesoft.com/resources/esb/what-mule-esb [Consulta: 25 de Setembre de 2017]

[9] Tien K. Microservices versus ESB. MuleSoft Blog. [En línia]
https://blogs.mulesoft.com/dev/microservices-dev/microservices-versus-esb/ [Consulta: 25 de Setembre de 2017]

[10] Hcltech. Everything you need to know about Enterprise Service Bus (ESB) [En línia]
https://www.hcltech.com/blogs/everything-you-need-know-about-enterpriseservice-bus-esb [Consulta: 25 de Setembre de 2017] 29

[11] Pàgina oficial de Kibana. [En línia]
http://www.elastic.co/products/kibana [Consulta: 25 de Setembre de 2017]

[12] Pàgina oficial d'ElasticSearch. [En línia]
http://www.elastic.co/products/elasticsearch [Consulta: 25 de Setembre de 2017]
[13] Create a Project Plan with Free Online Gantt Chart Software | GanttPRO. GanttPRO. 2017. [En línia]
https://app.ganttpro.com/ [Consulta: 2 d'Octubre de 2017]
[14] Ratolí Targus AMU30EUZ. Misco. [En línia]
http://www.misco.co.uk/product/Q180515/Targus-3-Button-USB-PS2-WiredMouse
[Consulta: 9 d'Octubre de 2017]
[15] Wired Keyboard 200 | Microsoft Accessories. Microsoftcom. [En línia]
http://www.microsoft.com/accessories/en-gb/products/keyboards/wiredkeyboard-200/jwd-00032 [Consulta: 9 d'Octubre de 2017]
[16] Dell Latitude E5570 Review. Laptopmagcom. [En línia]
http://www.laptopmag.com/reviews/laptops/dell-latitude-e5570 [Consulta: 9 d'Octubre de 2017]
[17] Monitor A. Acer V223HQV 21.5inch 5ms Dsub LCD Widescreen Black TFT Monitor [ET.WV3HE.017] from Overclock.co.uk. Overclockcouk. [En línia]
http://www.overclock.co.uk/product/Acer-V223HQV-215inch-5ms-Dsub-LCDWidescreen-Black-TFT-Monitor_41971.html [Consulta: 9 d'Octubre de 2017]
[18] Power Energy Consumption. Mypowerconsumptioncom. [En línia]
http://www.mypowerconsumption.com/product/ [Consulta: 9 d'Octubre de 2017]
[19] Calculo de las emisiones de CO2. Arbolizaes. [En línia]
http://arboliza.es/compensar-co2/calculo-co2.html [Consulta: 9 d'Octubre de 2017]