Fonaments i plataformes de *cloud computing*

PID_00286163

Remo Suppi Boldrito

Temps mínim de dedicació recomanat: 5 hores

Universitat Oberta de Catalunya

Remo Suppi Boldrito

Enginyer de Telecomunicacions. Doctor en Informàtica per la Universitat Autònoma de Barcelona (UAB). Professor del Departament d'Arquitectura de Computadors i Sistemes Operatius de la UAB.

L'encàrrec i la creació d'aquest recurs d'aprenentatge UOC han estat coordinats pel professor: Josep Jorba Esteve

Segona edició: febrer 2022

Av. Tibidabo, 39-43, 08035 Barcelona
Autoria: Remo Suppi Boldrito
Producció: FUOC

Índex

Introducció

El *cloud computing* (o els seus equivalents en català, com ara *informàtica en núvol*, *computació en núvol*, *serveis en núvol*, *núvol de còmput* i *núvol de conceptes*) és una proposta tecnològica adoptada, avui dia, per la societat en general com a forma d'interacció entre proveïdors de serveis, gestors, empreses, administracions i usuaris finals per prestar serveis i utilitzar recursos en l'àmbit de les tecnologies de la informació i la comunicació, que està sustentada per un model de negoci viable econòmicament.

Una definició interessant que aporta més precisió sobre el *cloud computing* és la que formula l'Institut Nacional d'Estàndards i Tecnologia (NIST) dels Estats Units, que afirma el següent:

> «el *cloud computing* és un model que permet l'accés ubic, a conveniència, a demanda i per xarxa a un conjunt compartit de recursos informàtics configurables (per exemple, xarxes, servidors, emmagatzematge, aplicacions i serveis) que es poden aprovisionar i alliberar ràpidament amb el mínim esforç d'administració o sense interacció del proveïdor de serveis» [Dcc11].

Aquest paradigma, inqüestionable pel que fa a la seva acceptació actual, vincula proveïdors, usuaris i tota la cadena intermèdia de transformació i processament de la informació, a través d'una xarxa que en la majoria dels casos és internet. És habitual que els usuaris utilitzin serveis en el *cloud* (ja siguin serveis empresarials o personals), com per exemple correu, xarxes socials i emmagatzematge d'arxius (fotos, vídeos, ofimàtica), i que els utilitzin des de diversos dispositius i amb diferents interfícies. Tan usual és aquest paradigma que els usuaris digitals (joves generacions) no coneixen una altra manera de fer-ho, ja que, d'una banda, han nascut amb aquesta tecnologia, que, juntament amb la mòbil, ha transformat la societat actual, i, de l'altra, la interconnectivitat i la usabilitat de diferents interfícies i mitjans d'accés són essencials per als seus afers quotidians (compres, oci, serveis administratius, negoci i una llarga llista més).

1. Breu història

Si bé tenim presents com a precursores les grans companyies que van oferir aquest tipus de serveis (per exemple, Yahoo! el 1994 i Google el 1998 [motor de cerca], Hotmail el 1997 [correu] i Amazon el 2002 i AWS el 2006), el *cloud computing* és un concepte que té les arrels als anys seixanta i que està basat en les idees de John McCarthy, professor emèrit de la Universitat de Stanford i cofundador del Laboratori d'Intel·ligència Artificial del MIT, on va predir, el 1961 durant un discurs per al centenari de la institució, que la tecnologia de temps compartit (que estava en auge en aquell moment) de les computadores podria conduir a un futur en què el poder del còmput i fins i tot aplicacions específiques es podrien vendre com un servei.

Aquesta idea, encara que des d'un altre punt de vista, també va ser expressada per altres investigadors, especialment Joseph Carl Robnett Licklider (conegut com a J. C. R. o simplement *Lick*), que va ser una figura molt important en l'àmbit de la ciència computacional, recordat particularment perquè va ser un dels primers que va imaginar la computació interactiva moderna i la seva aplicació a diferents activitats amb una visió de la interconnexió global d'ordinadors (molt abans que es construís). J. C. R. va treballar i va aportar finançament a projectes com ara ARPANET (predecessor d'internet) o la interfície gràfica d'usuari, que va permetre l'accés a serveis i recursos a persones no especialistes; són idees que els experts actuals coincideixen a afirmar que s'assemblen molt a la idea de *cloud computing* tal com la coneixem avui dia.

Però la història del *cloud computing* també se sosté en visionaris com ara John Burdette Gage, que quan treballava a Sun Microsystems va vaticinar «The network is the computer», o, més recentment, George Gilder (cofundador del Discovery Institute), que, més enllà de les seves controvertides opinions en altres àmbits, el 2006, a l'article «The Information Factories» [Tif06], va afirmar que el PC ja no era el centre d'atenció i d'emmagatzematge i processament de la informació, sinó que ara ho era el *cloud*, i que aquest serà el responsable d'emmagatzemar les dades de cada individu al planeta i que s'hi accedirà almenys una vegada a la vida. El 2013, Gilder va publicar un llibre, *Knowledge and Power. The Information Theory of Capitalism and How It Is Revolutionizing Our World*, en què relaciona l'economia, el capitalisme i la teoria de la informació d'Alan Turing i Claude Shannon i com afecten la societat actual.

Les evolucions des dels anys seixanta han estat notables i intrèpides, com es pot veure al *Timeline of Computer History* [Tch15], però probablement els desenvolupaments més vinculats al *cloud* actual són el progrés del web (Tim Berners-Lee proposa les idees el 1989 però fins al 1990 no hi ha un prototip del

World Wide Web) i la seva evolució més recent al web 2.0 i el desplegament d'internet (que a Espanya no té impacte fins a principis dels anys noranta, amb la transformació de RedIRIS i la connexió plena a internet).

A partir d'aleshores comencen a sorgir una sèrie de serveis, bàsicament aplicacions centrades en la cerca i el correu i pàgines com per exemple Wandex el 1993, que volia ser un programa per mesurar el volum d'internet, desenvolupat al MIT, però que va acabar sent el primer cercador, i després Yahoo! el 1994, Altavista el 1995 (que ha anat canviant de propietaris: Overture, Yahoo! i ara Microsoft), Hotmail el 1996 (comprat posteriorment per Microsoft el 1997, quan ja tenia 6 milions d'usuaris de correu), o més recentment també les cerques de Google el 1998.

Per a alguns autors, una de les grans fites vinculades al *cloud* és l'arribada el 1999 de Salesforce.com (empresa que ha estat considerada la més innovadora del món per Forbes en els últims quatre anys), que va ser una de les primeres a oferir als clients el que avui s'anomena *programari com a servei* (SaaS), i que era un programari per automatitzar les vendes mitjançant una simple pàgina web. Alguns experts pensen que aquest servei actualment pot ser una *plataforma com a servei* (PaaS) i uns altres ho consideren una plataforma híbrida, ja que ofereix les dues variants. Això va permetre mostrar una «nova» forma de negoci a través d'internet i generar el que avui és una realitat en diferents nivells d'aplicacions, serveis i recursos a la xarxa, i que s'anomena *cloud computing*.

El 2002 Amazon anuncia la seva infraestructura a la xarxa, que s'amplia el 2006 amb el llançament d'Amazon Web Services (AWS): Elastic Computer i Storage Cloud (EC2 i S3) són els serveis més importants d'AWS, per proveir instàncies d'un servidor a demanda per al còmput i l'emmagatzematge, respectivament. Es tractava d'un servei orientat al negoci que va permetre a les petites empreses i als particulars usar equips en què s'executessin les seves aplicacions i amb un model de monetització basat en el pagament per ús. El 2009 Google i altres grans proveïdors van començar a oferir aplicacions basades en el navegador: el seu ús es va popularitzar i van guanyar en seguretat i servei. També van implementar les seves pròpies infraestructures *cloud*, per exemple, Microsoft Azure, que es va anunciar l'octubre de 2008 (però no va començar a prestar serveis fins al 2010); IBM el 2011 (va llançar SmartCloud Framework per donar suport al seu projecte Smarter Planet), o Oracle el 2012 (Oracle Cloud).

Pel que fa a les plataformes més representatives (no les úniques) per desplegar *clouds*, el 2005 es comença a treballar en un projecte de recerca orientat a aquest àmbit basat en programari de codi obert, i el 2008 neix OpenNebula. Aquell any es forma el consorci Opennebula.org i és finançat per un projecte de la UE (7è. Programa Marc): va arribar a tenir, el 2010, 16.000 màquines virtuals configurades. Les dades actuals per a la versió actual d'OpenNebula

(V6.x) són 100.000 baixades en l'últim any, 2.500 *clouds* connectats al *marketplace* i 300.000 *cores* gestionats a la instal·lació més important amb aquesta plataforma.

El 2010 Rackspace i la NASA van llançar una iniciativa de *cloud-software* de codi obert i la van anomenar OpenStack, amb l'objectiu d'oferir a les organitzacions la possibilitat de muntar els seus serveis de *cloud* sobre maquinari estàndard. El 2012 es va crear la Fundació OpenStack per promoure aquest programari a la comunitat, amb 200 socis (una gran part de tots els actors importants del món TIC). Openstack s'ha popularitzat i avui es pot trobar a les distribucions més conegudes (Redhat, Ubuntu, SuSE...) o també en empreses que n'ofereixen les integracions, com ara Mirantis.

També el 2010, una empresa anomenada Cloud.com allibera el seu producte orientat a la creació de *clouds*, anomenat CloudStack (sobre el qual havia estat treballant en secret durant els anys anteriors), amb GPLv3. El 2011 Cloud.com és adquirida per Citrix Systems, i CloudStack passa a ser un projecte de la Fundació Apache i es distribueix mitjançant la llicència Apache Software License. CloudStack disposa d'un gran conjunt d'usuaris usuaris i és una plataforma sòlida i actualitzada (la versió 4.15 és de 2021).

Com a resum, es pot argumentar que avui dia és poc freqüent que una institució o empresa mitjana o gran no tingui externalitzats alguns serveis al *cloud* per als treballadors o usuaris finals, o no utilitzi el *cloud* per a alguns dels aspectes vinculats al negoci (correu, web, intranet, etc.), excepte aquelles en què l'activitat principal depengui de la privadesa de les dades o estiguin especialment protegides.

2. Característiques, avantatges i desavantatges del *cloud computing*

Tenint en compte les opinions dels experts (per exemple, Jamie Turner), a més del web 2.0, les grans revolucions que han facilitat l'evolució del *cloud computing* són l'avenç de les tecnologies de virtualització, la proliferació a costos acceptables de l'amplada de banda i els estàndards d'interoperabilitat de programari. A parer d'aquests experts, tots aquests aspectes han facilitat que avui dia qualsevol recurs o servei pugui tenir com a base el *cloud*.

No obstant això, aquestes opinions tan favorables envers el *cloud* s'han de considerar dins del marc adequat i no s'ha de pensar que el *cloud* ho pot contenir i proveir tot, ja que, com tot paradigma, té elements a favor (bàsicament, l'accés a recursos i serveis per part dels usuaris sense ser experts en la instal·lació o administració, la flexibilitat d'ús i adaptativa i els preus acceptables), però també punts febles que s'han de tenir presents, especialment el seu taló d'Aquil·les, que és la disponibilitat 24/7 de l'accés a la xarxa i l'amplada de banda: *sense xarxa o amb una xarxa deficient, el cloud no existeix*.

Són interessants l'article *15 Ways to Tell Its Not Cloud Computing* [Wtn08], que expressa clarament què no és *cloud computing*, i els tres criteris expressats per George Reese [Cca09] sobre quan un servei és *cloud*: accessibilitat per navegador web (no propietari) o una API de servei web, inici sense aportar capital i pagament únicament pel que s'usa quan s'usi.

De manera descriptiva, podem enunciar les següents **característiques** clau del *cloud computing* [Dcc11]:

1) Agilitat i autoservei: es tracta de la rapidesa i capacitat de proveir recursos (de manera gairebé immediata) sense grans intervencions ni accions de cap de les dues parts. Això s'aconsegueix tenint un flux de treball predefinit molt ben ajustat i automatitzant la provisió de recursos. És a dir, l'usuari pot, de manera no assistida, obtenir capacitats de còmput, emmagatzematge o xarxes, segons el que requereixi, automàticament sense necessitat d'interacció humana amb el proveïdor de serveis. Aquests recursos estaran disponibles, en un interval curt de temps (segons o com a màxim uns pocs minuts), per mitjà de la xarxa.

2) Cost: a causa de l'eficiència i l'automatització en la gestió i administració de recursos, els preus resultants per als proveïdors de *cloud* són reduïts i els poden traslladar a l'usuari final, que no ha de fer front a les despeses derivades de la implantació d'infraestructura i serveis ni a la inversió en màquines, programari i recursos humans, per la qual cosa comptabilitzant-ho tot el model surt favorable en relació amb el *cloud*. A més, com que el control i el monitoratge són automàtics, es poden aplicar mesures amb cert nivell d'abstracció apropiat per

al tipus de servei (per exemple, comptes / comptes actius, emmagatzematge, processament o amplada de banda) i és possible ajustar el cost a models, per exemple, de pagament per ús, pagament per peticions, etc., i això proporciona transparència, tant per al proveïdor com per al consumidor del servei utilitzat.

3) Escalabilitat i elasticitat: aquests conceptes es refereixen a l'aprovisionament de recursos en (gairebé) temps real i l'adaptabilitat a les necessitats de càrrega i ús. És a dir, si puc preveure la càrrega o la utilització, no és necessari aprovisionar tots els recursos des de l'inici com si es tractés d'una inversió local, sinó planificar-los per a quan la demanda ho requereixi, amb la reducció consegüent del cost.

4) Independència entre el dispositiu i la ubicació: el recurs s'independitza de l'accés i es facilita que aquest accés pugui ser des d'una xarxa local, corporativa o d'un altre tipus, i des de diferents dispositius (de diverses capacitats i tipologies).

5) Manteniment i llicències: el model d'actualitzacions i llicències se simplifica, ja que el programari estarà als servidors del *cloud* i no hi haurà res d'instal·lat al dispositiu de l'usuari final.

6) Rendiment i gestió dels recursos: es refereix al control exhaustiu i el monitoratge eficient dels serveis per aconseguir una alta disponibilitat i una utilització òptima dels recursos de manera automàtica. Aquestes característiques permeten reduir al màxim les ineficiències, aportant control i notificació immediata, així com transparència i seguiment tant al proveïdor com a l'usuari final. A més, com que els recursos es poden configurar per servir múltiples usuaris en un model de multitinença, els recursos físics i virtuals es poden assignar dinàmicament i reassignar-se d'acord amb la demanda dels consumidors. Això proporciona un sentit d'independència de la ubicació i el client no tindrà cap control o coneixement sobre la ubicació exacta dels recursos proporcionats (només en un nivell d'abstracció molt alt: per exemple, el país o el centre de dades).

7) Seguretat: es pot incrementar, atesa la particularitat de tenir les dades centralitzades. Al responsable de l'aplicació li correspondrà la seguretat d'aquesta, però el proveïdor s'haurà d'ocupar de la seguretat física. Amb això, la seguretat serà almenys igual que en els sistemes tradicionals, però es podrà millorar (i s'haurà d'estipular quin nivell es vol a l'acord de nivell de servei, o SLA), ja que al proveïdor li interessa que la seva infraestructura sigui segura per als clients, com a base de la qualitat del negoci, i hi podrà invertir globalment, mentre que aquesta inversió pot ser inassumible per a un client si l'ha d'emprendre sobre la infraestructura local.

8) Virtualització com a base per a l'aprovisionament: això permet compartir i optimitzar l'ús del maquinari, reduint els costos, l'energia consumida pel servidor i l'espai, la qual cosa facilita el desplegament ràpid de serveis i ga-

ranteix una alta disponibilitat (serveis en *stand by*) i mobilitat per a la càrrega (moviment de màquines virtuals a servidors més alliberats quan la càrrega augmenta).

Per convèncer els més reticents, es pot enumerar un conjunt d'**avantatges** que aporta el *cloud* com a paradigma. Són els següents:

9) Fàcil integració i acceptació: tenint en compte el seu desenvolupament i base en estàndards, es pot integrar amb molta més facilitat i rapidesa amb la resta de les aplicacions empresarials. L'usuari final queda totalment convençut quan se li permet accedir des de qualsevol dispositiu sense requeriments especials.

10) Servei global: proporciona ubiqüitat i accés als serveis des de qualsevol lloc, amb temps d'inactivitat reduït al mínim i amb una alta disponibilitat dels recursos.

11) Simplicitat: els rols i les responsabilitats estan molt ben definits. La separació de les activitats està ben estipulada (per exemple, proveïdor de continguts, proveïdor d'aplicació, proveïdor d'infraestructura i usuari final), la qual cosa es tradueix en una manera òptima de treballar i una inversió inferior per a totes les parts.

12) Reducció de riscos i rapidesa: no es necessiten ni una gran inversió ni l'adequació d'entorns. És possible accelerar al màxim la implantació de nous serveis en les diferents etapes de desenvolupament fins a la producció.

13) Reducció de l'ús d'energia (consum eficient): a causa de la tecnologia utilitzada i l'ús eficient dels recursos (afavorit per la virtualització), només es consumeix el que és necessari, a diferència dels centres de dades tradicionals, en què hi ha un consum fix independent de la càrrega o utilització.

Si bé els avantatges són evidents i molts actors d'aquesta tecnologia la basen sobretot en l'abaratiment dels costos de servei, es comencen a difondre opinions en contra que consideren que un *cloud públic* potser no és l'opció més adequada per a un determinat tipus de serveis o infraestructures. Entre els principals **desavantatges** que esgrimeixen alguns experts hi ha els següents:

1) Centralització: tant de les dades com de les aplicacions. Genera una dependència en relació amb el proveïdor, que si no disposa de la tecnologia adequada (monitoratge i detecció) i els recursos apropiats (alta disponibilitat), pot generar talls o inestabilitats en el servei. En aquests casos pren una rellevància especial l'SLA, que ha d'especificar les obligacions del proveïdor i les indemnitzacions a què hauria de fer front.

2) Confiabilitat: la «salut» tecnològica i financera del proveïdor serà un element clau en la continuïtat del seu negoci i del nostre, per la qual cosa les decisions que prengui el proveïdor afectaran directament el negoci del client, i si no són les adequades, impactaran directament en l'empresa. També l'empresa quedaria a la mercè d'un mercat molt dinàmic pel que fa a les fusions i monopolis (o pseudomonopolis), amb l'impacte que podria tenir en els costos dels serveis. Un problema important és que el proveïdor, per diversos motius (legals, financers, econòmics), tanqui la seva activitat de manera abrupta i es produeixi el *data lock-in* de l'empresa als servidors del proveïdor, sense possibilitat de poder recuperar les dades (són situacions que s'han esdevingut arran del bloqueig judicial d'un proveïdor per diverses raons, amb el bloqueig de les dades dels clients fins que es resolgui el conflicte judicial).

3) Escalabilitat: a mesura que el proveïdor disposi de més clients, hi haurà més usuaris sobre el maquinari, la sobrecàrrega augmentarà, i si el proveïdor no té un pla d'escalabilitat a mitjà i llarg termini per assegurar un creixement sostenible des del punt de vista de les necessitats dels clients, es pot arribar a la saturació dels serveis, amb la conseqüent degradació i pèrdua de prestacions.

4) Especialització o qualificació: la necessitat de serveis «especials» o qualificats podria tenir una prioritat molt baixa (i el retard conseqüent en el desplegament) per al proveïdor si no els demanen prou clients, la qual cosa pot afectar el negoci d'un en particular que sí que els necessiti.

5) Disponibilitat: el principal punt feble d'una infraestructura en *cloud* és l'accés a internet. Si no se'n disposa de manera fiable i amb una amplada de banda acceptable, el *cloud* deixa de tenir efectivitat.

6) Risc i privadesa: les dades «sensibles» del negoci no estan a les instal·lacions de les empreses i la seguretat no depèn dels seus recursos humans, sinó del proveïdor del servei. Si s'assumeix que aquestes dades són d'alt valor, en un context vulnerable, el risc pot ser molt alt per la possibilitat de robatori (còpia), accés a la informació (lectura) o destrucció (esborrament).

7) Seguretat: atès que la informació haurà de passar per diferents canals i serveis, pot resultar que cadascun d'ells sigui un focus d'inseguretat. Si bé això es pot resoldre mitjançant canals i serveis segurs, la possibilitat d'una fallada a la cadena de xifratge de la informació és real, amb els problemes conseqüents que representa. A més, en aquest cas, el propietari de la informació pot desconèixer totalment què ha passat i on ha estat la fallada.

8) *Vendor lock-in*: és un gran problema (avui dia s'ha demostrat que és un dels més freqüents) que origina que un client depengui d'un proveïdor de productes i serveis i sigui incapaç d'usar un altre proveïdor sense costos de canvi substancials, encara que el canvi signifiqui una reducció de costos o prestaci-

ons millors. Molts desenvolupadors o usuaris finals són reticents al *cloud* per aquest motiu, ja que significa un compromís adquirit que, si després es vol canviar, comportarà costos molt elevats.

A [Vcc10] s'amplien aquests «desavantatges» i es demostren amb xifres de casos d'ús en un proveïdor real algunes d'aquestes qüestions, per apuntar quins són els riscos que s'han de considerar abans de prendre les decisions o de signar un SLA.

Com es pot veure, entre característiques, avantatges i desavantatges hi pot haver elements que entren en contradicció o que des d'un altre punt de vista són oposats (per exemple, el tema de la seguretat), i l'equip de presa de decisions de l'empresa haurà d'analitzar acuradament les possibilitats i quins avantatges hi ha enfront dels riscos que assumirà. Des d'un punt de vista global, és una tecnologia que aporta beneficis i que, amb una planificació i presa de decisions adequades, valorant tots els factors que influeixen en el negoci i escapant-se de conceptes superficials (tots ho tenen, tots ho utilitzen, baix cost, expansió il·limitada, etc.), pot ser una elecció apropiada per als objectius de l'empresa, el negoci i la prestació de serveis TIC que necessita o que forma part de les seves finalitats empresarials.

La figura següent mostra una visió general dels **actors i elements** en joc dins del *cloud computing*. De manera abstracta, podem identificar:

- **Infraestructura**: recursos de maquinari i programari que gestiona el proveïdor i que seran objecte d'utilització per part de les empreses i clients d'aquest.

- **Catàleg de serveis**: elements oferts pel *cloud* i que seran seleccionables per categories o prestacions pels clients del proveïdor de serveis.

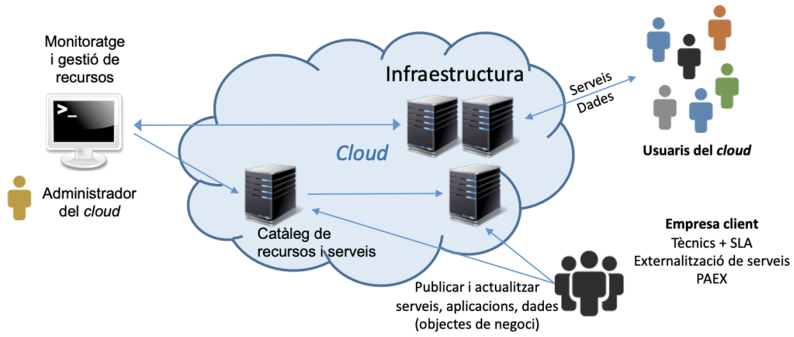
- **Administració del *cloud***: cos tècnic de professionals que garantirà les prestacions, seguretat, estabilitat, disponibilitat, etc., dels serveis comercialitzats en funció de l'SLA signat amb cada client.

- **Empresa client**: usuaris que externalitzen els serveis TIC i publiquen i actualitzen les seves aplicacions o dades al *cloud*. Es basen en una garantia del servei que ofereix el proveïdor, que forma part d'un SLA. Generalment, el cost es fonamentarà en polítiques semblants al pagament per ús.

- **Usuaris**: clients de l'empresa que ha posat els serveis al *cloud* i que poden saber o no que aquests es proveeixen des del *cloud*. Només accedeixen a un servei, aplicació o dades com si entressin als servidors de l'empresa que proveeix el servei.

És important fer notar que els usuaris (de vegades indicats com a usuaris finals) generalment no són clients del proveïdor de serveis del *cloud*, sinó de l'empresa que proveeix els serveis al *cloud*.

Per exemple, si considerem els principals clients d'Amazon AWS [Acs] (és a dir, les empreses que tenen l'etiqueta «milions de dòlars» associada a alguna característica [ingressos, actius totals, finançament, valoració o beneficis]), hi trobem Adobe Systems, Airbnb, Alcatel-Lucent, Aon, Autodesk, BMW, Bristol-Myers Canon, Capital One, Comcast i Docker, entre molts altres. Si prenem, per exemple, Adobe Systems, podrem veure el rol de cada actor: el **proveïdor del *cloud*** i la infraestructura és **AWS**, l'**empresa** que els contracta és **Adobe**, i aquesta comercialitza els productes per mitjà de **Creative Cloud**, que permet que **usuaris** (finals) accedeixin a programari de disseny gràfic, edició de vídeo, disseny web i serveis al *cloud* per una quota econòmica mensual.



3. Classificació

Hi ha diferents taxonomies per descriure els serveis i la forma de desplegament del *cloud*. Al document *The NIST Definition of Cloud Computing* del NIST [Dcc11], es defineixen quatre models de desplegament i tres models de serveis. Com a models de desplegament, es poden enumerar:

1) *Cloud* pública: en aquest cas, el sistema està obert per a l'ús general mitjançant diferents models de negoci (pagament per recursos, per ús, quota o lliure). Aquest tipus de recurs és mantingut i gestionat per un tercer i les dades i aplicacions dels diferents clients comparteixen els servidors, sistemes d'emmagatzematge, xarxes i altres infraestructures comunes; normalment, els recursos s'utilitzen i gestionen a través d'internet. Generalment, un client no sap amb quin altres usuaris comparteix la infraestructura, la utilització es contracta per mitjà d'un catàleg de recursos disponibles i l'aprovisionament és automàtic (o semiautomàtic) després d'haver complert amb una sèrie de tràmits respecte al model de pagament i l'SLA.

El propietari del *cloud* (proveïdor de recursos i serveis) pot ser una empresa privada, una institució acadèmica, una organització governamental o una combinació d'aquestes, generalment, en funció del tipus de clients i dels recursos que s'hauran de proveir. Tècnicament, hi pot haver poques diferències (o cap) amb algun dels altres models (especialment amb el privat); no obstant això, hi pot haver diferències substancials en relació amb la seguretat: el *cloud* públic pot estar disponible en una xarxa oberta com internet i sense mecanismes de xifratge.

> Com a exemples d'aquests serveis, podem esmentar AWS, Microsoft Azure, Google Compute, Rackspace i IBM SoftLayer (fins al 2014, IBM SmartCloud), que operen en la seva pròpia infraestructura i els recursos dels quals són accessibles a internet, i el Consorci de Serveis Universitaris de Catalunya (CSUC), que ofereix, amb diferents models d'explotació, recursos de *cloud* per a institucions acadèmiques i de recerca de Catalunya (https://www.csuc.cat/ca/serveis/infraestructura-en-nuvol).

És interessant esmentar que alguns proveïdors intenten reduir la probable inseguretat mitjançant serveis de connexió directa (per exemple, AWS Direct Connect i Azure ExpressRoute), que permeten que els clients puguin comprar o llogar una connexió privada per connectar-la a un punt privat del proveïdor, amb l'increment consegüent de la privadesa que significa un recurs d'aquesta mena. Per als clients, tots els seus costos són operatius (OPEX). Una visió de conjunt de proveïdors i serveis es pot analitzar a [Cis16] (on examinen els punts forts i febles de cada operador; es va elaborar l'agost de 2016) i a [Cmh15] (aquest informe està orientat a *cloud-enabled managed hosting*, que indica la qualitat de les empreses que ofereixen serveis de migració i adaptació d'altres proveïdors).

2) *Cloud* privat: en aquest supòsit, la infraestructura funciona exclusivament per a una organització o empresa i és utilitzada per les seves unitats de negoci o departaments. L'organització o empresa pot ser propietària, administradora i operadora, o pot subcontractar a un tercer alguna d'aquestes funcions. És l'opció més favorable per a les companyies que necessiten una alta protecció de dades i un alt nivell de servei, perquè els recursos i la gestió estan sota el control i la cura de la mateixa empresa o organització.

Si bé resol els problemes d'algun tipus d'empresa (per exemple, aquelles amb legislació especial, com ara les empreses públiques), aquesta ha d'assumir el cost de la inversió (CAPEX), però com a contrapartida disposa d'una infraestructura a demanda; gestionada per personal propi (o extern, però seguint els seus criteris) i que controla quines aplicacions s'han d'executar i on ho han de fer, i que manté la privadesa de la seva informació, la qual cosa permet definir les polítiques d'accés i evita el *data lock-in* i el *vendor lock-in* esmentats anteriorment.

3) *Cloud* híbrid: en aquesta organització del cloud, es combinen models públics i privats. El propietari disposa d'una part privada (amb accés restringit i sota el seu control) i en comparteix unes altres, encara que d'una manera controlada. Els *clouds* híbrids ofereixen la possibilitat d'escalar molt ràpidament amb aprovisionament extern a demanda, però impliquen una gran complexitat a l'hora de decidir com es distribueixen les dades i les aplicacions en una banda o una altra. Si bé és una proposta atractiva per a les empreses, els casos habituals d'ús no passen d'aplicacions simples sense restriccions de sincronització amb bases de dades sofisticades.

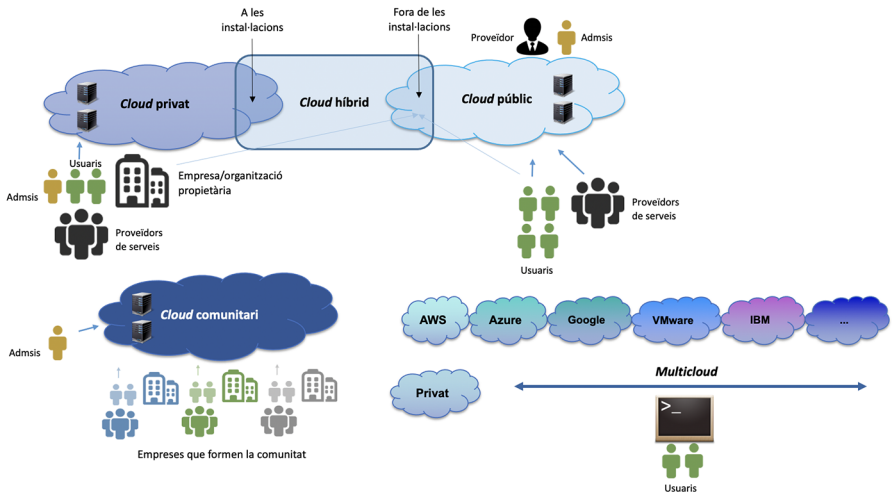
Un exemple en són algunes implementacions de correu empresarial o aplicacions d'ofimàtica.

4) *Cloud* comunitari: els serveis estan dissenyats perquè es puguin utilitzar per part d'una comunitat, organitzacions o empreses amb objectius o negocis alineats (per exemple, bancs, distribuïdors, arquitectes...) o que requereixin unes característiques específiques (per exemple, seguretat i privadesa). Les empreses que formen la comunitat poden ser els propietaris de la infraestructura, així com els gestors o operadors, o alguns d'aquests rols es poden subcontractar a tercers, però seguint les indicacions i regles que s'apliquen a la comunitat.

5) *Multicloud*: si bé aquest model de desplegament no està considerat en la taxonomia del NIST, es tracta d'una variació que té molta acceptació actualment. El *multicloud* representa la implementació del *cloud* per diferents proveïdors o de diverses solucions dins del *cloud* privat, integrats i administrats per un sol panell de control. Els avantatges comentats per als diferents *clouds* s'amplien: augmenten la flexibilitat, l'escalabilitat, la seguretat i l'agilitat per fer front a les diferents necessitats de l'organització.

A més, aquest tipus d'organització ofereix avantatges en el cost dels serveis: com que cada proveïdor aportarà diferents preus sobre l'ús del còmput o emmagatzematge, això permetrà a l'usuari utilitzar els recursos més adequats i amb un cost inferior en el desplegament de solucions, per fer un ús més eficient de la inversió. En relació amb la seguretat, s'incrementa per la distribució de les dades, la qual cosa mitiga l'efecte d'atacs de denegació de servei o uns altres que només afectarien un únic proveïdor (és molt poc probable que tots els proveïdors de *cloud* es vegin afectats per un mateix incident de seguretat). Funcionalment, no s'observaran gaires diferències respecte a un *cloud* híbrid, més enllà de la utilització múltiple tant de proveïdors públics com de privats.

La figura següent mostra un esquema de les cinc formes d'implantació i desenvolupament dels *clouds*.



Una altra de les classificacions habituals és pel nivell de servei que es presta. Tradicionalment, es distingien tres categories: *infraestructura com a servei* (**IaaS**), *programari com a servei* (**SaaS**) i *plataforma com a servei* (**PaaS**). Avui es poden trobar altres extensions o derivades de les habituals: *business as a service* (**BaaS**), *storage as a service* (**StaaS**), *desktop as a service* (**DaaS**), *disaster recovery as a service* (**DRaaS**), *marketing as a service* (**MaaS**), o alguns autors li donen un nom global amb *everything as a service* (**XaaS**), per referir-se a la creixent diversitat de serveis disponibles al *cloud* a través d'internet (es pot consultar [Idd09]). Un cas que afirma aquestes consideracions és el d'*application platform as a service* (**aPaaS**), que permet un ràpid desenvolupament i aprovisionament d'aplicacions, com a plataforma específica per codificar, aprovisionar i desplegar *apps*, i que és compatible amb el cicle de vida complet, proporcionant una forma més ràpida de crear aplicacions.

Exemples d'aPaaS

La seva acceptació es comprova per l'àmplia proposta del mercat: AWS amb Elastic Beanstalk, CenturyLink amb AppFog, Google amb App Engine i IBM amb BlueMix, entre altres.

A la **infraestructura com a servei** (IaaS) hi ha la capa de recursos bàsica (generalment, màquines virtuals, xarxes i emmagatzematge) on el client podrà col·locar els seus SO i les seves aplicacions i implementar un servei, o utilitzar-la per executar les seves aplicacions. El client no podrà manejar o controlar el maquinari subjacent, però, a partir de l'SO, sí que podrà manejar

l'emmagatzematge i les aplicacions o serveis implementats sobre aquestes màquines, així com alguns aspectes de la connectivitat (subxarxes, tallafocs, dominis...).

Exemples de grans proveïdors d'IaaS (desenes de milers de màquines virtuals) són Amazon EC2, Google Compute Engine i Digital Ocean, com a servei representatiu per a pimes (en unitats de màquines virtuals).

La reflexió de l'usuari en aquesta modalitat seria: «Per què he de comprar, instal·lar i provar infraestructura cada X anys [obsolescència], i no la puc llogar i pagar per ús?» Els recursos necessaris s'obtenen sense emprendre obres (centre de dades), amb els mínims recursos humans (no especialistes en TIC), sense una gran inversió inicial, de manera escalable i eficient i a costos acceptables.

La **plataforma com a servei** (PaaS) és l'encapsulació d'un entorn de desenvolupament i la provisió d'un seguit de mòduls que proporcionen una funcionalitat horitzontal (persistència de dades, autenticació, missatgeria, etc.). D'aquesta manera, una estructura bàsica d'aquest tipus podria consistir en un entorn que contingui serveis o aplicacions, biblioteques i API per a una finalitat específica.

Per exemple, per a una tecnologia de desenvolupament en particular, sobre Linux, un servidor web, una base de dades i un entorn de programació com ara Perl i Ruby.

És a dir, el client no gestiona ni controla la infraestructura (ni servidors, ni sistemes operatius, ni emmagatzematge, ni cap tipus d'elements de xarxa o seguretat, etc.), però té el control de les aplicacions desplegades i de la configuració del seu entorn d'execució (bases de dades i *middlewares*). És habitual que una PaaS pugui prestar serveis en tots els aspectes del cicle de desenvolupament i les proves de programari o per desenvolupar, gestionar i publicar continguts.

Exemples d'això serien Cloud9, Google App Engine, Heroku i OpenShift.

Com a resum, en aquesta modalitat, el desig de l'usuari seria: «Necessito aquest programari instal·lat sobre aquest sistema operatiu, aquesta base de dades i aquesta API», i rebria tot el conjunt (HW, SO, base de dades, biblioteques, API, seguretat, GUI i altres eines) preparat per usar-lo al cap de pocs segons, amb la finalitat de desenvolupar aplicacions empresarials o mòbils, pàgines web, continguts, etc.

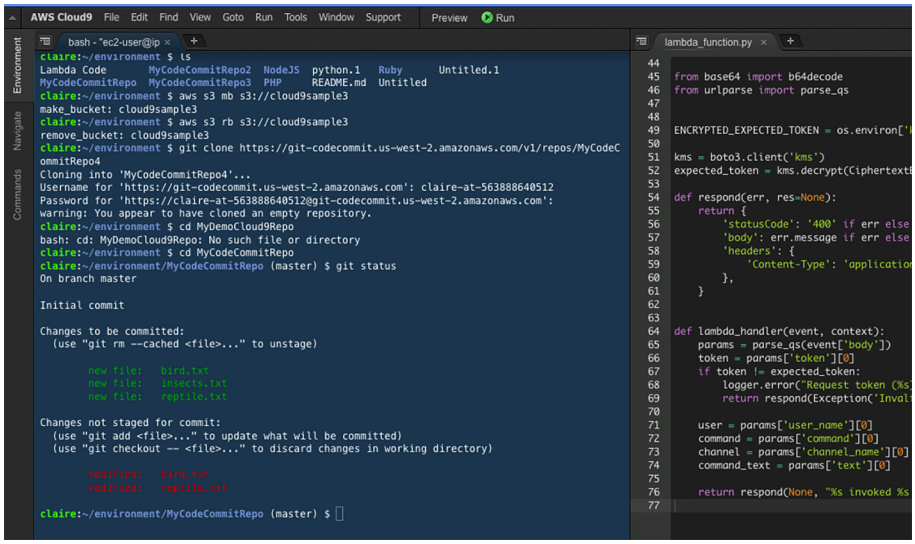
Finalment, el **programari com a servei** (SaaS) es troba a la capa abstracta (si bé hi ha una tendència bastant estesa a considerar que el SaaS és la capa més simple del PaaS, o la més baixa) i caracteritza una aplicació completa que s'ofereix com un servei, generalment a demanda amb multitinença (arquitectura de programari en què una sola instància de l'aplicació s'executa al servidor, però servint múltiples clients o organitzacions). En general, les aplicacions amb aquest model de servei són accessibles a través d'un navegador web i l'usuari no hi té cap control, encara que en alguns casos se li permet aplicar-hi algunes

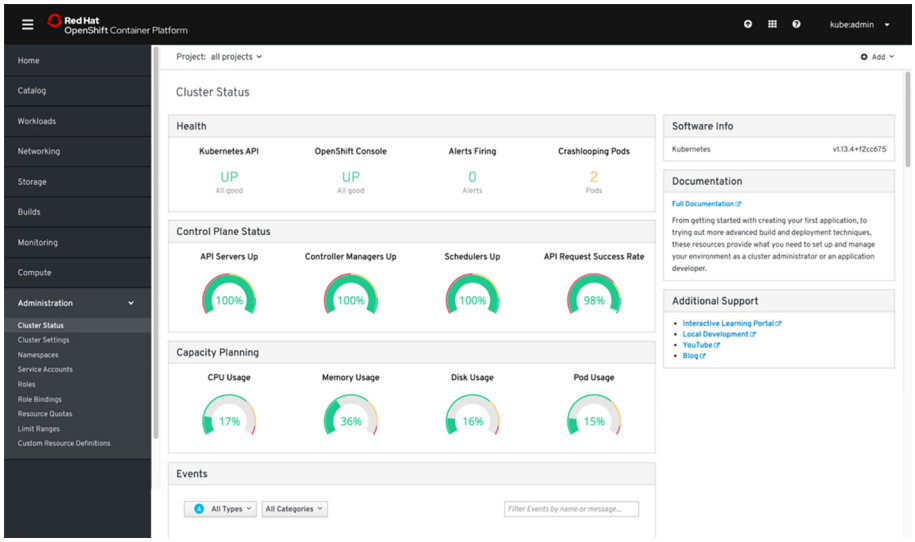
configuracions. Això elimina per al client la necessitat d'instal·lar l'aplicació als seus ordinadors, suprimeix el suport i el manteniment del maquinari i programari i millora el control i la gestió de les llicències, si són necessàries.

> Com a exemples representatius de SaaS, es poden esmentar els correus web (Gmail, Outlook, Yahoo!...), SalesForce, Google Docs, Office 365 i SiteBuilder.

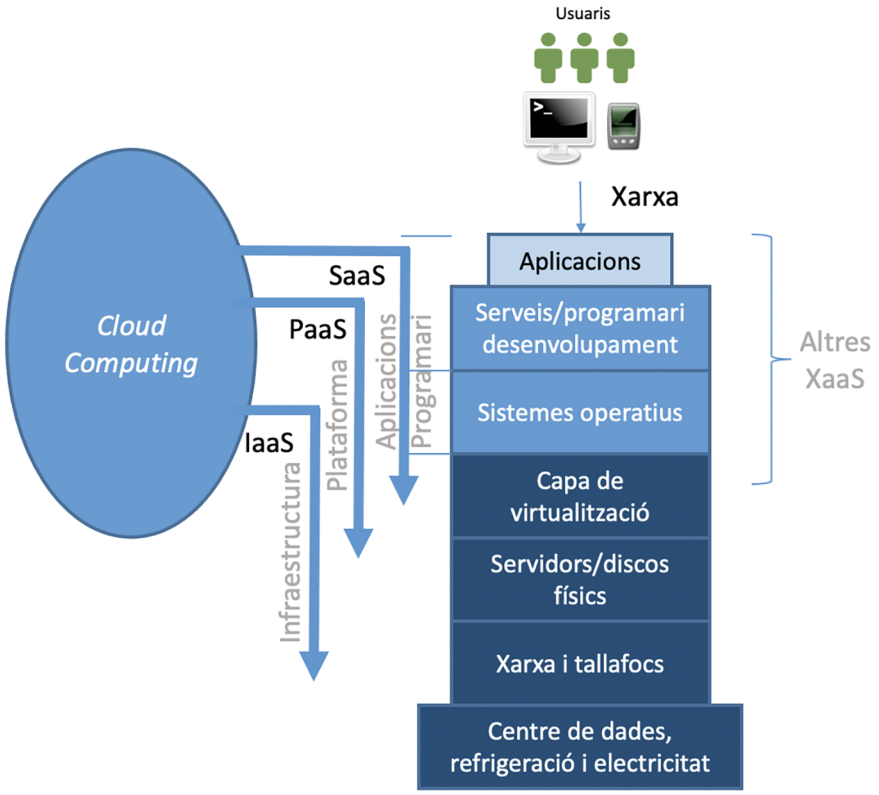
En aquesta modalitat, el pensament de l'usuari es podria resumir així: «Executi això per mi», sense responsabilitats en la gestió de maquinari o programari, utilitzant un navegador web o evitant la instal·lació de programari client, basat en un servei a demanda, amb escalabilitat gairebé immediata, accés des de qualsevol lloc i amb qualsevol dispositiu, amb un model de suport de 24/7 i un model de pagament com *pay-as-they-go* i *pay-as-they-grow*.

Les dues figures següents mostren les pantalles de Cloud9 (propietat d'AWS des de 2016) i d'Openshift (on es poden obtenir comptes gratuïts per a proves de concepte amb diferents modalitats de funcionament i desenvolupament), com a exemples de la facilitat i simplicitat d'engegar entorns complexos tan sols en uns segons, preparats per desenvolupar-hi aplicacions o serveis.





La figura següent resumeix en un diagrama les diferents modalitats de servei del *cloud* i què implica cadascuna d'elles.



Si es parteix de la base que una infraestructura de *cloud computing* és un sistema distribuït en el sentit general del terme, entenent-lo com una col·lecció de recursos connectats entre si per una xarxa de comunicacions en què cada màquina té els seus components de maquinari i programari i que l'usuari percep com un únic sistema, és necessari comparar-lo amb altres tecnologies dins d'aquest apartat. Entre elles es poden esmentar:

1) Clúster: s'aplica a agrupacions d'ordinadors units entre si (generalment) per una xarxa d'alta velocitat i que es comporten com si fossin un únic ordinador. S'utilitzen com a entorns de processament d'altes prestacions (*high performance computing*) i serveis per a aplicacions crítiques o d'alt rendiment, entre altres usos. El seu ús s'ha estès per a aplicacions que necessiten un temps de còmput elevat (per exemple, científiques), i estan basats en infraestructura comercial (*blades* o *slices*) juntament amb una xarxa d'altes prestacions (p. ex., InfiniBand), que, utilitzant majoritàriament programari de codi obert (Linux; NFS —encara que cada vegada més s'està reemplaçant per altres sistemes d'arxius distribuïts com Ceph, Lustre i GlusterFS—; sistemes de gestió de cues com SGE i Slurm, i biblioteques com per exemple OpenMPI, per executar tasques distribuïdes sobre l'arquitectura, i OpenMP, per aprofitar la potencialitat dels sistemes *multicore*), permet desenvolupar i executar aplicacions concurrents o distribuïdes d'altes prestacions. Això comporta disposar d'una infraestructura que proveeix d'alt rendiment, alta disponibilitat, amb equilibri de càrrega, eficient, escalable i a costos raonables. Existeix una classificació dels clústers en funció de quina característica hi predomina: alt rendiment (*high performance computing*, o HPC), alta disponibilitat (*high availability computing*, o HA/HAC) i alta eficiència (*high throughput computing*, o HT/HTC).

2) Superordinador: és una evolució funcional a gran escala d'un clúster (si bé pot tenir diferents arquitectures) que té una capacitat de còmput molt més elevada que aquest (i, per tant, que un ordinador de propòsit general). El seu rendiment es mesura en teraflops (10^{12} operacions de coma flotant per segon) i el més potent el juny de 2021, publicat a la llista del *top* 500 (els 500 superordinadors més potents del món), és el superordinador Fugaku, al Centre de Ciències de la Computació RIKEN (Japó), amb 7.630.848 *cores*, 442,010 teraflops i un consum de 29,9 megawatts.

La seva història s'inicia a la fi dels cinquanta. IBM, Sperry Rand i Cray van ser els grans actors d'aquella època amb arquitectures específiques (inicialment amb Univac, com el primer ordinador comercial, i posteriorment IBM7030 o Cray-1). Els van seguir les arquitectures vectorials (dècada dels setanta), amb un mercat dominat per Control Data Corporation (CDC) i Cray Research. Amb l'auge dels microprocessadors (Intel), a la dècada dels vuitanta comencen a emergir multiprocessadors escalars amb compartició de memòria (*symmetric multiprocessor*, o SMP) i després sense compartició de memòria (*massively parallel processor*, o MPP). Després, als noranta, deriven en clústers de milers de processadors, i a la fi de la dècada i massivament, en els superordinadors amb milions de *cores* dels sistemes actuals.

És important esmentar que un superordinador pot tenir dos enfocaments diferents:

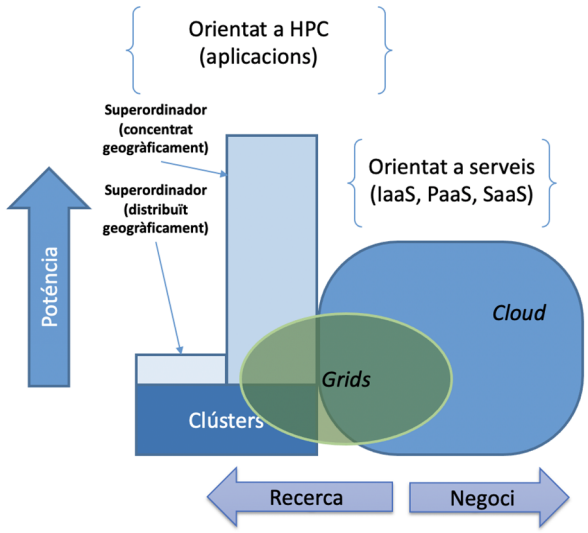
a) L'evolució del clúster (com per exemple el MareNostrum, al BSC [Catalunya]; el Finisterrae, al CESGA [Galícia], o el Juwels, al Jülich Supercomputing Centre [Alemanya]). Es dedica al còmput d'altes prestacions, comparteix un espai físic i disposa d'un seguit de recursos comuns com ara xarxes de comunicació d'alta velocitat i espai d'emmagatzematge compartit i distribuït.

b) El superordinador distribuït a través d'una xarxa, format per centenars o milers d'ordinadors discrets distribuïts mitjançant una xarxa (internet), que es dediquen de manera no exclusiva a solucionar un problema comú (generalment cada equip rep i processa un conjunt de tasques petites i trasllada els resultats a un servidor central que integra els resultats en la solució global).

Exemples representatius d'aquest últim són projectes com ara Seti@home, Folding@home, World Community Grid i Climateprediction.net. Molts d'ells estan basats en Boinc (una arquitectura de programari de codi obert que permet que voluntaris aportin els seus recursos a un problema comú, o paradigma conegut com a *volunteer computing*).

3) ***Grid***: es refereix a una infraestructura que possibilita la integració i l'ús col·lectiu d'ordinadors d'alt rendiment, xarxes i bases de dades propietat de diferents institucions, però units per una capa de programari (*middleware*) comú que els permet utilitzar com si fossin un únic superordinador. Per a això, i amb la finalitat de col·laborar aportant els recursos de còmput gestionats per cada institució, les universitats, els laboratoris de recerca o les empreses s'associen per formar *grids* i així cedir els seus recursos temporalment, però també per disposar del còmput equivalent a la unió de totes aquestes infraestructures. Les idees dels *grids* van ser establertes per Ian Foster i Carl Kesselman, que en van ser els precursors amb la creació de Globus Toolkit (avui dia obsolet), considerat com la primera eina per construir *grids*. Entre els grans projectes de *grid*, es poden trobar CrossGrid, EU-DataGrid o el recent EGI-InSPIRE (o simplement EGI, avui dia transformat en un servei als investigadors com a EOSC-Hub), tots ells finançats per la UE. És fonamental destacar que el *grid* és un tipus d'infraestructura de còmput que té com a àmbit d'utilització i propòsit natural el científic (*e-science*), a les universitats i els laboratoris de recerca, mentre que el *cloud* neix de la prestació de serveis i negocis a les empreses i usuaris (*e-business*) des d'altres empreses, però tots dos s'assemblen en paradigmes i estructures per manejar grans quantitats de recursos distribuïts. Són considerats per alguns experts com les dues vies de l'ús massiu de recursos (amb diferents models): una exclusivament per a la ciència (*grid*) i l'altra per a les empreses (*cloud*). És important destacar que es comencen a oferir serveis creuats, és a dir, altes prestacions (HPC) al *cloud* (per exemple, instàncies amb base en maquinari de GPU).

Amb aquesta caracterització dels recursos i la seva utilització, es pot situar gràficament cadascuna d'aquestes arquitectures en funció de la potència de còmput implicada i el seu tipus de dedicació (es pot veure a la figura següent).



Des del punt de vista dels paradigmes precursors que són essencials per al *cloud computing*, podem enumerar els següents:

1) Client-servidor: el model client-servidor es refereix de manera general als serveis implementats sobre un sistema de còmput distribuït (en els orígens, implementats per Sockets i RPC). Aquest tipus de paradigma s'utilitzarà en totes les aplicacions basades en serveis, així com en una gran part del codi que s'executi dins del *cloud* com a aplicacions d'alt rendiment.

2) SOA i WS: no són conceptes nous (es van definir a la dècada dels noranta). L'arquitectura orientada a serveis (SOA) es defineix com una arquitectura per dissenyar i desenvolupar sistemes distribuïts i que s'utilitza per al descobriment dinàmic i l'ús de serveis en una xarxa. Els serveis web (WS) són serveis prestats a través d'internet usant tecnologies com ara XML, Web Services Description Language (WSDL), Simple Object Access Protocol (SOAP) i Universal Description, Discovery, and Integration (UDDI). Els serveis de *cloud computing* es presten generalment per mitjà d'aquestes tecnologies, les quals també es poden utilitzar per a l'organització i la gestió interna.

3) Web 2.0: indubtablement, és la tecnologia que facilita la compartició d'informació, la interoperabilitat, el disseny centrat en l'usuari i la col·laboració al World Wide Web. Evidentment, aquesta tecnologia aporta molt al *cloud computing*, perquè la interacció i la prestació de serveis a l'usuari i els clients (i entre aplicacions) seran a través de les possibilitats que ofereix mitjançant els estils (CSS), llenguatges (HTML5, XML, XHTML, JavaScript, RoR...), aplicacions RIA (aplicacions d'internet enriquides, AJAX), agregació (RSS, ATOM), JavaScript Client Communication, interoperabilitat (transferència d'estat de representació, o REST), API (WebGL, DOM...), formats de dades (JSON, XML) o *mashups* (aplicacions web híbrides). Per a alguns investigadors i experts, ja s'ha arribat al web 3.0, però encara hi ha moltes opinions sobre què és i què no és 3.0.

Dins d'aquesta comparació, es pot argumentar que la tecnologia principal que sosté el *cloud computing* és la **virtualització**. La capa de virtualització (hipervisor; per exemple, Xen, VirtualBox, OracleVM, KVM, VMware ESX/ESXi i Hyper-V, entre altres) separa un dispositiu físic en un o més dispositius «virtuals» (anomenats *màquines virtuals*): cadascun d'ells pot ser assignat, utilitzat i gestionat pel client de manera molt simple, com si es tractés de màquines físiques, però amb els avantatges consegüents (aïllament, fàcil manteniment, engegada, etc). Avui dia tots els sistemes utilitzen tècniques de virtualització assistides per maquinari (extensions del processador VT-x o AMD-V), de tal manera que és possible tenir màquines virtualitzades molt eficients, disponibles i configurades (mitjançant tècniques de clonació o *pool of VM in stand-by*) per assignar-se a un usuari a demanda i en pocs segons.

Un pas addicional que ha permès un increment notable de l'eficiència i la disponibilitat i una revolució en els entorns de desenvolupament ha estat la **virtualització a escala de sistema operatiu**, per crear un sistema escalable de múltiples entorns independents amb un mínim impacte en la utilització dels recursos. La virtualització de l'SO permet que el nucli d'un SO (*kernel*) pugui acollir múltiples instàncies d'usuari aïllades: cadascuna d'elles actua com un contenidor amb les seves pròpies biblioteques i serveis, però utilitzant l'SO subjacent. Els noms donats a aquestes instàncies són *contenidors*, *virtualization engines* (VE) o *gàbies* (per exemple, FreeBSD Jails i Chroot Jail). Als ulls d'un usuari d'aquest contenidor, és similar a un servidor real i tindrà tots els elements necessaris, com si es tractés del servidor real/virtualitzat, però amb només una petita despesa de memòria, CPU i disc. Sobre els sistemes Linux, són una evolució del mecanisme de Chroot més un sistema de control o limitació de la utilització de recursos (d'un contenidor respecte a un altre). Un dels desavantatges dels contenidors és que, en compartir l'SO, no es poden tenir contenidors amb sistemes operatius que no comparteixin el mateix nucli (per exemple, si el *host* és Linux, no es disposarà d'un contenidor amb Windows, cosa que sí que seria possible si fos una màquina virtual). Cal tenir en compte que Windows inclou Windows Subsystem for Linux, que es pot utilitzar per executar Linux sobre Windows o disposar entorns de màquines virtuals i contenidors Linux sobre el sistema operatiu Windows 10. Entre el programari de virtualització de l'SO de codi obert podem trobar Docker, LXC/LXD, OpenVZ i FreeBSD Jails, i en programari propietari, Virtuozzo (basat en OpenVZ).

El *cloud computing* també comparteix característiques amb altres models o paradigmes de còmput distribuït (alguns de futur), com els següents:

4) ***Dew computing***: és un nou paradigma de còmput distribuït que considera que els equips locals (ordinadors de sobretaula, portàtils i dispositius mòbils) concedeixen un entorn propici per a microserveis independents dels serveis *cloud*, i que aquests hi poden col·laborar. És a dir, aquest paradigma planteja la distribució de les càrregues de treball entre servidors del *cloud* i els dispositius discrets o locals per utilitzar plenament el potencial de tots dos. Alguns autors parlen més de *mobile cloud computing* [Mcc13]: s'integren i distribueixen els

serveis i l'emmagatzematge de les dades entre els dispositius mòbils i els proveïdors de serveis *cloud* (algunes experiències acosten més aquest paradigma al *volunteer computing*).

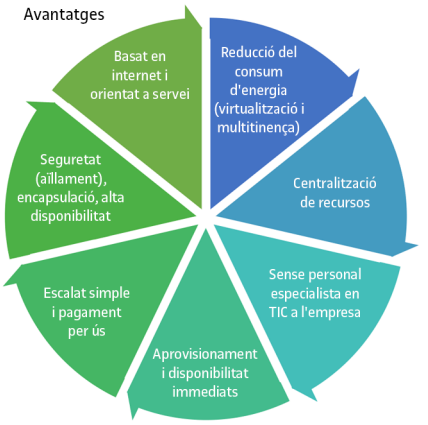
5) ***Edge computing***: és un paradigma de computació distribuïda que proporciona serveis de dades, computació, emmagatzematge i aplicacions més a prop de dispositius client o als dispositius pròxims a l'usuari. És a dir, processa i emmagatzema les dades als dispositius en un extrem de la xarxa (per exemple, dispositius mòbils) en lloc d'enviar les dades a un lloc del *cloud* per a això. Es considera una forma de computació distribuïda de proximitat, en què cadascun dels dispositius connectats a la xarxa pot processar les dades i transmetre'n només un conjunt que sigui d'interès, o fer-ho únicament en situacions excepcionals (alarmes o notificacions), tenint una capacitat autònoma i sense dependències del servidor al *cloud*. És un model que, amb el creixement que s'espera de la IdC (internet de les coses), permetrà que els sistemes *cloud* i les xarxes no es col·lapsin a causa de l'increment exponencial de dades per transmetre, emmagatzemar i processar. Alguns autors el diferencien del ***fog computing***, en què s'utilitzen agrupacions o multituds d'usuaris finals (o dispositius pròxims a l'usuari) per emmagatzemar dades (en lloc de fer-ho al *cloud*), comunicar (en lloc de fer-ho a través d'internet), controlar, configurar, mesurar i gestionar (en lloc que controlin principalment els *gateways* de xarxa, com passa a la xarxa LTE, per exemple). Aquest paradigma ha generat un moviment important de les grans empreses de tecnologia (OpenFog Consortium).

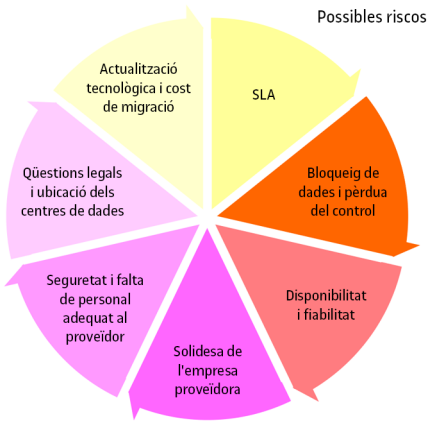
6) ***Peer-to-peer computing***: aquest paradigma planteja l'ús d'una arquitectura distribuïda sense la necessitat d'una coordinació central per dur a terme el còmput i l'emmagatzematge de dades. Els participants són els proveïdors i consumidors de recursos (en contrast amb el model tradicional de client-servidor) i tots s'ajuden per processar, emmagatzemar i comunicar les dades de manera igualitària. Els *peers* posen una part dels seus recursos (potència de processament, emmagatzematge o amplada de banda de xarxa) directament a la disposició d'uns altres *peers* de la xarxa, sense la necessitat d'una coordinació central per part dels servidors. Aquest tipus de paradigma ha tingut molta acceptació en serveis de gestió de continguts (BitTorrent, Spotify…), compartició d'arxius (Gnutella, eDonkey), diners digitals (Bitcoin, Peercoin) i anonimització (I2P), entre altres.

7) ***Volunteer computing***: com s'ha esmentat anteriorment en el *grid*, aquest tipus de processament distribuït es basa en el fet que els usuaris aporten els seus recursos (bàsicament, processament i emmagatzematge) per a un projecte determinat de què formen part. El primer registre que es té d'aquest tipus de còmput distribuït és de 1996, amb Great Internet Mersenne Prime Search (cerca de nombres primers que compleixin $2^n - 1$; per exemple, 3, 7, 31…), seguit el 1997 per distributed.net, i a continuació molts altres, entre ells Bayanihan, els desenvolupadors del qual van proposar el nom de *volunteer computing*. Avui dia, molts d'aquests projectes estan desplegats a partir de l'arquitectura Boinc esmentada anteriorment (desenvolupada per la Universitat de Califòr-

nia a Berkeley), com per exemple Seti@home, i també n'hi ha que han desplegat la seva pròpia arquitectura de programari, com Folding@home (desenvolupat inicialment a la Universitat de Stanford).

A les figures següents es mostren, de manera resumida, tant les característiques a favor del *cloud computing* com les que no són tan favorables, que poden representar un risc i que s'hauran d'analitzar amb compte.





4. Plataformes més representatives

Més enllà de l'extens catàleg a les diferents plataformes i opcions propietàries, hi ha un gran conjunt de plataformes basades en el programari lliure que permeten desenvolupar i engegar infraestructura de *cloud computing* sobre maquinari comercial, però sense grans requeriments. Aquests aspectes s'han de valorar molt bé, ja que són opcions totalment vàlides per a sistemes en producció i algunes permeten vincular-se als operadors *cloud* públics per estendre la potencialitat dels privats. Algunes d'aquestes plataformes disposen d'un gran nombre d'instal·lacions operatives; depenent de la grandària de l'empresa, les necessitats i el control de la informació que es vulgui tenir, són totalment viables, i moltes d'elles són utilitzades fins i tot pels operadors propietaris. Moltes de les opcions que es mostraran a continuació són productes amb una gran comunitat de desenvolupadors i usuaris i algunes s'han transformat en empreses (o empreses subsidiàries), o han nascut com a projectes de codi obert a les empreses, que treballen amb una versió *community* i una altra versió empresarial amb més suport o adaptacions a les necessitats específiques del client, o amb SLA o QoS diferenciats, però utilitzant el mateix programari que la versió de codi obert. Generalment, s'observa a les versions de codi obert un factor d'innovació i creixement continu que presenta noves adaptacions, serveis i millores en un cicle d'evolució més ràpid i constant que el dels seus corresponents propietaris, amb la producció de nous estàndards i noves API públiques, la qual cosa afavoreix la generació de coneixement i possibilitats adaptades a totes les necessitats [Bcs14].

No obstant això, aquesta proliferació de plataformes i serveis pot desconcertar, originar certa confusió i complicar la tria, però s'ha de prendre des d'un punt de vista pragmàtic per analitzar quin d'ells s'acosta més a les necessitats i s'adequa a la infraestructura de maquinari de què es disposa. És important fer proves d'implantació i desenvolupar sistemes de preproducció per veure com es comporta la plataforma escollida i si és adequada a les necessitats reals de l'organització.

El mercat, d'altra banda, ha evolucionat en tots els aspectes i avui és possible trobar una gran quantitat de proveïdors (alguns d'ells ja s'han esmentat), com es pot observar a les recomanacions i la llista *50 Best Cloud Computing Companies to Work for in 2021 Based on Glassdoor*, de Forbes, o la *The 100 Coolest Cloud Computing Companies of 2020*, de CRN.

4.1. Hipervisors

L'hipervisor (monitor de màquina virtual, *virtual machine monitor* o VMM) és la primera capa que s'ha de col·locar o bé sobre el sistema operatiu *host*, o integrada en aquest i funcionant com un conjunt indivisible. Aquesta capa, que en el sentit més ampli pot ser de programari, microprogramari o maquinari, és la responsable de crear i executar màquines virtuals (anomenades *guest*) sobre l'ordinador base (anomenat *host*). L'hipervisor permet (i gestiona) que les màquines virtuals es puguin executar simultàniament, cadascuna amb el seu sistema operatiu, i compartir els recursos de maquinari de base presentant a cadascun d'ells una plataforma virtual.

Les operacions en cada màquina virtual s'executen com si estiguessin sobre un maquinari determinat en un entorn tancat i només amb alguns recursos compartits (els que habiliti l'hipervisor) amb el sistema operatiu *host* (per exemple, directoris compartits entre el disc de l'SO *host* i l'SO *guest*).

És important destacar que els SO *guest* poden ser de diferents tipus (Linux, Unix, Windows, etc), en contraposició amb la virtualització del nivell del sistema operatiu, en què totes les instàncies (contenidors) han de compartir un únic nucli (*kernel*), encara que els SO *guest* poden diferir en l'espai d'usuari, com les diferents distribucions de Linux, però amb el mateix *kernel* (tot i que aquest concepte s'ha transformat en el cas de Windows 10 des de la integració de WSL al *kernel* del sistema operatiu).

Més enllà de l'evolució i història dels hipervisors des de la dècada dels seixanta, el punt que marca una diferència és quan els dos fabricants de processadors d'arquitectura x86/x86-64 introdueixen extensions de maquinari per donar suport a la virtualització (virtualització assistida per maquinari), fet que en millora les prestacions i permet una evolució notable dels hipervisors sobre processadors Intel, amb les extensions Intel VT-x (Vanderpool), i AMD, amb AMD-V (Pacifica) (cal destacar que, segons les dades del primer trimestre de 2021, provinents de MindFactory a Europa, AMD, amb el seu processador Ryzen 5 3600/3700, està liderant el mercat europeu i s'ha reduït molt el mercat d'Intel, que havia estat dominant fins al llançament de la nova arquitectura d'AMD basada en Zen 3).

Una altra tècnica alternativa per a la virtualització és l'anomenada **paravirtualització**, la qual presenta una interfície de programari per a les màquines virtuals similar (però no idèntica) al maquinari subjacent. La paravirtualització proporciona *hooks* especialment definits per permetre que l'SO *guest* i el *host* reconeguin tasques privilegiades que si s'executen en el domini virtual degradaran les prestacions (bàsicament E/S). Això permet que, en una plataforma paravirtualitzada, el monitor de màquina virtual (VMM) sigui més simple i

pugui ressituar l'execució de tasques crítiques des del domini virtual al domini de *host*, per reduir així la degradació del rendiment de l'SO *guest*. L'únic problema d'aquesta opció és que l'SO *guest* s'haurà d'estendre amb aquesta *paraAPI* perquè disposi de les extensions necessàries per comunicar-se amb l'hipervisor. Xen, la plataforma més coneguda i utilitzada pels grans proveïdors, implementa aquesta tècnica i es pot executar amb dos tipus diferents de *guest*: sistemes paravirtualitzats i virtualització completa amb assistència de maquinari. Tots dos tipus es poden utilitzar simultàniament en un mateix sistema Xen i també es poden usar tècniques de paravirtualització en un *guest* amb assistència de maquinari.

Els diferents tipus de plataformes de virtualització utilitzades avui dia es poden classificar en dos tipus: aquelles en què l'hipervisor gestiona el maquinari directament, és a dir, hipervisor i SO *host* formen un conjunt indivisible (*bare-metal hypervisor*, o tipus 1), o aquelles en què l'hipervisor s'instal·la sobre un SO *host* i s'executa com una aplicació més d'aquest (*hosted hypervisor*, o tipus 2). En ordre alfabètic:

Tipus 1	**Tipus 2**
Citrix XenServer	Parallels Desktop for Mac
Linux + extensions Xen	QEMU
Microsoft Hyper-V	VirtualBox
Oracle VM Server	VMware Workstation Player
VMware vSphere Hypervisor (ESXi)	
Linux + KVM, FreeBSD + bhyve	

Com es pot apreciar, hi ha algunes opcions, com Linux + Kernel-based Virtual Machine (KVM) i FreeBSD + bhyve, que són mòduls del *kernel*. Per tant, si s'instal·len sobre Linux o sobre BSD converteixen l'SO en hipervisor de la primera categoria, però alguns autors, com que són mòduls, els classifiquen a la segona categoria. A continuació, estendrem algunes consideracions sobre els hipervisors més representatius.

1) Xen Project Code: aquest hipervisor ha estat utilitzat pels grans proveïdors, com ara AWS, Rackspace Hosting i Verizon Cloud, entre altres, i es pot instal·lar des de les distribucions de Linux que contenen els paquets, instal·larne les extensions o des d'una imatge ISO per instal·lar-lo directament (i fins i tot algunes distribucions inclouen extensions LiveCD per provar-lo sense necessitat d'instal·lar-lo, i també hi ha versions comercials de l'hipervisor: per exemple, Citrix).

Des dels orígens a la Universitat de Cambridge, l'empresa XenSource, la compra per Citrix i la tornada a la Fundació Linux com a projecte col·laboratiu, l'hipervisor ha passat per diferents fases (avui a https://xenproject.org/). En re-

lació amb el *cloud*, el 2009 neix XCP (Xen Cloud Platform), una distribució binària de Xen Project Management API (o XAPI), i sorgeixen diferents integracions del projecte Xen utilitzant XAPI, com ara Eucalyptus, Apache CloudStack, OpenNebula i OpenStack; apareixen els primers *clouds* públics amb XAPI i OpenStack. El 2012 els paquets XAPI s'inclouen a Debian i Ubuntu Server, la qual cosa permet crear els primers *clouds* utilitzant distribucions Linux. El 2013 XenServer (un *superset* d'XCP) és alliberat com a codi obert per Citrix (des de 2019 es considera obsolet; http://xenserver.org/). Avui dia, el projecte Xen, a més d'una nova versió 4.15 el 2021, té una àrea centrada en els sistemes operatius de *cloud*. Aquests sistemes operatius lleugers i especials (també coneguts com a *unikernels*) no estan dissenyats per funcionar sobre maquinari, sinó per produir petites màquines virtuals que poden generar *clouds* massius amb un maquinari mínim. Els projectes d'*unikernels* Mirage US, Unikraft i OSv són alguns dels més representatius i coneguts com a *Xen Project-powered*.

2) KVM: és la sigla de Kernel-based Virtual Machine i és una opció de virtualització total (*full virtualization*) per a Linux sobre arquitectures x86 que disposin d'extensions Intel VT o AMD-V. S'instal·la per mitjà de mòduls que s'insereixen al nucli de l'SO *host* i que proveeixen de virtualització l'entorn(kvm-intel.ko o kvm-amd.ko). Amb KVM es poden executar múltiples màquines virtuals (Linux o Windows) sense aplicar-hi cap modificació, i cadascuna podrà accedir al maquinari virtualitzat (xarxa, disc, targeta gràfica…). Hi ha una certa discussió sobre la vinculació entre KVM i QEMU (Quick Emulator) i els rols que exerceixen cadascun d'ells. QEMU és un paquet de codi obert per emular i virtualitzar un maquinari determinat. L'emulació permet que un programa o SO per a una arquitectura (per exemple, ARM) es pugui executar en una altra (per exemple, x86), mentre que la virtualització permet l'execució del codi natiu directament sobre el CPU *host*. QEMU admet la virtualització quan s'executa amb un hipervisor Xen o quan fa servir el mòdul de KVM al *kernel* (utilitzant-los com a acceleradors i sense usar l'emulació). La raó d'això s'explica per com es virtualitza el CPU: amb les extensions de maquinari dels processadors, es crea una *physical CPU slice* que es pot mapar directament al CPU virtual, la qual cosa permet una considerable millora, i això és el que ocorre, per exemple, amb el mòdul de KVM, i és utilitzat per QEMU quan escull KVM com a tipus de virtualització. Si no s'utilitza aquesta «acceleració» (per exemple, perquè el processador no disposa de les extensions de maquinari), QEMU utilitza TCG (Tiny Code Generator) per traslladar i executar instruccions del CPU virtual com a instruccions del CPU físic (emulació), amb la reducció consegüent de prestacions. Pel que fa a KVM, utilitza QEMU com a programa per crear una instància de la màquina virtual, i una vegada en execució aquesta serà com un procés regular de l'SO, a què es pot aplicar el conjunt de comandaments habituals, com ara *top*, *kill*, *kill taskset* i altres eines habituals per gestionar màquines virtuals.

3) VirtualBox: és un hipervisor de codi obert i de tipus 2; molt flexible per a arquitectures x86 i AMD64/Intel64, tant per a ús empresarial com domèstic; s'instal·la sobre Linux, Windows, Macintosh i Solaris, i és compatible amb un

gran nombre d'SO *guests* (Windows, DOS, Linux, OpenSolaris, US/2, OpenBSD, Android, ChromiunOS i més). VirtualBox es pot complementar amb HyperBox (permet gestionar diferents hipervisors i representa una alternativa lliure a productes comercials com ara VMware vCenter/ESXi i Citrix XenCenter) i phpVirtualBox, per gestionar les instàncies de VirtualBox remotament. És una opció molt acceptable per a proves i petites o mitjanes instal·lacions i possibilita que les màquines virtuals es puguin convertir a altres formats (per exemple, vmdk de VMware) i reutilitzar-se en altres infraestructures. L'opció de VirtualBox està tan estesa que hi ha diverses pàgines mantingudes per la comunitat que permeten baixar-se les imatges de diferents SO o distribucions ja instal·lats i preparats per carregar-se i executar-se; entre ells es poden esmentar OSBoxes, Virtualboxes i Oracle.

4) VMware Workstation Player i ESXi: VMware Workstation Pro i VMware Workstation Player (versió gratuïta per a ús no comercial o domèstic) s'han transformat en un estàndard a les empreses per executar múltiples SO com a màquines virtuals sobre un PC. Aquestes poden ser útils en diferents àrees, com ara el desenvolupament d'aplicacions i la replicació d'entorns, i també per a l'homogeneïtzació de recursos i l'accés controlat a aquests, amb la finalitat de tenir un entorn corporatiu homogeni i que es pot controlar i gestionar des del departament de TIC de l'empresa amb eines com VMware VSphere o Horizon FLEX. VMware vSphere Hypervisor (ESXi) és un hipervisor *bare-metal* gratuït que permet virtualitzar servidors i consolidar aplicacions amb tots els avantatges de la virtualització (reducció de consum, espai i inversió en maquinari, optimització...). ESXi disposa d'una consola per a la seva gestió i control i un entorn web com a eina per a la gestió i control de l'entorn virtual. Per a la gestió avançada de diferents servidors ESXi, gestió en calent de les màquines virtuals, gestió avançada de l'emmagatzematge i altres opcions de monitoratge i control, s'ha de recórrer a les eines propietàries de VMware, que estan disponibles amb llicència.

5) Microsoft Hyper-V: és un hipervisor per a sistemes de 64 bits amb processadors que disposen d'extensions de maquinari VT-x i AMD-V (no obstant això, els programes de gestió es poden instal·lar sobre sistemes x86). Des de la versió inclosa a Windows Server 2008 R2, l'hipervisor incorpora funcionalitats ampliades, com ara migració en calent de les màquines virtuals (*live migration*), emmagatzematge en màquines virtuals dinàmiques, compatibilitat millorada amb processadors i xarxes i compatibilitat amb Linux (Debian, Ubuntu, Centos/RH, Oracle, SuSE), FreeBSD i òbviament Windows (des de W10 i incloent-hi els servidors des de 2008). Hi ha algunes diferències si Hyper-V s'executa, per exemple, a Windows 10 i sobre Windows Server, que se centren en la gestió de la memòria, la virtualització de GPU, la migració en calent, les replicacions i la compartició de VHDX (discos virtuals), però per a la resta són iguals (sobre Windows 10 aquestes extensions reemplacen un producte anterior de virtualització anomenat Virtual PC). La seva instal·lació és simple tant

a W10 com a W2019, i permet que les màquines creades es puguin exportar a altres entorns Hyper-V o directament a Azure (plataforma de *cloud* públic de Microsoft).

6) Proxmox: amb les facilitats de KVM i altres tecnologies, han sorgit diferents solucions, com el Proxmox VE, que possibilita tenir un entorn virtualitzat amb KVM i Linux Containers (LXC). Aquesta solució completa de virtualització de codi obert és adequada per a moltes empreses, ja que permet gestionar i configurar màquines virtuals a partir de KVM, LXC, emmagatzematge i xarxes virtualizats i clústers d'alta disponibilitat. S'administra i monitora per mitjà d'una interfície web i admet com a SO *guest* tant Linux com Windows en màquines virtuals o contenidors com a virtualització del sistema Linux (a escala de l'SO). Aquesta distribució és un hipervisor tipus 1 (*bare-metal*) i està basada en la combinació de Debian, KVM i LXC; admet la migració en calent (*live migration*); permet configurar clústers d'alta disponibilitat, i pot interactuar amb diferents sistemes d'arxius locals o remots (NFS, iSCSI, Ceph, GlusterFS…).

Un punt important sobre els hipervisors, com ja s'ha comentat anteriorment, és la seva evolució cap a la **virtualització del sistema operatiu i els contenidors**. Des de fa uns quants anys hi ha una disputa entre els hipervisors i els contenidors en relació amb les prestacions i altres paràmetres d'eficiència, sobretot al *cloud* [Cvh14]. A més, en aquest sentit, Ubuntu ha apostat per un contenidor (LXD), aplicant la idea del contenidor al *cloud* i promocionant-lo com The LXD Container hypervisor, amb més densitat d'ESX, un 25 % més de rapidesa i sense latència. Indica que les màquines virtuals es poden moure als contenidors, fàcilment i sense modificar-ne les aplicacions i les operacions; LXD és un hipervisor de contenidor pur que executa sistemes operatius i aplicacions Linux no modificats amb operacions a l'estil de les màquines virtuals a una velocitat i densitat increïbles.

Un exemple d'aquesta tendència (però no l'únic) és Google, que ha invertit en contenidors des del principi: qualsevol cosa que es faci en la seva plataforma (cerca, Gmail, Google Docs) és un contenidor per a cada servei. Es podria pensar que hi ha molts tipus diferents de contenidors per provar, però, segons els experts, la gran majoria tenen el mateix codi a la part inferior (des de Google, LXD i Docker amb *cgroups* o *namespaces*, o *bean-counters* a OpenVZ), i, a més, han anat convergint i avui dia no hi ha pràcticament diferències entre ells.

Dos dels entorns de contenidors més representatius són Docker i LXC/LXD: disposen d'ecosistemes que permeten obtenir els contenidors com a VA (*Virtual Appliances*), però sense la càrrega que representen la màquina virtual i tot el sistema operatiu *guest* i les seves dependències. Segons Ubuntu, Docker i LXD són complementaris, perquè Docker està centrat en les aplicacions i LXD en *host machine containers* (actua com una màquina virtual): això permet executar i gestionar contenidors Docker dins de contenidors LXD, la qual cosa fa Docker encara millor, ja que LXD, en relació amb KVM, aconsegueix 10 vegades més prestacions, 14,5 vegades més densitat, un 57 % menys de latència i

un 94 % més de rapidesa en la càrrega. És habitual que els desenvolupadors d'aplicacions generin un contenidor per a la seva aplicació i les instruccions per al seu funcionament.

D'aquesta manera, amb Docker a la part superior d'LXC/LXD, i atesos els avantatges del seu codi obert, es pot empaquetar, enviar i executar qualsevol aplicació com un contenidor LXC/LXD lleuger, portàtil i autosuficient que s'executa pràcticament en qualsevol lloc. Amb això, els desenvolupadors tenen una eina potent per distribuir el seu codi, el qual podrà ser executat per l'usuari sense pràcticament cap intervenció ni configuració, i es desplegarà ràpidament al *cloud* amb una portabilitat total i sense invertir el temps que costa tenir una màquina virtual configurada correctament i enviar-la o desplegar-la. Segons els experts, aquesta tendència inicia un nou paradigma de virtualització (que alguns, com Ubuntu, anomenen *container hypervisors*) que aporta una nova manera d'empaquetar aplicacions, tenint serveis al *cloud* que poden entrar en qualsevol dispositiu, la qual cosa permetrà en un futur pròxim moure aplicacions des d'una plataforma a una altra fent realitat l'objectiu —poques vegades real— de la interoperabilitat.

Una mostra d'això és el creixement dels ecosistemes creats a l'entorn dels contenidors. Per exemple, Docker disposa d'un ecosistema oficial, DockerHub, que actua com a repositori de les distribucions oficials o els contenidors de la mateixa comunitat, que es poden baixar i engegar en qüestió de minuts. Podem trobar el mateix per a LCC/LXD, i també contenidors creats per particulars i posats a la disposició de la comunitat: per exemple, Flockport, en què es demostra que LXD també es pot centrar en les aplicacions, les quals no són un terreny totalment propi de Docker (que, no obstant això, hi continua tenint un paper protagonista).

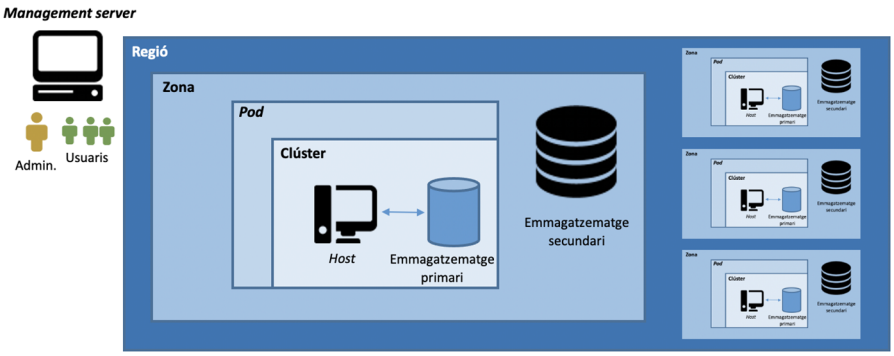
4.2. IaaS (infraestructura com a servei)

Dins de les plataformes de codi obert que podem trobar amb llicència Apache, GPL o similars, tenim les següents:

1) Apache CloudStack: és una plataforma IaaS de codi obert (des de 2010 parcialment sota GPL i des de 2011 sota ASL; vegeu-ne la història a l'apartat 1), orientada a crear, manejar i aprovisionar infraestructura *cloud* (recursos de còmput, *pools* d'emmagatzematge i xarxes) amb diferents supervisors (KVM, LXC, vSphere —via vCenter—, XenServer, Xen Project, Hyper-V). Amb això permet inicialitzar un servei elàstic de *cloud clomputing* i que els usuaris finals n'obtinguin els recursos; és capaç de manejar centenars de servidors físics distribuïts geogràficament en centres de dades i escalant molt bé (gairebé linealment) de servidor d'administració, la qual cosa evita la necessitat de servidors de gestió a escala de clústers. CloudStack configura automàticament les xarxes i l'emmagatzematge per a cada desplegament de màquines virtuals, les quals són proveïdes d'un *pool* de VA dins del mateix servidor. Aquestes VA disposen de serveis de *firewalling*, *routing*, DHCP, VPN, *console proxy*, *storage access* i *storage replication*; una interfície web (que es pot afinar) per aprovisionar i administrar el *cloud*, així com una interfície d'usuari (també web) per executar i gestionar les màquines virtuals.

Aquesta plataforma proveeix una *REST-like API* per operar i administrar el *cloud* i és compatible amb l'EC2 API, la qual cosa permet que eines d'EC2 s'hi puguin utilitzar. La instal·lació mínima consisteix en una màquina que executa CloudStack Management Server i una altra màquina que actua com a infraestructura *cloud* (que no és més que una màquina executant l'hipervisor), encara que per a proves de concepte es pot muntar tot en una única màquina (*management server* i *KVM hypervisor host*). El *management server* és l'únic punt de configuració: es pot executar en una màquina dedicada o en una màquina virtual, controla el desplegament de màquines virtuals als *hosts* (assigna IP i emmagatzematge a cada instància), s'executa sobre Apache Tomcat i requereix MySQL. L'arquitectura d'Apache CloudStack diferencia entre **regions** (col·leccions de zones pròximes geogràficament manejades per un o més *management servers*), **zones** (equivalents a un únic centre de dades amb un o més clústers, o *pods*, i emmagatzematge secundari), ***pods*** (bastidors amb commutadors de capa 2 i un o més clústers), **clústers** (un o més *hosts* homogenis i emmagatzematge primari), ***host*** (únic ordinador en un clúster, o hipervisor), **emmagatzematge primari** (el que proveeix un clúster per a l'execució de les imatges) i **emmagatzematge secundari** (recurs massiu d'emmagatzematge). Pel que fa a la xarxa, se n'ofereixen diferents tipus, però es poden classificar en dos escenaris concrets: **bàsica** (similar a la clàssica d'AWS, en què es proveeix una xarxa *layer-2* i s'aïlla el *guest* en *layer-3*, amb el *bridge* sobre els hipervisors) i **avançada** (utilitza aïllament en *layer-2*, o VLAN).

Com a requeriment, aquesta plataforma es pot instal·lar sobre CentOS/RHEL o Ubuntu. La figura següent mostra l'arquitectura d'aquesta plataforma.

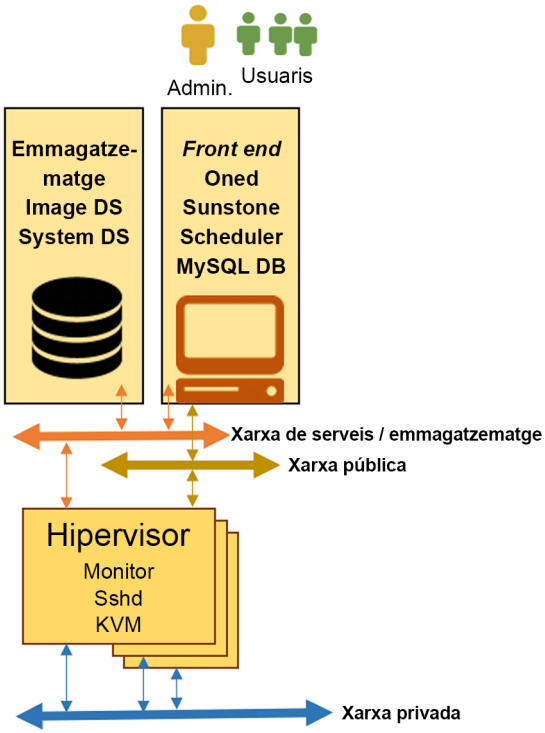


2) OpenNebula: plataforma de codi obert (ASL) per construir *cloud* IaaS tant privat com públic i híbrid, i que permet gestionar infraestructures de centres de dades heterogènies. La plataforma combina les tecnologies de gestió d'emmagatzematge, xarxa, virtualització, monitoratge i seguretat per desplegar serveis en múltiples nivells (per exemple, sobre clústers) com a màquines virtuals. A més, inclou funcions d'integració, gestió, escalabilitat, seguretat i comptabilitat; es basa en els estàndards actuals pel que fa a interoperabilitat i portabilitat, i permet que els usuaris es puguin interconnectar amb diferents

interfícies (EC2 Query, OGF Open Cloud Computing Interface i vCloud) i interactuar amb diversos hipervisors (KVM i VMware), contenidors (Docker) i micro màquines virtuals (Firecracker).

OpenNebula disposa d'un *marketplace*, que és un catàleg en què els usuaris i les organitzacions poden distribuir i desplegar ràpidament *appliances ready-to-run* sobre *clouds* OpenNebula. L'estabilitat de la plataforma i la qualitat dels serveis que proveeix l'han consolidada com una de les opcions IaaS actuals tant per a grans empreses com per a pimes, però també per a proveïdors de *hosting*, operadors de telecomunicacions, proveïdors de serveis de TIC, centres de supercomputació, laboratoris de recerca i projectes de recerca internacionals.

L'esquema d'interacció és a través d'un portal basat en rols i d'autoservei intuïtiu, amb un catàleg de serveis automatitzats, interfícies d'administració i l'*appliance* AppMarket, que permet gestionar i administrar tots els recursos per als diferents nivells o rols des d'un únic punt. Els requisits per instal·lar la plataforma es diferencien en instal·lacions bàsiques o mitjanes (desenes d'hipervisors) o avançades (centenars d'hipervisors). En relació amb les xarxes, les de serveis entre el *front end* i l'emmagatzematge poden ser compartides (atès el baix volum de comunicacions dels serveis), i les màquines virtuals necessitaran una xarxa pública i una altra de privada (per implementar VLAN aïllades). Pel que fa als sistemes operatius, poden ser Debian/Ubuntu o CentOS/RHEL (a totes les màquines), amb paquets específics per a cada distribució, amb KVM per als hipervisors i VLAN 802.1Q per a les instal·lacions bàsiques i VXLAN per a les avançades. Per a l'emmagatzematge, es pot utilitzar NFS/GlusterFS per a les primeres (amb imatges en format qcow2) i Ceph per a les segones; respecte a l'autenticació, es pot configurar el sistema natiu d'autenticació del que disposa OpenNebula, o Active Directory. La figura següent mostra l'arquitectura d'OpenNebula [Ona21].



OpenNebula utilitza KVM com a hipervisor, però, a causa de la seva flexibilitat i interoperabilitat, algunes organitzacions i empreses l'usen com a *cloud management* a VMware vCenter, anomenat vOneCloud, per proporcionar una capa d'aprovisionament multitinença per sobre de VMware vCenter. Aquests desplegaments busquen característiques de provisió, elasticitat i multitinença per a l'aprovisionament de centres de dades virtuals i la federació de centres de dades, mentre que la infraestructura és administrada per eines ja conegudes, com ara vSphere i vCenter Operations Manager. OpenNebula ha desenvolupat MiniONE: a partir de màquines virtuals sobre KVM, permet engegar una infraestructura de test sobre un node per avaluar-ne i analitzar-ne les característiques; és recomanable com a primer pas abans d'una instal·lació completa.

3) openQRM (*community edition*): és una plataforma de codi obert (GPL) de *cloud computing* per manejar infraestructura de centres de dades; que hi permet construir *clouds* privats, públics i híbrids, i que gestiona la virtualització (mitjançant KVM, Linux-VServer, OpenVZ, VMware ESX o Xen), l'emmagatzematge, la xarxa, el monitoratge i la seguretat. Amb això, permet desplegar serveis *multi-tier* sobre clústers de còmput com a màquines virtuals, combinant tant recursos locals com remots i gestionant-ne l'assignació mitjançant polítiques preestablertes.

La plataforma openQRM estableix un aïllament perfecte entre el maquinari (servidors físics o virtuals) i el programari (imatges dels SO i serveis), de manera que el maquinari es pot substituir sense necessitat de reconfigurar el programari i permet diferents models de migració de màquines virtuals (*physical to virtual* [P2V], *virtual to physical* [V2P] i *virtual to virtual* [V2V]), i també és

possible fer una transició d'un hipervisor a uns altres amb la mateixa màquina virtual. L'arquitectura d'aquesta plataforma és extensible mitjançant connectors (*plugins*), que permeten inserir API o connectors cap al *cloud* públic (per exemple, AWS) o uns altres *clouds* (per exemple, OpenStack). Els requeriments per fer una prova funcional són un servidor (per al servidor openQRM, virtualització i emmagatzematge) amb Intel/AMD de 64 bits, *dual/quad core*, extensions VT-x/AMD-V i Linux (Debian, Ubuntu o CentOS) amb accés a internet. L'última versió s'ha de baixar des del lloc del desenvolupador, però les versions anteriors s'han de baixar directament des de SourceForge, on també es pot accedir a la documentació de la comunitat (si bé està poc actualitzada).

4) OpenStack: és una plataforma de codi obert (ASL) de *cloud computing*, que permet desplegar IaaS mitjançant un conjunt de components interrelacionats que controlen recursos de maquinari (de diferents proveïdors), de còmput, de xarxa i d'emmagatzematge d'un centre (o més) de dades. Els usuaris poden gestionar i administrar el *cloud* utilitzant un panell de control (*dashboard*) al web o en CLI, o utilitzant una API RESTful. OpenStack es va iniciar el 2010 com un projecte conjunt de Rackspace Hosting i la NASA, i el 2014 va passar a ser gestionat per la Fundació OpenStack, amb l'objectiu de promoure aquesta plataforma i la seva comunitat, de la qual avui formen part més de 500 empreses, organitzacions i institucions.

La comunitat està organitzada en cicles de desenvolupament i actualitzacions semestrals, i es reuneix en sessions de treball (multitudinàries) a l'OpenStack Summit, per facilitar aquest desenvolupament i gestar plans de futur. A la sessió d'octubre de 2020 es van reunir (virtualment) 10.000 persones vinculades a la plataforma; tota la informació que es va generar (vídeos, tutorials, casos d'ús, etc.) es pot trobar en aquest enllaç. Avui es considera que la comunitat OpenStack està formada per més de 62.000 desenvolupadors i usuaris i unes 640 companyies de 187 països, i inclou més de 20 milions de línies de codi.

El primer codi d'OpenStack (2010) prové de la plataforma Nebula de la NASA (no s'ha de confondre amb OpenNebula) i de la plataforma Rackspace Cloud, i el 2011 Ubuntu adopta OpenStack per a l'estratègia de *cloud* (reemplaçant Eucaliptus) i dona suport per a *clouds* basats en Ubuntu 11.04 i la versió OpenStack Cactus. Passa igualment amb Debian 7.0, que inclou la versió Essex, i el mateix amb SuSE, i a partir de 2012 totes les grans companyies de tecnologia el comencen a incloure en les seves estratègies i a fer-lo disponible per als clients, integrat o adaptat juntament amb els seus productes (RHEL, Oracle, HP, etc.); avui dia ha arribat a tenir instal·lacions molt particulars, com la del CERN, China Mobile o Walmart, entre altres.

L'arquitectura d'OpenStack és modular amb diversos components, entre els quals es poden esmentar els següents [Osd]:

- ***Compute* (Nova)**: és la part principal de la IaaS i actua com a controlador del *cloud*. Gestiona els conjunts de recursos (*pools*) i pot treballar amb di-

ferents tecnologies de virtualització, maquinari de base (*bare metal*) i configuracions d'HPC, amb KVM, VMware, Xen i Hyper-V com a possibles hipervisors. Aquest mòdul no necessita cap maquinari propietari (està escrit en Python), escala horitzontalment i utilitza diferents biblioteques externes, com ara Eventlet, Kombu i SQLAlchemy.

- ***Compute* (Zun)**: proporciona una API OpenStack per llançar i administrar contenidors amb el suport de diferents tecnologies. A diferència de Magnum (Container Orchestration Engine Provisioning), Zun és per a usuaris que volen tractar els contenidors com un recurs administrat per OpenStack. Se suposa que els contenidors administrats per Zun s'integren bé amb altres recursos d'OpenStack, com ara la xarxa (Neutron) i el *block storage* (Cinder). Els usuaris disposen d'una API simplificada per administrar contenidors sense necessitat d'explorar les complexitats de les diferents tecnologies de contenidors.

- ***Networking* (Neutron)**: gestiona les xarxes i els seus paràmetres perquè no generin colls d'ampolla en una instal·lació, mitjançant l'autoservei. OpenStack proporciona diferents estratègies d'interconnexió (com ara xarxes planes o VLAN, IP estàtiques o DHCP reservats, IP flotants, etc.), la qual cosa permet als usuaris crear les seves pròpies xarxes, controlar el trànsit i connectar els servidors i els dispositius a una o més xarxes. Els administradors poden aprofitar les xarxes definides per programari de tecnologia (SDN), com OpenFlow, i annexar-hi serveis de xarxa addicionals, com ara els sistemes de detecció d'intrusos (IDS), equilibri de càrrega, tallafocs o xarxes privades virtuals (VPN).

- ***Block storage* (Cinder)**: aquest mòdul proporciona dispositius d'emmagatzematge permanents a escala de bloc que poden ser utilitzats per les instàncies de Nova, i gestiona la creació, muntatge i desmuntatge dels dispositius de bloc als servidors, els quals s'integren plenament a Nova i el panell de control (*dashboard*) perquè els usuaris puguin gestionar les seves pròpies necessitats d'emmagatzematge. A més de l'emmagatzematge local de Linux, pot utilitzar plataformes d'emmagatzematge com ara Ceph, CloudByte, Coraid, EMC GlusterFS, Hitachi, IBM, LIO, NetApp, Nexenta, Scality, SolidFire i HP, entre altres. L'emmagatzematge de blocs és apropiat per a escenaris on el rendiment és sensible, com ara bases de dades i sistemes d'arxius de creixement dinàmic, o per permetre al servidor accedir al bloc «en brut» (*raw*).

- ***Object storage* (Swift)**: és un sistema d'emmagatzematge redundant i escalable en què els objectes i arxius es gestionen en diverses unitats de disc repartides pels servidors del centre de dades, tenint-ne en compte la replicació i la integritat. Aquest servei pot escalar horitzontalment afegint nous servidors i gestiona la redundància des de la capa de programari, per la qual cosa pot utilitzar discos i servidors de baix cost.

- ***Identity* (Keystone)**: implementa un directori d'identitats per gestionar els serveis i els permisos de què disposen els usuaris, i actua, a més, com un sistema d'autenticació comú a tot el *cloud* integrant serveis de directori *back-end* existents, com LDAP. És compatible amb múltiples formes d'autenticació, incloent-hi usuari i contrasenyes, sistemes basats en *tokens* i inicis de sessió (a l'estil AWS). També inclou una llista de tots els serveis existents al *cloud* OpenStack en un sol registre perquè els usuaris i serveis de tercers puguin consultar quins recursos hi ha i amb quins permisos hi poden accedir.

- ***Image* (Glance)**: proporciona gestió sobre les imatges dels discos i servidors (les quals es poden utilitzar com una plantilla per a nous desplegaments). Entre les seves funcions més importants hi ha emmagatzemar i gestionar imatges de discos i de servidors en una varietat de *back-ends* (incloent-hi OpenStack Swift), i disposa d'una API REST per gestionar les imatges (per exemple, amb Heat per tenir metadades sobre les imatges, o amb Nova, per configurar una variació d'una imatge i generar una nova instància). Aquest mòdul és l'únic que pot agregar, esborrar, duplicar o compartir una imatge i atén les peticions dels altres mòduls segons es necessiti.

- ***Dashboard* (Horizon)**: interfície al panell de control que serveix als administradors i usuaris per accedir a la provisió i l'automatització de recursos basats en el *cloud* i per gestionar-les. Té un disseny obert i permet que s'hi puguin incloure noves funcionalitats (facturació, monitoratge, etc.), per transformar-se en punt d'accés a la infraestructura *cloud* (a més de l'API nativa d'OpenStack o l'API de compatibilitat EC2).

- ***Orchestration* (Heat)**: permet mitjançant plantilles gestionar configuracions i desplegaments complexos de *cloud* a través de l'API REST nativa d'OpenStack o l'API *cloudformation-compatible query*.

- ***Workflow* (Mistral)**: és un servei que gestiona els fluxos de tasques (*workflows*) generats pels usuaris (escrits en YAML). Pot interactuar mitjançant l'API REST: l'execució del *workflow* es pot iniciar a través de la mateixa API o programant-ne l'inici amb un esdeveniment (*trigger*).

- ***Database* (Trove)**: és un servei de *database-as-a-service* que aprovisiona un motor de bases de dades relacionals i no relacionals.

- ***Elastic map reduce* (Sahara)**: component que permet la provisió ràpida de clústers Hadoop mitjançant l'especificació simplificada per part de l'usuari.

- ***Bare metal* (Ironic)**: aprovisiona màquines *bare-metal* en lloc d'instàncies de màquines virtuals, utilitzant PXE i IPMI per gestionar les màquines de maquinari.

- ***Messaging* (Zaqar)**: servei de missatges multitinença disponible per a desenvolupadors web que funciona a través d'una API RESTful i que pot enviar missatges entre els diversos components d'un SaaS i aplicacions mòbils.

- ***Shared file system* (Manila)**: proveeix una API per gestionar compartició ns d'objectes (independents del proveïdor), i permet crear, esborrar i denegar objectes i donar-los accés a diferents entorns d'emmagatzematge, com ara EMC, NetApp, HP, IBM, Oracle, Quobyte i Hitachi, i sistemes d'arxius, com RHEL GlusterFS.

- **DNS (Designate)**: proveeix una API REST per manejar *DNS-as-a-service* i és compatible amb altres tecnologies, com PowerDNS i BIND. No proveeix el servei com a tal, sinó que actua com a interfície de DNS existents per manejar zones de DNS.

- ***Key manager* (Barbican)**: és una API REST dissenyada per a l'emmagatzematge segur, l'aprovisionament i la gestió de claus (secretes).

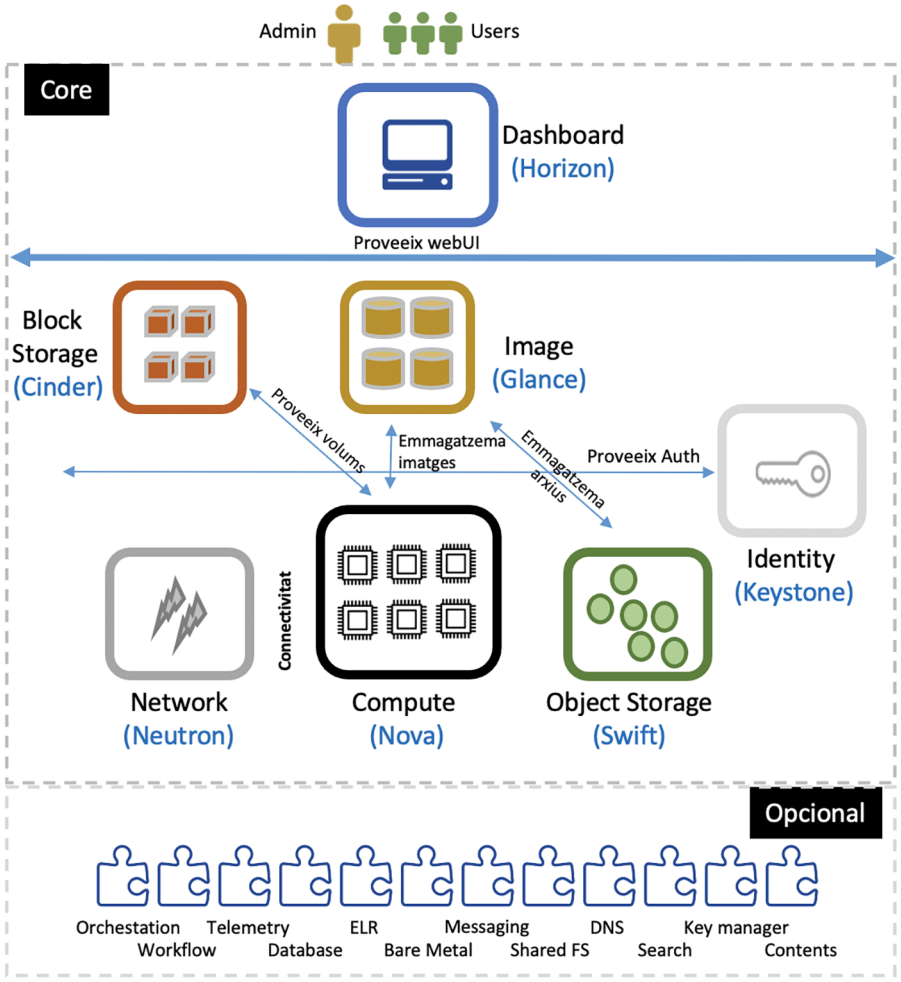
- ***Containers* (Magnum)**: Magnum fa que els motors d'orquestració de contenidors, com ara Docker Swarm, Kubernetes i Apache Mesos, estiguin disponibles com a recursos d'OpenStack. Magnum usa Heat (Orchestration) per orquestrar una imatge del sistema operatiu que conté Docker i Kubernetes i executa aquesta imatge en màquines virtuals o *bare metal* en una configuració de clúster.

Es pot obtenir una llista completa de tots els components i serveis d'OpenStack en aquest enllaç.

L'última versió, d'abril de 2021, és **Wallaby**, i a la seva pàgina web s'hi poden veure tots els components i serveis que inclou. Consultant el *marketplace*, es poden veure les distribucions o *appliances* que l'admeten, els *clouds* públics i privats que l'implementen, els proveïdors de formació o les consultories relacionades amb OpenStack o els controladors per a dispositius específics.

Com es pot observar al *marketplace*, existeix una gran quantitat d'*appliances* d'OpenStack amb diferents objectius i s'ha de tenir en compte que alguns d'aquests *appliances* són versions de prova per un temps limitat.

La figura següent mostra alguns dels components d'OpenStack i les seves interrelacions.



La manera més simple de provar OpenStack és mitjançant DevStack, que és una sèrie de scripts que s'utilitzen per desplegar ràpidament un entorn OpenStack complet basat en les últimes versions des del *git master*. S'utilitza de manera interactiva com a entorn de desenvolupament i com a base per a gran part de les proves funcionals del projecte OpenStack.

5) oVirt: és una plataforma de codi obert (ASL), creada per RH com a projecte de la comunitat i a partir de la seva distribució i tecnologia, de gestió i provisió d'entorns virtualitzats (amb KVM) i formada per dos components: oVirt Engine i oVirt Node. Permet, mitjançant una interfície web molt senzilla, gestionar centralitzadament màquines virtuals i recursos de còmput, xarxa i emmagatzematge de manera remota i sense accedir als recursos físics (és l'equivalent en codi obert a VMware vSphere).

Els seus requeriments són molt bàsics, però està tot orientat a RH o CentOS, amb processadors amb extensions AMD-V i VT-x i emmagatzematge com NFS, iSCSI, FCP, Local, POSIX FS o GlusterFS. A més, es necessitaran les imatges dels discos d'instal·lació (Windows, RHEL, Ubuntu, Fedora, OpenSuSE, CentOS), de les quals després es crearan les màquines virtuals.

El nucli d'oVirt Engine està escrit en Java; la interfície en GWT Web Toolkit i s'executa sobre el servidor d'aplicacions WildFly (abans JBoss), i l'arquitectura es pot veure en aquest enllaç. Els usuaris poden ser gestionats internament o connectats a un LDAP o AD. Per a l'emmagatzematge, oVirt utilitza PostgreSQL i proveeix una API RESTful com a interfície a les funcionalitats de nucli. oVirt Node és un servidor que executa RHEL o CentOS amb hipervisor KVM i un *daemon* VDSM (Virtual Desktop and Server Manager). Tota la gestió i l'administració es duen a terme mitjançant el portal web d'oVirt Engine, que es connecta amb VDSM per fer les tasques indicades. oVirt Engine es pot instal·lar en un node separat o sobre un node del clúster com una màquina virtual (*self-hosted engine*), la qual es podrà instal·lar manualment o utilitzar un *appliance* preparat per executar-se.

4.3. PaaS (plataforma com a servei)

Les PaaS proporcionen als usuaris eines per desenvolupar, executar i administrar aplicacions (generalment web), i atès que aquestes s'ofereixen com un servei a través d'internet, els desenvolupadors hi poden treballar sense haver de gestionar la infraestructura subjacent, la qual cosa els permet concentrar-se únicament a desenvolupar la seva aplicació. Les ofertes de les plataformes inclouen components de molts serveis utilitzats per desenvolupar programari, com ara bases de dades, servidors web, sistemes operatius i emmagatzematge: això possibilita que en algunes hi treballin múltiples usuaris, que col·laborin i que comparteixin codi i recursos.

Entre les més destacades (o entre les més ben classificades en alguns indicadors, com ara satisfacció, quota de mercat, suport i servei, per exemple, a G2 Crowd), hi ha **Heroku**, **Amazon Lamda**, **ServiceNow Now Platform**, **OpenShift i Platform.sh**, entre altres.

Si es consideren les plataformes amb un alt grau de satisfacció per part dels seus usuaris però que encara no han aconseguit una quota elevada de mercat, es pot esmentar **ServiceNow**. Entre els productes que tenen presència i recursos, però valoracions una mica inferiors a les del primer grup, hi ha **Google App Engine**, **AWS Elastic Beanstalk** i **Plesk**. Entre les plataformes molt especialitzades o per segments molt verticals del mercat, hi trobem **Cloud Foundry** i **Dokku**.

És interessant consultar llistes de les PaaS, com per exemple l'esmentada, ja que elaboren una valoració de la seva adequació per a diferents tipus d'empreses, i en un quadre on es consideren el grau de satisfacció i les preferències del mercat.

Hi ha diverses raons per seleccionar les plataformes PaaS, entre les quals es poden indicar les següents:

- Llenguatges de programació: atès que serà un projecte de programari, les plataformes triades seran les que siguin compatibles amb el llenguatge en què es desenvoluparà el projecte i estiguin dins dels seus requeriments.

- Públic enfront de privat: la PaaS pública representa beneficis de disponibilitat i immediatesa, perquè després del registre es comença a treballar. La PaaS privada requereix l'adequació i el desplegament d'infraestructura i provisió del servei, amb recursos addicionals, i un departament de TIC adequat als objectius, però té com a contrapartides el control, la seguretat i la privadesa.

- Temps de funcionament del servei: n'hi ha una dependència total, en el cas públic, i el temps està controlat, en el cas privat; això pot afectar seriosament la planificació del projecte.

- Estructura de preus: cal seleccionar el model adequat a les necessitats del projecte i amb equilibri entre serveis, suport i recursos.

- Recursos: s'ha de decidir si la plataforma aporta els seus propis recursos o si s'han de gestionar externament (per exemple, sobre EC2/S3).

- Integració: el projecte s'haurà d'integrar després amb altres serveis, per la qual cosa cal analitzar el temps necessari per a aquesta integració i d'acord amb quina política, per evitar inconvenients després que s'iniciï el treball.

Entre les plataformes de codi obert que es poden instal·lar sobre infraestructura pròpia (o *cloud*), podem esmentar les següents (cal tenir en compte que, si bé totes comparteixen objectius semblants, la visió i la missió de cadascuna d'elles són diferents i s'escapen al detall d'aquest apartat):

- **OKD**: és una distribució de Kubernetes optimitzada per al desenvolupament continu d'aplicacions i amb una implementació multitinença. OKD afegeix eines a Kubernetes per permetre el desenvolupament ràpid d'aplicacions, la implementació i l'escalat fàcils, i el manteniment del cicle de vida a llarg termini per a equips petits i grans. OKD és una distribució de Kubernetes (germana de Red Hat OpenShift) i l'amplia amb seguretat i altres conceptes integrats. OKD també es coneix com a Origen a Github i a la documentació. Si es busca suport empresarial o informació, Red Hat també ofereix Red Hat OpenShift Container Platform.

- **Apache Stratos** (abans W02): és un entorn PaaS extensible que permet executar aplicacions Apache Tomcat, PHP i MySQL, amb la possibilitat d'estendre's a altres serveis sobre les infraestructures *cloud* més importants. Per als desenvolupadors, Stratos proporciona un entorn basat en el *cloud* per desenvolupar, provar i executar aplicacions escalables, i els proveïdors de TIC obtenen benefici de les altes taxes d'utilització, la gestió automa-

titzada de recursos i la visió general de la plataforma, incloent-hi la supervisió i la facturació.

- **Cloudify** (Cloudify Community Version): és un entorn de codi obert d'orquestració que permet modelar aplicacions i serveis i automatitzar-ne tot el cicle de vida, incloent-hi la implementació en qualsevol entorn de *cloud* o centre de dades, monitorant tots els aspectes de l'aplicació desplegada, detectant problemes i fallades, permetent que es despleguin solucions (manualment o automàticament) i manejant les tasques de manteniment.

- **Tsuru**: PaaS de codi obert, disponible a GitHub, que facilita el desplegament ràpid i el control d'aplicacions en maquinari propi. Els desenvolupadors poden utilitzar Git, Tsuru Cli o el mateix panell de control (*dashboard*) per desenvolupar aplicacions web en diferents llenguatges, i es poden fer proves i tests instal·lant les màquines virtuals proveïdes per Vagrant.

- **Cloud Foundry**: plataforma PaaS de codi obert desenvolupada originalment per VMware, transferida a Pivotal Software (empresa conjunta entre EMC, VMware i General Electric) i gestionada ara per la Fundació Cloud Foundry. Aquesta plataforma comprèn tot el cicle de vida de les aplicacions, des del desenvolupament inicial i les etapes de proves fins a la publicació, ajustant-se a una estratègia de lliurament continu. Els usuaris tenen accés a un o més espais, que corresponen a una etapa del cicle de vida, i cada usuari adopta un rol diferent en funció de l'espai al qual tingui permisos per accedir.

- **Dokku**: per a alguns experts, la PaaS més petita i eficient per gestionar el cicle de vida de les aplicacions.

- **AppScale**: plataforma de codi obert que desplega i escala automàticament sobre *clouds* privats i públics aplicacions Google App Engine sense modificar-les.

Si bé algunes d'elles **no** es poden qualificar com a plataformes PaaS, és útil esmentar plataformes orientades a desenvolupadors (*cloud IDE*), però cal recordar que no totes ofereixen accés gratuït a una sèrie de recursos:

- **Cloud9.io** (avui forma part d'AWS): aquesta plataforma ofereix 40 llenguatges, control sobre l'aplicació, espai de treball (*workspaces*), públics il·limitats gratuïts i depuració d'aplicacions en diferents entorns (Django, Rails, WordPress...) en més de 300 combinacions, entre altres característiques.

- **Red Hat CodeReady Workspaces** (abans **Codenvy**): es basa en el projecte obert Eclipse Che i utilitza Kubernetes i contenidors per proporcionar a qualsevol desenvolupador un entorn coherent, segur i sense configuració.

L'experiència és ràpida i senzilla, com un entorn de desenvolupament integrat en un ordinador portàtil.

- **Koding**: plataforma de desenvolupament que orquestra tot l'entorn i d'on els desenvolupadors obtenen tot el que necessiten per crear entorns complets i específics de projectes en pocs segons, utilitzant una interfície senzilla per compartir, actualitzar i gestionar la infraestructura. Aquesta plataforma es pot provar des de koding.com. Un projecte similar és Codebox.

- **Codiad**: plataforma IDE basada al web amb uns requeriments molt reduïts, que permet disposar d'un entorn de desenvolupament al *cloud* sense els potents recursos que necessiten els entorns IDE habituals. Per això, els usuaris d'eines com ara Eclipse, NetBeans i Aptana troben en aquesta plataforma simplicitat i prestacions.

- **Codeanywhere**: plataforma *cloud* que permet disposar d'un entorn de desenvolupament amb el suport de més de 72 llenguatges i entorns preconfigurats.

- **Ideone**: compilador i depurador en línia en un model de *write-&-run*, amb suport per a més de 60 llenguatges i basat en tecnologia Sphere Engine.

- **Eclipse Che**: IDE nadiu de Kubernetes que permet que aquest sigui accessible per als equips de desenvolupadors, proporcionant espais de treball amb un sol clic i eliminant la configuració de l'entorn local per a tot l'equip. Che trasllada l'aplicació Kubernetes a l'entorn de desenvolupament i proporciona un IDE al navegador, la qual cosa permet codificar, construir, provar i executar aplicacions exactament com s'executen en producció des de qualsevol màquina.

També podem trobar entorns basats en el *cloud* d'objectius generalistes: per exemple, **Coding Ground**, que presenta un gran conjunt de terminals en línia i IDE interactius per editar, compilar, executar i compartir programes en línia en una plataforma *cloud*.

4.4. SaaS (programari com a servei)

Les plataformes SaaS són un model de distribució de programari en què el suport lògic i les dades que es manegen s'allotgen en servidors d'una companyia de TIC (no necessàriament la proveïdora del servei), i el client hi accedeix a través d'un programari específic (programari client, avui en desús) o a través d'un servei web (avui dia és l'opció més utilitzada, perquè no s'ha d'instal·lar res en l'equip local i només cal disposar d'un navegador).

L'empresa proveïdora s'ocupa del manteniment, l'operació diària i el suport del programari usat pel client, i en la majoria dels casos aquest hi pot accedir des de qualsevol lloc i des de qualsevol dispositiu. Dins d'aquest model de plataforma, l'execució i l'emmagatzematge de les dades del client sempre es fan als servidors del proveïdor i el client no s'ha de preocupar per la disponibilitat, accés, manteniment o seguretat, ja que tot això recau sobre el proveïdor en els termes de l'SLA signat amb el client.

Aquest tipus de model presenta **avantatges** com els següents:

- El client no necessita ni inversions en infraestructura ni personal tècnic especialitzat en el servei, la qual cosa redueix els costos i riscos d'inversió.

- L'operació (disponibilitat, funcionalitat, seguretat...) recau en l'empresa proveïdora d'acord amb un SLA.

- No hi ha abandonament del servei, ja que el proveïdor funciona mitjançant una quota de pagament per ús, per la qual cosa és necessari atendre el servei perquè es continuï pagant.

- No és necessari preocupar-se per llicències de màquines (les que es paguen tant si s'utilitzen com si no), ja que només es paga per l'ús.

- No és necessari modificar la màquina del client perquè accedeixi al servei, ni adequar-ne la potència ni les prestacions, perquè tot s'executarà a la màquina del proveïdor.

Així mateix, es pot enumerar un conjunt de **desavantatges**, com els següents:

- No es té accés a les dades (excepte si el servei preveu un mètode de baixada).

- Si no es compta amb mecanismes de xifratge i control, disminueixen l'índex de privadesa, el control i la seguretat.

- L'usuari accedeix a un servei i no el pot modificar o configurar més enllà de les opcions que li permet el proveïdor.

- Es poden produir el *data lock-in* (en cas de tancament d'activitats de l'empresa proveïdora) o el *vendor lock-in* (costos més alts que la competència, perquè els de migració encara són superiors).

- Es pot perdre el servei o reduir-se'n la qualitat si la connexió és deficient.

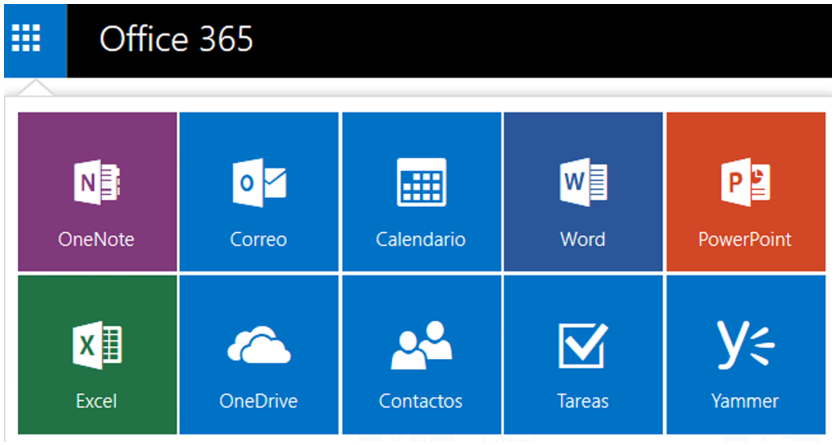
Mitjançant quatre preguntes, podem determinar si el servei al qual estem accedint és SaaS:

- Es pot accedir al servei a través d'un navegador o de protocols normalitzats com REST?
- Es pot accedir a un model de pagament com *pay-as-you-go*?
- El programari escala a demanda?
- El proveïdor assumeix la responsabilitat de la infraestructura, la gestió, el manteniment, la millora i la supervisió del programari, i deixa al client només l'ús del programari?

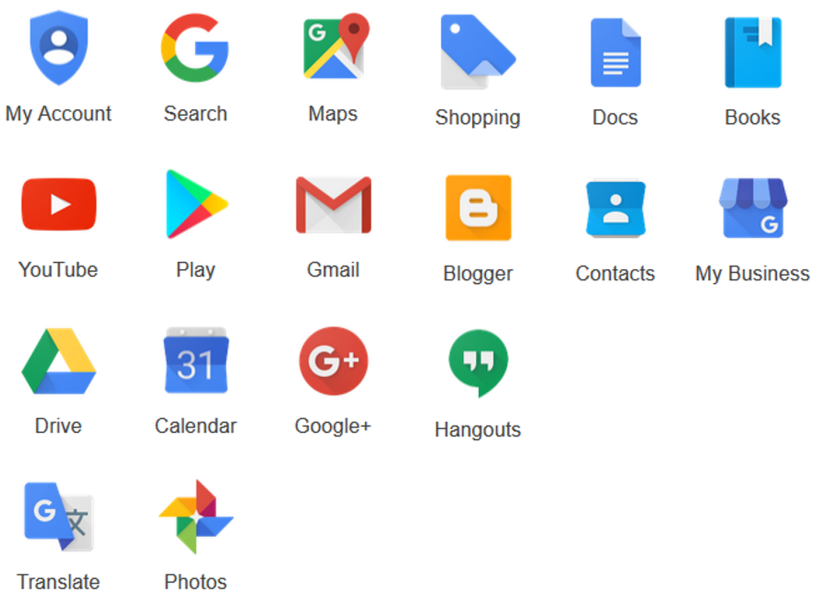
Si la resposta és un rotund **sí**, **podem** assegurar gairebé amb total seguretat que estem davant d'un SaaS.

Google i Microsoft

Dos exemples de SaaS que es mostren habitualment són dues de les grans companyies TIC: Google i Microsoft. Les imatges a continuació mostren els serveis oferts dins de les plataformes corresponents.



A Microsoft no són els únics serveis que s'ofereixen dins de la llicència SaaS (tant per a ordinador d'escriptori com per a dispositiu mòbil), que inclou eines d'edició de documents (Office365), col·laboració (SharePoint, Yammer), missatges instantanis (Lync), correu (Exchange) i emmagatzematge (OneDrive), però també CRM, comunicació (Skype) i BMI, entre els més destacats. Cal tenir en compte que alguns d'ells són gratuïts (no s'han de confondre amb el codi obert), com per exemple el correu i l'emmagatzematge (fins a un volum determinat), i uns altres tenen diferents models de llicència (per exemple, per als centres educatius, Office365 s'inclou a la llicència de l'SO Windows).



Google també ofereix els seus recursos, amb diferents models de negoci, encara que disposant d'una versió d'ús «gratuïta».

A més dels que es mostren, de cerca, navegadors, mapes, correu, documents, vídeos i fotos, emmagatzematge, calendaris, comunicació i traducció, ofereix xarxes socials, compres i serveis per a les empreses. En el cas de Google, la gran majoria de serveis són d'ús gratuït i el model de negoci (encara que no es coneix totalment) està en eines com Adwords i Adsense, que funcionen amb les dades que es recullen des de les diferents eines (concepte anomenat *dades massives* o *big data*).

Entre altres exemples de plataformes SaaS d'èxit, podem esmentar els següents (clicant a l'enllaç s'accedeix a la pàgina del servei):

Categories	Plataformes
Emmagatzematge	Dropbox MediaFire Mega
Col·laboració	Asana Todoist Trello Evernote
Documents	Pixlr, Polarr (edició de fotos) SiteBuilder (creació web) SlideRocket, Prezi (creació de presentacions) Slideshare (presentacions i documents) StackEdit (edició al web)
Mitjans	Netflix Flickr (fotos) OutBrain (continguts) Spotify (música) Wikipedia (enciclopèdia)
Serveis	bit.ly (URL) Flood (tests) Ionic (aplicacions) Mailchimp (correu) Panda (antivirus)

Categories	Plataformes
Social	BuzzStream Facebook Instagram LinkedIn LuckyOrange Twitter

Alguns autors consideren que el SaaS és l'evolució d'un model de negoci àmpliament conegut, anomenat **ASP** (proveïdor d'aplicacions), que proporciona serveis als clients a través d'una xarxa i permet l'accés a una aplicació de programari en particular. Aquest model va sorgir a partir dels costos creixents del programari especialitzat i de la complexitat d'instal·lar-lo i mantenir-lo, que es redueixen i es fan accessibles quan es funciona en mode ASP.

En principi, podem afirmar que és un concepte semblant al **SaaS**, però hi ha algunes diferències:

- L'ASP és un model de negoci especialitzat (de vegades amb programari de tercers adaptat a mida per a un únic client) i es basa en una relació «de tu a tu», mentre que el SaaS és un model global, flexible i que en principi es basa en la ubiqüitat i l'àmplia distribució (encara que la mateixa empresa o tercers puguin fer adaptacions i configuracions especialitzades).

- Generalment, l'ASP s'allotja en llocs independents del desenvolupador, mentre que en el SaaS els desenvolupadors ofereixen el programari i l'allotjament en un sol paquet (encara que hi comença a haver desenvolupadors que permeten utilitzar les IaaS de què disposi el client).

- La majoria d'aplicacions ASP s'executen mitjançant models client-servidor, per la qual cosa es requereix un programari específic en un SO determinat i es perden la ubiqüitat i la independència de l'SO (en el SaaS només es necessita un navegador); atès que és tecnologia d'un tercer, les actualitzacions i les versions depenen de la instal·lació que s'hagi dut a terme. Això no passa amb el SaaS, ja que tots els clients tenen sempre l'última versió i el suport és unificat amb un servei uniforme, amb la reducció consegüent dels costos.

- L'ASP és unitari, és a dir, són instàncies independents i molt adaptades o personalitzades per a empreses diferents, mentre que el SaaS, generalment, es tracta de múltiples clients amb una sola instància (multitinença).

- L'ASP pot integrar diferents serveis, com ara ofimàtica, directori i CRM, en un de sol, en funció dels acords amb els diversos proveïdors, mentre que un SaaS està orientat a un únic producte (encara que alguns proveïdors ofereixen SaaS multiproducte).

- Finalment, pel que fa a la relació amb el desenvolupador o proveïdor, en el SaaS la relació és més directa, ja que es parla directament amb qui ha desenvolupat el programari i no amb un tercer que actua d'intermediari. Es podria pensar, pel que s'ha descrit abans, que un programari complex, com un ERP, només pot funcionar en mode ASP, per la gran adaptació i personalització que es necessita, però hi ha casos, com per exemple OpenBravo (ERP, TPV, POS *retail*), que presten aquest servei com a SaaS o amb un IaaS extern (AWS) per a les pimes.

Respecte als models de monetització del SaaS, n'hi ha de diferents tipus i són bastant més difusos que en altres segments del *cloud*. Els habituals són *pay-per-user* (model semblant al de llicències en instal·lacions locals), o *pay-as-you-go* (per quantitat de recursos —emmagatzematge, CPU...— que s'han consumit en un període —per exemple, per hora—) o el seu equivalent *pay-what-you-want*. Pel que fa als models *freemium* i prèmium, el *freemium* representa un percentatge molt alt d'usuaris en relació amb el prèmium, però té limitacions en les prestacions o opcions i és la font que alimenta el prèmium. Els usuaris que realment provin i usin l'aplicació s'acabaran convertint en prèmium, pagaran per això i aquest fet sostindrà el model de negoci, que el proveïdor «cuidarà», en el sentit de pertinença a un grup amb privilegis. Els *freemium* tindran incentius per convertir-se en prèmium i, en moments determinats, reduccions dinàmiques de les prestacions per afavorir aquest pas. Altres models de monetització són els següents:

- *Long tail*, o llarga cua: catàleg de molts productes dels quals es venen poques unitats (és el model Amazon).

- *Bait & hook*, o esquer i ham: és semblant al *freemium*, es dona accés a uns recursos, però limitats, per incentivar el pas al pagament (per exemple, *pay-as-you-go*).

- *Paywall*, o subscripció: distribució dels costos en una gran quantitat d'usuaris i distribució de l'impacte inicial per una llarga subscripció.

- Anuncis: mètode indirecte en què la publicitat pròpia o de tercers monetitza el servei.

- Microtransaccions: generalment s'aplica a l'oci, s'adquireixen recursos o serveis amb inversions de valor molt baix però que en conjunt representen un gran volum.

- *Tiered service*: variant del *pay-as-you-go*, s'incrementa el preu de manera progressiva a mesura que es necessiten més recursos.

En relació amb l'estructura del SaaS, la majoria es basen en una arquitectura de multitinença, en què una única versió de l'aplicació, amb una única configuració (maquinari, xarxa, sistema operatiu), s'utilitza per a tots els clients,

i per afrontar l'escalabilitat, l'aplicació s'instal·la en diverses màquines a fi de permetre l'escalat horitzontal. Alguns SaaS, però, no utilitzen la multitinença, sinó models més aïllats, com els contenidors (model de Google) o les màquines virtuals, perquè milloren la privadesa i el funcionament i usen menys recursos. No obstant això, és un tema de gran discussió en fòrums especialitzats i cada proveïdor té la seva visió particular actualment entre la multitinença o els contenidors.

És important assenyalar que una plataforma SaaS es pot desplegar sobre una plataforma IaaS o PaaS (per exemple, OpenStack, OpenNebula, Cloudify, OpenShift, CloudStack...), perquè significa proveir una màquina virtual o contenidor de l'aplicació que executarà l'usuari sobre aquest recurs. No obstant això, existeixen plataformes específiques per construir infraestructures SaaS, entre les quals podem esmentar:

- **Appserver.io**: si bé últimament no s'ha actualitzat (https://github.com/appserver-io/appserver), és una plataforma interessant (*multithreaded application server for PHP*) per crear aplicacions multitinença basades en PHP, té una funcionalitat estable i no presenta grans inconvenients.

- **Drupal**: CMS en SaaS (openSaaS).

- **Elgg**: motor de xarxa social de codi obert perquè les empreses i institucions puguin crear xarxes socials específiques amb l'objecte de fidelitzar clients o mobilitzar entorns socials, en paral·lel amb una presència a les xarxes socials convencionals.

- **ERP5** (plataforma TioLive): servei ERP en SaaS de codi obert, ben documentat i dissenyat per a petites empreses.

- **Innomatic**: entorn de codi obert per construir plataformes SaaS de multitinença, productes i aplicacions empresarials en PHP. Amb aquesta plataforma, les empreses poden construir aplicacions multitinença preparades per executar-se al *cloud* i basades en els entorns Composer i Symfony2.

- **Magento** (comerç electrònic en SaaS): és una plataforma de codi obert per al comerç electrònic escrita en PHP i que utilitza diferents entorns PHP, com ara Laminas i Symfony. El 2018 Magento va ser adquirida per Adobe, però continua tenint una versió *open source*, si bé a parer d'alguns experts ha perdut la potencialitat de les versions anteriors.

- **OwnCloud**: entorn per a l'allotjament d'arxius, que permet l'emmagatzematge en línia i les aplicacions en línia (*cloud computing*). Es pot combinar amb serveis de LibreOffice per editar arxius d'ofimàtica simplement disposant d'un navegador.

- **Piwigo**: entorn de galeria de fotos de codi obert totalment reconfigurable i que inclou la possibilitat de gestionar també vídeos, aplicacions web i jocs, entre altres.

- **WordPress**: plataforma SaaS de codi obert per publicar i gestionar contingut (inclou extensions per al comerç electrònic), que disposa d'un model de negoci també amb llicència o *cloud*.

En model propietari o amb llicència, es poden esmentar Multihoster, SurPaaS, SaaS OpenBox, SaaS Maker (gratuïta per a desenvolupadors) i Koros, entre altres.

4.5. Altres serveis *cloud*

Ja s'ha esmentat abans que hi ha una gran quantitat de XaaS. En aquest subapartat en revisarem dos, que tenen un alt impacte sobre els serveis anteriors i que es poden comercialitzar per separat o units als anteriors.

1) ***Storage as a service (StaaS)***: en aquest apartat han sorgit diferents tipus de negocis, com ja s'ha comentat anteriorment, com per exemple Amazon S3, Dropbox i Drive (alguns autors els engloben tots en el *cloud storage*), que, tenint com a premissa l'emmagatzematge, presenten models d'explotació diferents; el primer ofereix els serveis d'emmagatzematge a internet per mitjà d'una interfície de servei web (REST, SOAP i BitTorrent), mentre que el segon i el tercer són explotats bàsicament mitjançant SaaS.

No obstant això, i més enllà dels serveis i models de negoci utilitzats, un aspecte important per a tota mena de *cloud* és el programari necessari per proporcionar el servei. En aquest sentit, descriurem els paquets de codi obert que permeten generar un servei d'emmagatzematge distribuït, fiable i segur.

- **GlusterFS**: utilitza el sistema d'arxius a l'espai d'usuari (FUSE) per generar un sistema d'arxius virtual (VFS), i permet crear un sistema d'arxius de xarxa en un clúster a l'espai d'usuari, utilitzant sistemes d'arxius existents, com ext3, ext4, xfs, etc., per emmagatzemar dades. La popularitat de GlusterFS ve de les capacitats d'escalat i accessibilitat, ja que pot proporcionar petabytes de dades en un únic punt de muntatge i distribueix els arxius a través d'una col·lecció de subvolums; fa unitats grans de la unió d'espais petits i distribuïts permetent la replicació i, per tant, la redundància i l'alta disponibilitat.

- **Ceph**: la base d'aquest sistema d'emmagatzematge és la Reliable Autonomic Distributed Object Store (RADOS), que proporciona aplicacions amb emmagatzematge d'objectes, blocs i arxius en un únic clúster d'emmagatzematge unificat. Amb les llibreries que s'ofereixen a les aplicacions client, els usuaris poden aprofitar RADOS Block Device (RBD) i RADOS Gateway, així com el sistema d'arxius Ceph. La RADOS Gateway

proporciona interfícies a Amazon S3 i OpenStack, compatibles amb el magatzem d'objectes RADOS. A més, Ceph proveeix una interfície Posix, per la qual cosa les aplicacions que utilitzen sistemes de fitxers compatibles amb aquest estàndard poden usar fàcilment el sistema d'emmagatzematge d'objectes de Ceph. Aquest sistema d'arxiu d'altes prestacions permet la lectura i l'escriptura parcials o completes, les instantànies, les assignacions de valor clau de nivell d'objecte i les transaccions atòmiques.

- **OpenStack**: aquesta arquitectura proporciona emmagatzematge d'objectes escalable i redundant utilitzant clústers de servidors, i pot escalar fins a petabytes de dades. Mitjançant aquest sistema d'emmagatzematge distribuït, ofereix als usuaris d'aquesta infraestructura escalabilitat i redundància sobre les dades emmagatzemades, i a més disposa d'interfícies amb Ceph, NetApp, Nexenta, SolidFire i Zadara. Les característiques inclouen instantànies, escalat en calent, suport per a emmagatzematge en bloc, *self-healing* i eines potents per a l'administració, ús, rendiment i auditoria.

- **Sheepdog**: permet emmagatzemar objectes distribuïts i es caracteritza pel seu petit volum, la simplicitat i la facilitat d'ús. Aquest sistema gestiona volums i serveis i maneja de forma intel·ligent els discos i nodes amb un escalat òptim que pot arribar a centenars o milers de dispositius. Sheepdog es pot annexar a la màquina virtual de QEMU i els *targets* SCSI de Linux, és compatible amb libvirt i OpenStack i pot interactuar amb HTTP Simple Storage. A escala del sistema d'arxius, permet fer instantànies, clonació i aprovisionament prim, i es pot annexar a màquines virtuals i SO que s'executin en *bare-metal* (amb interfície iSCSI).

2) ***Network as a service (NaaS)***: freqüentment s'associa el terme a altres serveis del *cloud* o també a *communication as a service* (CaaS), però en un sentit ampli inclou la provisió d'un servei de xarxa virtual des de la xarxa específica cap a tercers i sovint utilitza protocols com OpenFlow. Entre el programari de codi obert utilitzat per proveir aquest servei al *cloud*, podem esmentar:

- **Floodlight**: és un controlador OpenFlow d'altes prestacions en Java i amb llicència d'Apache. És un controlador SDN (xarxa definida per programari) obert que funciona amb dispositius (*switches*) físics i virtuals que interactuen a través del protocol OpenFlow, i es pot triar el protocol per interactuar amb els altres dispositius remots de xarxa (commutadors, encaminadors, commutadors virtuals o punts d'accés). En utilitzar OpenFlow, permet explotar-ne tota la funcionalitat: controlar remotament (taules de reexpedició de paquets, regles de flux, reexpedició o bloqueig de trànsit) i aprofitar interfícies personalitzades i llenguatges de seqüències de comandaments.

- **OpenStack Networking «Neutron»**: és una part del projecte OpenStack i proporciona una «xarxa com un servei» entre els NIC gestionats per Nova. Si bé és part del nucli d'OpenStack, «Neutron» es pot considerar, pel seu vo-

lum i funcionalitat, com un producte NaaS, en què els usuaris poden crear topologies d'aplicacions de múltiples nivells; utilitzar capacitats avançades de xarxa, com la supervisió de QoS o NetFlow d'extrem a extrem, i hi poden afegir serveis avançats mitjançant l'ús de connectors (*plugins*).

- **Open vSwitch**: és un commutador virtual multicapa de codi obert (ASL), d'altes prestacions i funcionalitat estesa amb una àmplia gamma de característiques, que inclouen VLAN 802.1Q amb ports de troncal i d'accés, enllaç NIC (amb i sense LACP *upstream*), NetFlow/sFlow, QoS, GRE, GRE sobre IPSEC, VXLAN i LISP *tunnling*, gestió de fallades de connectivitat 802.1ag, OpenFlow, reexpedició a través del *kernel* de Linux i una base de dades de configuració transaccional. Open vSwitch pot funcionar completament a l'espai d'usuari amb un mòdul de *kernel* o com un commutador basat en el *kernel* que admet múltiples tecnologies de virtualització, com ara Xen/XenServer, KVM i VirtualBox.

Resum

Com s'ha pogut veure en aquest mòdul, el món *cloud* gaudeix d'una perspectiva encoratjadora de futur i les tendències (informe de RightScale) van cap a estratègies *multiclouds*, centrant-se en una integració (*clouds* híbrids) en què els entorns DevOps prenen força dins de l'ecosistema *cloud* com a PaaS, i amb actors molt definits dins de cada categoria. A Eurostat es poden observar les tendències: com estan usant les empreses aquesta tecnologia i com es poden extreure les tendències del mercat per al futur pròxim.

També és necessari tenir en compte les recomanacions de la Comissió Europea, ja que, si bé tenen uns quants anys, moltes d'elles encara són vàlides (Estratègia *cloud* a Europa), i el finançament d'aquesta tecnologia en les propostes en curs o en les que encara estan obertes.

Activitats

1. Analitza dos hipervisors diferents i extreu conclusions dels seus avantatges i desavantatges.

2. Utilitza dues plataformes IaaS públiques i analitza'n les prestacions, avantatges i desavantatges.

3. Repeteix l'activitat 2 amb dues plataformes PaaS i dues plataformes SaaS (si pot ser, dins del mateix àmbit de negoci o funcionalitat).

4. Compara una plataforma PaaS orientada al desenvolupament d'aplicacions i un *cloud IDE*, i extreu conclusions sobre els avantatges de cadascun.

5. Elabora un quadre comparatiu sobre les diferents opcions del *cloud*, amb els seus avantatges i desavantatges, i proposa un model de negoci basat en una d'elles, tenint en compte els condicionants, el model de monetització, la infraestructura, la veta de mercat i tots els elements que consideris necessaris i que poden condicionar un negoci basat en aquesta tecnologia.

Glossari

acord de nivell de servei *m* Acord escrit entre un proveïdor de servei i el seu client amb l'objecte de fixar el nivell acordat per a la qualitat d'aquest servei.
en service level agreement
sigla **SLA**

aplicació d'internet enriquida *f* Aplicació web que té la majoria de les característiques de les aplicacions d'escriptori tradicionals (per exemple, implementades en AJAX).
en rich internet application
sigla **RIA**

appliance *m* Entorn que inclou tot el programari preconfigurat i llest per utilitzar, incloent-hi l'SO, que es pot executar sobre una màquina *bare-metal* o sobre un hipervisor.

arquitectura orientada a serveis *f* Arquitectura per dissenyar i desenvolupar sistemes distribuïts i que s'utilitza per al descobriment dinàmic i l'ús de serveis en una xarxa.
en service-oriented architecture
sigla **SOA**

bare-metal *loc* Dit d'una màquina sense SO instal·lat.

cloud computing *m* Proposta tecnològica que permet accedir a aplicacions o serveis remots de manera ubiqua a través d'un navegador: es poden aprovisionar i alliberar a demanda i en un temps reduït. Els seus equivalents en català són *informàtica en núvol*, *computació en núvol* i *serveis en núvol*.

còmput d'altes prestacions *m* Conjunt de recursos dedicats a executar aplicacions complexes amb grans necessitats de potència de còmput.
en high performance computing

contenidor *m* Tècnica de virtualització del sistema operatiu que permet altes prestacions (exemples: Docker, LXC/LXD).
sin. gàbia
en container, jail

data lock-in *m* Situació quan una empresa proveïdora tanca la seva activitat sense avís previ i les dades del client queden «atrapades» als servidors del proveïdor sense possibilitat d'accedir-hi.

despesa de funcionament *f* Cost d'operació.
sigla **OPEX**

hipervisor *m* Capa de programari que s'ha de posar entre l'SO de la màquina en què es volen virtualitzar els recursos.

InfiniBand *f* Xarxa de comunicació d'altes prestacions (per exemple, 40 Gbit).

infraestructura com a servei *f* Mitjà que és a la capa inferior i aprovisiona còmput, emmagatzematge i xarxa com a serveis estandarditzats a la xarxa.
sigla **IaaS**

inversió en béns de capital *f* Cost d'inversió.
sigla **CAPEX**

JavaScript asíncron i XML *m* Tècnica de desenvolupament web per crear aplicacions interactives o RIA.
en asynchronous Javascript and XML
sigla **AJAX**

kernel *m* Nucli de l'aplicació o del sistema operatiu, és a dir, les funcions essencials que componen l'aplicació o l'SO.

llicència de codi obert *f* Llicència que dona suport al codi *open-source*. Alguns exemples en són GPL (GNU General Public License), ASL (Apache Software License), CC (Creative Commons) i BSDL (Berkeley Software Distribution License).

multitinença *f* Mètode que permet executar una sola instància del programari a la infraestructura del proveïdor i serveix múltiples clients mantenint l'aïllament de les dades de cadascun.
en multitenency

OpenMP *f* Biblioteca per programar aplicacions de memòria compartida.

OpenMPI *f* Biblioteca per programar aplicacions distribuïdes.

plataforma com a servei *f* Encapsulació d'un entorn de desenvolupament (SO + aplicació + entorn) i un seguit de mòduls o complements que proporcionen tots els elements perquè un desenvolupador, per exemple, pugui treballar immediatament sense preocupar-se per la infraestructura de maquinari o programari.
sigla **PaaS**

programari com a servei *m* Aplicació completa que s'ofereix com un servei, a demanda, accessible a través d'un navegador i que l'usuari no controla; elimina la necessitat d'instal·lar qualsevol programari i de preocupar-se pel manteniment de l'aplicació.
sigla **SaaS**

programari de codi obert *m* Programari que disposa una llicència de manera que els usuaris en poden estudiar, modificar i millorar el disseny mitjançant la disponibilitat del codi font.
en open-source software

servei web *m* Tecnologia que utilitza un conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions desenvolupades en llenguatges de programació diferents.
en web service

standby service *m* Servei preparat i llest per posar-se en execució i que estarà disponible en un temps mínim.

transferència d'estat de representació *f* Interfície entre sistemes que utilitzen HTTP per obtenir dades o indicar l'execució d'operacions sobre les dades, en qualsevol format (XML, JSON, etc).
en representational state transfer
sigla **REST**

vendor lock-in *m* Dependència d'un proveïdor de productes i serveis, que comporta que no sigui possible canviar a un altre proveïdor sense costos de canvi substancials, encara que el canvi signifiqui una reducció de costos o prestacions millors.

virtualització *f* Tècnica mitjançant la qual es presenta una visió virtual de les capes subjacents a l'aplicació o a l'SO. En el cas de la virtualització del maquinari, l'SO «veu» una màquina de maquinari equivalent però que és virtual.

Bibliografia

Tots els enllaços s'han visitat el juliol de 2021.

[Acs] All AWS Customer Stories. <https://aws.amazon.com/solutions/case-studies/all/>

[Bcs14] The Best Cloud Computing Services of 2020. <https://www.business.com/categories/cloud-computing-services/>

[Cca09] Cloud Application Architectures. George Reese. 2009. O'Reilly Media, Inc.

[Cca11] Considering Cloud Appliances for Private Cloud Deployments. Would you prefer to build a cloud from components or plug it in and go? Vince Vasquez. Cloudbook Journal. Vol 2 Issue 3, 2011.

[Ccp13] Cloud Computing: Paradigms and Technologies. A. Shawish, M. Salama. Inter-cooperative Collective Intelligence: Techniques and Applications. 2013. Springer. 495. pp 39-67. <http://link.springer.com/chapter/10.1007%2F978-3-642-35016-0_2>

[Cis16] Magic Quadrant for Cloud Infrastructure as a Service. L. Leong, G. Petri, B. Gill, M. Dorosh. 2016. ID: G00278620. Gartner.

[Cmh15] Magic Quadrant for Cloud-Enabled Managed Hosting. D. Toombs, B. Gill, M. Dorosh. 2015. ID: G00269143. Gartner.

[Cvh14] Containers vs Hypervisors: The Battle Has Just Begun. Russell Pavlicek. 2014. Linux.Com. <https://www.linux.com/news/containers-vs-hypervisors-battle-has-just-begun>

[Dcc11] The NIST Definition of Cloud Computing. P. Mell and T. Grance. National Institute of Standards & Technology. Special Publication 800-145. 2011. <http://dx.doi.org/10.6028/NIST.SP.800-145>

[Eoc] The Evolution of the Cloud. The Work, Progress and Outlook of Cloud Infrastructure. Ari Liberman García. MIT. 2015. <https://dspace.mit.edu/bitstream/handle/1721.1/100311/932065967-MIT.pdf?sequence=1>

[Idd09] Info World Cloud Computing Deep Dive. 2009. <http://www.infoworld.com/resources/15675/cloud-security/download-the-cloud-security-deep-dive>

[Mcc13] A Survey of Mobile Cloud Computing Application Models. A. Khan, M. Othman, S. Madani, S. Khan. IEEE Communications Surveys & Tutorials. 2013. Vol 16 (1). <http://dx.doi.org/10.1109/SURV.2013.062613.00160>

[Ona21] OpenNebula. Open Cloud Reference Architecture. 2015. OpenNebulaSystems. <https://support.opennebula.pro/hc/en-us/articles/204210319-Open-Cloud-Reference-Architecture-White-Paper>

[Osd] OpenStack Documentation. <http://docs.openstack.org/#docs-main-body>

[Ovf15] Open Virtualization Format Specification. DSP0243. 2015. <https://www.dmtf.org/sites/default/files/standards/documents/DSP0243_2.1.1.pdf>

[SaaSA] SaaS. Apprenda. <https://apprenda.com/library/software-on-demand/>

[Tch15] Timeline of Computer History. <http://www.computerhistory.org/timeline/2015/>

[Tif06] The Information Factories. George Gilder. 2006. Wired. <https://www.wired.com/2006/10/cloudware/>

[Vcc10] A View of Cloud Computing. M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. 2010. Communications of the ACM, Vol. 53 No. 4, Pages 50-58. <https://doi.org/10.1145/1721654.1721672>

[Wtn08] 15 Ways to Tell Its Not Cloud Computing. James Governor. 2008. RedMonk. <http://redmonk.com/jgovernor/2008/03/13/15-ways-to-tell-its-not-cloud-computing/>

[Iar] Icones amb llicència d'ús lliure. <http://www.iconarchive.com> <http://www.customicondesign.com> <http://icons8.com>