DESPLEGAMENT D'APLICACIONS WEB AL NÚVOL

TREBALL FINAL DE GRAU

Autor: Francesc Xavier Puig Ventosa

Tutor: José Manuel Castillo Pedrosa

Àrea: Administració de xarxes i sistemes operatius

Grau en Enginyeria Informàtica

Universitat Oberta Catalunya

Gener 2019

Resum

El present Treball Final de Grau tracta d'explicar els diferents components que formen l'arquitectura d'una aplicació web, des dels llenguatges fins a l'arquitectura dissenyada. Ens hem basat en les noves plataformes de serveis d'internet, principalment en Amazon Web Services, per desenvolupar l'arquitectura bàsica d'una aplicació web molt senzilla partint inicialment d'una arquitectura monolítica, on tots els serveis es troben en un únic servidor, fins a tenir una arquitectura elàstica i escalable dinàmicament utilitzant els serveis de la plataforma d'Amazon, on cada servei s'encarrega de la seva funció per obtenir un millor rendiment en la nostra aplicació. A través d'exemples he volgut explicar les millores realitzades pas a pas.

Abstract

The present TFG tries to explain different components of the architecture of a web application, from code languages to the final designed architecture. For this, we have used the new Internet Services Platform, mainly Amazon Web Services, to develop a basic architecture of a very simple web application. Initially, we have designed a monolithic architecture, where all services are in a single server, and, at the end, we have created an elastic and dynamically scalable architecture using Amazon platform services, where each service is responsible for its function to obtain the better performance in our application. We have wanted to explain step by step through different examples and benchmarks.

Taula de continguts

Taula d’imatges

Taula de taules

Glosari

Bucket: és un contenidor d'objectes en el servei d'Amazon S3.

Framework. Entorn de treball predifinit amb una estructura conceptual i tecnològica, normalment amb mòduls de software concrets en base a la qual una altre projecte pot ser organitzat i desenvolupat.

Hooks. Són accions que es podem configurar depenen de l'esdeveniment que es produeixi.

Instància. A les noves tecnologies, és un servidor virtual al núvol.

Memória cau. És el que mal anomenem memòria caché.

Repositori de codi font. És el lloc on es guarda el codi de la nostra aplicació. Des d'on es poden consultar les diferents versions i que els desenvolupadors es poden descarregar.

Tags. Etiquetes. Utilitzades a Amazon per etiquetar serveis i poder controlar costos.

Acrònims

AMI	Amazon Machine Image
ASG	Auto Scaling Group
ASP	Active Server Pages
AWS	Amazon Web Services
bbdd	Bases de dades
CDN	Content Delivery Network
DNS	Domain Name Server
EC2	Elastic Compute Cloud
ELB	Elastic Load Balancer
GCP	Gloogle Cloud Platform
GCS	Gloogle Cloud Storage
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPs	Hyper Text Transfer Protocol Secure
PHP	HyperText Pre-Processor
RDS	Relational Database Service
S3	Simple Storage Service
SPA	Single Page Application
SSL/TLS	Secure Socket Layer / Transport Layer Security
TI	Tecnologia de la Informació
TTFB	Time to First Byte
vCPU	Virtual Central Processing Unit

Introducció

Aquest projecte és un Treball Final de Grau (TFG) que s'emmarca en els estudis del Grau d'Enginyeria Informàtica de la Universitat Oberta de Catalunya (UOC). El projecte pertany a l'especialitat d'Administració de xarxes i sistemes operatius.

L'objectiu del treball és visualitzar les diferents parts d'una aplicació web i explicar-ne el seu funcionament. Aplicar canvis respecte al disseny inicial

Per una banda, es parlarà del disseny de l'aplicació respecte a quin llenguatge es pot utilitzar (PHP, java, …), la utilització d'algun framework (symphony, JSF, …), i més característiques sobre el disseny i creació de l'aplicació.

També es parlarà del disseny i infraestructura de la plataforma i les possibles tecnologies que podem aplicar: protocol SSL, tipus de bases de dades (relacional, noSql, …), servei d'elements estàtics, tipus de servidor web (Apache, nginx,...), utilització de servei CDN, utilització d'elements guardats en memòria cau (varnish, squid,..), balancejadors de càrrega.

Amb tot, s'explicarà com es pot mesurar la càrrega en les aplicacions per que la plataforma sigui escalable i pugui suportar el pics de càrrega.

L'evolució en l'ús de serveis web en les organitzacions està fortament lligada al desenvolupament de la xarxa d'internet com a donant de serveis. Entre els factors que han impulsat l'ús de serveis web es troben:

- L'ample de banda és barat. A mesura que creix l'ample de banda, els serveis web s'han d'adaptar a nous tipus de continguts.

- Emmagatzemament de dades és barat. Una aplicació web ha de ser capaç de traficar quantitats grans de dades, i fer-ho de manera intel·ligent.

- Continguts més dinàmics. Les aplicacions webs actuals proporcionen continguts instantanis. A través d'un servei web, hem de poder ser capaços de combinar contingut provinent de fonts diferents.

Per arribar a veure una aplicació web finalitzada, primer s'han hagut de prendre moltes decisions. Molts cops, els usuaris finals no som conscients de moltes de les cruïlles en que s'han trobat els desenvolupadors i arquitectes de la plataforma per arribar al punt de finalització d'una aplicació

Llenguatges de programació web

Quan es comença un projecte web, hi ha dues parts que s'han de tenir en compte. Per una banda, hi ha el disseny gràfic que implica tota la imatge gràfica visible en la web: el logotip, les tipografies del text, les imatges, … i com aniran tots aquests elements distribuïts dins de la web.

Un segon punt a tenir en compte seria la programació: el llenguatge i eines a utilitzar.

Actualment existeixen molts llenguatges de programació, que han anat apareixent per les necessitats de les plataformes i segons les tendències.

Podem separar els llenguatges de programació depenent de la seva funció. Tenim llenguatges de programació de la banda del servidor (server-side) i llenguatges de programació de la banda del client (client-side).

Es coneix com a programació server-side a aquella que s'executa en el servidor web, immediatament abans que l'aplicació web enviï la informació a l'usuari.

En els inicis de la programació web, els servidors web només treballaven amb contingut estàtic, però van anar apareixent llenguatges que permetien fer connexions a

bases de dades i poder realitzar consultes i cerques. D'aquesta manera es van aconseguir aplicacions dinàmiques.

Característiques de codi server-side:

- Qualsevol codi que pugui córrer en una computadora i respongui a peticions HTTP pot ser un servidor web.
- Es pot emmagatzemar dades persistents en bases de dades.
- El codi no pot ser vist pel client. A no ser que passi alguna cosa terrible.
- Es crea la pàgina que l'usuari visualitza ja que es renderitza en el servidor web.

Els llenguatges més utilitzats en aquest tipus de programació són PHP, Java, ASP, Python

La programació en client-side implica que tot el programa corre a la banda del client, a la banda de l'usuari.

Característiques de codi client-side:

- reacciona a les entrada i accions de l'usuari
- l'usuari pot veure el codi totalment.
- no es guarden dades persistents
- no pot llegir fitxer del servidor directament, s'ha de fer mitjançant peticions HTTP

Tenim com a clàssics llenguatges d'aquest tipus: HTML, css, javascript.

Decidir que ha de fer el codi en el servidor en contra de que s'ha d'executar en el navegador és una de les preguntes que un desenvolupador web ha de decidir quan es comença una aplicació web.

De totes maneres, hi ha la pràctica generalitzada d'utilitzar els dos tipus de programació conjuntament. En una aplicació accessible des de un navegador, normalment el servidor és qui gestiona les pàgines, l'emmagatzematge i la càrrega de dades persistents. Però totes les reaccions de l'usuari i tot lo relacionat amb l'aparença visual canviant són controlades pel codi del client. Aquest tipus de pràctica es coneix com Aplicació de pàgina única (SPA, Single Page Application)

Sistema de control de versions

Quan es realitza una aplicació web és normal que fem proves, que canviem el codi sovint, que afegim noves funcionalitats. Molts cops, quan s'ha de fer front a un canvi important és necessari tocar molt de codi, és important fer els canvis a partir d'una còpia que sabem que funciona i aplicar-hi els canvis.

Si no s'utilitza cap eina que ens ajudi a fer això, el més fàcil és fer una còpia del codi font i treballar sobre un directori separat. Però no és la millor manera de treballar. Existeixen els sistemes de control de versions.

Aquestes eines són programes pel control de versions ideades per gestionar àgilment els canvis en el codi font dels programes i poder-los revertir. Més endavant, s'han ampliat les funcionalitats.

Ara mateix, en projectes de certa envergadura i amb varis desenvolupadors s'ha fet imprescindible utilitzar un sistema de control de versions, tot i que també és útil per a un únic desenvolupador, de manera que sempre tindrà totes les versions del seu programa controlat.

Hi ha dos tipus principals de sistemes de control de versions:

- Centralitzats: Es tracta d'un únic repositori on s'emmagatzema tot el codi del projecte. Exemple: CSV, subversion
- Distribuïts: Cada usuari té el seu propi repositori, i es poden intercanviar i barrejar revisions. Exemple: Git, mercurial.

Actualment, el sistema de control de versions més utilitzat és Git. Aquest va ser desenvolupat per Linus Torvalds i el seu equip, i és un sistema ràpid i molt eficient amb grans projectes i té un increïble sistema de branques. A més a més, és un sistema de codi obert.

En el moment en què ens referim a "desenvolupar aplicacions al núvol" hem de puntualitzar de quina manera ho anirem a fer.

El concepte núvol té diferents aplicacions de com fer-ho que ens aporten diferents valors de flexibilitat o facilitat a l'hora del desplegament.

En les diferents formes que es pot adoptar al núvol tenim:

- **Software-as-a-Service (SaaS):** permet als usuaris connectar-se a aplicacions basades en el núvol a través d'internet i utilitzar-les. Generalment es paga pel seu ús. Exemples comuns en són: Dropbox, Salesforce, Gmail,...

- **Platform-as-a-Service (PaaS):** És un entorn de desenvolupament i implementació complet al núvol. És un model que redueix la complexitat a l'hora de desplegar i mantenir aplicacions. Està pensat per facilitar als desenvolupadors la creació ràpida d'aplicacions web sense necessitat de preocupar-se de la configuració i infraestructura. Exemples populars seria Google App Engine que permet desenvolupar aplicacions en java o Python i desplegar-les en la infraestructura de Google.

- **Infraestructure-as-a Service (IaaS):** És la categoria més bàsica de serveis informàtics al núvol. Es lloga la infraestructura de TI (servidors, màquines virtuals, emmagatzemament, …) a un proveïdor de serveis al núvol i es paga pel seu ús. Exemples comuns: Amazon Web Services (AWS), Google Cloud Platform (GCP) i Azure de Microsoft.

Il·lustració 1. Esquema de plataformes al núvol

En aquest TFG, ens centrarem principalment en la última, que és la que ens permet fer canvis a la infraestructura i tenir més serveis aplicats a l'aplicació.

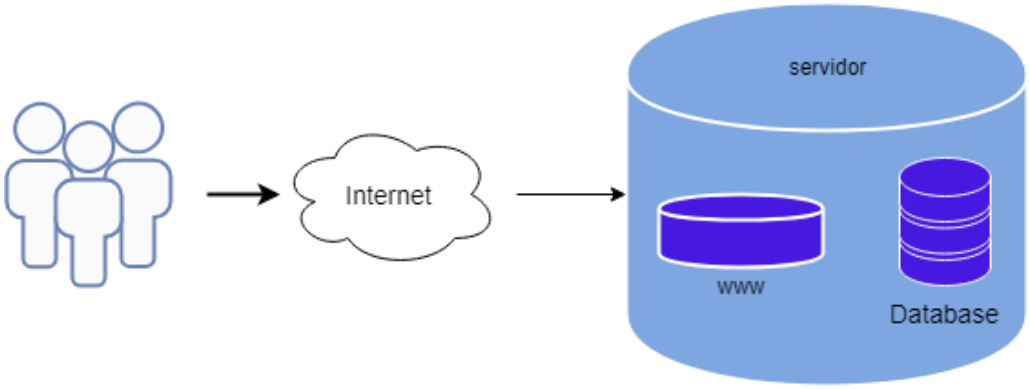
Arquitectura monolítica

Quan parlem d'arquitectura monolítica en el disseny d'aplicacions webs volem dir que tota l'aplicació està en el mateix servidor. Totes les parts estan acoblades i no hi ha separació entre serveis.

Es pot dir sense por, que és el tipus d'arquitectura menys nou. Tot i que no deixa de tenir una sèrie de característiques que durant molt de temps ha estat una bona solució. I ho seguirà sent per aplicacions webs petites.

Una de les grans avantatges que té és que totes les necessitats i serveis necessaris per l'aplicació es troben de manera estable en el mateix servidor.

Un altre avantatge pot ser l'eficiència. L'entorn on es munta aquest tipus de solució està molt ben definit, però aquesta característica crea entorns molt rígids i pot provocar que no es puguin actualitzar fàcilment. Un dels avantatges es converteix també en un dels principals desavantatges d'aquest tipus d'arquitectura.



Il·lustració 2. Arquitectura Monolítica

Arquitectures escalables i tolerants a fallades

En el moment de disseny d'una aplicació web i de la seva arquitectura se li ha de donar molta importància a les següents característiques:

- L'escalabilitat
- L'elasticitat
- Tolerància a fallades. Si el hardware falla, l'aplicació web ha de seguir donant servei als usuaris.

Escalabilitat

És la capacitat i habilitat de l'aplicació per poder créixer i adaptar-se sense perdre qualitat.

Dins de l'escalabilitat hi incloc la possibilitat d'augmentar la mida de càrrega dins de la infraestructura existent (hardware i software) sense afectar el rendiment.

Aquests recursos necessaris acostumen a estar pre-planificats per poder suportar els pics de demanda.

En resum, és la quantitat de recursos assignats per a poder assumir la càrrega més alta possible sense notar degradació en el rendiment.

Un cas pràctic d'escalabilitat és tenir dimensionada, per exemple, una plataforma de venta d'entrades online. En el moment en que apareixen entrades d'un gran concert han de tenir preparada la plataforma per poder donar servei a una gran quantitat de seguidors "histèrics". Però durant l'activitat d'un dia normal, la major part de la infraestructura podria estar parada.

Elasticitat

És la capacitat d'ampliar o reduir els recursos informàtics d'infraestructura per complir la demanda sense haver de preocupar-se de tenir la infraestructura preparada sempre per períodes màxims.

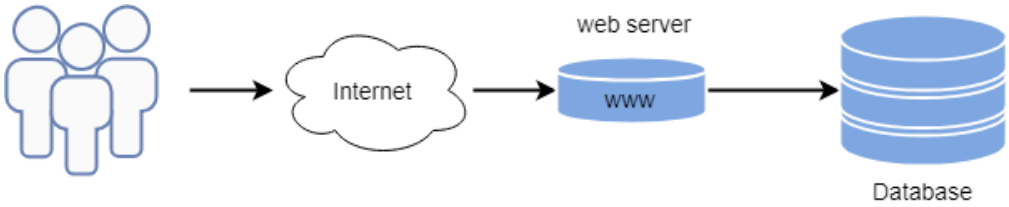
L'ús de serveis elàstics implica generalment que tots els recursos en la infraestructura siguin elàstics.

Un cas pràctic d'elasticitat el podem trobar en les botigues online amb una clara activitat estacional. Per exemple, en el període de rebaixes o en els famosos "black fridays" l'activitat en totes aquests botigues pot créixer de manera molt considerable durant aquests dies. Amb una infraestructura elàstica, es poden aguantar aquests dies de màxima audiència sense problemes.

Bones pràctiques i solucions

Per afrontar les característiques d'escalabilitat i elasticitat en la plataforma web, el que cal és separar els diferents serveis de la plataforma.

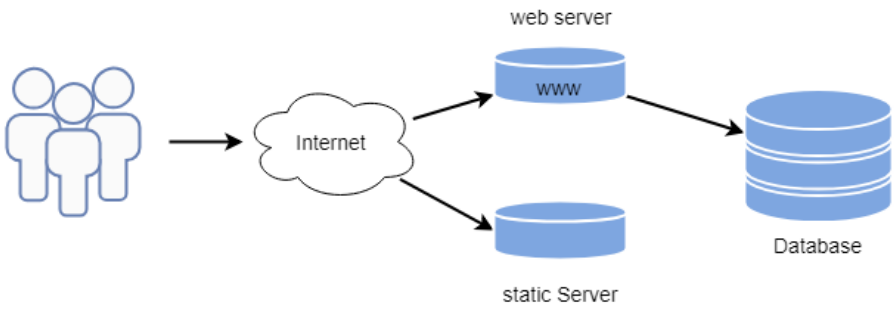
- **Separar el servidor de base de dades**: amb això aconseguim que dos servidors treballin conjuntament per servir peticions. Depèn molt de l'ús que faci la nostra aplicació de la base de dades, però normalment cada petició web fa varies consultes. D'aquesta manera aconseguim que part de la càrrega computacional la faci un altre servidor.



Il·lustració 3. Separació base de dades del servidor web

La nostra aplicació pot utilitzar varis sistemes de base de dades, tan SQL com noSQL. Depenen de la càrrega que la nostra aplicació faci a la base de dades, aquest servidor pot ser un dels colls d'ampolla. De les possibles solucions, en parlarem més endavant.

- **Separació elements estàtics:** quan servim una web no només servim la pàgina principal, s'han d'enviar imatges, css, tipus de lletra,... Separar aquest tipus d'arxius que són peticions que no necessiten processament i servir-los des de un servidor web únicament dedicat a elements estàtics, ens pots alliberar molta càrrega del servidor d'aplicacions.

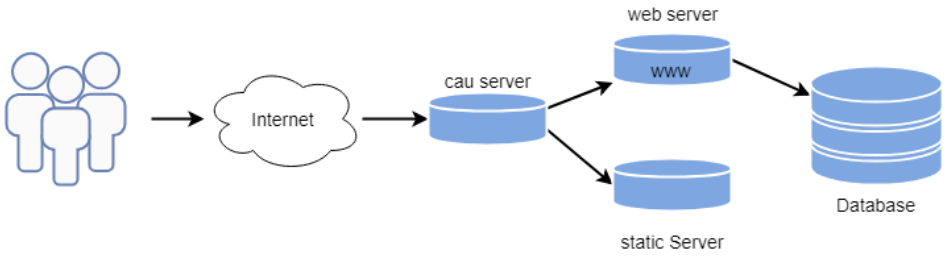


Il·lustració 4. Separació d'elements estàtics del servidor web

Si ens fixem, quan naveguem per pàgines grans (facebook, google,...) molts dels seus contingut no són servits des de el seu domini principal (facebook.com, google.com,...) sinó que parteixen d'un subdomini. I és que les imatges, arxius d'estils d'extensió css, i arxius de javascript (.js) són gran part de les peticions web.
Si per exemple, utilitzem el servidor web Apache amb el mòdul de PHP, cada petició que es faci al web server carrega el subsistema PHP encara que no el fem servir, i això perjudica a la nostra infraestructura. Al servir els estàtics per separat, ens assegurem que el web server serveix únicament les peticions que ha de processar.

I el servidor d'elements estàtics estarà correctament configurat per servir elements estàtics i no tindrà carregat el mòdul PHP.

- **Servidor memòria cau** Normalment les aplicacions web guarden informació temporal com poden ser els resultats de consultes a bbdd, pàgines web ja generades. És informació temporal que no acostuma a ser persistent i el seu emmagatzematge beneficia si tenim moltes visites.
  Imaginem la pàgina web d'un diari, si la portada està generada i guardada en cau durant un període de temps curt (5minuts), aquesta pàgina només s'hauria de generar des de el servidor d'aplicacions 1 cop cada 5 minuts.



Il·lustració 5. Inclusió d'un servidor de memòria cau

Serveis al núvol

Un cop ja hem vist l'estructura que volem aplicar en l'aplicació web, em centraré en com aplicar-ho en els proveïdors de servei al núvol més importants. Parlaré principalment d'Amazon Web Services (AWS), tot i que Google Cloud Platform (GCP) i Azure de Microsoft donen serveis similars.

La inclusió d'aquestes tres empreses com a empreses de serveis informàtics fa canviar el model econòmic de la informàtica.

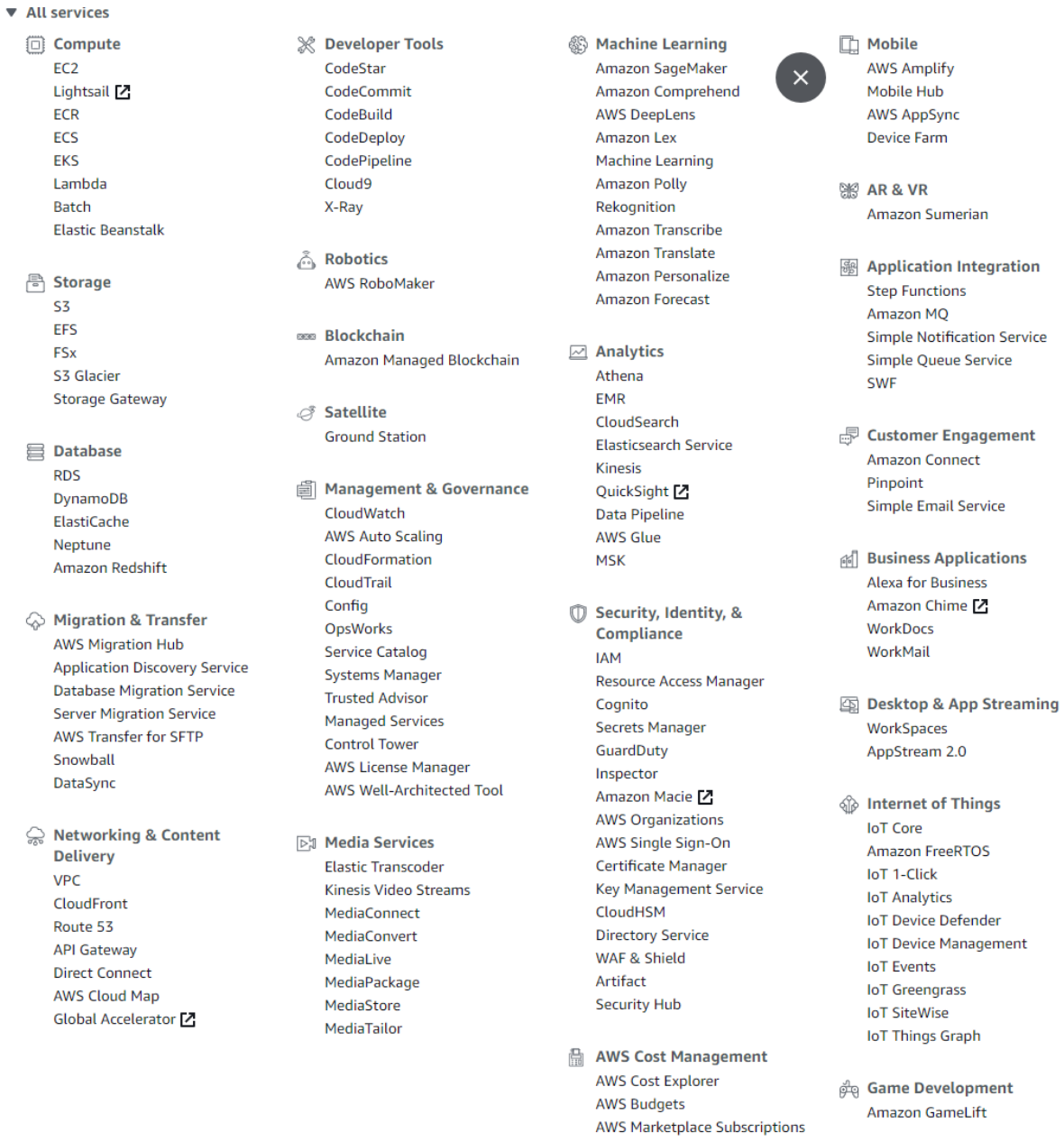
No he trobat millor solució per definir la plataforma d'Amazon que la que utilitzen ells en la seva presentació:

"Amazon Web Services (AWS) és una plataforma segura de serveis al núvol que ofereix potència de computació, emmagatzemament de bases de dades, entrega de contingut i altres funcionalitat per ajudar a les empreses a ajustar la seva escala i créixer.

Conegui de quina manera milions de clients aprofiten els productes i solucions del núvol d'AWS per crear aplicacions sofisticades i cada més flexibles, escalables i fiables.

El núvol d'AWS proporciona un ampli conjunt de infraestructura, como potencia de computació, opcions d'emmagatzemament, xarxes i bases de dades, que s'ofereixen com una utilitat: sota demanda, disponibles en qüestió de segons i pagant només pel que s'utilitza."

A continuació una imatge on es poden veure tots els serveis que ofereix AWS.



Il·lustració 6. Serveis d'AWS

De totes maneres, en el nostre cas en centrarem en les eines que utilitzarem per muntar l'arquitectura que comentàvem en apartats anteriors:

- Amazon EC2 (Amazon Elastic Cloud Computing)
- Amazon RDS
- Amazon S3
- Amazon Cloudfront

Serveis de computació

Aquestes plataformes tenen serveis que proporcionen capacitat informàtica en el núvol de manera segura i de mida modificable. Amb l'ús d'aquests serveis es redueix el temps necessari per obtenir i iniciar instàncies de servidor en qüestió de minuts, i això permet escalar (augmentant o reduint) ràpidament la capacitat actual a la necessària en aquest moment.

A AWS, aquest servei s'anomena Amazon EC2. A GCP, s'anomena Compute Engine.

Per utilitzar Amazon Ec2 es parteix d'una AMI (imatge de màquina Amazon) d'una plantilla pre-configurada per poder entrar en funcionament ràpidament. O també podem crear una AMI que contingui les aplicacions instal·lades, biblioteques, i dades i valors configurats necessaris per l'aplicació que s'hi vol muntar..

En la configuració de la instància, s'escull en quina zona volem aixecar la nostra instància. Amazon ofereix la possibilitat de col·locar instàncies en diferents ubicacions. Les ubicacions a Amazon es componen de regions i zones de disponibilitat. Les zones de disponibilitat són zones diferents dissenyades per estar aïllades d'errors que es puguin produir en altres zones de disponibilitat i que proporcionen connectivitat de xarxa de baixa latència a altres zones de disponibilitat de la mateixa regió.

Al llançar instàncies en zones de disponibilitat diferents, podem protegir les nostres aplicacions en cas de produir-se qualsevol error en una única ubicació.

Aquestes són les zones disponibles:

US East (N. Virginia)	Asia Pacific (Tokyo)
US East (Ohio)	Canada (Central)
US West (N. California)	EU (Frankfurt)
US West (Oregon)	EU (Ireland)
Asia Pacific (Mumbai)	EU (London)
Asia Pacific (Seoul)	**EU (Paris)**
Asia Pacific (Singapore)	EU (Stockholm)
Asia Pacific (Sydney)	South America (São Paulo)

Il·lustració 7. Zones disponibles en Amazon AWS

La flota de tipus de instàncies que ofereix Amazon Ec2 és molt amplia, i permet, en cada cas, obtenir la capacitat informàtica necessària. Dins del ventall ofertat, podem escollir des de les més petites:

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☑	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes

Il·lustració 8. Instàncies Ec2 petites

fins les més potents:

☐	General purpose	m5.12xlarge	48	192	EBS only	Yes	10 Gigabit	Yes
☐	General purpose	m5.24xlarge	96	384	EBS only	Yes	25 Gigabit	Yes

Il·lustració 9. Instàncies Ec2 potents

La seguretat en el núvol és una de les prioritats que tenen aquestes plataformes. Es poden arribar a crear arquitectures de xarxa molt complexes per arribar a donar una seguretat màxima en l'accés a les dades que s'hi dipositin.

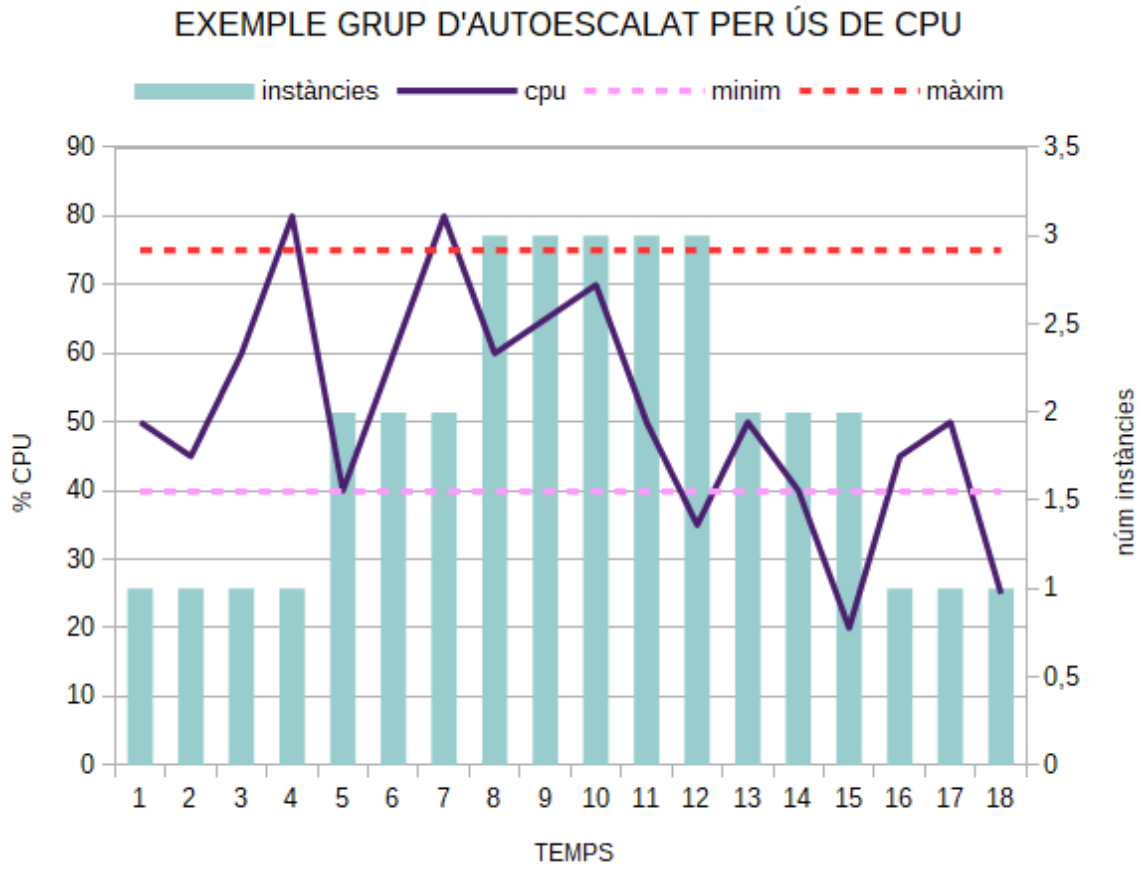
Auto-escalat

De les noves plataformes de serveis que estic parlant, l'Auto-escalat és una de les característiques més importants per donar elasticitat a una plataforma.

L'auto-escalat és una característica d'un grup d'instàncies creades a partir de la mateixa plantilla d'instància, sigui una AMI d'Amazon o una AMI creada per nosaltres. Amb el sistema d'auto-escalat, es permet ajustar la capacitat automàticament d'acord amb les condicions que es defineixin.

Tan a AWS com a GCP, les condicions d'auto-escalat es basen en la mitjana d'ús de CPU de totes les instàncies del grup, o bé en mitjana de bytes d'entrada o de sortida. Les pròpies eines que ens donen aquestes plataformes serveixen per gestionar els escalats.

La configuració més bàsica i més comú és la basada en ús de CPU. Se li marquen un paràmetres mínims i màxims i són aquests amb els que es basa la plataforma per auto escalar amunt per augmentar les instàncies o per auto-escalar avall per reduir-les.



Il·lustració 10. Gràfic d'exemple de comportament d'auto-escalat

Serveis de base de dades

Les tres grans plataformes de serveis informàtics al núvol ofereixen els serveis de bases de dades. Tant per bases de dades relacionals per aplicacions de transaccions com per bases de dades no relacionals per aplicacions d'escala d'Internet, o bé un magatzem de dades per emmagatzemament en memòria cau i càrregues de treball en temps real.

A Amazon AWS hi ha el servei d'Amazon RDS que serveix per a bases de dades relacionals. Es pot escollir el motor de MySQL, MariaDB, PostgreSQL, Oracle o Microsoft SQL Server.

Quan es vol aixecar una instància, igual que en el servei ec2, s'escull el tipus de instància que es cregui necessari segons nombre de vCPU i la memòria RAM de la instància. Es pot produir el cas, que interessi augmentar o reduir el tipus d'instància i es pot fer sense cap tipus de problema.

Per altra banda, Amazon també té el servei que anomenen Aurora. És un servei de DBaaS (Database-as-a-Service) d'alta disponibilitat, únicament accessible a través dels serveis d'AWS.

A més a més, també hi ha el servei de bases de dades no relacionals (Amazon DynamoDB), o de memòria en cau (elasticache).

Serveis d'emmagatzemament

En l'actualitat, les empreses necessiten la capacitat de recopilar, emmagatzemar i analitzar les seves dades de manera simple, segura i a gran escala. Tant Amazon S3 (Simple Storage Service) com GCS (Google Cloud Storage) són serveis d'emmagatzematge d'objectes creats per guardar i recuperar dades des de qualsevol ubicació: aplicacions mòbils, aplicacions webs, aplicacions corporatives,..

Ofereixen alts nivells de disponibilitat, durabilitat i escalabilitat.

A més, també es pot escollir en diferents nivells d'emmagatzematge: standard, d'ús poc freqüent, i Glacier (AWS) o Coldline(GCS) . Depenen de quin tipus de nivell es guardi, té una variació important en el cost.

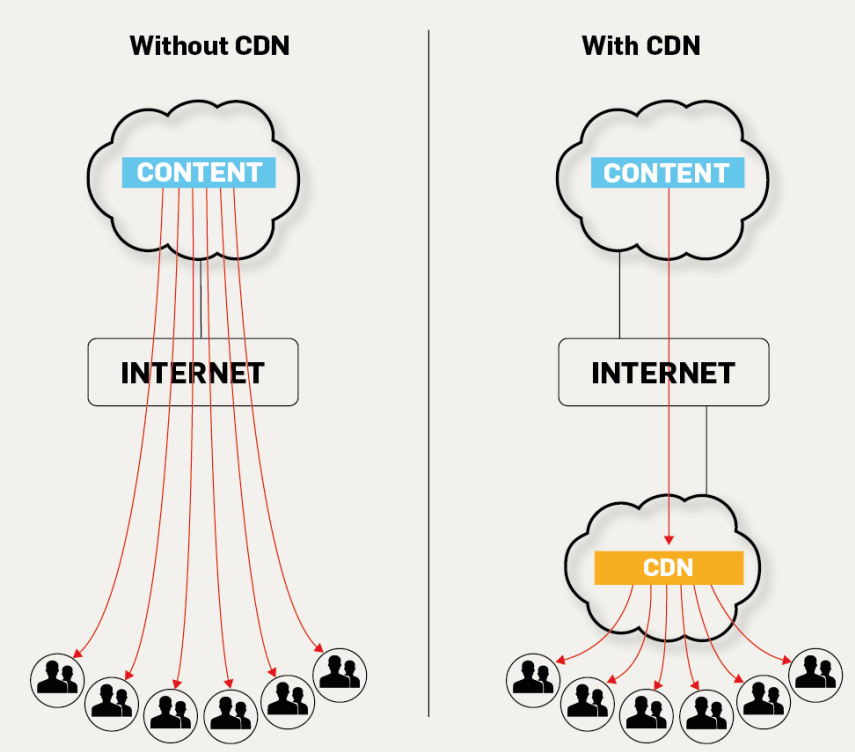
Serveis CDN

Content Delivery Network (CDN) és una xarxa de distribució de continguts, que conté còpia de les dades i col·locada en varis punts de la xarxa, de manera que l'ample de banda sigui el màxim i més proper possible al client. Del que es tracta és aconseguir uns temps baixos de resposta.

En les plataformes d'AWS i GCP també ofereixen aquests serveis, coneguts com Cloudfront i Google CDN respectivament.

Tot i que no tot el contingut és compatible amb CDN. Principalment es pot servir

- contingut estàtic: pàgines en HTML, imatges,
- àudio i vídeo



Il·lustració 11. Explicació de CDN

Creació d'una aplicació en arquitectura monolítica

Un cop vist els serveis més comuns que ofereixen les plataformes més importants de Infraestructura-as-a-Service, aixecaré una plataforma utilitzant els serveis d'Amazon AWS.

Partiré d'una arquitectura monolítica, on en el mateix servidor hi residirà el servidor web, la bases de dades Mysql, i unes imatges.

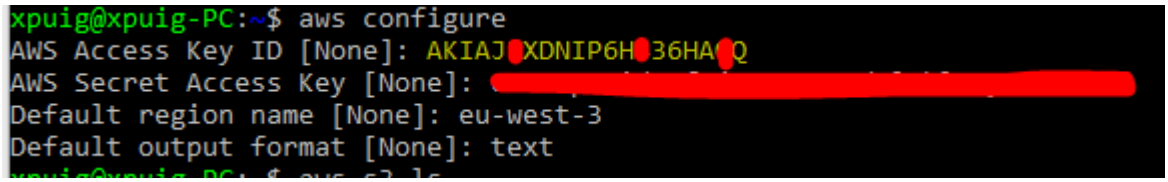
El servei d'Amazon que necessitem per aixecar un servidor d'aplicacions és EC2.

Com que el que volem és un petit servidor escullo una mida d'instància petit en la regió de EU-París que per proximitat és la que més ens pot interessar.

Per aixecar la plataforma ho farem per consola a partir d'un terminal en una màquina de linux, on hem instal·lat l'aplicació awscli que és el client per poder fer les crides als serveis d'AWS.

A través de la consola web d'Amazon, he donat permisos al meu usuari perquè pugui accedir a través del client d'aws que acabem d'instal·lar.

Per configurar el client, es fa em la següent comanda:



Taula 1. configuració client aws: awscli

Abans d'aixecar una instància, s'ha de configurar els accessos que ha de tenir des de l'exterior. En el nostre cas, com que volem que sigui un servidor web, donem accés a través del port 80, i per poder administrar donem accés per ssh pel port 22 des de la ip de casa meva.

Per habilitar aquesta seguretat, creem un grup de Seguretat que anomeno TFGgroup, i li afegeixo les regles que comentava.

```
xpuig@xpuig-PC:~$ aws ec2 create-security-group --group-name TFGgroup --description "TFG group" --profile xpuigv-UOC
{
  "GroupId": "sg-0bb66dc763b330335"
}
```

```
xpuig@xpuig-PC:~$ aws ec2 authorize-security-group-ingress --group-name TFGgroup  --protocol tcp --port 22 --cidr 88.13.107.158/32 --profile xpuigv-UOC
xpuig@xpuig-PC:~$ aws ec2 authorize-security-group-ingress --group-name TFGgroup  --protocol tcp --port 80  --cidr 0.0.0.0/0 --profile xpuigv-UOC
```

Taula 2. awscli: creació grups de seguretat

Per accedir, a la instància que aixecarem, creo un parell de claus, que seran les que ens permetran accés per ssh.

```
xpuig@xpuig-PC:~$ aws ec2 create-key-pair --key-name uoc-key --profile xpuigv-UOC
{
    "KeyFingerprint": "39:07:cd:c4:31:10:7b:9a:6e:ed:e0:4c:2a:8d:e8:da:1c:b7:b9:0e",
    "KeyMaterial": "-----BEGIN RSA PRIVATE KEY-----\n
**************
s=\n-----END RSA PRIVATE KEY-----",
    "KeyName": "uoc-key"
}
```

Taula 3. awscli: creació parells de claus d'accés.

I aixequem la instància. Al executar la instrucció, inicialment la instància no té IP pública per poder connectar-nos, però podem veure-ho amb les següents comandes.

La instància la he creat a partir d'una AMI del marketplace d'Amazon AWS, i he escollit una imatge d'un Ubuntu 18.04

```
xpuig@xpuig-PC:~$ aws ec2 run-instances --profile xpuigv-UOC --image-id ami-08182c55a1c188dee --count 1 --instance-type t2.micro --key-name  uoc-key --security-group-ids
{    "Groups": [ ],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-08182c55a1c188dee",
            "InstanceId": "i-064f307af815605ae",
            "InstanceType": "t2.micro",
            "KeyName": "uoc-key",
            "LaunchTime": "2018-12-21T16:42:14.000Z",
            "Monitoring": {
                "State": "disabled"
            },
 …
}
```

```
xpuig@xpuig-PC:~$ aws ec2 describe-instances --instance-ids i-064f307af815605ae  --
profile xpuigv-UOC
{
…
        "PrivateDnsName": "ip-172-31-36-71.eu-west-3.compute.internal",
        "PrivateIpAddress": "172.31.36.71",
        "ProductCodes": [],
        "PublicDnsName":
              "ec2-35-180-152-180.eu-west-3.compute.amazonaws.com",
        "PublicIpAddress": "35.180.152.180",
…
}
```

Taula 4. awscli: primera execució instància

Li assigno un ip fixa a la instància per poder configurar correctament els DNS.

```
xpuig@xpuig-PC:~$ aws ec2 allocate-address --profile xpuigv-UOC
{
   "PublicIp": "35.180.240.78",
   "AllocationId": "eipalloc-0cf728eebe3599121",
   "Domain": "vpc"
}

xpuig@xpuig-PC:~$ aws ec2 associate-address --instance-id i-064f307af815605ae --
public-ip 35.180.240.78 --profile xpuigv-UOC
{
   "AssociationId": "eipassoc-02a0c35ed5d0fe787"
}
```

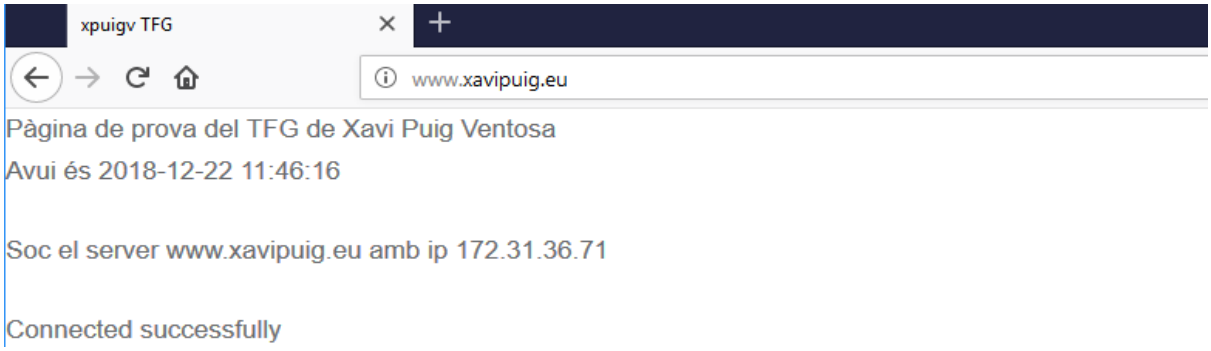
Taula 5. awscli: assignació IP fixe a una instància.

En aquesta instància, que és de les més petites s'hi ha instal·lat un servidor web amb un Apache2, que s'ha configurat amb la configuració bàsica perquè respongui a la direcció URL www.xavipuig.eu

S'ha instal·lat també un servidor mysql, amb una base de dades amb un taula amb diferents hash.

En el servidor s'hi ha instal·lat el client i el mòdul PHP per poder servir pàgines amb consultes a una base de dades.

Per fer proves de rendiment, he configurat una senzilla web amb una cerca d'un hash aleatori en una base de dades, i amb la càrrega d'unes imatges una mica grans.



Il·lustració 12. Captura pantalla de la pàgina exemple.

Les proves que s'han realitzat per revisar el rendiment, les he realitzat amb les aplicacions htop del propi sistema d'Ubuntu, la pròpia monitorització d'AWS i les aplicacions ab (apache benchmark) i siege, una utilitat de rendiment multi-thread per protocol HTTP, i també la consola de desenvolupador del google chrome.

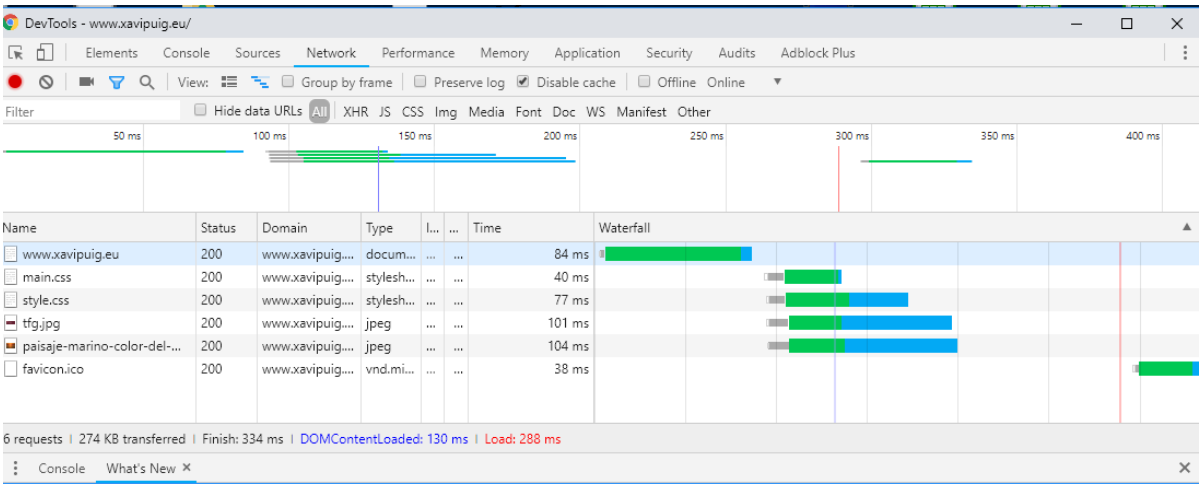
Al llançar les proves amb l'aplicació ab, les tiro amb 1000 peticions i 60 de concurrents. Es tracta és de veure fins quins paràmetres podem aguantar.

```
xpuig@xpuig-PC:~$ ab -n 1000 -c 60 -l www.xavipuig.eu/index.php
Benchmarking www.xavipuig.eu (be patient)
Server Software:        Apache/2.4.29
Server Hostname:        www.xavipuig.eu
Server Port:            80
Document Path:          /index.php
Document Length:        Variable
Concurrency Level:      60
Time taken for tests:   48.167 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      3425663 bytes
HTML transferred:       3208779 bytes
Requests per second:    20.76 [#/sec] (mean)
Time per request:       2890.005 [ms] (mean)
Time per request:       48.167 [ms] (mean, across all concurrent requests)
Transfer rate:          69.45 [Kbytes/sec] received
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       28   29   2.9     29      75
Processing:  1183 2833 320.3   2844    5435
Waiting:     1183 2831 320.3   2840    5434
Total:       1212 2862 320.8   2873    5476
Percentage of the requests served within a certain time (ms)
  50%   2873
...
 100%   5476 (longest request)
```

Taula 6. execució apache Benchmark

En aquesta arquitectura monolítica, l'ús de CPU pel procés de mysql creix moltíssim, fins a posar-se al 100%. Puc afirma que la instància es dedica a processar les consultes sql.

A través de la consola de google chrome, estudio com es produeix la càrrega dels diferents elements.



Il·lustració 13. Consola de desenvolupador a GoogleChrome

El que es pot apreciar és que tots els elements venen del mateix domini (www.xavipuig.eu), i el temps de càrrega dels diferents elements.

Dins dels temps de resposta, és bo tenir un temps baix de TTFB (Time To First Byte). Aquest temps ve donat per la latència . Més endavant, miraré de millorar-lo .

Traspàs cap a una arquitectura elàstica i escalable

Com havíem comentat abans, el traspàs cap a una arquitectura elàstica i escalable ve donat perquè tots els seus elements siguin elàstics i escalables.

El primer pas és separar la bases de dades del propi servidor web. Com que hem estat treballant amb Amazon AWS, trobo adient utilitzar les seves pròpies eines. Per tant, aixeco una instància del servei Amazon RDS amb un motor de bases de dates Mysql.

A la instància RDS se li assignen permisos d'accés únicament des de la pròpia VPC on tenim el servidor web. No és necessari donar-li accés des de enlloc més. Si tenim necessitat de connectar-nos-hi ho podrem fer des la instància anterior.

```
xpuig@xpuig-PC:~$ aws rds create-db-instance --profile xpuigv-UOC --db-instance-identifier tfg-uoc --allocated-storage 20 --db-instance-class db.t2.micro --engine mysql --master-username xpuigvUOC --master-user-password passwordDB --no-multi-az --engine-version 5.6.40
{
    "DBInstance": {
        "DBInstanceIdentifier": "tfg-uoc",
        "DBInstanceClass": "db.t2.micro",
        "Engine": "mysql",
        "DBInstanceStatus": "creating",
        "MasterUsername": "xpuigvUOC",
        "AllocatedStorage": 20,
        "PreferredBackupWindow": "10:46-11:16",
        "BackupRetentionPeriod": 1,
        "DBSecurityGroups": [],
        "VpcSecurityGroups": [
            {
                "VpcSecurityGroupId": "sg-bc0f13d4",
                "Status": "active"
            }
        ],
        "DBParameterGroups": [
            {
                "DBParameterGroupName": "default.mysql5.6",
                "ParameterApplyStatus": "in-sync"
            }
        ],
        "DBSubnetGroup": {
            "DBSubnetGroupName": "default",
            "DBSubnetGroupDescription": "default",
            "VpcId": "vpc-5994cd30",
            "SubnetGroupStatus": "Complete",
            "Subnets": [
                {
                    "SubnetIdentifier": "subnet-b10699ca",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-west-3b"
                    },
                    "SubnetStatus": "Active"
                },
                {
                    "SubnetIdentifier": "subnet-bb2787f6",
                    "SubnetAvailabilityZone": {
                        "Name": "eu-west-3c"
                    },
                    "SubnetStatus": "Active"
                },
```

```
            {
                "SubnetIdentifier": "subnet-d8bffbb1",
                "SubnetAvailabilityZone": {
                    "Name": "eu-west-3a"
                },
                "SubnetStatus": "Active"
            }
        ]
    },
    "PreferredMaintenanceWindow": "wed:01:10-wed:01:40",
    "PendingModifiedValues": {
        "MasterUserPassword": "****"
    },
    "MultiAZ": false,
    "EngineVersion": "5.6.40",
    "AutoMinorVersionUpgrade": true,
    "ReadReplicaDBInstanceIdentifiers": [],
    "LicenseModel": "general-public-license",
    "OptionGroupMemberships": [
        {
            "OptionGroupName": "default:mysql-5-6",
            "Status": "in-sync"
        }
    ],
    "PubliclyAccessible": true,
    "StorageType": "gp2",
    "DbInstancePort": 0,
    "StorageEncrypted": false,
    "DbiResourceId": "db-LVGLDELYY6KLG5VI56AEDXFTVA",
    "CACertificateIdentifier": "rds-ca-2015",
    "DomainMemberships": [],
    "CopyTagsToSnapshot": false,
    "MonitoringInterval": 0,
    "DBInstanceArn": "arn:aws:rds:eu-west-3:025974803533:db:tfg-uoc",
    "IAMDatabaseAuthenticationEnabled": false,
    "PerformanceInsightsEnabled": false
  }
}
```


Taula 7. awscli: creacio instància de bases de dades a RDS

La creació de la instància triga uns 5 minuts. I després de fer el traspàs de les dades de la base de dades que teníem en el propi servidor web cap al nou servidor de Mysql.

Repeteixo les proves que havia fet anteriorment per veure si hi ha millora en els resultats.

```
xpuig@xpuig-PC:~$ ab -n 1000 -c 60 -l www2.xavipuig.eu/index.php
Server Software:        Apache/2.4.29
Server Hostname:        www2.xavipuig.eu
Server Port:            80
Document Path:          /index.php
Document Length:        Variable
Concurrency Level:      60
Time taken for tests:   45.321 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      3804638 bytes
HTML transferred:       3588150 bytes
Requests per second:    22.06 [#/sec] (mean)
Time per request:       2719.265 [ms] (mean)
Time per request:       45.321 [ms] (mean, across all concurrent requests)
Transfer rate:          81.98 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:       28   29   2.3     29      48
Processing:   454 2663 469.3   2679    4330
Waiting:      454 2660 469.1   2674    4330
Total:        485 2692 468.8   2708    4358

Percentage of the requests served within a certain time (ms)
  50%   2708
  66%   2780
  75%   2830
  80%   2882
  90%   3005
  95%   3401
  98%   3846
  99%   4052
 100%   4358 (longest request)
```

Taula 8. resultat test apache benchmark

Els temps de resposta no són gaire concloents, i són molts semblants entre les dues execucions. De totes maneres, el temps total és una mica inferior en la segona execució respecte a la 1a. On es nota molt és en l'ús de CPU. Al tenir un servei servidor de bases de dades, l'ús de CPU en el servidor Apache ha passat d'un 100% anteriorment a 4% essent el procés d'Apache el que n'utilitza més.

Veient aquest resultats, em plantejo rebaixar la mida de la instància actual d'una t2.micro a un t2.nano.

```
xpuig@xpuig-PC:~$ aws ec2 stop-instances --instance-ids i-064f307af815605ae --profile
xpuigv-UOC
{
    "StoppingInstances": [
        {
            "CurrentState": {
                "Code": 64,
                "Name": "stopping"
            },
            "InstanceId": "i-064f307af815605ae",
            "PreviousState": {
                "Code": 16,
                "Name": "running"
            }
        }
    ]
}

xpuig@xpuig-PC:~$ aws ec2 modify-instance-attribute  --instance-id i-064f307af815605ae
--instance-type t2.nano --profile xpuigv-UOC

xpuig@xpuig-PC:~$ aws ec2 start-instances --instance-ids i-064f307af815605ae --profile
xpuigv-UOC
{
    "StartingInstances": [
        {
            "CurrentState": {
                "Code": 0,
                "Name": "pending"
            },
            "InstanceId": "i-064f307af815605ae",
            "PreviousState": {
                "Code": 80,
                "Name": "stopped"
            }
        }
    ]
}
```

Taula 9. awscli: canvi mida instància

El temps del canvi de mida d'instància quasi bé és el temps de fer un reinici de la mateixa. S'ha de parar, indicar la nova mida, i aixecar-la de nou.

Els resultats d'haver executat el mateix test que havíem fet amb anterioritat és quasi bé el mateix. L'ús de CPU del servidor web es manté estable i respon en els temps esperats, per tant el canvi de tipus de instància cap a una de mida més petita el dono per bo i el mantenim.

El següent pas és treure els elements estàtics perquè no siguin servits pel servidor web amb el PHP instal·lat.

Per fer aquest pas, he estat pensant varies opcions. La primera era muntar una instància amb servidor web nginx, configurar-lo amb un altre domini per servir estàtics (p.e. http://media.xavipuig.eu) i utilitzar una de les propietats que té el servidor nginx que és la de servir contingut estàtic.

De totes maneres, ja que he estat parlant dels serveis d'AWS, he decidit utilitzar el servei S3 per emmagatzemament d'objectes. La primera entitat on es guarden els objectes, en S3, s'anomena bucket i els noms dels buckets són únics en tota la plataforma d'Amazon. Com que l'ús que li vull donar és per a servir estàtic, el bucket que creo l'anomeno static.xavipuig.eu

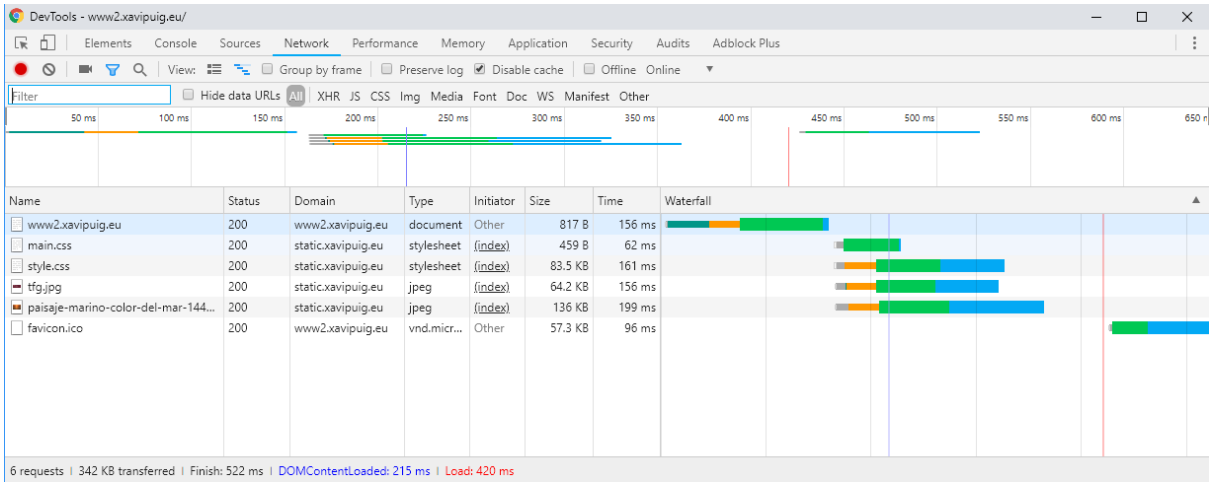
```
xpuig@xpuig-PC:~$ aws s3api create-bucket --bucket static.xavipuig.eu --region eu-west-3 --create-bucket-configuration LocationConstraint=eu-west-3 --acl public-read --profile xpuigv-UOC
{
    "Location": "http://static.xavipuig.eu.s3.amazonaws.com/"
}

xpuig@xpuig-PC:~$ aws s3 cp css/main.css s3://static.xavipuig.eu/css/main.css --acl public-read --profile xpuigv-UOC
upload: css/main.css to s3://static.xavipuig.eu/css/main.css
xpuig@xpuig-PC:~$
```

Taula 10. awscli: creació bucket a S3

D'aquesta manera, el bucket creat s'ha creat amb permisos perquè tothom pugui llegir, i aquest servei de S3 fa ens fa de servidor d'objectes estàtics.

Si creem, en el servidor DNS, una entrada CNAME amb el nom del bucket i que apunti a static.xavipuig.eu.s3.amazonaws.com, podem servir el contingut estàtic amb el nostre domini.



Il·lustració 14. Resultat d'estàtics a S3

Un altre servei que aplicarem a continuació és el de CDN. El que ens aporta aquest servei és de proximitat dels objectes i millorar amb les latències i els temps de transport. Des de l'origen a nosaltres es crea una capa intermèdia que és la que ens serveix a nosaltres el contingut.

La primera petició d'un objecte potser que trigui més que abans, ja que la petició la fem al CDN i si no té l'objecte el demana a l'origen i llavors ens el serveix.
Si l'objecte ja el té, ens el serveix directament.

En AWS, el servei s'anomena Cloudfront.
Creem una distribució per servir els nostres elements estàtics on l'origen és el bucket creat anteriorment.

```
xpuig@xpuig-PC:~$ aws cloudfront create-distribution --origin-domain-name
static.xavipuig.eu.s3.amazonaws.com --default-root-object index.html --profile xpuigv-UOC
{
    "Location": "https://cloudfront.amazonaws.com/2017-03-
25/distribution/E3DQ0MNN7J74VS",
    "ETag": "E15D40OM0NNVR6",
    "Distribution": {
        "Id": "ERWYCL29COYE5",
 ...
}

xpuig@xpuig-PC:~$ aws cloudfront get-distribution --id ERWYCL29COYE5 --profile
xpuigv-UOC
{
    "ETag": "ELVZK2CBKFURG",
    "Distribution": {
        "Id": "ERWYCL29COYE5",
...
        "DomainName": "d36bxrymgsuq2v.cloudfront.net",
```

```
...
        "DistributionConfig": {
...
            "Aliases": {
                "Quantity": 1,
                "Items": [
                    "static-cdn.xavipuig.eu"
                ]
            },
…
}
```

Taula 11.awscli: creació distribució Cloudfront

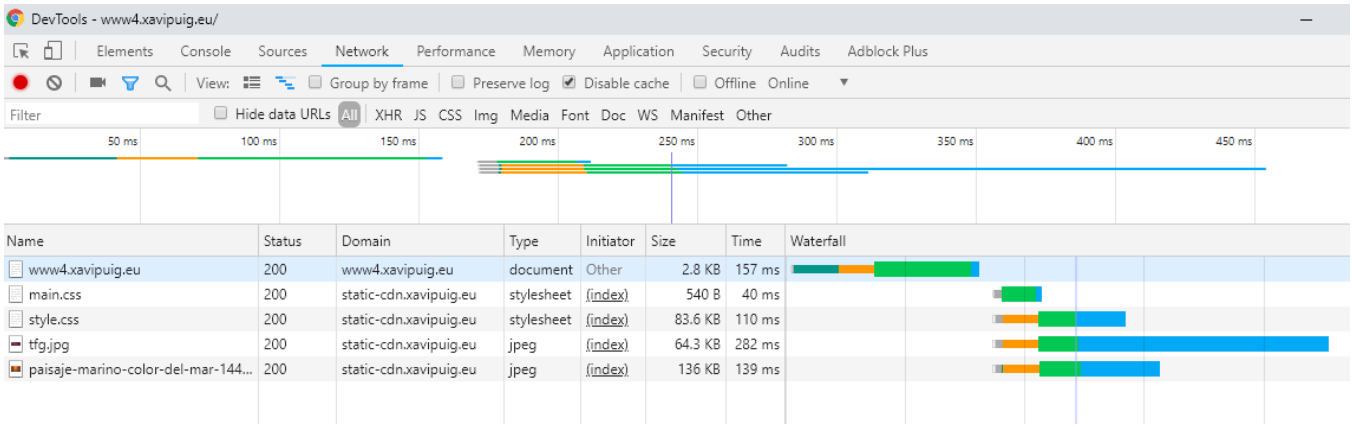
Per utilitzar el servei de Cloudfront, els comandes de la api ens retornen un domain name, que és el que podem utilitzar per servir el contingut estàtic.
Sinó també es pot configurar un CNAME a la nostra zona del domini que apunti a la distribució de Cloudfront.

Les crides a un fitxer main.css de la web d'exemple des dels diferents entrades que hem anat creant per servir estàtics serien:

Servidor web	http://www.xapuig.eu/css/main.css
S3	http://static.xavipuig.eu.s3.amazonaws.com/css/main.css
S3 (amb dns personalitzat)	http://static.xavipuig.eu/css/main.css
Cloudfront	http://d36bxrymgsuq2v.cloudfront.net/css/main.css
Cloudfront (amb dns personalitzat)	http://static-cdn.xavipuig.eu/css/main.css

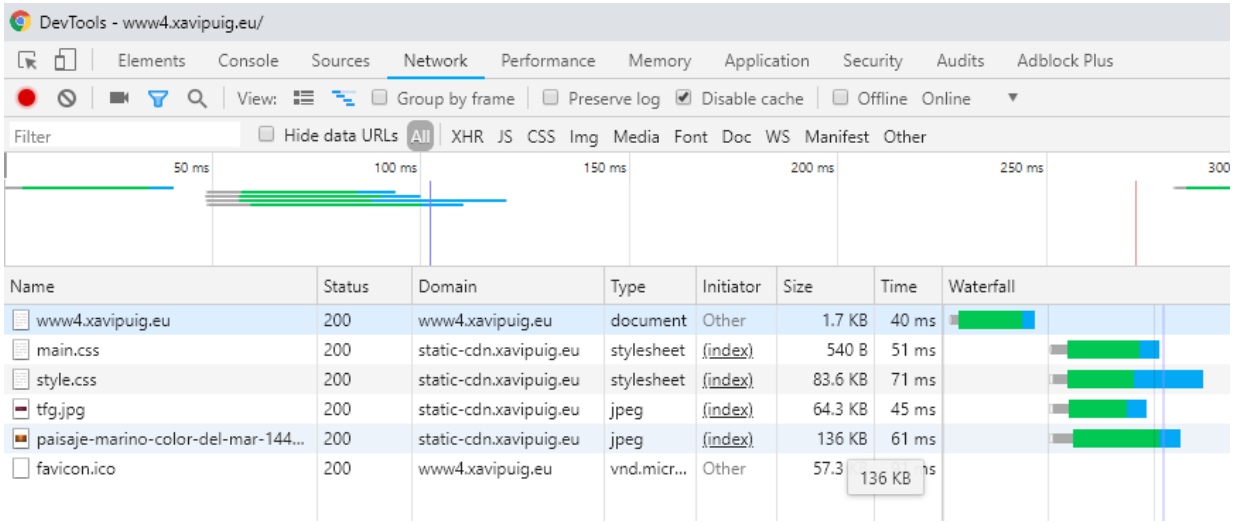
Taula 12. diferents crides a main.css segons origen.

Els temps de resposta dels estàtics són diferents en la primera petició que en la segona a través de CDN. La raó és perquè el primer cop que sol·licitem un objecte al CDN, com que no el té guardat, llavors el solicita a l'origen, en aquest cas a S3.



Il·lustració 15.Cloudfront: temps de resposta primera petició

En la segona petició, els objectes estàtics ja estan al CDN, i es serveixen més ràpid. De totes maneres, és normal que els serveis de CDN tinguin diversos servidors, per tant també es pot donar el cas que les primeres sol·licituds dels mateix objecte no vagin tot lo ràpid que esperem.



Il·lustració 16. Cloudfront: temps de resposta segona petició

Com es pot veure en les imatges, el temps de resposta quan s'utilitza CDN és molt inferior que si ho sol·licitem a un servidors més llunyans.

En el nostre cas, com que l'exemple és molt senzill el guany entre una opció o l'altre no deixa de ser de 100-200 ms, però en aplicacions web amb molts més elements estàtics el guany és més pronunciat.

Una altra opció per millorar seria separar en diversos subdominis de CDN. Tenir un subdomini per a servir els css, tenir-ne un altre per servir les imatges, tenir-ne un tercer per servir els fitxers js,... D'aquesta manera es podrien fer les càrregues d'elements estàtics en paral·lel.

He realitzat varies comprovacions amb l'eina curl per consola per comprovar el TTFB i el temps total d'alguns elements estàtics.

```
xpuig@xpuig-PC:~$ curl -o /dev/null -w "Connect: %{time_connect} TTFB: %{time_starttransfer} Total time: %{time_total} \n" http://static-cdn.xavipuig.eu/image/paisaje-marino-color-del-mar-1440x810.jpg
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  135k  100  135k    0     0   680k      0 --:--:-- --:--:-- --:--:--  677k

Connect: 0.100200 TTFB: 0.131815 Total time: 0.199593
```

Taula 13. Comprovació TTFB

Aquests són els resultats obtinguts, de mitjana sobre 50 mostres.
Per una imatge:

image/paisaje-marino-color-del-mar-1440x810.jpg (136KB)	TTFB	Total Time
servidor web en repós	150ms	245ms
servidor web (no repós)	200ms	290ms
servidor estàtics (s3)	185ms	275ms
cdn (Cloudfront)	125ms	192ms

Taula 14. Comprovació TTFB per una imatge

Per un fitxer d'estils:

css/style.css (83.5 KB)	TTFB	Total Time
servidor web en repós	111ms	172ms
servidor web (no repós)	180ms	242ms
servidor estàtics (s3)	169ms	229ms
cdn (Cloudfront)	103ms	163ms

Taula 15. Comprovació TTFB per style.css

Amb aquests resultats encara es fa més palesa la situació que explicàvem. Fins i tot m'estranyen els resultats obtinguts per S3 essent resultats semblants als obtingut en el servidor web amb arquitectura monolítica, tot i que es pot arribar a comprendre al pensar que tots dos són serveis d'AWS i es troben en xarxes similars.

Aplicació d'un grup d'auto-escalat

Una de les característiques que ens aporten les plataformes de serveis web és l'aplicació de grups d'auto-escalat i que ens permet donar una elasticitat dinàmica a la plataforma que estem aixecant.

El grup d'auto-escalat el creem sobre la capa del servidor web, que és que pateix més i sobre el que estem treballant més per que no sigui el coll d'ampolla de la nostra plataforma.

Per crear el grup d'autos escalat ho fem a partir de la instància que tenim aixecada com a servidor web. Farem una AMI a partir d'aquesta instància perquè la resta de instàncies que aixequem ho facin a partir d'aquesta AMI que ja té tot el software configurat per respondre als dominis xavipuig.eu.

Amb la següent comanda creem una AMI de la instància que tenim aixecada.

```
xpuig@xpuig-PC:~$ aws ec2 create-image --instance-id i-064f307af815605ae --
description "ami for ASG" --name "ASG-AMI-v1" --no-reboot --profile xpuigv-UOC
{
    "ImageId": "ami-0dfd2e71b26383e94"
}
xpuig@xpuig-PC:~$
```

Taula 16. awscli: creació AMI

I per anar bé hem d'aixecar una instància nova amb la nova AMI, perquè el grup d'auto-escalat es crea a partir de la instància aixecada i aquesta ja ha estat aixecada amb la nova AMI creada.

Amb la següent comanda creo la configuració del grup, i la descripció

```
xpuig@xpuig-PC:~$ aws autoscaling create-auto-scaling-group --auto-scaling-group-name
TFG-xpuigv --instance-id i-0043c0912535df892  --min-size 1 --max-size 10 --desired-
capacity 1 --profile xpuigv-UOC
xpuig@xpuig-PC:~$ aws autoscaling describe-auto-scaling-groups --auto-scaling-group-
name TFG-xpuigv --profile xpuigv-UOC
{
    "AutoScalingGroups": [
        {
            "AutoScalingGroupName": "TFG-xpuigv",
            "AutoScalingGroupARN": "arn:aws:autoscaling:eu-west-
3:025974803533:autoScalingGroup:2a77d7d4-1891-4643-b8fe-
77298fa9c980:autoScalingGroupName/TFG-xpuigv",
            "LaunchConfigurationName": "TFG-xpuigv",
            "MinSize": 1,
            "MaxSize": 10,
```

```
            "DesiredCapacity": 1,
            "DefaultCooldown": 300,
            "AvailabilityZones": [
                "eu-west-3c"
            ],
            "LoadBalancerNames": [],
            "TargetGroupARNs": [],
            "HealthCheckType": "EC2",
            "HealthCheckGracePeriod": 0,
            "Instances": [
                {
                    "InstanceId": "i-0e7b2b1a69c1b8f74",
                    "AvailabilityZone": "eu-west-3c",
                    "LifecycleState": "InService",
                    "HealthStatus": "Healthy",
                    "LaunchConfigurationName": "TFG-xpuigv",
                    "ProtectedFromScaleIn": false
                }
            ],
            "CreatedTime": "2018-12-27T17:26:58.812Z",
            "SuspendedProcesses": [],
            "VPCZoneIdentifier": "subnet-bb2787f6",
            "EnabledMetrics": [],
            "Tags": [],
            "TerminationPolicies": [
                "Default"
            ],
            "NewInstancesProtectedFromScaleIn": false,
            "ServiceLinkedRoleARN": "arn:aws:iam::025974803533:role/aws-service-
role/autoscaling.amazonaws.com/AWSServiceRoleForAutoScaling"
        }
    ]
}
xpuig@xpuig-PC:~$
```

Taula 17.awscli: creació del grup auto-escalat

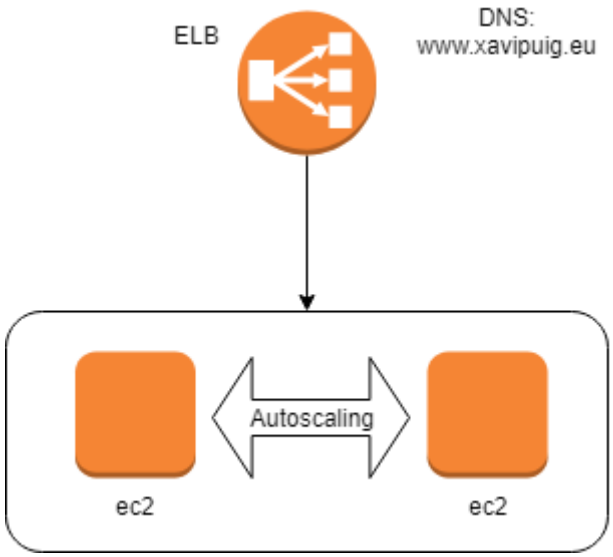
Com que el que volem és que l'ASG sigui el servidor web, hem de donar-li un punt d'entrada en comú i que reparteixi el tràfic. Pensem que cada instància té una ip interna diferent, i el DNS de www.xavipuig.eu apunta a una IP externa. Ens interessa configurar un Elastic Load Balancer (ELB) perquè sigui qui reparteixi el tràfic a l'ASG. Si tenim una única instància aixecada, el tràfic anirà únicament cap aquesta instància, però si l'ASG té més instàncies l'ELB repartirà el tràfic per totes elles.

```
xpuig@xpuig-PC:~$ aws elb create-load-balancer --load-balancer-name elb-TFG --
listeners "Protocol=HTTP,LoadBalancerPort=80,InstanceProtocol=HTTP,InstancePort=80"
--availability-zones eu-west-3a eu-west-3b eu-west-3c
```

```
--security-groups sg-0bb66dc763b330335 --profile xpuigv-UOC
{
    "DNSName": "elb-TFG-1094346005.eu-west-3.elb.amazonaws.com"
}
xpuig@xpuig-PC:~$ aws autoscaling attach-load-balancers --load-balancer-names elb-TFG --auto-scaling-group-name  TFG-xpuigv --profile xpuigv-UOC
```

Taula 18. awscli: creació ELB

Amb la última instrucció assignem l'ELB al grup d'auto-escalat, i d'aquesta manera podem canviar en la zona del nostre domini que el registre www.xavipuig.eu apunti com a registre CNAME a l'adreça de l'ELB.



Il·lustració 17.Auto-escalat

Si consultem el DNS, un cop canviat respon de la següent manera:

```
xpuig@xpuig-PC:~$ dig www.xavipuig.eu
...
www.xavipuig.eu.        300     IN      CNAME   elb-tfg-1094346005.eu-west-3.elb.amazonaws.com.
elb-tfg-1094346005.eu-west-3.elb.amazonaws.com. 60 IN A 35.180.235.16
elb-tfg-1094346005.eu-west-3.elb.amazonaws.com. 60 IN A 52.47.72.180
...
xpuig@xpuig-PC:~$
```

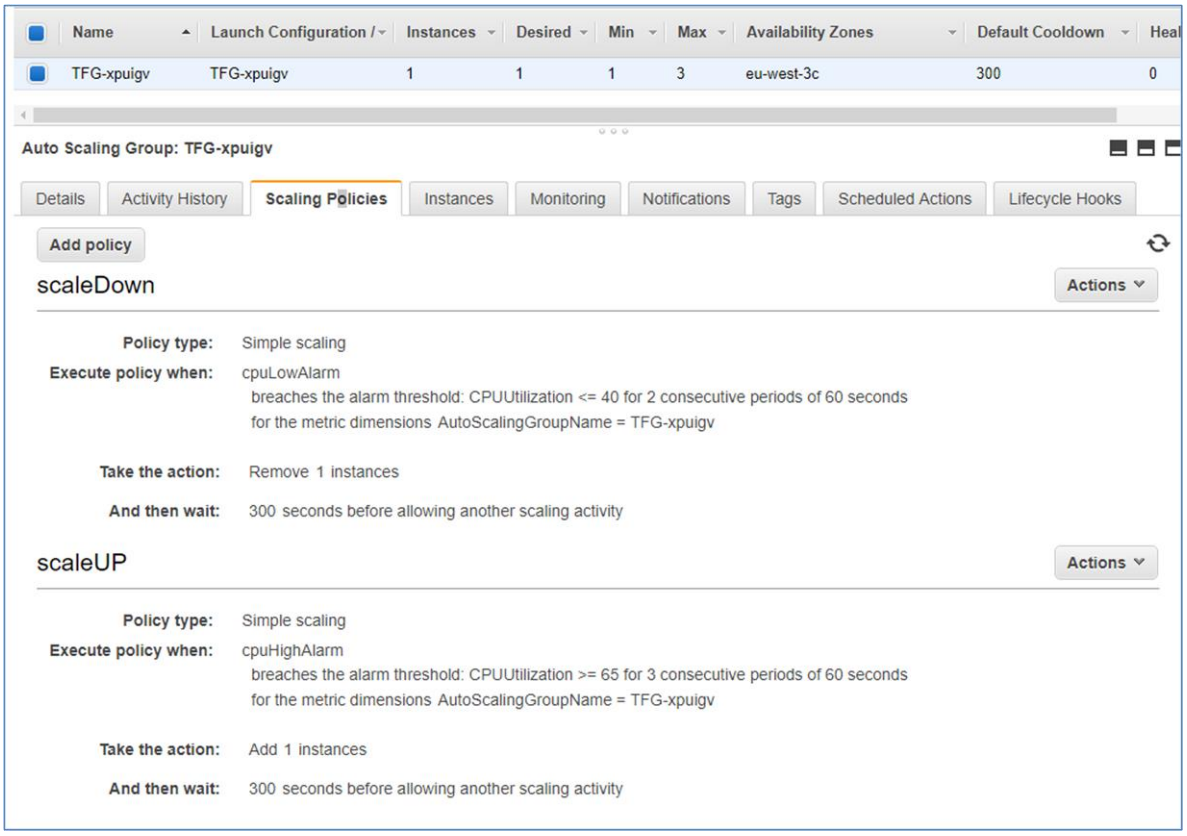
Taula 19.Consulta del registre www.xavipuig.eu

Perquè la plataforma sigui dinàmicament elàstica s'han de crear els esdeveniments perquè l'ASG pugui aixecar instàncies automàticament i també les pugui parar .

Per fer-ho s'utilitzen les alarmes de l'eina de monitorització Amazon Cloudwatch.

En el nostre cas, he creat dues polítiques per auto-escalar:

- Si CPU mig de totes les instàncies de l'ASG supera el 65% d'ús de CPU durant 3 minuts consecutius llavors s'aixecarà una nova instància
- si CPU mig de totes les instàncies de l'ASG està per sota del 40% d'ús de CPU durant 2 minuts consecutius s'apaga una instància.



Il·lustració 18. Polítiques en l'auto-escalat

Hem testejat amb els mateixos paràmetres que hem provat abans amb l'eina d'Apache benchmark ab, però al tenir una base de dades de poca potència el coll d'ampolla era la pròpia base de dades i no podia provar l'autoscaling.

He creat un petit script a la web, del que he tret la connexió a base de dades i que calcula deu hash per pantalla. Al ser les instàncies de mida petita i fer crides concurrents de la mateixa URL provoca un alt ús de CPU que fa saltar l'auto-escalat per introduir noves instàncies al nostre grup.

He llençat la següent comanda

```
xpuig@xpuig-PC:~$ ab -n 1000000 -c 60 -l www.xavipuig.eu/xp.php
```

i això ha provocat, tal com esperàvem, un alt ús de CPU. L'ASG s'ha auto-escalat fins a haver-hi 5 instàncies aixecades, mantenint-se allà mentre han durant les proves.

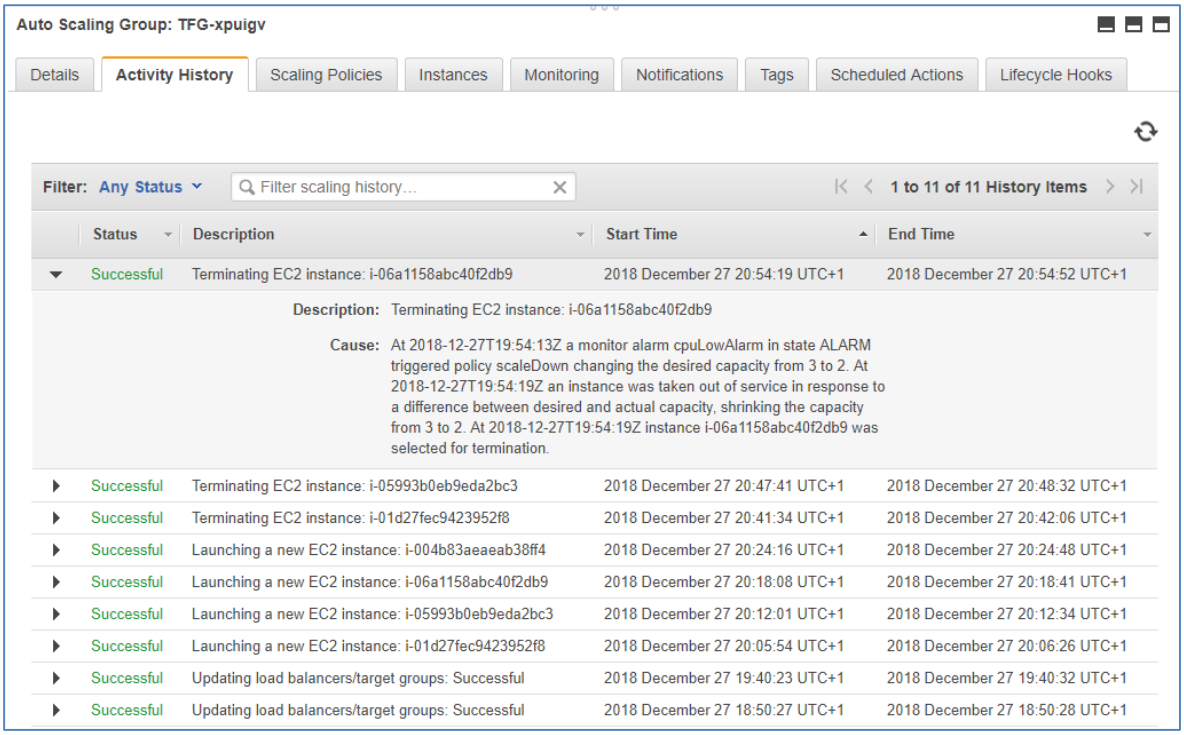
En la gràfica es pot apreciar quan l'ús de CPU és elevat que es van aixecant instàncies, A mesura que s'aixequen instàncies la càrrega es va repartint entre les diferents

instàncies i es pot arribar a servir amb normalitat sense haver de patir per la càrrega en l'aplicació.

En la gràfica es pot veure la línia blava com a ús de CPU i la línia taronja el nombre de instàncies aixecades en aquell moment.

Il·lustració 19. gràfica d'ús de CPU en auto-escalat

En els logs d'activitat del panel d'AWS es pot veure com s'han anat aixecant i apagant instàncies automàticament



Il·lustració 20. Historial d'esdeveniments en l'auto-escalat

En el cas que sabéssim amb anterioritat que la càrrega ha d'incrementar molt de cop, es pot augmentar manual el mínim de instàncies aixecades per estar preparats per rebre una allau de càrrega. Per exemple, ens podem trobar que en una determinada hora es posi a la venta algun producte molt demandat

O bé es pot posar una política d'auto-escalat més restrictiva o més flexible segons les necessitats.

Una altra possible configuració es que l'increment d'instàncies en l'auto-escalat es faci de dos en dos, o de tres en tres instàncies cada vegada. Així, podem obtenir certa tranquil·litat en la plataforma. De totes maneres, l'auto-escalat decreixent es pot mantenir que es pari d'una en una.

Aplicar un capa de cau

Una altra capa que podem aplicar a la nostra arquitectura per poder reduir el treball en el servidor web és el fet d'afegir una capa de cau abans de que el servidor web respongui.

Si el servidor de cau té la pàgina guardada la servirà sense que el servidor web l'hagi de processar. Sinó la té guardada o bé el max-age de la pàgina està superat doncs la tornarà a demanar.

S'ha d'anar amb molt de compte amb quins temps de guardat de pàgina es posen, ja que hi ha pàgines que poden ser més susceptibles a canvis que d'altres.

Imaginem, per exemple, la portada d'un diari. En aquest cas, el temps ha de ser curt, potser se li pot donar un minut per què vagi refrescant la portada. De totes maneres, a les notícies se'ls hi pot assignar un temps més alt, ja que la noticia un cop escrita ja no té perquè canviar.

Per aplicar aquesta capa, he afegit una nova instància amb el software de varnish instal·lat.

La definició del software extreta de la seva documentació és:
"Varnish Cache és un accelerador d'aplicacions web també conegut com un proxy invers HTTP de cau. L'instal·la davant de qualsevol servidor que parli HTTP i el configura per emmagatzemar en memòria cau el contingut. La cau de vernís és molt, molt ràpida. En general, accelera amb un factor multiplicatiu de 300 a 1000, depenent de la seva arquitectura.

He configurat varnish amb una configuració super bàsica. De totes maneres, Varnish es pot configurar molt finament tocant el fitxer en format vlc per a gestionar les cookies, els mètodes de petició (GET, POST, DELETE, HEAD,...) i tenir configuracions per a cada cas.

S'ha d'anar amb cura, perquè si es treballa amb alguna cookie de sessió d'usuari no es guardi la informació en varnish, ja que es podria servir informació igual a diferents usuaris.

A la que apliquem aquesta capa, l´ús de CPU del servidor web es redueix moltíssim.

Els tests que hem llançat han consistit en llençar per forçar l'ús de CPU.

```
xpuig@xpuig-PC:~$ ab -n 1000000 -c 14 -l www.xavipuig.eu/xp.php
```

Amb aquests paràmetres vull aconseguir que s'aixequin unes 3-4 instàncies de l'ASG.

El script.php té un max-age de 10s per tant varnish guarda en memòria aquesta pàgina durant 10 segons, i llavors la demana al servidor web perquè la carregui de nou.

Un cop estiguin aquestes instàncies aixecades, posarem la capa de varnish pel mig, i veurem com podem augmentar la concurrència.

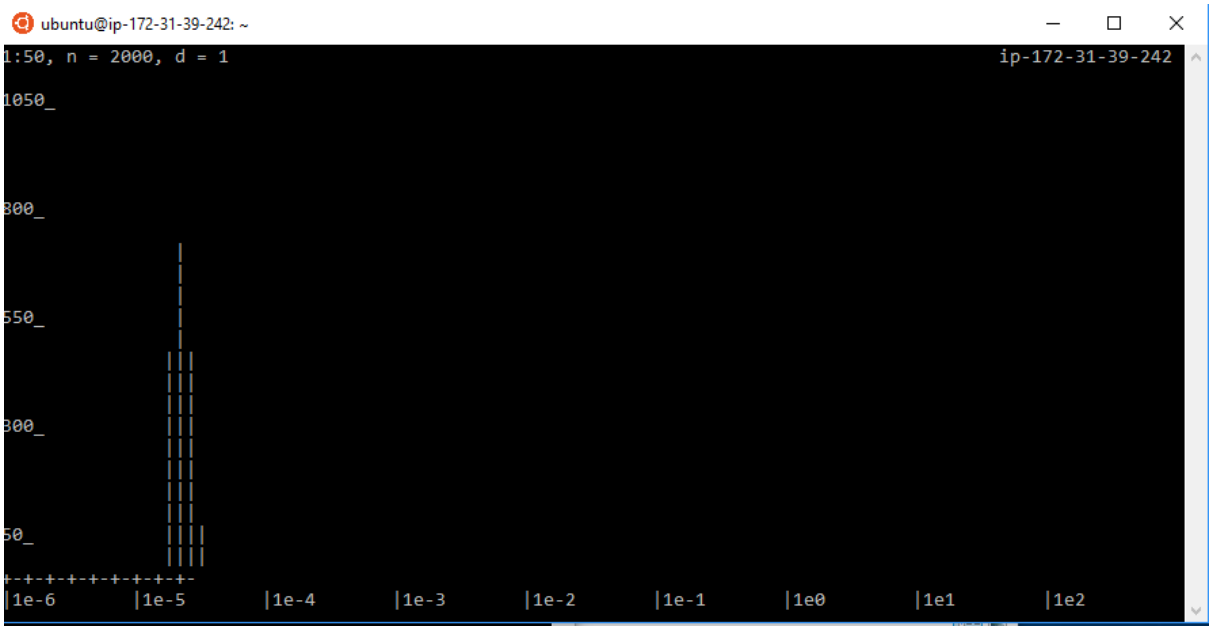
Adjunto captura de pantalla on es poden veure les capçaleres de la URL que peticiono. Es pot veure que la petició té un antiguitat de 8 segons, inferior al max-age de 10 segons que té configurada la pàgina, i que en aquests 8 segons ha tingut 2052 hits a varnish.

```
xpuig@xpuig-PC:~$ curl -I www4.xavipuig.eu/xp.php
HTTP/1.1 200 OK
Cache-Control: max-age=10
Content-Type: text/html; charset=UTF-8
Date: Fri, 28 Dec 2018 01:26:21 GMT
Server: Apache/2.4.29 (Ubuntu)
Vary: Accept-Encoding
X-Varnish: 694905 888055
Age: 8
Via: 1.1 varnish (Varnish/5.2)
X-Cache: HIT
X-Cache-Hits: 2052
Accept-Ranges: bytes
Connection: keep-alive
xpuig@xpuig-PC:~$
```

Taula 20. Capçaleres obtingudes al fer una petició contra el servidor Varnish

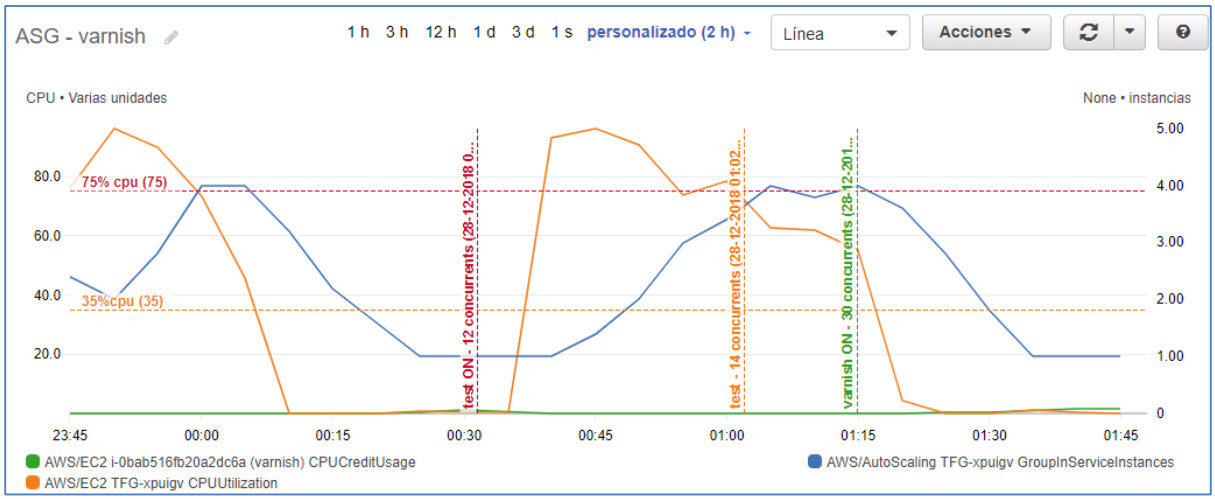
Varnish té una aplicació estadística per veure el nombre de peticions encertades i fallades, i ho mostra en forma de histograma on es pot veure el temps de resposta de les peticions.

L'eix de les X són les unitats de temps en forma exponencial. En la gràfica es pot veure com totes les peticions són hits (|). En cas contrari, apareixerien amb el símbol(#) indicant que és una fallada (miss), i que responen amb un temps de 1x10^-5 segons.



Il·lustració 21. varnish: histograma de hits - misses

A continuació la gràfica de la seqüència extret de l'eina de monitorització proporcionada per AWS. Es pot veure l'ús de CPU de l'ASG, el nombre de instàncies aixecades, i l'ús de CPU de la instàncies



Il·lustració 22. Gràfic on és veu ús de CPU sense cau i amb cau

.

Seguretat en la web (HTTPs)

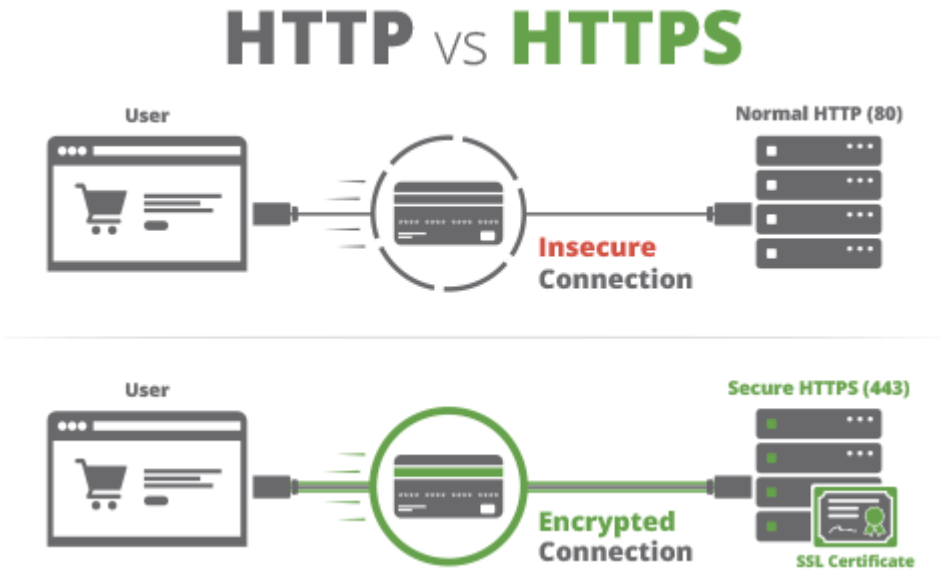
El protocol de transferència de Hipertext (HTTP) utilitzat en les aplicacions web és un protocol utilitzat en sistemes de xarxes, dissenyat amb el propòsit de definir i estandarditzar la sintaxis i la semàntica de les transaccions que es duen a terme entre els equips que conformen una xarxa. Per posar un exemple, la manera com es comunica un servidor web amb els navegadors.

La característica més important d'aquest protocol és que és un sistema orientat a una comunicació petició-resposta. Nosaltres escrivim l'adreça URL i la demanem al servidor web que ens respon.

El protocol HTTPs és el protocol resultant de la combinació entre el protocol HTTP i el protocol criptogràfic SSL/TLS. HTTPs utilitza el protocol HTTP sobre una capa de protecció segura, de manera que el missatge HTTP és xifra abans de la transmissió i desxifrat quan es rep.

Amb tot això, el benefici principal d'utilitzar HTTPs és que el lloc sigui més segurs pels usuaris. Actualment, és imprescindible per a qualsevol aplicació web que utilitzi dades de caràcter personal.

A més a més, amb el protocol HTTPs, l'usuari visitant pot verificar que el domini és nostre i que som propietaris del domini.



Il·lustració 23. HTTP vs HTTPs.

Imatge extreta de https://bit.ly/2LDNcgh

Tot i els avantatges que hem indicat, l'ús d'aquest protocol necessita més recursos que HTTP, ja que cada comunicació s'ha d'encriptar i desencriptar, per tant el rendiment és pot veure una mica afectat.

A més a més, al utilitzar HTTPs también ens millora en l'apartat de SEO (Search Engine Optimization). A ulls de Google, està molt més ben vist un lloc segur que un que no ho sigui, i és un factor de millora en el posicionament web, tot i que aquesta raó no és oficial.

Aplicar HTTPs

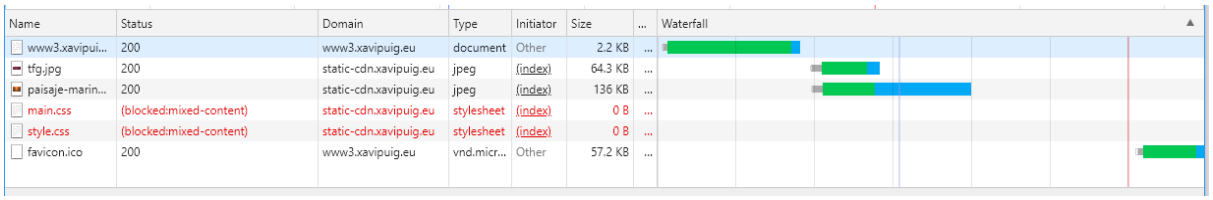
Per poder aplicar aquesta capa de seguretat en la nostra plataforma necessitem d'un certificat SSL. De certificats SSL n'hi ha de pagament i de gratuïts. Pagar a una autoritat certificadora té els beneficis de tenir suport tècnic, però l'encriptació entre els tipus de certificat és la mateixa.

De totes maneres, en el nostre cas, al tenir tota la plataforma creada en AWS, Amazon ens ofereix certificats gratuïts per aplicar-los en els seus serveis.
Els serveis d'AWS que permeten l'ús de certificats generats per ells no són tots. Únicament Elastic Load Balancer dins del servei EC2 i Amazon Cloudfront són els serveis que hem utilitzat que permeten aquest ús.

Així doncs, sol·licito un certificat wildcard per xavipuig.eu a AWS i l'aplico a la nostra plataforma. AWS no permet.

HTTPs treballa pel port 443, per lo que he hagut d'obrir les regles del tallafocs corresponent al port 443 perquè en pugui entrar tràfic.

Actualment, els navegadors acostumen a bloquejar contingut HTTP en pàgines que es serveixen amb HTTPs pel motiu de que s'intenta servir una pàgina segura pero no amb totes les garanties, per tant és important que un cop es decideix aplicar el protocol segur tot el contingut es serveixi en protocol HTTPs.



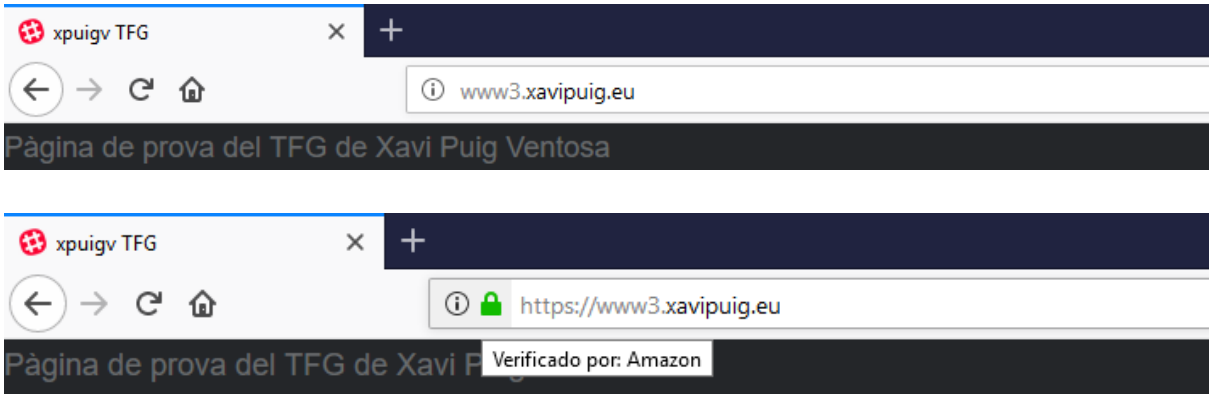
Il·lustració 24. pàgina HTTPs amb contingut no segur

He configurat l'ELB que gestiona el tràfic cap al grup d'auto-escalat perquè escolti també pel port 443 i configurat amb el certificat d'AWS.
També he configurat el servei de Cloudflront amb el certificat generat i d'aquesta manera podrem servir el contingut en HTTPs.

Varnish únicament treballa amb el protocol HTTP, per tant per poder aplicar la capa de varnish en la nova arquitectura he hagut d'afegir un nou ELB configurat com l'anterior.

Els ELB són els dos accessos des de l'exterior de la plataforma que no estan bloquejats al tallafocs per IP. El tràfic de l'ELB cap a dins ja no va encriptat sinó que va utilitza el protocol HTTP. Al no tenir punts oberts, ens assegurem que tot el tràfic a l'ASG prové del ELB del varnish o de l'ELB de l'ASG.

Quan la pàgina se serveix totalment segura amb HTTPs es pot veure pel cadenat que apareix al costat de la URL en els navegadors està de color verd i tancat.



Il·lustració 25. URL servida en HTTP vs HTTPs

Gestió de logs

Cada cop més, les empreses es recolzen en sistemes que generen gran quantitat de dades en format de traces textuals, que tècnicament s'anomenen logs. Aquesta informació no és visible per l'usuari però acostuma a estar relacionada amb la seva activitat informàtica. o amb els sistemes informàtics.

Els logs es solen escriure en fitxers, i ens poden explicar el comportament dels nostres sistemes. De totes maneres, els logs s'han d'analitzar perquè els fitxers sense anàlisis no ens detectaran si es produeix un error o no.

```
2018-12-30T18:07:36.850783Z elb-TFG-varnish 88.13.107.158:59873 172.31.39.242:80 0.000043 0.412128 0.000042 200 200 0 396 "GET http://www3.xavipuig.eu:80/xp.php HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0" - -

2018-12-30T18:07:37.506862Z elb-TFG-varnish 88.13.107.158:59873 172.31.39.242:80 0.000048 0.004229 0.000023 200 200 0 58323 "GET http://www3.xavipuig.eu:80/favicon.ico HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0" - -

2018-12-30T18:07:39.535706Z elb-TFG-varnish 88.13.107.158:59883 172.31.39.242:80 0.000043 0.001037 0.000023 200 200 0 396 "GET http://www3.xavipuig.eu:80/index.php HTTP/1.1" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:64.0) Gecko/20100101 Firefox/64.0" - -
```

Il·lustració 26. varis logs de l'ELB

Amb tota la informació que podem aconseguir dels logs és convenient emmagatzemar correctament els logs, sinó es poden produir varis problemes:

- visibilitat nul·la d'errors per als equips desenvolupament i sistemes.
- increment de temps de resposta davant d'incidències a nivell de servei.
- accessos i informació descentralitzada.

Així doncs, és important centralitzar els logs per facilitar l'anàlisi del sistema i plantejar possibles millores com detecció d'errors, configuració d'alarmes, extracció d'informació, ...

Alguns dels serveis d'AWS ja tenen la possibilitat de que els logs es guardin al servei d'emmagatzemament de S3.

En el cas dels ELB, la pròpia configuració dels ELB permet guarda els logs a S3, únicament cal configurar-ho correctament a través de la consola d'Amazon

En el servei RDS de bases de dades, als logs s'hi pot accedir des de la consola dins dels servei RDS però només hi ha els logs de les últimes 24, per tant si es volen guardar s'ha gestionar mitjançant un script que els descarregui del servidor de base de dades i els copiï al bucket de S3.

Pel que fa al logs d'Apache de les instàncies d'EC2 de l'auto-scalat, s'hauria de configurar algun recol·lector de logs per que els passi a S3.

Una de les solucions més comuns es configurar el recol·lector Fluentd, una solució gratuïta i de codi obert. Amb Fluentd es poden configurar diferents fonts de dades (en el nostre cas, els logs d'Apache) perquè es guardin a S3.

Un cop tinguem els logs centralitzats podem configurar en alguna instància d'EC2 la solució ElasticStack. Es tracta d'un paquet de tres tecnologies (elasticsearch, Logstash i Kibana) i d'altres components útils per monitoritzar logs i fer-ne una tasca molt més senzilla.

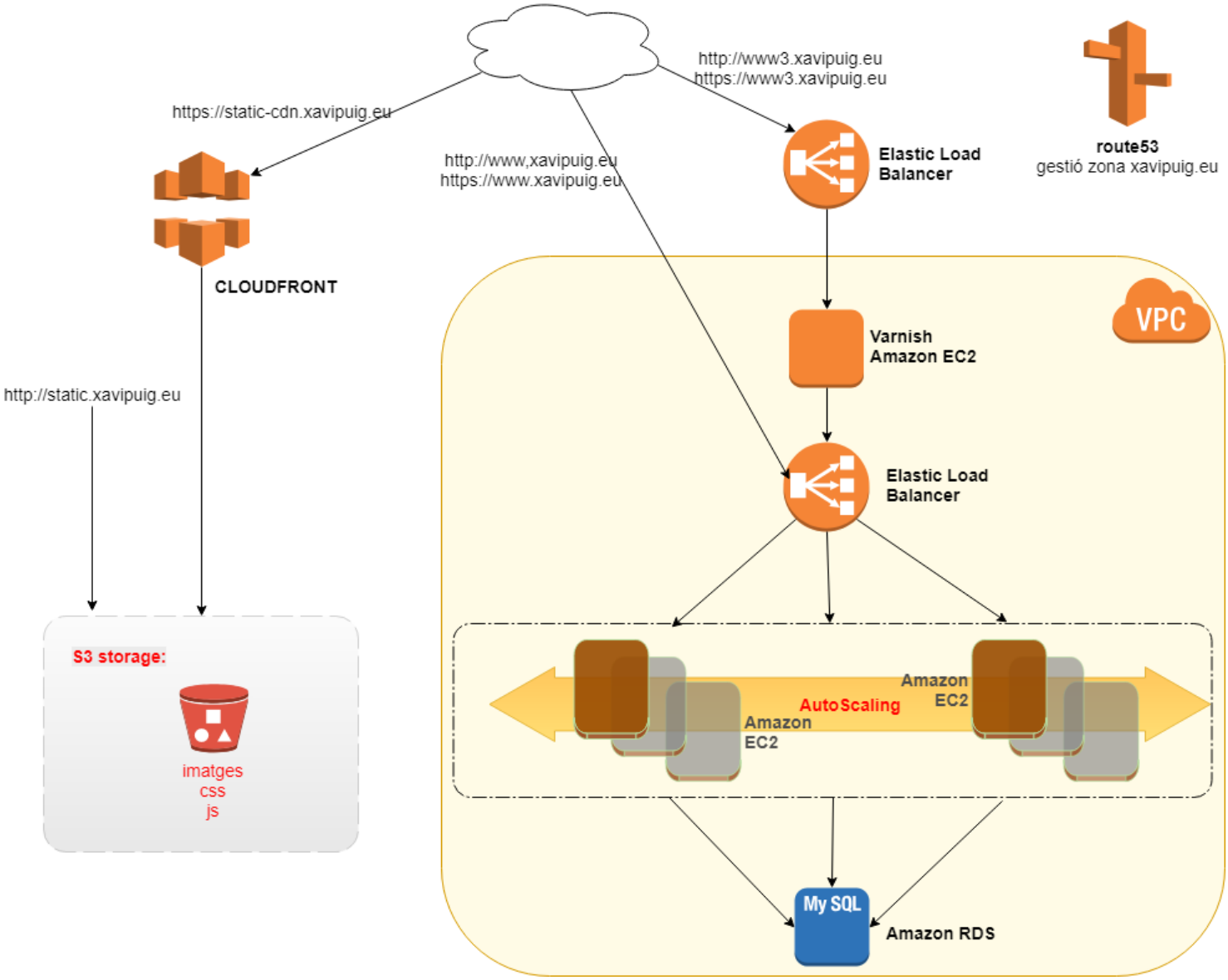
Elasticsearch és la part principal d'aquesta infraestructura i s'encarrega de guardar i realitzar cerques de informació guardada de manera molt ràpida.

Logstash és el servei que s'encarrega de agafar els logs transformar-los perquè es puguin emmagatzemar i mostrar de manera més senzilla en Elasticsearch.

Kibana és el panel que ens mostra la informació que tenim emmagatzemada en Elasticsearch

Esquema de la plataforma creada

Aquest és l'arquitectura creada com a exemple:



Il·lustració 27. Esquema de la plataforma creada

Desplegament de codi

Fins ara, hem vist com obtenir una arquitectura elàstica i escalable, però respecte al codi de la aplicació no n'hem parlat.
Les aplicacions estan vives i el desenvolupadors han de poder aplicar millores i noves funcionalitats, i també arreglar errors que s'hagin pogut aplicar sense adonar-se. Per aquest motiu, és molt important poder fer desplegament de codi en la plataforma que acabem d'aixecar.

AWS té el seu propi servei per fer aquesta funció, CodeDeploy. Aquest servei permet fer el desplegament de codi en instàncies i en grups d'auto-escalat.

A partir d'un "grup de desplegament" , en el nostre cas, seria el grup d'auto-escalat, es pot escollir quin tipus de desplegament volem:

- **blue/green.** Reemplaça les instàncies del grup per unes noves instàncies amb l'última versió del codi. Quan les noves instàncies s'han registrat en l'ELB, les velles en són tretes i es poden eliminar.
- **in-place.** Ens permet actualitzar el codi de l'aplicació dins de les instàncies del grup de desplegament. Mentre es fa l'actualització de codi, cada instància deixa de rebre tràfic i està fora de servei fins que finalitza i fa la següent.
  En aquest tipus de desplegament, es pot donar la circumstància que conviuen dos versions de codi diferents, ja que un cop s'ha fet l'actualització en la primera instància, la resta encara tenen el codi antic.
  Per aquesta raó, el servei dona la opció d'escollir quantes es pot decidir quantes instàncies es volen desplegar a l'hora. S

Per que els desplegaments a través d'aquest servei funcionin bé, és necessari que les instàncies estiguin darrera un ELB. És el ELB qui posa les instàncies fora de servei mentre es fa l'actualització de codi.

El desplegament automàtic es pot realitzar agafant el codi de S3 o bé des de el servei extern a AWS de github.

Aplicar un nou desplegament

Per poder utilitzar el servei de codeDeploy d'AWS, primer s'han de donar permisos perquè l'aplicació pugui fer el desplegament.

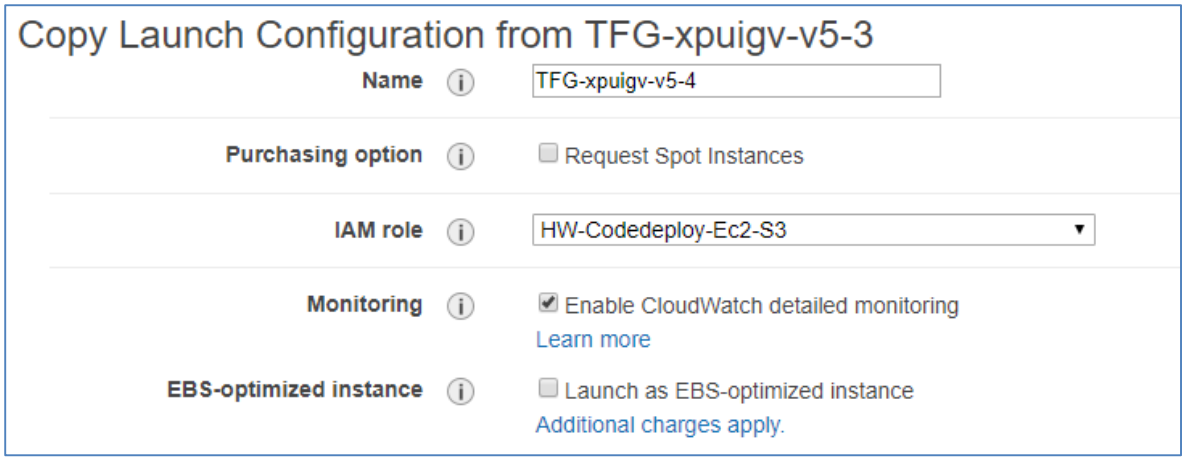
Lo primer és configurar l'agent de codeDeploy en les instàncies aixecades. Per fer-ho, ens connectem a una d'elles, instal·lem l'agent que ens facilita AWS i generem una nova AMI a partir d'aquesta instància.:

```
xpuig@xpuig-PC:~$ssh -i .aws/uoc-key.pem ubuntu@ec2-52-47-81-175.eu-west-3.compute.amazonaws.com
Welcome to Ubuntu 18.04.1 LTS (GNU/Linux 4.15.0-1031-aws x86_64)
ubuntu@ip-172-31-38-13:~$ wget https://aws-codedeploy-eu-west-3.s3.amazonaws.com/latest/install
ubuntu@ip-172-31-38-13:~$ chmod a+x ./install
ubuntu@ip-172-31-38-13:~$ sudo ./install auto
I, [2019-01-03T19:58:26.628243 #6558]  INFO -- : Starting Ruby version check.
I, [2019-01-03T19:58:26.628315 #6558]  INFO -- : Starting update check.
I, [2019-01-03T19:58:26.628342 #6558]  INFO -- : Attempting to automatically detect supported package manager type for system...
I, [2019-01-03T19:58:26.636881 #6558]  INFO -- : Checking AWS_REGION environment variable for region information...
I, [2019-01-03T19:58:26.636932 #6558]  INFO -- : Checking EC2 metadata service for region information...
I, [2019-01-03T19:58:26.685093 #6558]  INFO -- : Running version 1.0-1.1597
I, [2019-01-03T19:58:26.685159 #6558]  INFO -- : Downloading version file from bucket aws-codedeploy-eu-west-3 and key latest/VERSION...
I, [2019-01-03T19:58:26.710223 #6558]  INFO -- : Running version matches target version, skipping install
I, [2019-01-03T19:58:26.710260 #6558]  INFO -- : Update check complete.
I, [2019-01-03T19:58:26.710281 #6558]  INFO -- : Stopping updater.
ubuntu@ip-172-31-38-13:~$ sudo service codedeploy-agent start
ubuntu@ip-172-31-38-13:~$ exit

xpuig@xpuig-PC:~$ aws ec2 create-image --instance-id i-0bd957c57867f0c94 --description "ami for ASG" --name "ASG-AMI-v5" --no-reboot --profile xpuigv-UOC
{
    "ImageId": "ami-092b4f25de2b0b83a"
}
xpuig@xpuig-PC:~$
```

Taula 21. instal.lació agent codeDeploy

A continuació, creem un rol per poder accedir a les aplicacions de EC2,CodeDeploy i S3. Aquest rol l'assignarem a les instàncies aixecades, i per això s'ha de crear un nova configuració de llançament de l'autoScaling, i indicar-li a l'actual que arranqui amb aquesta configuració. La nova configuració arrancarà amb la AMI que acabem de crear.
El rol creat l'he anomenat HW-CodeDeploy-Ec2-S3.

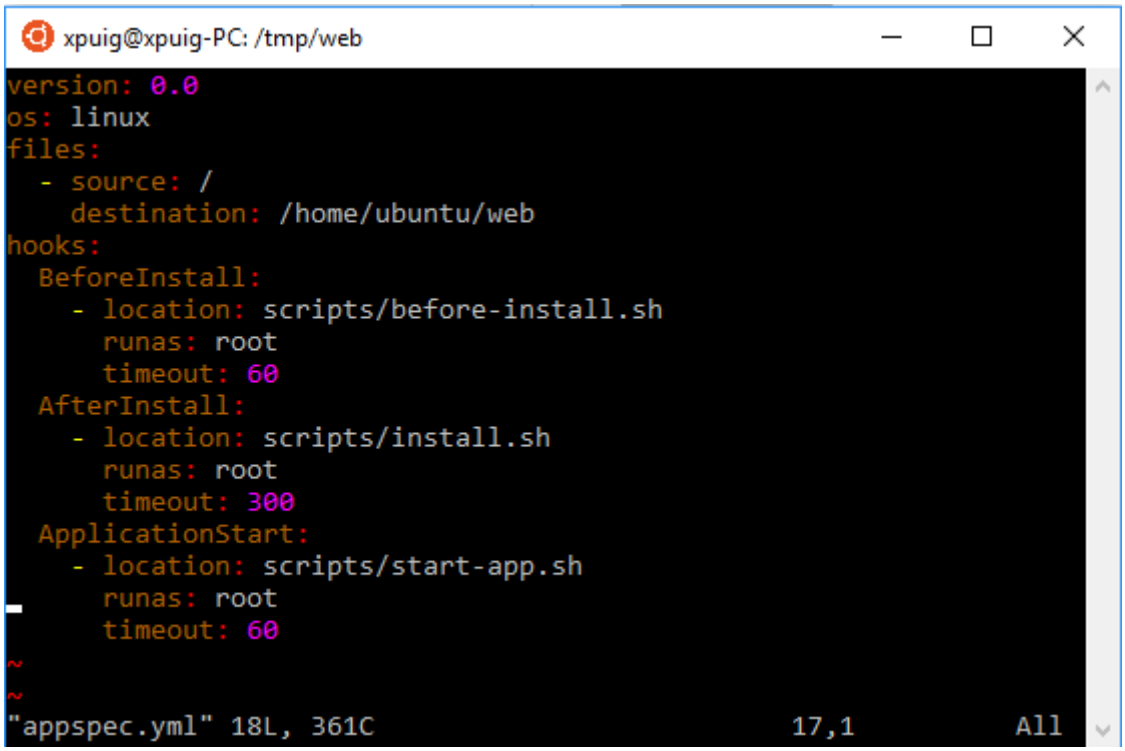


Il·lustració 28. Assignació de IAM Role a la configuració de la instància

Com mencionava anteriorment, CodeDeploy només suporta S3 i Github com a repositori de codi. En el nostre cas, ho farem des de S3, on he deixat un fitxer zip en la ruta s3://tfg-xpuigv-code/AWS-TFG.zip

Jo hem configurat el bucket perquè guardi les versions dels fitxers, per tant puc anar pujant versions de codi amb el mateix nom, i el S3 s'encarrega de guardar-les.

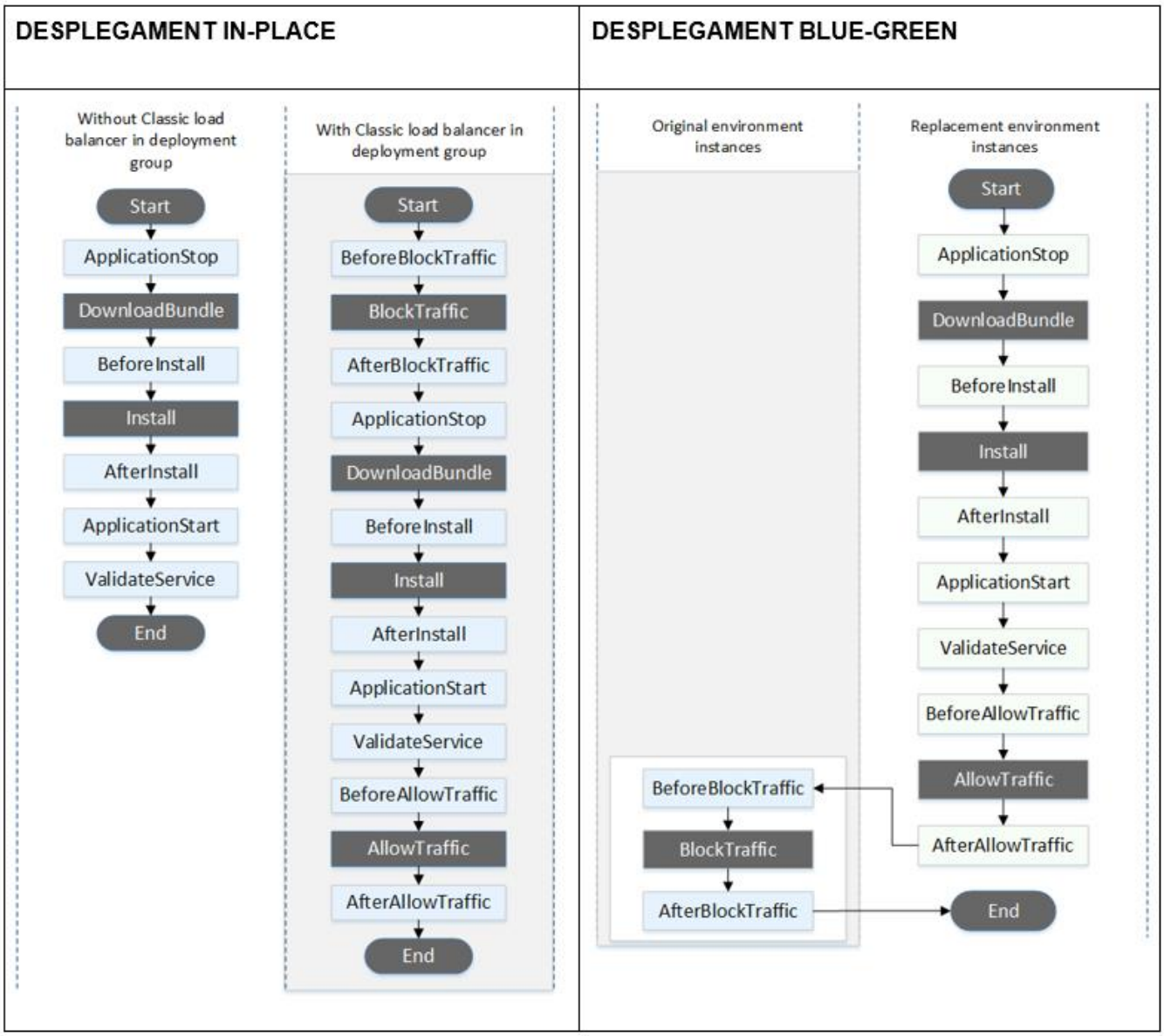
En el codi de la nostra aplicació s'hi ha d'afegir un fitxer appsec.yml, on hi ha la configuració i els passos a seguir durant el desplegament amb codeDeploy.



Il·lustració 29. fitxer appsec.yml de configuració de codeDeploy

Els hooks beforeInstall, AfterInstall i applicationStart, són scripts que es poden configurar per que la nostra aplicació funcioni correctament.
Son scripts que es poden aprofitar per esborrar alguna carpeta antiga, reiniciar Apache un cop l'aplicació està descomprimida, es podria fer un pre-escalfament de cau, …
A més a més d'aquests hooks, se'n poden configurar més.

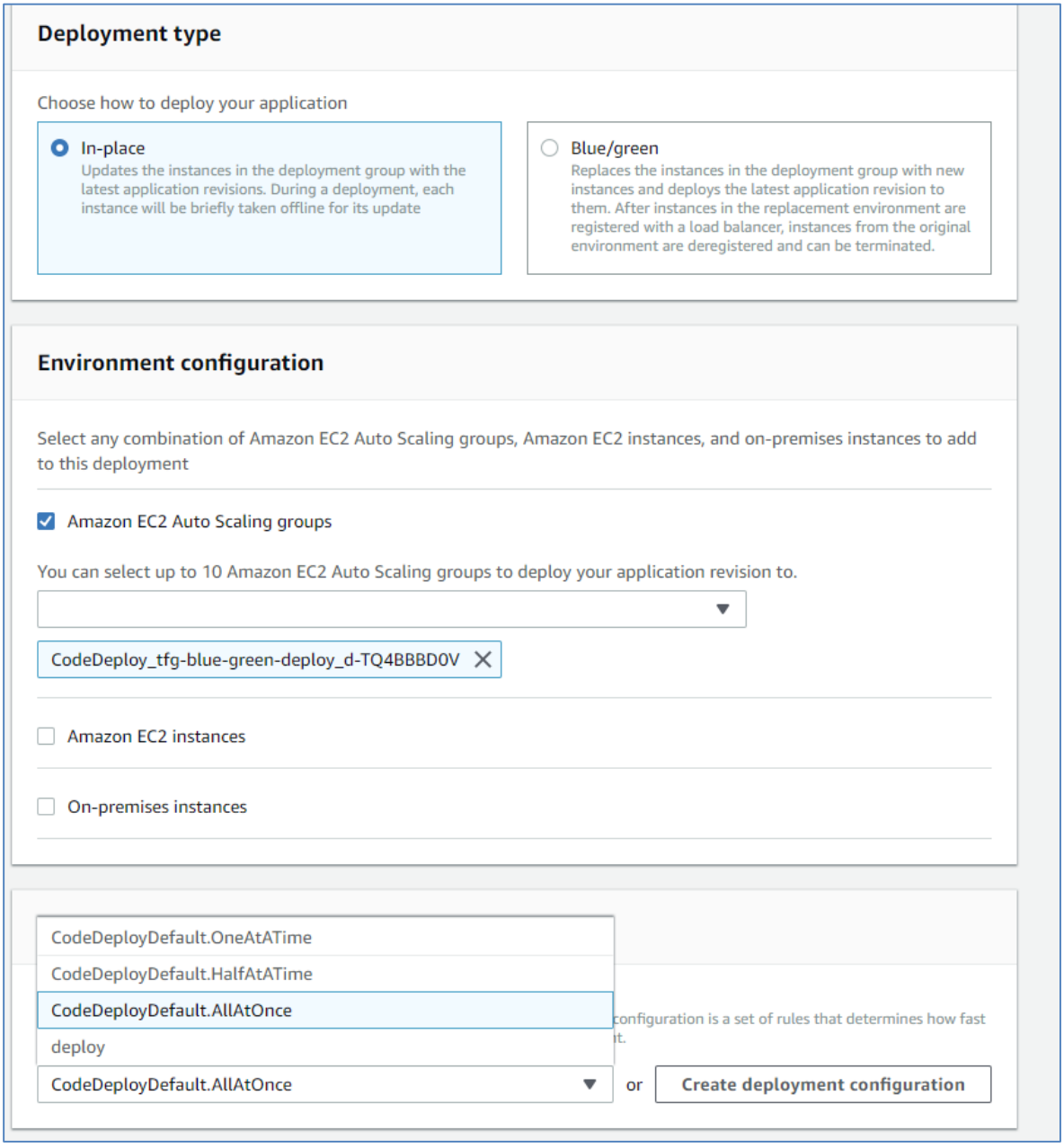
Adjunto dues imatges extretes de la documentació d'AWS on es pot veure el cicle de vida del desplegament:



Il·lustració 30. CodeDeploy: esquemes de desplegament

El primer pas en la configuració del desplegament és crear una aplicació. En el meu cas, l'he anomenat tfg-xpuigv.

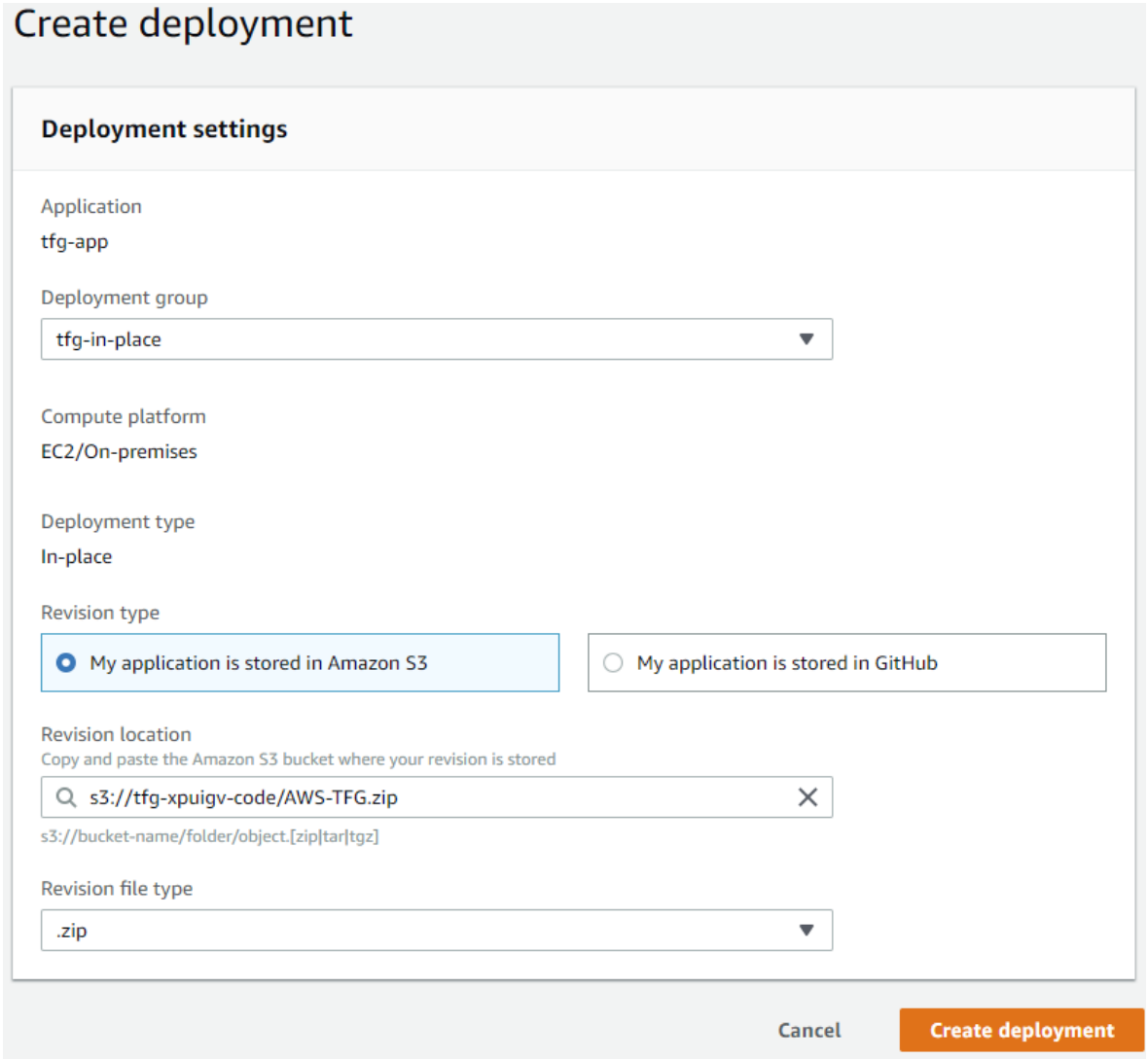
Després s'ha de crear el "grup de desplegament", aquí és on s'escull quin tipus de desplegament volem (in-place, blue-green). Si és in-place, és pot escollir si es fa una darrera l'altra, o bé primer una meitat i després l'altre, o bé totes a la vegada.
En el cas de blue-green, s'ha d'escollir que es fa amb les instàncies velles: es poden deixar enceses per analitzar-les, es poden apagar al cap d'una estona o bé immediatament.



Il·lustració 31. CodeDeploy: configuració de l'entorn

En el meu exemple, he realitzat dos grups. Un in-place i l'altre blue-green.

A partir dels grups creats, és on podem decidir fer el desplegament, s'indica la ruta d'origen a S3, i es crea el l'execució.



Il·lustració 32. CodeDeploy: creació d'un desplegament.

Depenen del nombre de instàncies que hi hagi aixecades, un tipus de desplegament pot ser més ràpid:

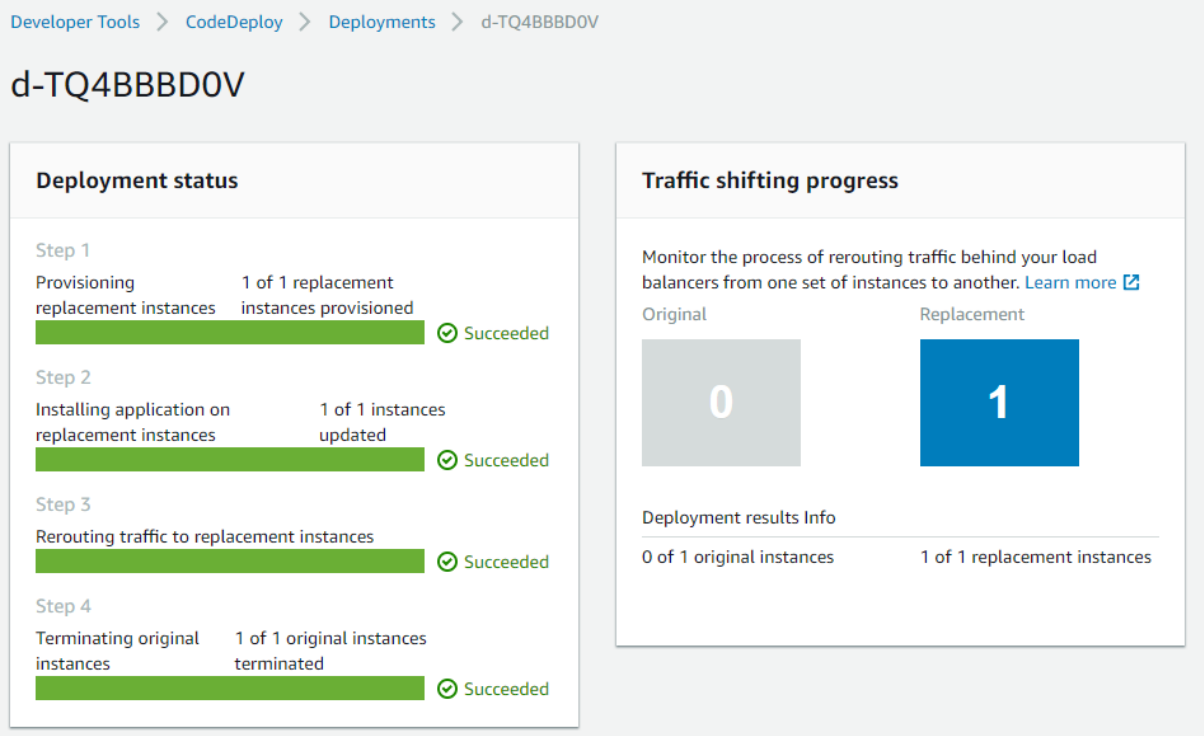
- In-place ha tardat en una instància prop de 1min30 segons per fer tot el procés.
- En el desplegament Blue-green, el temps de desplegament ha estat de 5 minuts.

◉	d-TQ4BBBD0V	Succeeded	Blue/green	EC2/On-premises	tfg-app	tfg-blue-green-deploy	s3://tfg-xpuigv-code/AWS-TFG.zip	user	Jan 3, 2019 8:35 PM	Jan 3, 2019 8:40 PM
○	d-GYQFHBI0V	Succeeded	In-place	EC2/On-premises	tfg-app	tfg-in-place	s3://tfg-xpuigv-code/AWS-TFG.zip	user	Jan 3, 2019 8:32 PM	Jan 3, 2019 8:34 PM

Il·lustració 33. CodeDeploy:Temps de desplegament

Si haguéssim tingut 5 instàncies aixecades, al fer-ho in-place ens hagués trigat de l'ordre de 7-8 minuts.
En canvi, el desplegament blue-green ens hagués trigat 5 minuts igualment.

Aquest és el resultat del desplegament blue-green (per una instància),



Il·lustració 34. CodeDeploy: desplegament blue-green

Cal comentar que si un desplegament no acaba bé, es pot fer un rollback fàcilment. I si volem desplegar una versió antiga de codi, com que a S3 tenim el fitxer amb les versions, podem indicar el fitxer a S3 amb la versió i es desplegarà una versió anterior.

Amb tot, crec que el tipus de desplegament a escollir depèn del tipus d'aplicació que tinguem i/o de la situació, ja que hem de preveure si podem tenir dues versions de codi corrent a la vegada.

Control dels costos

Un dels temes més importants que s'han de tenir en compte quan ens plantegem una aplicació web, i més en una arquitectura escalable i elàstica dinàmicament és el cost que pot suposar.

Sinó és va amb una mica de cura, el fet que creixi i decreixi automàticament pot fer augmentar el cost total. És clar, de totes maneres, que si s'ha realitzat una bona tasca en la configuració de l'aplicació i l'arquitectura, i s'ha previst uns límits correctes, que l'arquitectura creixi vol dir que tenim tràfic i que per tant el negoci que oferim funciona.

Pel que fa als preus, cal dir que en AWS i en la resta de plataformes de serveis d'internet, els preus depenen de l'ús individual de cada un dels serveis utilitzats. L'ús total combinat de cada servei ens donarà la factura mensual.

En el cas del servei de Amazon S3, els preus es basen en cinc components:

- tipus d'emmagatzemament utilitzat
- ubicació del contingut , varia segons la zona.
- volum de dades guardades.
- nombre de sol·licituds per emmagatzemar o retirar el contingut.
- volum de dades transferides des de S3 cap fora.

En el servei de computació EC2, els preus van determinats per les següents característiques:

- el tipus d'instància escollida.
- la regió on es troba la instància.
- el software executat (Linux, Windows, …)
- model de preus (sota demanda, en subhasta, capacitat reservada).

Depenen de totes aquestes opcions, tenim un preu x hora utilitzada.

Comentar que el servei d'auto-escalat no té un valor afegit. Com hem vist, es defineixen una política de instàncies mínimes i màximes, i el servei va posant i traient. El propi ús d'aquest servei implica que ho gastem en instàncies ec2.

El servei Elastic Load Balancing, que en el nostre exemple n'utilitzem dos, es basen en dos components principals:

- el nombre d'hores que executem el ELB
- volum de dades en GB de dades transferit a través de cadascun dels ELB.

Amazon RDS, com a servei de bases de dades relacional, té un cost basat en:

- la classe de instància de base de dades
- les zones de disponibilitat
- la mida de dades que es pot necessitar
- la transferència de dades.

En el cas del servei de Cloudfront, el preus es basen en:

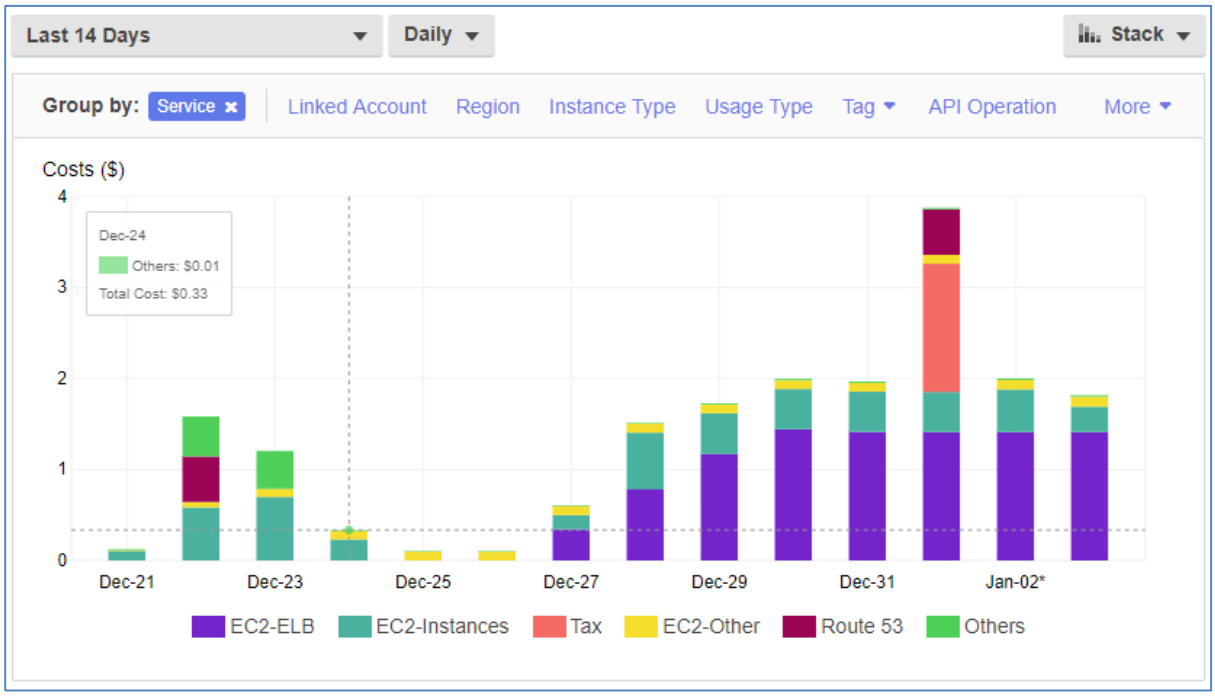
- volum de dades de CDN cap a internet
- nombre de peticions
- tipus de peticions (http o https)

En aquest cas, el tràfic des de l'origen de l'objecte a S3 cap al CDN és gratuït.

Com podem comprovar, cada servei té el seu propi càlcul del cost, però la pròpia plataforma ens ofereixen eines per tenir-ho tot sota control.

Conjuntament amb el servei d'AWS Cloudwatch i del panel de costos, podem saber quin cost estem tenint en cada moment agrupat per serveis o etiquetes.

De les etiquetes (tags) no n'he parlat fins ara, però AWS aconsella fer-ne ús ja que facilita per tenir una visió clara del cost que pot tenir cada servei. I més, en el cas que tinguem varies aplicacions en els serveis d'AWS.



Il·lustració 35. Gràfic del panel de costos

Per altra banda, i per tenir un control molt més exhaustiu durant els períodes de facturació, AWS permet fer pressupostos i crear-ne alertes per què ens avisi en cas que es superi la previsió que hem fet.

Amb tantes variables, pot semblar difícil realitzar un pressupost del que pot costar tenir la infraestructura a AWS, però ens faciliten una eina per fer-ne càlculs (https://calculator.s3.amazonaws.com/index.html).

A més a més, AWS permet tenir grans descomptes si reservem l'ús els serveis durant un any. Si, per exemple, tenim clar que sempre tindrem dues instàncies en l'auto-escalat aixecades, aquestes dues instàncies les podem reservar anualment. El mateix pot

passar amb la instància de bases de dades. Si ho féssim, obtenim uns descomptes d'un 30-35% respecte el pagament sota demanda per hores.

AWS intenta seguir una filosofia de preus orientada al client. Si és bo pel client, acaba essent bo per ells. En la imatge següent, es pot veure clarament:

Il·lustració 36. Filosofia de costos d'AWS.

Imatge copiada de: https://bit.ly/2LT3uSk

Conclusions

Un cop acabat el treball, penso que s'ha complert amb les especificacions sol·licitades i que s'ha assolit amb els objectius plantejats.

En aquests moments, recordo una de les primeres converses amb el tutor un cop ja vàrem enfocar el tema cap el que ha estat finalment. Li vaig comentar que veia el tema molt teòric i que trobava a faltar un objectiu més pràctic. La seva resposta va ser que ja anirien sortint coses i que no patís per no trobar exemples on aplicar-ho. Doncs, finalment, trobo que n'han sortit moltes de coses.

Durant el desenvolupament del projecte, m'he adonat que temes que inicialment creia que serien més importants o que donarien per més del que finalment he acabat exposant. Això em satisfà ja que, sense perdre l'objectiu final, t'adones que el treball no ha estat una línia recta des de el principi, sinó que l'he anat creant a mesura que anava avançant.

Personalment, el treball m'ha ajudat a posar en pràctica coneixements aconseguits a través de les assignatures del grau, però també cal comentar els coneixements adquirits mentre desenvolupava aquest estudi.

En els inicis, els meus coneixements sobre desplegaments al núvol, sobre les possibilitats que permeten les noves plataformes de serveis i com aprofitar aquests serveis eren molts petits.

Un cop finalitzat, puc dir que he aprés molt. I m'emporto la sensació que encara tinc molt més aprendre. Jo únicament he treballat sobre els serveis més indispensables per muntar arquitectures escalables i elàstiques, però aquestes grans plataformes ofereixen una gran quantitat de serveis extres que es poden arribar a aplicar per fer una aplicació molt robusta i tolerant a fallades.

Durant l'execució, procés de muntatge i proves realitzades durant el desenvolupament, m'han fet estar atent al control de costos de la infraestructura muntada. Tot i ser una infraestructura petita, ha tingut un petit cost que sempre volia que fos el més petit possible. Per tant, he aconseguit certa destresa en aixecar i parar la plataforma mentre no m'hi dedicava.

Bibliografia

[1] Lloc web documentació dels serveis d'Amazon AWS
https://aws.amazon.com/es/

[2] Lloc web documentació sobre el client d'aws: awscli
https://docs.aws.amazon.com/cli/latest/reference/index.html#cli-aws

[3] Lloc web documentació dels serveis de Gloogle Cloud Platform
https://cloud.google.com/products/?hl=es

[4] Lloc web sobre explicació dels serveis d'AWS en termes senzills.
https://www.expeditedssl.com/aws-in-plain-english

[5] Lloc web comparativa sobre client-side vs server-side
https://spin.atomicobject.com/2015/04/06/web-app-client-side-server-side/

[6] Lloc web sobre els costos d'un lloc web.
https://aws.amazon.com/es/getting-started/projects/build-wordpress-website/services-costs/

[7] Lloc web explicació de llenguatges web
https://www.piensasolutions.com/blog/principales-lenguajes-programacion-web/

[8] Lloc web documentació sobre el software Fluentd
https://docs.fluentd.org/v1.0/articles/apache-to-s3

[9] Lloc web sobre introducció a l'eina de gestió de logs ELK
https://www.elastic.co/webinars/introduction-elk-stack

[10] Lloc web sobre fer un desplegament amb codeDeploy
https://read.acloud.guru/zero-downtime-deployment-with-aws-codedeploy-and-auto-scaling-groups-8ed002dd2d42

[11] Lloc web documentació del servei codeDeploy
https://aws.amazon.com/es/blogs/devops/use-aws-codedeploy-to-deploy-to-amazon-ec2-instances-behind-an-elastic-load-balancer-2/

[12] Lloc web calculadora de costos a AWS
https://calculator.s3.amazonaws.com/index.html

[13] Lloc web sobre els costos d'un lloc web.
https://www.ticportal.es/temas/cloud-computing/amazon-web-services/precios-aws

[14] Lloc web documentació sobre apache benchmark
https://httpd.apache.org/docs/2.4/programs/ab.html