Fonaments i plataformes de *cloud computing*

PID_00269375

Remo Suppi Boldrito

Temps mínim de dedicació recomanat: 6 hores

Universitat Oberta de Catalunya

Remo Suppi Boldrito

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

Primera edició: febrer 2020
Autoria: Remo Suppi Boldrito
Llicència CC BY-SA d'aquesta edició, FUOC, 2020
Av. Tibidabo, 39-43, 08035 Barcelona
Realització editorial: FUOC

Índex

Introducció

El *cloud computing* (o, en les seves accepcions en català, computació en el núvol, serveis en el núvol, informàtica en el núvol, núvol de còmput o núvol de conceptes) és una proposta tecnològica adoptada, avui dia, per la societat en general com a forma d'interacció entre els proveïdors de serveis, els gestors, les empreses/Administració i els usuaris finals per a la prestació de serveis i la utilització de recursos en l'àmbit de les TIC (tecnologies de la informació i la comunicació), i sustentada per un model de negoci viable econòmicament.

Una definició interessant que aporta més definició sobre el *cloud computing* és la que realitza el National Institute of Standards & Technology (NIST), que expressa el següent:

> «El *cloud computing* és un model que permet l'accés ubic, a conveniència, sota demanda i per xarxa a un conjunt compartit de recursos informàtics configurables (per exemple, xarxes, servidors, emmagatzematge, aplicacions i serveis) que poden ser ràpidament aprovisionats i alliberats 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 els proveïdors i els usuaris, i tota la cadena intermèdia de transformació i processament de la informació, per mitjà d'una xarxa que en la majoria dels casos és internet. És habitual que els usuaris utilitzin serveis en el *cloud* (sia serveis empresarials o personals), per exemple, el correu electrònic, les xarxes socials, l'emmagatzematge d'arxius (fotos, vídeos, ofimàtica), i que els utilitzin des de diversos dispositius i amb diferents interfícies. Aquest paradigma és tan usual que els usuaris digitals (joves generacions) no coneixen una altra forma 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 o usabilitat de diferents interfícies o mitjans d'accés són essencials per al seu quefer quotidià (compres, oci, serveis administratius, negoci i una llarga llista més).

1. Breu història

Si bé tenim present com a precursors les grans companyies que van oferir aquest tipus de serveis (Yahoo! el 1994 i Google el 1998 –motor de cerca–, Hotmail el 1997 –correu– i Amazon el 2002 i AWS el 2006), és un concepte que té les seves arrels en els anys seixanta. Està basat en les idees de John McCarthy (professor emèrit de la Universitat d'Stanford i cofundador del Laboratori d'Intel·ligència artificial del MIT), que va predir, el 1961, durant un discurs per al centenari del MIT, 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, les 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 J.C.R. o simplement Lick), que ha estat una figura molt important en l'àmbit de la ciència computacional i recordat, particularment, per 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 fos construïda). J.C.R. va treballar i va aportar finançament a projectes com ara Arpanet (predecessor d'internet) o la interfície gràfica d'usuari que van permetre l'accés a serveis i recursos a persones no especialistes. Idees que, segons les opinions dels experts actuals, coincideixen que s'assemblen molt a la idea del *cloud computing* tal com el coneixem avui.

Però la història del *cloud computing* també se sustenta en visionaris com 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, en l'article «The Information Factories» [Tif06], afirma que el PC ja no és el centre d'atenció i emmagatzematge o processament de la informació, sinó que ho és el *cloud*, i que aquest serà el responsable d'emmagatzemar les dades de cada individu al planeta i que accedirà a aquestes almenys una vegada a la vida. El 2013, Gilder va publicar el 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.

Des dels anys seixanta, les evolucions han estat notables i intrèpides, com es pot veure en el *Timeline of Computer History* [Tch15], però probablement, pel que fa als desenvolupaments més vinculats al *cloud* actual, cal destacar el desenvolupament de la web (Tim Berners-Lee proposa les idees el 1989, però no és fins l'any 1990 que hi ha un prototip del WorldWideWeb) i la seva evo-

lució més recent, Web 2.0, i el desenvolupament d'internet (que a Espanya no té impacte fins al començament dels anys noranta amb la transformació de RedIris i la connexió plena a internet).

A partir de llavors comencen a sorgir una sèrie de serveis, bàsicament aplicacions centrades en cerca, correu i pàgines com Wandex, el 1993, que pretenia ser un programa per a mesurar la grandària d'internet i que, desenvolupat en el MIT, va acabar sent el primer cercador, i després Yahoo! el 1994, Altavista el 1995 (que ha anat canviant de propietaris des d'Overture, Yahoo i ara Microsoft) o Hotmail el 1996 (posteriorment comprat per Microsoft, el 1997, quan ja comptava amb sis milions d'usuaris de correu) o, més recentment, també en 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 en els últims quatre anys «Forbes' Most Innovative Company»), que va ser una de les primeres a oferir als seus clients el que avui s'anomena SaaS (*software as a service*) i que era un programari per a l'automatització de les vendes per mitjà d'una simple pàgina web. Actualment, alguns experts consideren que aquest servei pot ser un PaaS (*platform as a service*) i uns altres el consideren una plataforma híbrida, ja que ofereix les dues variants. Això va permetre mostrar una «nova» forma de negoci per mitjà d'internet i generar el que avui és una realitat a 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 i la seva ampliació el 2006 amb el llançament d'Amazon Web Service, AWS (sent *Elastic Compute/Storage Cloud* –EC2/S3– els serveis més importants d'AWS per a proveir instàncies d'un servidor sota demanda per a còmput i emmagatzematge respectivament) com un servei orientat al negoci que va permetre a les petites empreses i particulars usar equips en què s'executen les seves pròpies 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, popularitzant-se i guanyant en seguretat i servei. També altres grans proveïdors implementen les seves pròpies infraestructures *cloud*. Per exemple, Microsoft Azure es va anunciar l'octubre de 2008 (però no és fins l'any 2010 que comença a prestar serveis), IBM el 2011 (llança *SmartCloud Framework* per a donar suport al seu projecte *Smarter Planet*) o Oracle el 2012 (Oracle Cloud).

Pel que fa a les plataformes més representatives (no les úniques) en la construcció de *clouds*, el 2005 es comença a treballar en un projecte de recerca orientat a aquest àmbit basat en programari *open source* i, el 2008, neix OpenNebula. En aquest any es forma el consorci Opennebula.org, que és finançat per un projecte de la UE (7th FP) arribant a tenir, el 2010, 16.000 MV configurades. El 2010, Rackspace i la NASA llancen un iniciativa *open source cloud-software*, que anomenen OpenStack, amb l'objectiu d'oferir a les organitzacions la possibilitat de muntar els seus serveis de *cloud* sobre maquinari estàndard (el 2012

es crea l'OpenStack Foundation per a promoure aquest programari en la comunitat amb dos-cents socis, gran part de tots els actors importants del món IT). Openstack s'ha popularitzat i avui es pot trobar en les distribucions més conegudes (Redhat, Ubuntu, SuSE...) o també empreses que ofereixen les seves integracions com 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 al llarg dels darrers anys) sota GPLv3. El 2011, Cloud.com és adquirida per Citrix Systems i CloudStack passa a ser un projecte de la Fundación Apache i a distribuir-se amb la llicència Apache *Software License*. El projecte *Virtual Grid Application Development Software* (liderat per la Rice University entre 2003-08) és el precursor de la plataforma per a generar i configurar *clouds* anomenada Eucalyptus. Eucalyptus és inclòs a Ubuntu 9.04 (2009) i es caracteritza per integrar-se amb AWS. Des de 2014, la companyia del mateix nom, i que desenvolupa i manté el paquet, va ser adquirida per Hewlett Packard i el paquet es va passar a anomenar HP Helion Eucalyptus.

En resum, es pot argumentar que en l'actualitat és poc freqüent que una institució, una empresa mitjana o una gran empresa no tingui externalitzats alguns serveis en el *cloud* per als seus treballadors o usuaris finals, o no utilitzi el *cloud* per a alguns dels aspectes vinculats al seu negoci (correu, web, intranet, etc.), a excepció d'aquelles que el *core business* depengui de la privadesa de les dades o estiguin especialment protegides.

2. Característiques, avantatges i desavantatges del CC

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

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

És interessant l'article 15 «Ways to Tell Its Not Cloud Computing» [Wtn08], que expressa clarament què no és *cloud computing* o els tres criteris per a definir-lo, expressats per George Reese [Cca09], de quan un servei és *cloud*: accessibilitat via *web browser* (no-propietari) o una API de *web services*, inici sense aportar capital i solament pagar pel que s'usa quan s'usi.

De forma descriptiva, podem enunciar les **característiques** clau següents per a la computació en núvol (ordre alfabètic) [Dcc11]:

1) Agilitat i autoservei: rapidesa i capacitat de proveir recursos (de forma gairebé immediata) sense grans intervencions ni accions per part 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, en forma no assistida, aprovisionar capacitats de còmput, emmagatzematge i xarxes segons li calgui de forma automàtica, 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 quants minuts), per mitjà de la xarxa.

2) Cost: atesa l'eficiència i 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. L'usuari final no ha de fer front a les despeses derivades de la implantació de la infraestructura civil o serveis, ni a la inversió en màquines, programari i recursos humans, per la qual cosa comptabilitzant tot el model surt favorable en relació amb el *cloud*. A més, com que el control i el monitoratge són automàtics, es poden realitzar mesures amb un cert nivell d'abstracció apropiat per al tipus de servei (per exemple, comptes o comptes

actius, emmagatzematge, processament o amplada de banda). Per exemple, és possible ajustar el cost a models de pagament-per-ús, pagament-per-peticions, etc. proporcionant 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 o ús d'aquests. És a dir, si es pot preveure la càrrega o ús no cal aprovisionar tots els recursos des de l'inici com si es tractés d'una inversió local, sinó de planificar-los per quan la demanda ho requereixi amb la conseqüent reducció del cost.

4) Independència entre el dispositiu i la ubicació: independitzar el recurs de l'accés i facilitar que aquests puguin ser des d'una xarxa local, corporativa o un altre tipus i des de diferents dispositius (capacitat o tipologia).

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 instal·lat al dispositiu de l'usuari final.

6) Rendiment i gestió de recursos: control exhaustiu i monitoratge eficient dels serveis per a aconseguir una alta disponibilitat i utilització òptima dels recursos de forma automàtica. Aquestes característiques permeten reduir al màxim les ineficiències aportant control i notificació immediata, i també transparència i seguiment tant al proveïdor com a l'usuari final. A més, com que els recursos es poden configurar per a servir a múltiples usuaris en un model de tinença múltiple, els recursos físics i virtuals poden ser assignats dinàmicament i reassignats 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 (solament a un nivell d'abstracció molt alt, per exemple, país o centre de dades).

7) Seguretat: es pot incrementar, atesa la particularitat de tenir les dades centralitzades. Correspondrà al responsable de l'aplicació la seguretat d'aquesta i, al proveïdor, la responsabilitat de la seguretat física. Amb això, la seguretat serà almenys igual que en els sistemes tradicionals, però podrà ser millorada (i s'haurà d'estipular quin nivell es desitja en l'SLA –*Service Level Agreement*–), ja que el proveïdor té interès en què la seva infraestructura sigui segura per als seus clients com a base de la qualitat del negoci i podrà invertir en aquesta globalment, mentre que aquesta inversió pot ser inacceptable per a un client si ha de fer el mateix 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 o servidor i l'espai, facilitant el desplegament ràpid de serveis i garantint l'alta disponibilitat (serveis en *stand by*) i mobilitat per càrrega (moviment de màquines virtuals a servidors més ociosos quan la càrrega augmenta).

Per a convèncer els més reticents, es poden enumerar un conjunt d'**avantatges** que aporta el *cloud* com a paradigma, que són (ordre alfabètic):

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 d' aplicacions empresarials. L'usuari final queda totalment convençut quan se li permet accedir des de qualsevol dispositiu, sense requeriments especials pel que fa a les necessitats del mateix.

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

11) Simplicitat: rols i responsabilitats molt ben definits, separació de les activitats ben estipulades (per exemple, proveïdor de continguts, proveïdor d'aplicacions, proveïdor d'infraestructura i usuari final), la qual cosa es tradueix en una forma òptima de treballar i menor inversió pertot arreu.

12) Reducció de riscos i rapidesa: no cal una gran inversió ni adequació de llocs o entorns, sent possible accelerar al màxim la implantació de nous serveis en les diferents etapes de desenvolupament fins a producció.

13) Reducció de l'ús d'energia (consum eficient): atesa la tecnologia utilitzada i la utilització eficient dels recursos (afavorit per la virtualització), solament es consumeix el necessari, a diferència dels centres de dades tradicionals en què hi ha un consum fix no depenent de la càrrega o utilització.

Si bé els avantatges són evidents i molts actors d'aquesta tecnologia la basen principalment en l'abaratiment dels costos de servei, hi comença a haver opinions en contra que consideren que un *cloud* públic pot no ser l'opció més adequada per a determinats tipus de serveis o infraestructures. Entre les causes principals de **desavantatges** que esgrimeixen alguns experts, hi ha (ordre alfabètic):

1) Centralització: tant de les dades com de les aplicacions. Genera una dependència en relació amb el proveïdor, ja que si no disposa de la tecnologia adequada (monitoratge i detecció) i dels recursos apropiats (alta disponibilitat), pot generar talls o inestabilitats en el servei. En aquests casos, pren especial rellevància l'SLA (*service level agreement*), que especificarà a què està obligat el proveïdor i les indemnitzacions que haurà de fer front per això.

2) Confiabilitat: la «salut» tecnològica i financera del proveïdor serà un element clau en la continuïtat del seu negoci i el del client, per la qual cosa les decisions que prengui el proveïdor afectaran directament el negoci client, i si aquestes no són les adequades, afectaran directament l'empresa. També l'empresa quedaria a la mercè d'un mercat molt dinàmic quant a fusions i monopoli (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 forma abrupta i es produeixi el *data lock-in* de l'empresa en els servidors del proveïdor sense possibilitat de poder recuperar les dades (situacions que han passat amb el 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 en aquests augmentarà, i si el proveïdor no disposa d'un pla d'escalabilitat a mitjà i llarg termini per a assegurar un creixement sostenible des del punt de vista de les necessitats dels seus 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 conseqüent retard en el seu desplegament) per al proveïdor si aquests no són requerits per altres clients, afectant el negoci d'un en particular que sí que els necessita per al seu.

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

6) Risc i privadesa: les dades «sensibles» de negoci no resideixen en les instal·lacions de les empreses i la seguretat no depèn dels recursos humans d'aquesta 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 pel seu possible robatori (còpia), accés a la informació (lectura) o destrucció (eliminació).

7) Seguretat: atès que la informació haurà de travessar diferents canals i serveis, pot resultar que cadascun d'aquests sigui un focus d'inseguretat. Si bé això pot ser resolt mitjançant canals i serveis segurs, la possibilitat d'una fallada en la cadena de xifrat de la informació pot ser possible amb els conseqüents problemes 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 (que en l'actualitat s'ha demostrat com un dels més freqüents) que fa que un client depengui d'un proveïdor de productes i serveis, incapaç d'usar un altre proveïdor sense costos de canvi substancials, encara que el canvi signifiqui una reducció de costos o millors

prestacions. Molts desenvolupadors o usuaris finals són reticents al *cloud* per aquest motiu, ja que significa un compromís adquirit. Després, si es desitja canviar, el cost serà molt elevat.

El [Vcc10] amplia aquests «desavantatges» i demostra algunes d'aquestes qüestions amb nombres reals de casos d'ús en un proveïdor, apuntant quins són els riscos que s'han de considerar abans de prendre les decisions o de signar un SLA.

Com es pot veure en les 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). L'equip de presa de decisions de l'empresa haurà d'analitzar acuradament les possibilitats i quins avantatges i riscos assumeix. Des d'un punt de vista global, és una tecnologia que aporta beneficis i que, amb una adequada planificació i presa de decisions, valorant tots els factors que influeixen en el negoci i escapant a conceptes superficials (tots ho tenen, tots ho utilitzen, baix cost, expansió il·limitada, etc.), pot ser una elecció adequada per als objectius de l'empresa, el seu negoci i la prestació de serveis en IT 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* on, de forma abstracta, podem identificar:

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

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

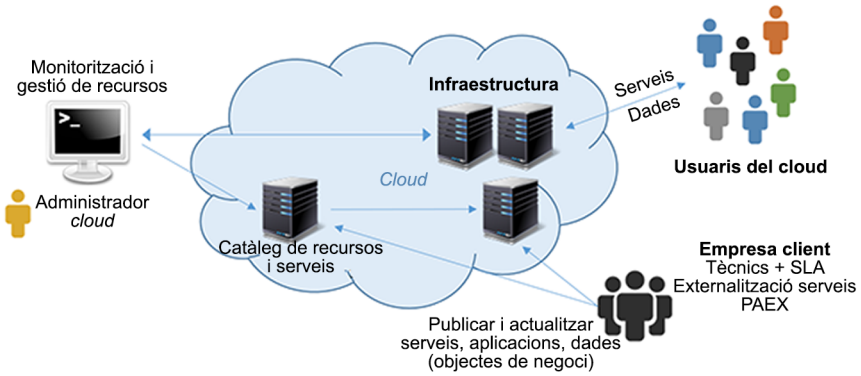
- **Administrador del *cloud***: cos tècnic de professionals que garantiran les prestacions, seguretat, estabilitat, disponibilitat, etc. dels serveis comercialitzats sobre la base d'un SLA signat amb cada client.

- **Empresa client**: usuaris que realitzen una externalització dels seus serveis d'IT i publiquen i actualitzen les seves aplicacions o dades en el *cloud*. Es basen en una garantia del servei que ofereix el proveïdor i formen part d'un acord de prestació de serveis (SLA). Generalment, el cost estarà basat en polítiques similars a les de «pagament per ús».

- **Usuaris**: clients de l'empresa que han posat els seus serveis en el *cloud* i que poden conèixer o no que els mateixos estan sent proveïts des del *cloud*. Solament accedeixen a un servei, aplicació o dades com si s'entrés en els servidors de l'empresa que proveeix el servei.

És important assenyalar 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 en el *cloud*.

Per exemple, si considerem els clients principals d'Amazon AWS [Acs16] (és a dir, aquelles empreses que tenen l'etiqueta «milions de dòlars» associada a alguna característica: ingressos, actius totals, finançament, valoració o beneficis), trobarem Adobe Systems, Airbnb, Alcatel-Lucent, Aon, Autodesk, BMW, Bristol-Myers Canon, Capital One, Comcast, Docker entre moltes altres. Si, per exemple, ens fixem en Adobe Systems, podrem veure el rol de cadascun d'aquests, en què el **proveïdor del *cloud*** i la infraestructura la proveeix **AWS**, l'**empresa** que ho contracta és **Adobe** i aquesta comercialitza els productes per mitjà de **Creative Cloud**, que permet que els **usuaris** (finals) tinguin accés a diferents programaris de disseny gràfic, edició de vídeo, disseny web i serveis en el *cloud* per una quota econòmica mensual.



3. Classificació

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

a) *Cloud* pública: en aquest cas, el sistema és obert per al seu ús general amb diferents models de negoci (des de 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 per mitjà d'internet. Generalment, un client no sap amb quins altres usuaris comparteix la infraestructura i la seva utilització es contracta mitjançant un catàleg de recursos disponibles. El seu aprovisionament és automàtic (o semiautomàtic) després d'haver complert amb una sèrie de tràmits quant al model de pagament i SLA. El propietari del *cloud* (proveïdor de recursos i serveis) pot ser una empresa privada, una institució acadèmica o una organització governamental, o una combinació d'aquestes, depenent del tipus de clients i 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, que en el *cloud* públic pot estar disponible en una xarxa oberta com ara internet i sense mecanismes de xifrat.

> Com a exemple d'aquests serveis podem esmentar AWS (Amazon Web Services), Microsoft Azure, Google Compute, Rackspace o IBM-SoftLayer (fins l'any 2014, IBMSmartCloud), que operen la seva pròpia infraestructura i els seus recursos són accessibles a internet, o el CSUC (Consorci de Serveis Universitaris de Catalunya), que ofereix, en diferents models d'explotació, recursos de *cloud* per a institucions acadèmiques i de recerca de Catalunya (http://www.csuc.cat/es/investigacion/infraestructura-en-la-nube).

És interessant conèixer que alguns proveïdors intenten reduir la probable inseguretat per mitjà de serveis de connexió directa (per exemple, AWS *Direct Connect* o Azure *ExpressRoute*), que permeten que els clients puguin comprar o llogar una connexió privada per a connectar-la a un punt privat del proveïdor, amb el consegüent increment de la privadesa que significa un recurs d'aquest tipus. Per als clients, tots els seus costos són operatius (OPEX). Una visió de conjunt de proveïdors i serveis es pot analitzar a [Cis16] (en què s'analitzen els punts forts i febles de cada operador), realitzat 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).

b) *Cloud* privat: en aquest cas, la infraestructura funciona exclusivament per a una organització o empresa i és utilitzada per les seves unitats de negoci o departaments. Aquesta pot ser la propietària, l'administradora i l'operadora, o alguns d'aquests apartats poden ser subcontractats a un tercer. És l'opció més favorable per a les companyies que necessiten una alta protecció per a les dades i un alt nivell de servei, ja que els recursos i la seva gestió estan sota el control i cura de la pròpia empresa o organització.

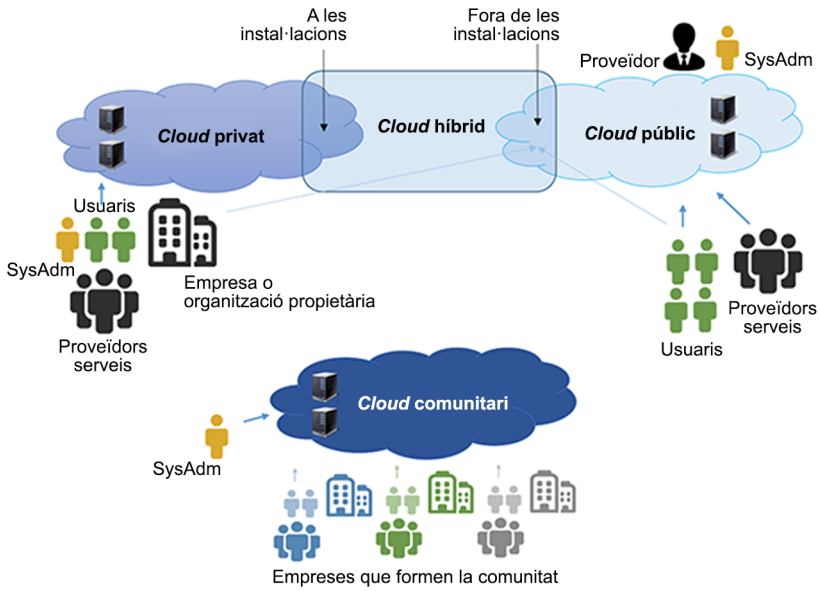
Si bé resol alguns problemes d'algun tipus d'empresa (per exemple, aquelles amb legislació especial com ara les empreses públiques), aquesta ha d'assumir el cost de la inversió (CAPEX), però com a contrapartida disposa d'una infraestructura sota demanda, gestionada per personal propi (o extern, però sota els seus criteris) i que controla què i on s'han d'executar les aplicacions i mantenint la privadesa de la seva informació, permetent definir les polítiques d'accés i evitant el *lock-in* de les dades, i també el *vendor lock-in* esmentat anteriorment.

c) *Cloud* híbrid: aquí es combinen models públics i privats. El propietari disposa d'una part privada (amb accés controlat i sota el seu control) i en comparteix d'altres, encara que d'una manera controlada. Els núvols híbrids ofereixen la possibilitat d'escalar molt ràpidament amb aprovisionament extern sota 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. Tot i que 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 d'això són algunes implementacions de correu empresarial o aplicacions d'ofimàtica.

d) *Cloud* comunitari: els serveis estan dissenyats perquè puguin ser utilitzats per una comunitat, organitzacions o empreses amb el mateix alineament d'objectius o negoci (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, i també els gestors o operadors. Alguns d'aquests rols poden ser subcontractats a tercers, però amb les indicacions i regles que s'apliquen a la comunitat.

La figura següent mostra un esquema d'aquestes quatre formes d'implantació i desenvolupament dels *clouds*.



Una altra de les classificacions habituals és pel **nivell de servei** que presten. Si bé els tradicionals són tres: **IaaS** (*infraestructure as a sevice*), **SaaS** (*software as a service*), **PaaS** (*platform as a service*), avui podem trobar altres extensions o derivades dels habituals com poden ser **BaaS** (*business as a service*), **StaaS** (*storage as a service*), **DaaS** (*desktop as a service*), **DRaaS** (*disaster recovery as a service*), **MaaS** (*marketing as a service*). Alguns autors li donen un nom global amb **XaaS** (*everything as a service*) per a referir-se a la creixent diversitat de serveis disponibles en el *cloud* per mitjà d'internet (consultar [Idd09]). Un cas que afirma aquestes consideracions és **aPaaS** (*application platform as a service*), que permet un ràpid desenvolupament i aprovisionament d'aplicacions (*apps*) com a plataforma específica per a la codificació, aprovisionament i desplegament d'aplicacions i que suporta el cicle de vida complet, proporcionant una forma més ràpida de crear aplicacions.

Exemple d'aPaaS

Exemples de la seva acceptació és l'àmplia proposta del mercat: AWS amb Elastic Beanstalk, CenturyLink amb AppFog, Google amb App Engine, IBM amb BlueMix, entre d'altres).

En la **infraestructura com a servei** (IaaS) hi ha la capa de recursos bàsica (generalment màquines virtuals, xarxes i emmagatzematge) on el client podrà posar els seus SO i les seves aplicacions, i implementar un servei o utilitzar-lo per a executar les seves aplicacions. El client no podrà manejar o controlar el maquinari subjacent, però sí, a partir de l'SO, l'emmagatzematge i les aplicacions o serveis implementats en aquestes màquines, i també alguns aspectes de la connectivitat (subxarxes, *firewalls*, dominis...).

> Un exemple de gran proveïdor IaaS (desenes a milers d'MV) podria ser Amazon EC2, Google Compute Engine o Digital Ocean com a servei representatiu per a pimes (en unitats d'MV).

La reflexió de l'usuari en aquesta modalitat seria «per què comprar, instal·lar i provar una infraestructura cada N anys (obsolescència) i no llogar-la i pagar per ús?». Els recursos són obtinguts sense fer obra civil (centre de dades), amb mínims recursos humans (no especialistes en IT), sense una gran inversió inicial, escalable i eficient, i a costos acceptables.

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

Per exemple, per a una tecnologia de desenvolupament en particular amb Linux i un servidor web més una base de dades i un ambient de programació com Perl o Ruby.

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

Exemple d'això podrien ser Cloud9, Google App Engine, Heroku o OpenShift.

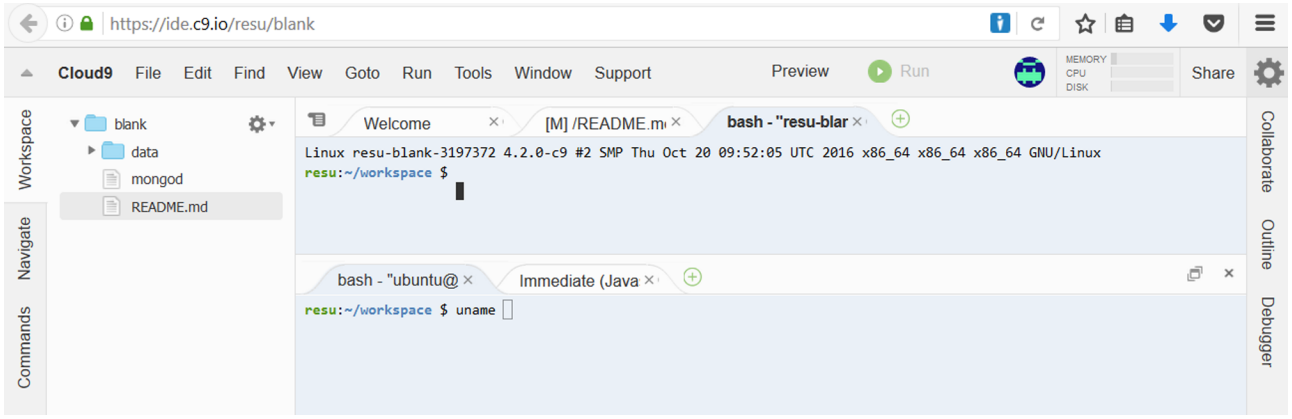
En resum, en aquesta modalitat el desig de l'usuari seria «Necessito aquest programari instal·lat en aquest operatiu i aquesta base de dades i amb aquesta API», i rebria tot el conjunt (hw, SO, base de dades, llibreries, API, seguretat, GUI i altres eines) llest per a usar en pocs segons i desenvolupar aplicacions empresarials o mòbils, pàgines web, continguts, etc.

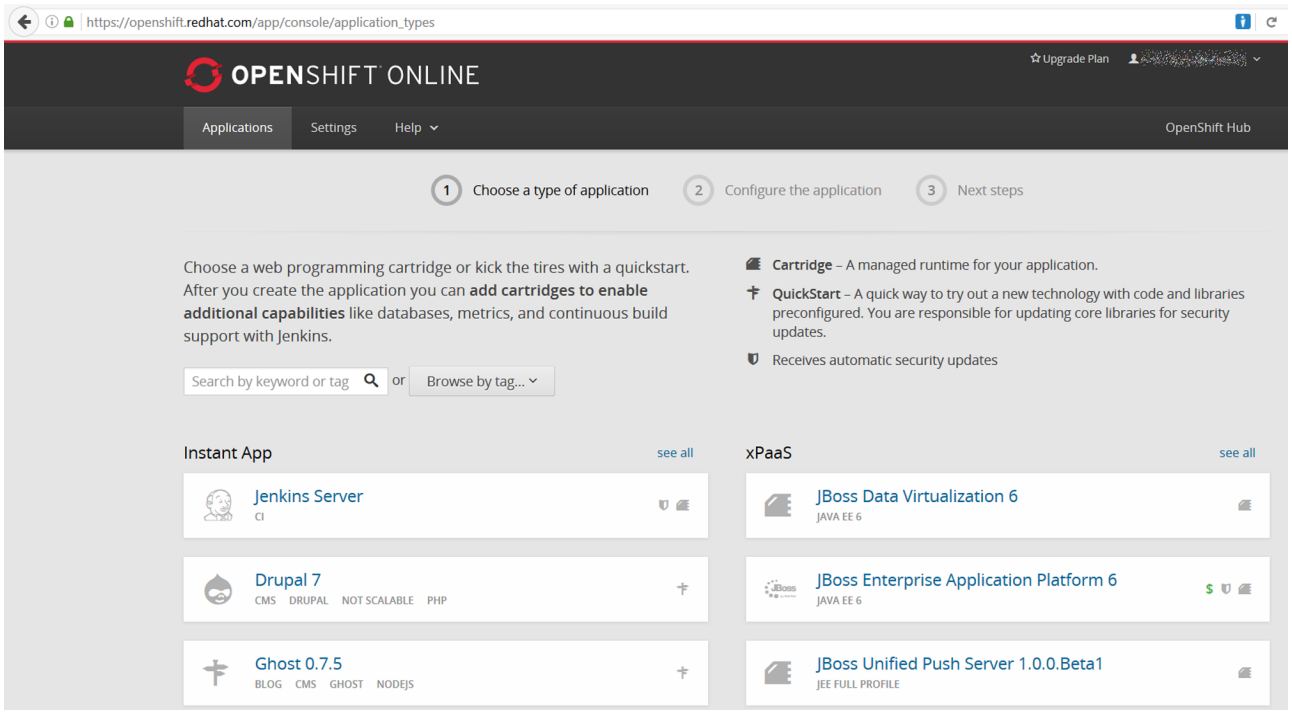
Finalment, el **programari com a servei** (SaaS) està en la capa abstracta (si bé hi ha una tendència força estesa a considerar que l'SaaS és la capa més simple del PaaS, o la més baixa) i caracteritza una aplicació completa oferta com a servei, sota demanda, generalment amb tinença múltiple (arquitectura de programari en què una sola instància de l'aplicació s'executa en el servidor, però servint a múltiples clients o organitzacions). En general, les aplicacions en aquest model de servei són accessibles per mitjà d'un navegador web i l'usuari no té control sobre aquestes, encara que en alguns casos se li permet realitzar algunes configuracions. Això fa que no calgui instal·lar l'aplicació en els ordinadors dels clients, eliminant el suport i manteniment del maquinari i programari, i millorant el control i gestió de les llicències, si calen.

Com a exemples representatius de SaaS es podria enumerar WebMail (Gmail, Outlook, Yahoo, etc.), SalesForce, Google Docs, Office 365 o SiteBuilder.

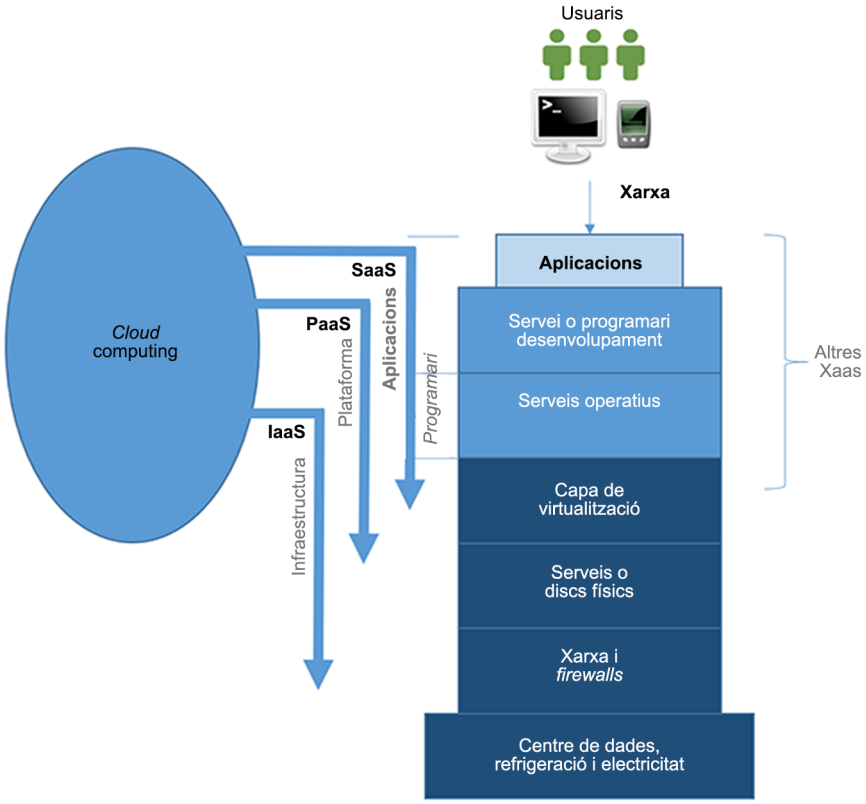
En aquesta modalitat es podria resumir que el pensament de l'usuari seria «Executi això per mi» sense responsabilitats en la gestió de maquinari o programari, utilitzant un navegador web o evitant la instal·lació de programari client, basat en un servei sota demanda, amb escalabilitat gairebé immediata,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

Les dues figures següents mostren les pantalles de Cloud9 i d'Openshift (on es poden obtenir comptes gratuïts per a proves de concepte amb diferents modalitats de funcionament i desenvolupament), com a exemple de la facilitat i simplicitat de posar en marxa un entorn complex només en uns segons i preparats per a desenvolupar aplicacions o serveis en aquests.





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



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

a) Clústers: 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. Són utilitzats com a entorns de processament d'altes prestacions (*high performance computing*) i serveis per a aplicacions crítiques o d'alt rendiment, entre d'altres usos. La seva utilització s'ha popularitzat per a aplicacions que necessiten un elevat temps de còmput (per exemple, científiques). Es basen en una infraestructura comercial (*blades* o *slices*) juntament amb un xarxa d'altes prestacions (per exemple, Infiniband). La major part de les vegades utilitzen programari *open source* (Linux, NFS –encara que cada vegada més aquest està sent reemplaçat per altres sistemes d'arxius distribuïts com ara Ceph, Lustre, GlusterFS– i sistemes de gestió de cues com, per exemple, SGE i Slurm i llibreries com, per exemple, OpenMPI per a executar tasques distribuïdes sobre l'arquitectura i OpenMP per a aprofitar la potencialitat dels sistemes *multicore*), cosa que permet desenvolupar i executar aplicacions concurrents o distribuïdes d'altes prestacions. Això permet disposar d'una infraestructura que proveeix un alt rendiment i una alta disponibilitat, amb un balanç de càrrega eficient i escalable, i amb uns costos raonables. Hi ha una classificació dels clústers

en funció de quina característica predomina sent: alt rendiment (HPC-*high performance computing*), alta disponibilitat (HA/HAC-*high availability computing*) i alta eficiència (HT/HTC-*high throughput computing*).

b) Superordinador: és una evolució funcional a gran escala d'un clúster (si bé hi ha diferents arquitectures) que posseeix una capacitat de còmput molt més elevada que aquests (i, per tant, que un ordinador de propòsit general). El seu rendiment es mesura en TeraFlops (10^{12} operacions de punt flotant per segon) i el més potent publicat en la llista del Top500 (els cinc-cents superordinadors més potents del món), el juny de l'any 2016, que és el Sunway TaihuLight amb 93.014 TeraFlops, que està compost de 10.649.600 *cores* de processament, disposa d'1,3 PetaBytes de memòria i consumeix 15.371 kW.

La seva història s'inicia a les acaballes dels anys cinquanta, sent IBM, Sperry Rand i Cray els grans actors d'aquella època amb arquitectures específiques (inicialment amb Univac com el primer ordinador comercial, i posteriorment IBM7030 o Cray-1), seguides per les arquitectures vectorials (dècada dels setanta) amb un mercat dominat per Control Data Corporation (CDC) i Cray Research. Amb l'auge dels microprocessadors (Intel), en la dècada dels vuitanta, comencen a emergir els multiprocessadors escalars amb compartició de memòria (SMP-*symmetric multiprocessor*) i després sense compartició de memòria (MPP-*massively parallel processor*), que, en els anys noranta i a les acaballes de la dècada, derivaran en clústers de milers de processadors i, de forma massiva, en superordinadors amb desenes de milers de processadors.

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

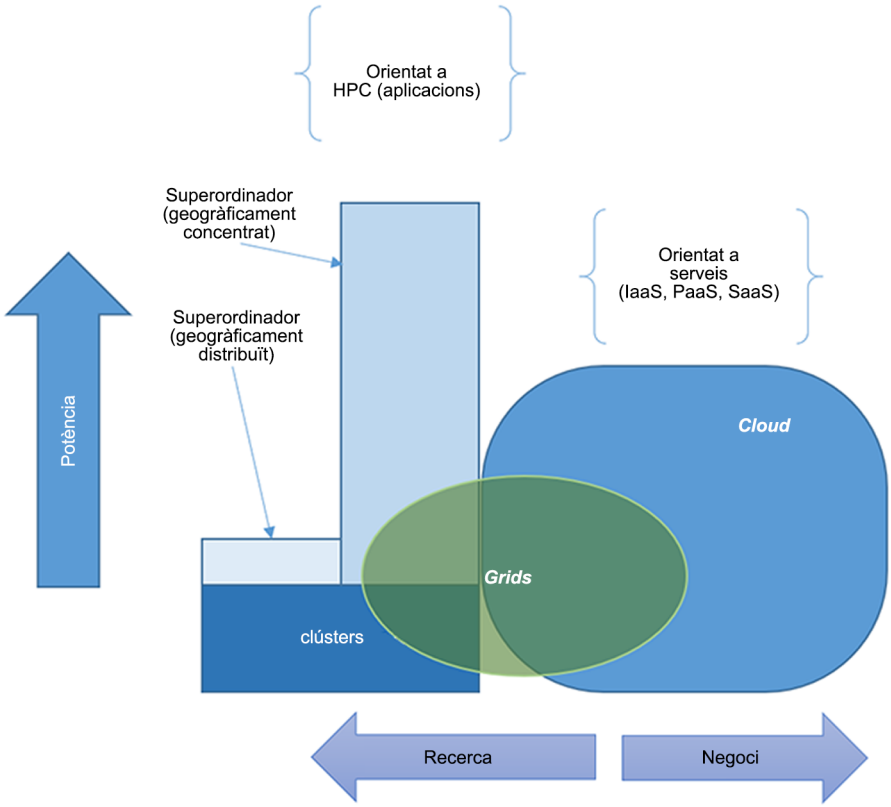
- l'evolució del clúster (com, per exemple, el MareNostrum a Catalunya, el Finisterrae a Galícia o el Juqueen a Alemanya), que s'utilitza de forma dedicada per al còmput d'altes prestacions, comparteix un espai físic i disposa d'una sèrie de recursos comuns com ara xarxes de comunicació d'alta velocitat o espai d'emmagatzematge compartit i distribuït, o

- superordinador distribuït per mitjà d'una xarxa formada per centenars o milers d'ordinadors discrets distribuïts mitjançant una xarxa (internet) que es dediquen de forma no exclusiva a la solució d'un problema comú (generalment cada equip rep i processa un conjunt de tasques petites traslladant els resultats a un servidor central que integra els resultats en la solució global).

  Exemples representatius d'aquests últims són projectes com Seti@home, Folding@home, World Community Grid o Climateprediction.net, molts dels quals estan basats en Boinc (una arquitectura programari *open source* que permet que voluntaris aportin els seus recursos a un problema comú –paradigma conegut com *volunteer computing*). A la pàgina dels seus desenvolupadors, es pot observar que la potència de còmput generada per aquesta arquitectura en tots els projectes que la utilitzen és de 12,2 PetaFlops, aconseguida mitjançant 278.493 voluntaris i 944.194 ordinadors (octubre de 2016).

c) ***Grid***: es refereix a una infraestructura que permet 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*) comuna que permet utilitzar-los com si fos 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 a formar *grids* i així cedir els seus recursos temporalment, però també per a disposar del còmput equivalent a la unió de totes aquestes infraestructures. Les idees de *grid* van ser establertes per Ian Foster, Carl Kesselman, que van ser els precursors en la creació de Globus Toolkit considerat com la primera eina per a construir *grids*. Entre els grans projectes *grid* podem trobar CrossGrid o EU-DataGrid o, més recentment, el projecte EGI-InSPIRE, tots ells finançats per la Comunitat Europea. És important destacar que el *grid* és un tipus d'infraestructura de còmput l'àmbit d'utilització i propòsit natural del qual és el científic (*e-science*), com ara universitats i laboratoris de recerca, mentre que el *cloud* neix des de la prestació de serveis i els negocis a les empreses i usuaris (*e-business*) des d'altres empreses, però tots dos són similars en paradigmes i estructures per a manejar grans quantitats de recursos distribuïts, sent considerats per alguns experts com les dues vies de l'ús massiu de recursos (en diferents models): un per a l'e-ciència exclusivament (*grid*) i un altre per a les empreses (*cloud*). Cal assenyalar que es comencen a oferir serveis creuats, és a dir, altes prestacions (*high performance computing*) en el *cloud* (per exemple, instàncies amb base en maquinari de GPU, SSD i acceleradors com Xeon Phi).

Amb aquesta caracterització dels recursos i la seva utilització podem situar gràficament cadascuna d'aquestes arquitectures en funció de la potència de còmput involucrada i el seu tipus de dedicació (vegeu figura següent).



Des del punt de vista dels **paradigmes precursors** que són essencials en el *cloud computing*, podem explicar:

a) Client-servidor: el model client-servidor es refereix de forma general als serveis implementats en un sistema de còmput distribuït (en els seus orígens implementats per Sockets i RPC. Vegeu exemples de programes en l'assignatura Administració avançada). Aquest tipus de paradigma serà utilitzat en totes les aplicacions basades en serveis, i també en una gran part del codi que s'executi dins del *cloud* com a aplicacions d'alt rendiment.

b) WS & SOA: no són conceptes nous (definits en la dècada dels noranta). SOA (*service-oriented architecture*) és definit com una arquitectura per a dissenyar i desenvolupar sistemes distribuïts i que s'utilitza per al descobriment dinàmic i l'ús de serveis en una xarxa i WS (*web services*) com a serveis prestats per mitjà 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* són prestats generalment per mitjà d'aquestes tecnologies, les quals també poden ser utilitzades internament per a la seva organització i gestió interna.

c) 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 en la WWW. Evidentment, aquesta tecnologia aporta molt al *cloud computing* ja que la interacció i prestació de serveis amb l'usuari i els clients (i entre aplicacions) serà per mitjà de les possibilitats que brinda mitjan-

çant estils (CSS), llenguatges (HTML5, XML, XHTML, JavaScript, RoR, etc.), aplicacions RIA (*Rich Internet Applications*, Ajax), agregació (RSS, ATOM), *JavaScript Client Communication*, interoperabilitat (*Representational State Transfer*, REST), APIS (WebGL, DOM, etc.), formats de dades (JSON, XML) o *Mashup* (aplicació web híbrida).

Dins d'aquesta comparació, es pot argumentar que la tecnologia principal que suporta el *cloud computing* és la **virtualització**. La capa de virtualització (hipervisor, per exemple, Xen, VirtualBox, OracleVM, KVM, VMware ESX/ESXi, Hyper-V, entre d'altres) separa un dispositiu físic en un o més dispositius «virtuals» (anomenats màquines virtuals –MV–), en què cadascun d'aquests poden ser assignats, utilitzats i gestionats pel client de forma molt simple, com si es tractés de màquines físiques, però amb els consegüents avantatges (aïllament, fàcil manteniment, posada en marxa, etc.). Avui dia tots els sistemes utilitzen tècniques de virtualització assistides per maquinari (extensions del processador VT-x o AMD-V), de manera que és possible tenir màquines virtualitzades molt eficients, disponibles i configurades (mitjançant tècniques de clonatge o *pool of VM in stand-by*) per a ser assignades a un usuari sota demanda i en pocs segons.

Un pas addicional que ha permès un increment notable de l'eficiència, disponibilitat i una revolució en els entorns de desenvolupament ha estat la **virtualització en el sistema operatiu** per a la creació d'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 albergar múltiples instàncies d'usuari aïllades, en què cadascuna d'aquestes actua com un «contenidor» amb les seves pròpies llibreries i serveis, però utilitzant l'SO subjacent. Els noms que reben aquestes instàncies són el de *containers*, *virtualization engines* (VE) o *jails* (per exemple, *FreeBSD jail* o *chroot jail*). A ulls d'un usuari, aquest «contenidor» és similar a un servidor real i tindrà tots els elements necessaris, com si es tractés del servidor real/virtualitzat, però amb solament una petita despesa de memòria, CPU i disc. En sistemes Linux (o *nix), és una evolució del mecanisme de *chroot* més un sistema de control o limitació de la utilització dels recursos d'un contenidor respecte d'un altre. Un dels desavantatges dels contenidors és que, com que comparteixen l'SO, no es poden tenir contenidors amb sistemes operatius que no comparteixin el mateix nucli (per exemple, si el *host* és Linux no podrem posar un contenidor amb Windows, cosa que sí que és possible si fos una màquina virtual). Entre el programari de virtualització de l'SO *open source* podem trobar Docker, LXC/LXD, OpenVZ, FreeBSD Jails, o 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 ara:

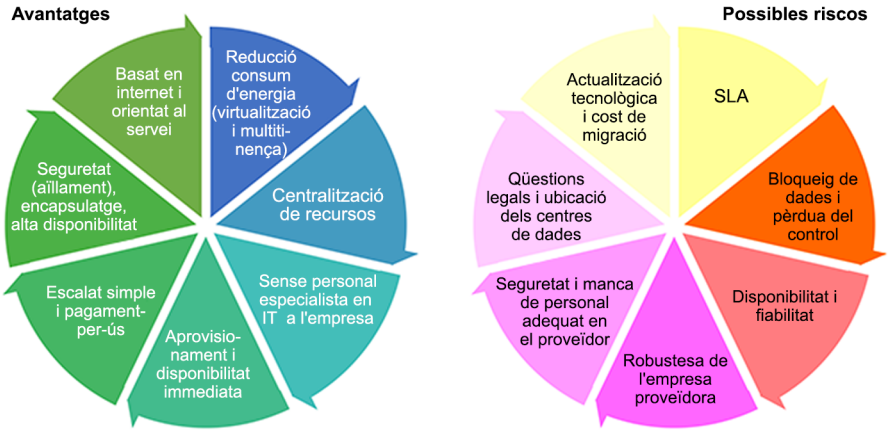
d) ***Dew computing***: nou paradigma de còmput distribuït que considera que els equips locals (ordinadors de sobretaula, portàtils i dispositius mòbils) proporcionen un entorn propici per als microserveis independents dels serveis *cloud* i aquests poden col·laborar amb ells. És a dir, aquest paradigma planteja la distribució de les càrregues de treball entre els servidors del *cloud* i els dispositius discrets o locals per a utilitzar plenament el potencial de tots dos. Alguns autors parlen més de *mobile cloud computing* [Mcc13], en què s'integren i distribueixen els serveis i l'emmagatzematge de les dades en els dispositius mòbils i els proveïdors de serveis *cloud* (algunes experiències apropen més aquest paradigma al *volunteer computing*).

e) ***Edge computing***: és un paradigma de computació distribuïda que proporciona serveis de dades, computació, emmagatzematge i aplicacions més a prop dels dispositius client o en els dispositius propers a l'usuari. És a dir, processa i emmagatzema les dades en els dispositius que estan a prop de la xarxa (per exemple, dispositius mòbils) en lloc d'enviar les dades a un lloc en el núvol per a això. És considerat una forma de computació distribuïda de proximitat, en què cadascun dels dispositius connectats a la xarxa pot processar les dades i solament transmetre un conjunt d'aquestes que siguin d'interès o fer-ho solament en situacions excepcionals (alarmes o notificacions) tenint una capacitat autònoma i sense dependències del servidor en el núvol. És un model que, amb el creixement que s'espera de la IoT (*internet of things*), permetrà que els sistemes *cloud* i les xarxes no es col·lapsin davant l'increment exponencial de dades a transmetre, emmagatzemar i processar. Alguns autors el diferencien del ***fog computing*** en què s'utilitza agrupacions o multituds d'usuaris finals (o dispositius propers a l'usuari) per a l'emmagatzematge de les dades (en lloc de fer-ho en el *cloud*), la comunicació (en lloc de fer-ho per mitjà d'internet), el control, la configuració, el mesurament i la gestió (en lloc de ser controlats principalment per *gateways* de xarxa com ocorre a la xarxa LTE, per exemple). Aquest paradigma ha generat un moviment important de les grans empreses de tecnologia (OpenFog).

f) ***Peer-to-peer computing***: aquest paradigma planteja l'ús d'una arquitectura distribuïda sense la necessitat d'una coordinació central per a realitzar 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 a processar, emmagatzemar i comunicar les dades d'una forma igual. Els *peer* posen una part dels seus recursos (potència de processament, emmagatzematge o amplada de banda de la xarxa) directament a la disposició d'altres *peers* de la xarxa, sense la necessitat d'una coordinació central pels servidors. Aquest tipus de paradigma ha tingut molta acceptació en serveis de gestió de continguts (Bittorrent, Spotify, Skype), compartició d'arxius (Gnutella, Edonkey), monedes digitals (Bitcoin, Peercoin), anonimització (I2P), entre d'altres.

g) ***Volunteer computing***: com hem esmentat anteriorment en l'apartat de Superordinadors, aquest tipus de processament distribuït es basa en el fet que els usuaris aporten els seus recursos (bàsicament processament i emmagatzematge) per a un determinat projecte del qual formen part. El primer registre que es té d'aquest tipus de còmput distribuït és de l'any 1996, amb Great Internet Mersenne Prime Search (cerca de nombres primers que compleixen amb 2^n – 1, per exemple, 3, 7, 31...) seguit per distributed.net (el 1997) i, a continuació, molts d'altres, entre els quals Bayanihan, els desenvolupadors del qual van proposar el nom de *volunteer computing*. En l'actualitat, molts d'aquests estan basats en l'arquitectura Boinc (desenvolupada per la Universitat de Califòrnia, Berkeley) amb projectes com els que s'han esmentat anteriorment (Seti@home) o aquells que han desenvolupat la seva pròpia arquitectura programari com Folding@home (Universitat d'Stanford) que el maig de 2016 van anunciar que havia arribat a la marca de 100 PetaFlops (x86).

En les figures següents es mostren, en forma resumida, tant les característiques a favor del *cloud computing* com aquelles que no són tan favorables i que poden representar un risc i que s'hauran d'analitzar amb cura.



4. Plataformes més representatives

Més enllà de l'extens catàleg de les diferents opcions de plataformes i opcions propietàries, hi ha un gran conjunt de plataformes basades en programari lliure que permeten desenvolupar i posar en marxa infraestructures 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 d'aquestes permeten vincular-se als operadors *cloud* públics per a estendre la potencialitat dels privats. Algunes d'aquestes plataformes compten amb un gran nombre d'instal·lacions operatives que, depenent de la grandària de l'empresa, les necessitats i el control de la informació que es desitgi tenir, és totalment viable i moltes d'aquestes són utilitzades per, inclús, els 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 *open source* a les empreses i 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 *open source*. Generalment, en les versions *open source* es detecta un factor d'innovació i creixement constant que presenta noves adaptacions, serveis i millores en un cicle d'evolució més ràpid i constant que els seus corresponents propietaris, que porten a la generació de nous estàndards, noves API públiques afavorint la creació de coneixement i possibilitats adaptades a totes les necessitats [Osc14].

No obstant això, aquesta proliferació de plataformes i serveis pot desconcertar i generar certa confusió i complicar l'elecció, que s'ha de prendre des d'un punt de vista pragmàtic i «mirar» quina d'aquestes s'apropa més a les necessitats i s'adequa a la infraestructura maquinari que es disposa. És important fer proves d'implantació i desenvolupar sistemes de preproducció per a veure com es comporta la plataforma escollida i si és adequada per a les necessitats reals de l'organització.

D'altra banda, el mercat ha evolucionat en tots els aspectes i avui és possible trobar una gran quantitat de proveïdors (alguns d'aquests ja s'han esmentat), com es pot observar en les recomanacions i la llista *The Best Cloud Computing Companies And CEOs To Work For In 2016* de Forbes, la comparació de *Cloud Computing Providers* de SoftwareInsider o la de *The 100 Coolest Cloud Computing Vendors Of 2016*.

4.1. Hipervisors

L'hipervisor (o monitor de màquina virtual, VMM, *virtual machine monitor*) és la primera capa que s'ha de posar o en el sistema operatiu *host* o integrat en aquest i funcionant com un conjunt indivisible. Aquesta capa, que en el sentit més ampli pot ser programari, *firmware* o maquinari, és el responsable de crear i executar les 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 maquinari base presentant a cadascun d'aquests una plataforma virtual.

Els operatius a cada màquina virtual s'executen com si estiguessin sobre un maquinari determinat en un entorn tancat i solament amb alguns recursos compartits (aquells que l'hipervisor habiliti) 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 a l'espai d'usuari, com a diferents distribucions de Linux, però amb el mateix *kernel*.

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 maquinari per a donar suport a la virtualització (virtualització assistida per maquinari), millorant les seves prestacions i permetent una evolució notable dels hipervisors en processadors Intel amb les extensions Intel VT-x (Vanderpool) i d'AMD amb AMD-V (Pacifica) (cal destacar que, a les acaballes de 2015, Intel representa el 87,7% del mercat de microprocessadors, i AMD el 12,1%).

Una altra tècnica alternativa per a la virtualització és l'anomenada **paravirtualització**, la qual presenta una interfície programari de les màquines virtuals similar (però no idèntica) al maquinari subjacent. La paravirtualització proporciona *hooks* especialment definits per a 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 la màquina virtual (VMM) sigui més simple i pugui resituar l'execució de les tasques crítiques des del domini virtual al domini del *host* i, així, reduir la degradació del rendiment de l'SO *guest*. L'únic problema d'aquesta opció és que l'SO *guest* haurà de ser estès amb aquesta para-API perquè disposi de les extensions necessàries per a comunicar-se amb l'hipervisor. Xen, la plataforma més coneguda i utilitzada pels grans proveïdors, implementa aquesta tècnica i suporta l'execució amb dos tipus de *guest*

diferents: sistemes paravirtualitzats i virtualització completa amb assistència de maquinari. Tots dos tipus poden ser utilitzats simultàniament en un mateix sistema Xen i també es poden usar tècniques de paravirtualització en un *guest* amb assistència maquinari.

Els diferents tipus de plataformes de virtualització utilitzades en l'actualitat es poden classificar en dos tipus: aquelles que l'hipervisor gestiona el maquinari directament, és a dir, l'hipervisor i l'SO *host* formen un conjunt indivisible (*bare-metal hypervisor* o *type 1*) o aquelles en què l'hipervisor s'instal·la en un SO *host* i s'executa com una aplicació més d'aquest (*hosted hypervisor* o *type 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, FreBSD+bhyve	

Com es pot apreciar, hi ha algunes opcions, com Linux's *Kernel-based Virtual Machine* (KVM) i FreeBSD's Bhyve, que són mòduls del *kernel*. La seva instal·lació sobre Linux o sobre BSD converteixen l'SO en hipervisor, per tant, de la primera categoria, però alguns autors, com que són mòduls, ho classifiquen com de la segona.

A continuació, farem algunes consideracions dels més representatius:

1) Xen Project Code: aquest hipervisor és utilitzat pels grans proveïdors, com ara Amazon Web Services (des de 2006), Rackspace Hosting, Verizon Cloud, entre d'altres, i pot ser instal·lat des de les distribucions de Linux que contenen els paquets o des d'una imatge ISO per a instal·lar-lo directament (fins i tot, algunes distribucions inclouen extensions LiveCD per a 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, la seva compra per Citrix i la seva tornada a la Linux Foundation com a projecte col·laboratiu, l'empresa XenSource ha passat per diferents fases (avui en https://www.xenproject.org). En relació 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 Eucalyptus, Apache CloudStack, OpenNebula i OpenStack, i apareixen els primers *clouds* públics amb XAPI +

Openstack. El 2012, els paquets XAPI s'inclouen a Debian i Ubuntu Server permetent fer els primers *clouds* utilitzant distribucions Linux. El 2013, XenServer (un *superset* d'XCP) és alliberat com a *open source* per Citrix i disponible en l'actualitat a Xenserver.org (fent obsolet el projecte XCP). Avui dia, el projecte Xen està centrat a l'àrea dels sistemes operatius de *cloud*. Aquests sistemes operatius lleugers i especials (també coneguts com a *unikernels*) no estan dissenyats per funcionar sobre maquinari sinó per a produir petites màquines virtuals que poden generar *clouds* massius amb un maquinari mínim. El projecte Mirage OS és un dels primers sistemes operatius de *cloud* en producció, a més de LING (Erlang-on-Xen) i OSv, també coneguts com a Xen Project-powered.

2) KVM: és l'acrònim de *Kernel-based Virtual Machine* i és una opció de virtualització total (*full virtualization*) per a Linux en arquitectures x86 que disposin d'extensions Intel VT o AMD-V. La seva instal·lació és per mitjà de mòduls que s'insereixen en el nucli de l'SO *host* i que proveeixen l'entorn de virtualització (*kvm-intel.ko* o *kvm-amd.ko*). Amb aquest es poden executar múltiples màquines virtuals (Linux o Windows) sense cap mena de modificació d'aquestes i en què cadascuna podrà accedir al maquinari virtualitzat (xarxa, disc, targeta gràfica, etc.). Hi ha una certa discussió sobre la vinculació de KVM i QEMU (*Quick Emulator*) i els rols que juguen cadascun d'aquests. QEMU és un paquet *open source* que permet l'emulació i la virtualització d'un determinat maquinari. Amb l'emulació permet que un programa o SO per a una arquitectura ARM, per exemple, es pugui executar en una altra, per exemple, x86, mentre que amb la virtualització permet l'execució del codi natiu directament en la CPU *host*. QEMU suporta la virtualització quan s'executa en un hipervisor Xen o quan utilitza el mòdul de KVM en el *kernel* (utilitzant-los com a acceleradors i no fent servir l'emulació).

La raó d'això s'explica en com es virtualitza la CPU: amb les extensions maquinari dels processadors es crea una *physicalCPU slice* que pot ser mapejada directament a la CPU virtual permetent una considerable millora. Això és el que ocorre, per exemple, amb el mòdul de KVM i és utilitzat per QEMU quan s'escull que el tipus de virtualització és KVM. Si no s'utilitza aquesta «acceleració» (per exemple, perquè el processador no disposa de les extensions maquinari), QEMU utilitza TCG (*Tiny Code Generator*) per a traslladar i executar les instruccions de la CPU virtual a la CPU física (emulació), amb la conseqüent reducció de prestacions. Pel que fa a KVM, utilitza QEMU com a programa quan es crea una instància des de la màquina virtual, i una vegada en execució, aquesta serà com un procés regular de l'SO al qual se li pot aplicar el conjunt d'ordres habituals com `top`, `kill`, `taskset` i altres eines habituals per a gestionar les màquines virtuals.

3) VirtualBox: és un hipervisor *open source*, de tipus 2, molt flexible per a arquitectures x86 i AMD64/Intel64, tant per a ús empresarial com domèstic, i s'instal·la sobre Linux, Windows, Macintosh i Solaris. Suporta un gran nombre d'SO *guest* (Windows, DOS, Linux, OpenSolaris, OS/2, OpenBSD, Android, ChromiunOS i d'altres. VirtualBox pot ser complementat amb Hyper-

Box (permet gestionar diferents hipervisors i representa una alternativa lliure per als productes comercials com VMware vCenter/ESXi i Citrix XenCenter) o phpVIrtualBox per a gestionar les instàncies de VirtualBox remotament. És una opció molt acceptable per a proves o petites/mitjanes instal·lacions i permet que les màquines virtuals puguin ser convertides a altres formats (per exemple, vmdk de VmWare) i ser reutilitzades en altres infraestructures. L'opció de VirtualBox està tan estesa que hi ha diverses pàgines mantingudes per la comunitat que permeten descarregar les imatges de diferents SO o distribucions ja instal·lades i llestes per a ser carregades i executades, entre les quals, es pot esmentar OSBoxes, Virtualboxes, 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 a executar múltiples SO com a màquines virtuals en un PC. Aquestes poden ser útils en diferents àrees com a desenvolupament d'aplicacions o replicació d'entorns, o també per a l'homogeneïtzació de recursos o accés controlat a aquests per a tenir un entorn homogeni corporatiu i que pot ser controlat i gestionat des del departament d'IT de l'empresa amb eines com VMware VSphere/Horizon FLEX. VMware vSphere Hypervisor (ESXi) és un hipervisor *bare-metal* gratuït que permet virtualitzar els servidors i consolidar aplicacions amb tots els avantatges de la virtualització (reducció de consum o espai, inversió de maquinari, optimització, etc.). ESXi disposa d'una consola per a la seva gestió i control, però no disposa d'eines per a la gestió i el control de l'entorn virtual. Per a aquests s'ha d'accedir per mitjà de vSphere Client (VMware Infrastructure Client), que es descarrega des de la pàgina de VMware (en les últimes versions, disposa d'una servei web que reemplaça el client i que es veurà en el mòdul següent). Per a la gestió i el control de diferents servidors ESXi, la gestió de calent en calent de les màquines virtuals, l'emmagatzematge i altres opcions de monitoratge i control s'ha de recórrer a les eines propietàries de VMware que estan disponibles sota llicència.

5) Microsoft Hiper-V: és un hipervisor per a sistemes de 64 bits amb processadors que disposen d'extensions maquinari: VT-x i AMD-V (no obstant això, els programes de gestió es poden instal·lar en sistemes x86). Des de la versió inclosa en Windows Server 2008 R2, l'hipervisor incorpora funcionalitats esteses com ara la migració de calent en calent de les màquines virtuals (*live migration*), l'emmagatzematge en màquines virtuals dinàmiques, la compatibilitat millorada amb processadors i xarxes, el suport per Linux (Debian, Ubuntu, Centos/RH, Oracle, SuSE), FreeBSD i, òbviament, Windows (des de Vista fins a W10 incloent *servers* des de 2008). Hi ha algunes diferències en l'Hyper-V executant-se, per exemple, en Windows 10 i en Windows Server 2012, que se centren en la gestió de la memòria, la virtualització de la GPU, la migració de calent en calent, les rèpliques o la compartició de VHDX (discos virtuals), però per la resta són iguals (en Windows 10, aquestes extensions reemplacen un producte anterior de virtualització anomenat Virtual PC). La seva instal·lació

és simple tant en W10 com en W2012 i permet que les màquines creades puguin ser exportades a altres entorns Hyper-V o directament a Azure (plataforma de *cloud* públic de Microsoft).

Microsoft també disposa d'una versió anomenada Hyper-V Server gratuïta que incorpora un Windows Server amb funcionalitat limitada, a més dels components d'Hyper-V (solament aquest rol està actiu). L'Hyper-V Server solament disposa d'interacció per mitjà de la línia d'ordres per a configurar el *host*, el maquinari i el programari, i suporta l'accés remot per mitjà de Remote Desktop Connection. La configuració del *host* i les màquines virtuals pot ser realitzada per mitjà de la xarxa, utilitzant programes com ara *System Center Virtual Machine Manager*, que permeten configurar i monitorizar el servidor de forma fàcil. L'Hyper-V Server requereix menys requeriments de memòria i disc que un Windows Server, però està limitat a 64 màquines virtuals quan en un Windows Server és de 1.024.

6) Proxmox: amb les facilitats de KVM i altres tecnologies, han sortit diferents solucions com el Proxmox VE, que permet tenir un entorn virtualitzat amb KVM i Linux Containers (LXC). Aquesta solució completa de virtualització *open source* és adequada per a moltes empreses, ja que permet gestionar i configurar màquines virtuals sobre la base de KVM, contenidors Linux (LXC), emmagatzematge i xarxes virtualitzades i clústers d'alta disponibilitat. La seva administració i monitoratge es realitza per mitjà d'una interfície web i suporta tant SO *guest* com Linux i Windows en màquines virtuals o contenidors com a virtualització del sistema Linux (a nivell de l'SO). Aquesta distribució és un hipervisor tipus 1 (*bare-metal*) i està basat en la combinació de Debian+KVM +LXC, suporta la migració de calent 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, etc.).

La taula següent mostra la distribució d'hipervisors (per defecte, ja que es poden negociar altres serveis com, per exemple, el *bare-metal servers ad-hoc*) sobre les deu empreses més significatives de l'informe de Gartner de 2016 [Cis16].

Proveïdor	Hipervisor
AWS	Xen (EC2 Container Service: Docker)
Century Link	VMware
Fujitsu-Global Fujitsu-K5	Xen KVM (Openstack)
Google Compute	KVM
IBM-Softlayer	Citrix-XenServer
Microsoft Azure	Hyper-V
NTT Com NTT Com Enterprise	KVM (CloudStack) VMware

Proveïdor	Hipervisor
Rackspace	Citrix-XenServer (default), VMware, KVM(Openstack), Hyper-V
Virtustream	VMware i KVM
VMware	VMware

Un punt important dels hipervisors, com ja hem esmentat anteriorment, són l'evolució d'aquests cap a la virtualització del sistema operatiu i els *containers*. Des de fa un parell d'anys, hi ha una disputa sobre els hipervisors i contenidors en relació amb les prestacions i altres paràmetres d'eficiència, sobretot en el *cloud*. [Cvh14]

A més, en aquest sentit, Ubuntu ha fet una aposta per un contenidor (LXD) aplicant la idea del «contenidor» en el *cloud* i promocionant aquest com *The LXD container hypervisor*, amb 10x de densitat d'ESX, un 25% més ràpid i 0 latència, i indicant que es poden moure les MV als contenidors, fàcilment i sense modificar les aplicacions o operacions i on LXD és un hipervisor de contenidor pur que executa sistemes operatius i aplicacions Linux no modificats amb operacions d'estil VM a una velocitat i densitat increïbles.

Un exemple d'aquesta tendència (però no és 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 a provar, però, segons els experts, en la seva gran majoria tots tenen el mateix codi en la part inferior (des de Google/LXD/Docker amb *cgroups/namespaces* o *bean-counters* en OpenVZ) i, a més, han anat coincidint i, en l'actualitat, no hi ha pràcticament diferències entre aquests.

Dos dels entorns de contenidors més representatius són Docker, LXC/LXD. Compten amb ecosistemes que permeten obtenir els *containers* com VA però sense la càrrega que representa l'MV i tot el sistema operatiu *guest* i les seves dependències. Segons Ubuntu, LXD i Docker són complementaris ja que Docker està centrat en les aplicacions i LXD en el *host machine containers* (aquest actua com una màquina virtual) permetent executar i gestionar els contenidors Docker dins dels contenidors LXD, fent que Docker encara sigui millor, ja que LXD (en relació amb KVM obté ×10 en prestacions, ×14,5 en relació amb la densitat, 57% menys de latència i 94% més ràpid en la càrrega). És habitual que els desenvolupadors d'aplicacions proveeixin el contenidor per a la seva aplicació i les instruccions per al seu funcionament.

D'aquesta forma, amb Docker a la part superior d'LXC/LXD i atesos els avantatges del seu *open source*, 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 a distribuir el seu codi, el qual podrà ser executat per l'usuari sense

pràcticament cap intervenció o configuració, i podrà ser desplegat ràpidament en el *cloud* amb una portabilitat total i sense invertir el temps que costa tenir una VM correctament configurada i enviar-la o desplegar-la. Segons els experts, aquesta nova tendència inicia un nou paradigma de virtualització (que alguns, com Ubuntu, denominen *container hypervisors*) que aporta una nova forma d'empaquetar les aplicacions tenint serveis en el *cloud* que poden entrar en qualsevol dispositiu, permetent en un futur proper 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 al voltant dels contenidors. Per exemple, Docker compta amb un ecosistema oficial, DockerHub, que actua com a repositori de les distribucions oficials o contenidors de la pròpia comunitat que poden ser descarregats i posats en marxa en uns minuts. El mateix podem trobar per LCC/LXD o també contenidors creats per particulars i llocs a la disposició de la comunitat, per exemple, Flockport, on es demostra que LXD també es pot centrar en aplicacions i no és un terreny totalment propi de Docker (que, no obstant això, continua tenint un paper protagonista).

4.2. Imatges virtuals (*virtual software appliances/cloud appliances*)

Una vegada instal·lat i configurat l'hipervisor, cal començar a desplegar (instal·lar, configurar i posar en marxa) les màquines virtuals (MV) que cobreixin les necessitats dels usuaris.

El primer pas és escollir la distribució del sistema operatiu (per exemple, des de Distrowatch) que millor s'adeqüi i en aquest instal·lar-hi i configurar-hi tots els serveis necessaris. Aquesta tasca implica una gran quantitat de temps i coneixements, ja que algunes aplicacions, servidors o entorns requereixen una gran quantitat de passos previs de prerequisits (SO, llibreries, aplicacions addicionals, etc.) que s'han de complir perquè després el programari desitjat funcioni correctament.

Considerem el cas de voler provar una versió de Liferay per a avaluar si s'adapta a les necessitats del client en la versió CE basada en el servidor d'aplicacions Tomcat. Liferay és un gestor d'alt acompliment de portals corporatius que permet la gestió de continguts i documents, la col·laboració entre els usuaris i la vinculació a les xarxes socials amb incorporació d'aplicacions i totalment adaptable i configurable a les necessitats dels clients.

Entre els passos de preinstal·lació, cal el *framework* de Java, un servidor de bases de dades, un servidor d'aplicacions (Tomcat o WildFly), a més d'un servidor de correu i la configuració d'ElasticSearch que s'haurà d'instal·lar per separat. A més, Tomcat necessita Ant i CVS i la seva instal·lació també consumeix temps.

Després d'instal·lar aquests paquets es pot seguir el procés d'instal·lació de Liferay i la seva configuració entre els diferents paquets i proves per a verificar que tot funciona. Òbviament, més enllà del temps consumit, no és una tasca

fàcil per a un no expert en Liferay-Tomcat-DB-ElasticSearch, que comportarà un cert temps i dedicació tenir-ho tot a punt per a fer una prova, inclús per a un expert en IT (que no tingui experiència específica en aquests paquets).

Una opció que ha cobrat força per a apropar el programari a empreses o organitzacions que no desitgin invertir tot aquest temps a fer una prova és la d'oferir als usuaris un paquet preconfigurat del programari (normalment anomenat *bundle*) amb totes les aplicacions i que solament s'ha de reconnectar i adequar, per exemple, en el cas de Liferay, a la base de dades, al servidor d'aplicacions i al servidor de cerques (vegeu *Deploying Liferay Portal*).

Més enllà dels avantatges de l'encapsulació de paquets (*bundles*) i amb les facilitats proveïdes per la virtualització i el *cloud*, ha irromput amb força una nova tendència, que és la de proveir imatges amb el programari configurat i llest per a usar. Aquestes imatges, anomenades ***appliances***, permeten de forma fàcil i simple disposar d'un entorn tancat (*sandbox*) perquè ràpidament (en qüestió de minuts) es pugui provar i avaluar el programari desitjat sense tot el procés d'instal·lació i configuració del programari i amb solament uns petits passos d'adequació a l'entorn, si cal.

Per exemple, per al cas de Liferay, podem accedir a Bitnami o SystemSector, que ofereixen una gran quantitat d'*appliances* de diferents programaris *open source* (o propietaris vinculats a una llicència o una versió de proves) realitzades per experts i per a diferents hipervisors (per exemple, en el cas de Liferay, sobre Bitnami o SystemSector per VMware i Virtualbox).

Aquest tipus de programari *appliances* està en un moment d'auge, ja que els responsables d'IT no desitgen gastar una gran quantitat de temps i esforç a enfrontar-se a qüestions específiques de la instal·lació del programari (*kernel* adequat, llibreries, dependències, etc.) que poden fer que un servei quedi inestable o insegur per omissió o no configuració d'una part crítica, ja que no s'és expert en tot el que significa les particularitats del mateix (i que, en alguns casos, són molt complexes).

Tot això significa que es desisteixi d'aquest paquet (per les dificultats afegides que representa), la qual cosa representa un problema per als desenvolupadors del paquet, ja que no té acceptació per les dificultats que comporta tenir una instal·lació de proves simple, fàcil i sense riscos. Això genera frustració per als possibles usuaris del paquet (fins i tot, per al personal d'IT experimentat) i un fracàs per als seus desenvolupadors.

Això se soluciona amb una *virtual appliance* (VA), que es pot desplegar en una infraestructura amb un hipervisor o en un *cloud* públic i que conté el sistema operatiu, el programari d'aplicació i les dependències necessàries, la configuració i els arxius de dades necessàries per al funcionament inicial i tot llest per a funcionar, i, a més, en una gran varietat de formats per a diferents hipervi-

sors (KVM, Xen, VMware, Virtualbox, etc.), o una imatge ISO a instal·lar, un USB o un *cloud* (per exemple, una *Amazon Machine Image* –AMI– perquè pugui ser desplegada en EC2, *Elastic Compute Cloud*).

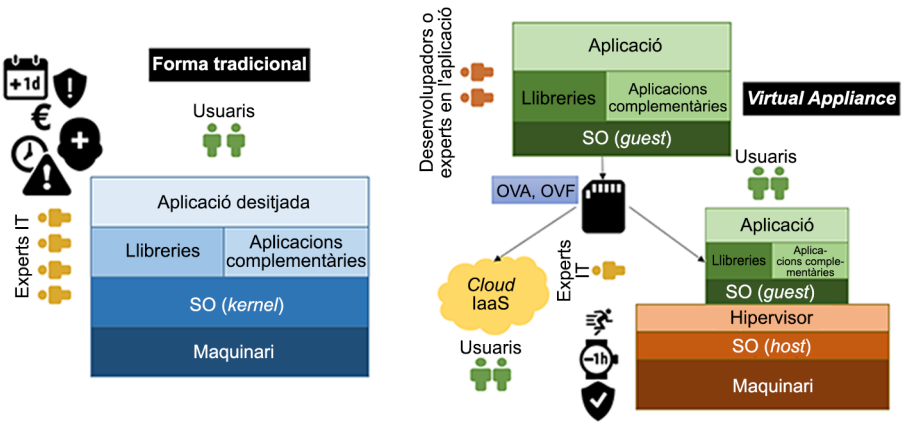
En el cas de les *virtual appliances* (VA), és el mateix desenvolupador o experts que empaqueten, configuren i cuiden els detalls del programari amb el qual desitgen que els seus clients facin les seves proves de concepte. El valor que això comporta és evident: qualsevol usuari, sense grans coneixements d'IT, pot provar un determinat paquet i prendre decisions si s'adapta a les seves necessitats o no.

Això representa, per als desenvolupadors, un notable increment de responsables o usuaris que proven i avaluen aquest programari, que si no fos així probablement haguessin desistit si no comptessin amb un grup d'IT expert i temps o recursos per a desplegar-ho, encara que solament fos com a prova de concepte.

> Alguns autors [Cca11] utilitzen una analogia equivalent al món de l'automotor: un usuari que desitgi un cotxe va a un concessionari i el compra d'acord amb les seves necessitats i possibilitats, i surt conduint-lo tenint la seguretat que tot funcionarà i no s'haurà de preocupar per la seguretat si, per exemple, l'ABS (sistema d'ajuda a la frenada) està ben configurat. Segurament, un usuari que volgués comprar totes les peces i construir-lo en funció dels seus coneixements també podria fer-ho, però és indubtable que el temps invertit seria molt major i molts aspectes dependrien de l'habilitat o coneixements d'aquests. La virtualització i el *cloud* han permès que la gran complexitat de la infraestructura quedi oculta i sigui fàcil «provar i conduir» noves opcions que ens poden ser útils deixant que la complexitat de la instal·lació i configuració quedi en mans del desenvolupador o expert.

Aquesta idea, com veurem més endavant, és la que dona suport a les propostes d'SaaS, que s'han transformat en una opció molt viable per als desenvolupadors de programari dins del món del *cloud*.

Si tenim en compte aquestes idees, es pot concloure que no és més que una evolució a la idea de l'SO, ja que no cal obtenir les fonts d'aquest i compilar-ho o configurar-ho, sinó que ja ve en una imatge (o directament en un disc virtual per a executar-ho des d'un hipervisor) amb totes les utilitats i programari bàsic necessari per al funcionament i, a més, el programari extra o específic necessari es pot obtenir des d'un repositori, creat per usuaris experts en aquesta distribució i en l'aplicació, per mitjà d'una ordre. Per exemple, per a instal·lar un servidor de base de dades amb Debian, solament hem d'executar `apt-get install mysql-server`, respondre les preguntes que ens farà l'instal·lador i tindrem el servei funcionant en aquesta distribució (solament s'hauran de fer uns petits ajustos per a acabar de sintonitzar l'aplicació a les necessitats dels usuaris). La figura següent resumeix les idees explicades en aquest apartat:



El format més habitual per a les *virtual* (*software*) *appliances* és l'OVF (*Open Virtualization Format*). Aquest format és un *open standard* per a empaquetar i distribuir *virtual appliances* en forma oberta, segura, eficient i portable sense dependències especials d'un hipervisor o arquitectura del processador. Inicialment (2007) va ser una proposta de VMware, Dell, HP, IBM, Microsoft i XenSource enviada a la DMTF (*Distributed Management Task Force*), la qual ha normalitzat el format i alliberat diferents versions (l'última el 2015 V2.1.1) incloent millores (per exemple, en la 2.0, pensada especialment pel *cloud*, amb l'especificació de la xarxa i encriptació per a millorar la seguretat en l'enviament d'*appliances*) i, posteriorment, ha estat adoptada com a ANSI *standard* (2010) i ISO (*International Organization for Standardization*) a l'IEC (*International Electrotechnical Commission*) el 2011.

Un paquet OVF consisteix en diversos arxius en un directori que inclou un arxiu XML amb extensió **.ovf**, el qual inclou metadades com ara el nom, el maquinari requerit, les referències a altres arxius i una (o més) imatge de disc (en format *raw*, que és una còpia binària sector-per-sector de la font). Aquest directori es pot empaquetar com un únic arxiu amb extensió **.ova** (*open virtual appliance or application*), el qual és un arxiu `tar` del directori OVF. Avui dia, és suportat per Amazon Elastic Compute Cloud, IBM SmartCloud, Microsoft System Center Virtual Machine Manager, OpenNode, Oracle VM, rPath, Xarxa Hat Virtualization, SUSE Studio, VMware, VirtualBox i XenServer Citrix, entre d'altres [Ovf15].

Un altre format acceptat per gran part dels proveïdors és el **VMDK** (*Virtual Machine Disk*), desenvolupat per VMware. Actualment és un format obert (VMware proveeix un paquet VDDK –llibreries, documentació i exemples– per a crear o accedir a arxius vmdk i una eina –`ovftool`– per a convertir vmdk a ovf). Inicialment, suportava imatges de 2TB, però en l'última actualització ha estat estès fins a 62TB i hi ha eines *open source* com `qemu-img` que permeten fer la conversió de formats per a adequar-los als diferents hipervisors.

El format VHD (*Virtual Hard Disk*) també utilitzat per alguns proveïdors de *virtual appliances* està orientat als sistemes Windows® i representa un virtual *hard disk drive* (HDD). Va ser creat per a Virtual PC per Connectix i va passar a

ser propietat de Microsoft el 2003 i des de l'any 2005, Microsoft té disponible a tercers l'especificació de VHD amb la promesa de fer-ho en un futur per a Microsoft Open Specification.

Finalment, un altre format utilitzat per algunes VA és el **qcow/qcow2** (QEMU *Copy On Write*) utilitzat per QEMU/KVM, que utilitza una estratègia d'optimització que demora l'emmagatzematge en el disc fins que calgui, per la qual cosa els formats són més petits que un *raw*, ja que no s'emmagatzema espai buit. Aquest format permet agregar noves dades a un arxiu ja existent, suporta còpies o instantànies (*snapshots*) parcials múltiples, la comprensió basada en `zlib` i el xifrat per mitjà d'AES. Un dels problemes d'aquest format és que no pot ser muntat directament com els discos *raw* i cal una utilitat que primer llegeixi l'arxiu abans que pugui ser muntat.

En el cas dels sistemes operatius, les imatges de la distribució es proveeixen generalment com a ISO (format basat en l'estàndard ISO 9660 o el protocol *Universal Disk Format* (UDF) creat com un format només de lectura i que manté l'estructura original i del qual hi ha diverses variants) i és l'utilitzat per a crear CD/DVD, però també es pot crear des d'un CD/DVD amb `dd if=/dev/cdrom of=/tmp/cd.iso` o amb `genisoimage`, si els arxius estan en un directori, o muntar com si fos un directori amb `mount -o loop arxiu.iso /mnt`.

Aquesta imatge ens serà útil per a arrencar el sistema operatiu des d'un CD, DVD, USB o des d'una MV i qualsevol modificació posterior que es faci en aquest SO no quedarà reflectida en la ISO (generalment són utilitzades per a instal·lar una distribució).

No obstant això, hi ha una gran quantitat de repositoris que publiquen SO ja instal·lats com a VA llestos per a funcionar, entre els quals podem enumerar OSBoxes (per a VirtualBox i VMware), virtualboxes.org, virtualboximages, VMA, a més del fet que la majoria de les distribucions ja ofereixen les seves imatges en diferents formats preparats pel *cloud* (per exemple, Ubuntu i Centos).

Amb l'auge de la VA han sorgit una sèrie de llocs que proveeixen VA amb diferents configuracions o, fins i tot, alguna d'aquestes permet la creació i adaptació de les VA d'acord amb unes necessitats específiques o amb una configuració predeterminada per a diferents hipervisors.

A continuació, mostrarem algunes d'aquestes.

a) Bitnami: és una plataforma que permet disposar de VA en diferents formats per a ser executades tant en el local (com VM) o *containers* o directament en el *cloud* (AWS, GoogleCloud, Azure, OracleCP, 1&1, CenturyLink o GoDaddy). Aquesta plataforma inclou més de cent cinquanta aplicacions o *stacks* (diferents paquets de programari agrupats com són LAMP, WAMP, WAPP, MAMP, LAPP o MAPP) actualitzades i segures, optimitzades, consistents (significa que

implementar una aplicació com una màquina virtual o iniciar-la en el núvol és pràcticament el mateix, per la qual cosa és summament fàcil passar del desenvolupament a producció en el *cloud*).

Bitnami ofereix *containers* que contenen les últimes versions de l'aplicació (aplicació + llibreries + dependències) que poden ser accedits de diferents maneres o, fins i tot, ser construïts d'acord amb les necessitats de l'usuari amb Stacksmith. També pot desplegar les VA en el *cloud* ajustant-les a les particularitats de cada proveïdor i donant la millor opció entre les arquitectures sofisticades i l'escalabilitat.

Però, a més, si es desitja executar en MV o instal·lar en un SO local, es poden utilitzar les imatges o els binaris que utilitzen els mateixos components que les versions de *cloud* i els contenidors, permetent una coherència entre totes les aplicacions.

b) SuSEstudio: és una plataforma en línia desenvolupada per SuSE per a la creació de VA basada en aquesta distribució de l'SO (es pot escollir en openSuSE i SuSEEnterprise, amb llicència). Els usuaris poden crear el seu propi SO, el seu *software appliance* o *virtual appliance*, seleccionant aquelles aplicacions o paquets que desitgen tenir en el seu Linux «personalitzat».

La plataforma també ofereix una sèrie d'imatges preconfigurades (jeOS, *minimal server*, escriptoris GNOME i KDE, entre d'altres) i diferents formats (Live CD/DVD/ISO, VMDK, VirtualBox, VHD/Azure, USB, Xen, KVM, OVF, Amazon EC2 (AMI), PXEBoot (*on site version only*) per a ser descarregats o enviats directament al *cloud* (Azure, Amazon EC2 SUSE Cloud).

Un aspecte interessant és que SuSESTudio també permet clonar i provar la imatge generada sobre el mateix lloc, validar el que desitja l'usuari, això és, si conté tots els elements necessaris abans de generar la imatge definitiva o baixar simplement la VA creada per altres usuaris en el format desitjat.

c) Turnkey: la Turnkey *Linux Virtual Appliance Library* és un projecte *open source* que proveeix de VA (100+) basades en Debian i que poden ser executades com a VM, contenidors o desplegades en el *cloud* (disponibles en un ampli rang de formats, per exemple, ISO, ova, vmdk, Openstack, Promox, Xen LXC, Docker).

Turnkey també està connectada a Amazon i permet, per mitjà del TurnkeyHub, desplegar les màquines amb un mínim cost i amb VA totalment adaptades a aquesta infraestructura. La base (TurkeyCore) per a totes les VA inclouen Debian 8, amb actualitzacions diàries, *1-clik backup and restore*, Dynamic DNS, NTP, LVM, AJAX web Shell –proveeix accés a una terminal des d'un navegador– *Web Management Interface* (*Webmin*).

d) OZ: si bé hi ha importants eines de desplegament (Ansible, Xef, Puppet), aquesta aplicació permet la creació automàtica de VA de sistemes operatius amb una mínima intervenció de l'usuari i adaptables a les necessitats que aquest indiqui en un arxiu de configuració en format xml.

Oz pot instal·lar l'SO, personalitzar-lo, o també generar el *package manifests*. OZ dona suport per a generar VA de RHEL, CentOS, Scientific Linux, OEL, Fedora, OpenSUSE, Debian, Ubuntu, Mandrake, Mandriva, Mageia, FreeBSD i Windows en diferents versions. Un aspecte molt important que presenta OZ, a diferència d'altres aplicacions similars, és la seva facilitat d'instal·lació i utilització i que la instal·lació realitzada serà idèntica al fet que si aquesta s'hagués realitzat des de la ISO de la distribució en una màquina física (*bare-metal*).

e) BoxGrinder: és un projecte per a crear VA des d'un arxiu de text, però la seva última versió estable és de 2012. L'usuari ha de crear l'*appliance definition file* a partir d'un esquema (arxiu escrit en YAML) i després executar *boxgrinder* que baixarà tots els elements necessaris, construirà la instància en el format seleccionat i el descarregarà en la plataforma indicada. BoxGrinder suporta plataformes *cloud* (EC2, Xen, KVM, VMware) i pot utilitzar com a SO base Fedora, RHE o CentOS o també es pot escriure el *plugin* per a donar suport a una altra plataforma de virtualització o SO. Si bé és una eina interessant, la seva manca d'informació i detall sobre l'activitat del projecte fa que calgui tenir certa precaució amb les VA generades, ja que poden estar desactualitzades o ser no adequades.

Com ja hem esmentat anteriorment, la virtualització del sistema operatiu ha portat a crear una evolució de les *virtual appliances*, ja que en aquest cas s'utilitzen els anomenats contenidors que solament inclouen el programari d'aplicació i les dependències utilitzant l'SO *host* com a SO del contenidor. Atesa la seva importància en el *cloud* i en els entorns DevOps, aquests temes seran tractats en futurs mòduls.

També en el mateix sentit de la VA és important esmentar Vagrant, que és una eina orientada als desenvolupadors que proporciona ambients de treball (basat en MV) fàcils de configurar, reproduïbles i portàtils construïts sobre la tecnologia estàndard i controlats per un sol flux de treball, ajudant a maximitzar la productivitat i flexibilitat.

Vagrant, que pot ser considerat com a administrador d'MV, utilitza *provisioners* (aprovisionadors) i *providers* (proveïdors) com a actors bàsics per a administrar els entorns de desenvolupament. Els *provisioners* són eines que permeten als usuaris personalitzar la configuració de les MV, com ara Puppet, Ansible o Xef, que són les més utilitzades en l'ecosistema Vagrant. Els *providers* són els serveis que Vagrant utilitza per a crear i inicialitzar els entorns virtuals (MV) i alguns dels proveïdors més utilitzats són VirtualBox, Amazon AWS, VMWare i Docker. És a dir, Vagrant està per sobre del programari de virtualització com un *wrapper* i ajuda el desenvolupador a interactuar fàcilment amb els *providers*.

Els requisits d'MV i programari s'escriuen en un arxiu denominat *Vagrantfile* que tindrà els passos a dur a terme per a crear un sistema integrat de desenvolupament llest per a usar en una MV (*development ready box*).

Box és el format (extensió **.box**) per als entorns Vagrant i pot ser utilitzat per qualsevol en qualsevol plataforma que estigui suportada per Vagrant, garantint que l'entorn serà exactament igual en un o altre lloc (és a dir, copiant el *box* tindré el mateix entorn en una màquina que a l'original). Els *boxes* poden ser compartits en un entorn oficial de Vagrant anomenat Atles que permet que els *boxes* (una espècie de VA) puguin ser classificats per *providers*, arquitectura o SO per a accedir ràpidament a aquests (consulteu els *Bentos boxes*).

Altres repositoris de VA

A més d'aquests repositoris (generalistes), hi ha altres repositoris de VA orientats cap a un hipervisor determinat: VMware Solution Exchange, Xenserver per a aquestes plataformes, o dels propis desenvolupadors, OpenVAS, Oracle, entre d'altres.

4.3. IaaS (*infrastructure as a service*)

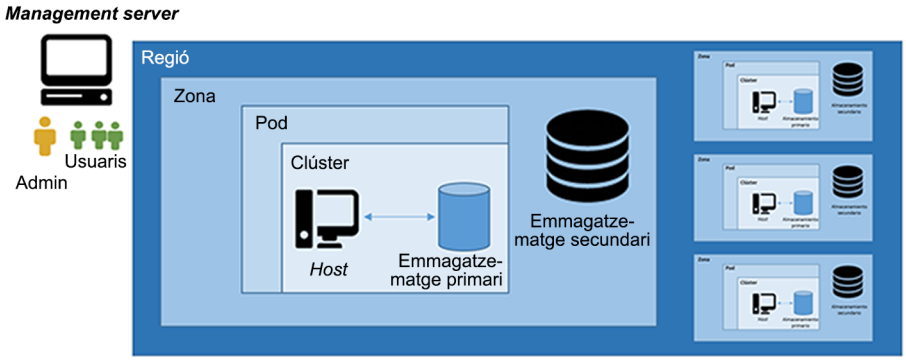
Dins de les plataformes *open source* que podem trobar amb llicència Apache, GPL o similars tenim (ordre alfabètic):

a) Apache Cloud Stack: és una plataforma IaaS *open source* (des de 2010 parcialment amb GPL i des de 2011 amb ASL, vegeu la seva història a l'inici) orientada a crear, manejar i aprovisionar infraestructura *cloud* (recursos de còmput, *pools* d'emmagatzematge i xarxes) en diferents supervisors (KVM, LXC, vSphere –via vCenter–, Xenserver, Xen Project, Hyper-V). Amb això, permet inicialitzar un servei elàstic de *cloud clomputing* i permet als usuaris finals aprovisionar els recursos sent capaç de manejar centenars de servidors físics geogràficament distribuïts en centres de dades i amb molt bon escalat (gairebé linealment) de servidor d'administració, evitant així la necessitat de servidors de gestió a nivell de clústers. Cloudstatck configura automàticament les xarxes i l'emmagatzematge per a cada desplegament d'MV, 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*, oferint, a més, una interfície web (que pot ser personalitzada) per a l'aprovisionament i administració del *cloud*, i també una interfície d'usuari (també web) utilitzada per a executar i gestionar les MV.

Aquesta plataforma proveeix una *REST-like API* per a operar i administrar el *cloud* i dona suport a l'*EC2 API* permetent que eines d'EC2 puguin ser utilitzades en aquesta. La instal·lació mínima consisteix en una màquina que executa el *CloudStackManagement Server* i una altra màquina que actua com a infraestructura *cloud* (que no és més que una màquina executadora de l'hipervisor), encara que, per a proves de concepte, es pot muntar tot en una única màquina (*ManagementServer* i *KVM hypervisor host*).

El *management server* és l'(únic) punt de configuració i es pot executar en una màquina dedicada o en una MV i controla el desplegament d'MV en els *hosts* (assigna l'IP i l'emmagatzematge a cada instància), s'executa amb Apache Tomcat i requereix MySQL. L'arquitectura d'Apache CloudStack diferencia entre: regions (col·lecció de zones geogràficament properes manejades per un o més *management servers*), zones (equivalent a un únic *datacenter* amb un o més clústers –*pods*– i emmagatzematge secundari), *pods* (*racks* amb *switchs layer-2* i un o més clústers), clústers (un o més *hosts* homogenis i emmagatzematge primari), *host* (únic ordinador en un clúster –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, s'ofereixen diferents tipus, però es poden classificar en dos escenaris concrets: bàsic (similar a la clàssica d'AWS on es proveeix una xarxa *layer-2* i s'aïlla el *guest* en *layer-3* amb el *bridge* sobre els hipervisors), avançat (utilitza aïllament en *layer-2*: VLANs).

Com a requeriments, aquesta plataforma es pot instal·lar (v4.9) sobre CentOS/RHEL 6.3+ o Ubuntu 14.04 de 64b, 4 GB RAM, 250 GB disc (500 GB recomanat), 1 NIC + FQDN i pel *host* o hipervisor ha de tenir extensions maquinari (Intel-VT, AMD-V), CPU X86 de 64 bits, 4 GB RAM, 36 GB de disc i un NIC. La figura següent mostra l'arquitectura d'aquesta plataforma:



b) Eucaliptus: aquesta plataforma *open source* (GPL/ASL) permet construir *clouds* privats i híbrids compatibles amb AWS (des de 2012). Eucalyptus és l'acrònim d'*Elastic Utility Computing Architecture for Linking Your Programs To Useful Systems* i permet que els recursos de còmput, emmagatzematge i xarxes puguin ser assignats i escalats en funció de les necessitats de càrrega. Aquesta plataforma comença a prendre forma dins del projecte *Virtual Grid Application Development Software* [2003-2008] a la Universitat de Rice. Va ocórrer una fita important dins del desenvolupament quan, l'any 2009, va ser inclosa dins del repositori d'Ubuntu 9.04, formant part de l'estratègia d'Ubuntu Enterprise Cloud (UEC), en què aquesta plataforma va ser la protagonista fins a la versió 11.10. El 2011, UEC és reemplaçat per Ubuntu Cloud Infrastructure i adopta com a plataforma OpenStack.

Eucalyptus Systems (empresa amb capital-risc) es crea el 2009 per a potenciar el seu desenvolupament i, el setembre de 2014, aquesta passa a ser propietat de Hewlett Packard. El programari parteix de l'estratègia HP Helion com a portafolis de *cloud computing* empresarials basats en *open source* (juntament amb HPE Helion OpenStack i HPE Helion Stackato com a PaaS basat en *Cloud Foundry*) i el producte passa a anomenar-se HPE Helion Eucalyptus. Com a requeriments, requereix CentOS 7 o RHEL 7 de 64 bits i tots els serveis han de ser instal·lats en servidors físics (no en MV).

Eucalyptus està basat en els següents components:

- *Cloud Controller* (CLC) és el punt d'entrada en el núvol per als administradors, desenvolupadors, gestors de projectes i usuaris finals. El CLC maneja la persistència i és el *backend* per l'UFS (serveis d'usuari, UFS, que hi poden haver a la mateixa màquina). Un *cloud* d'Eucalyptus pot tenir el mateix CLC. És un programa Java que ofereix interfícies compatibles amb EC2 o web i representa la interfície administrativa, permetent la gestió de recursos i el sistema d'*accounting*, i també processar les peticions a l'API per CLI (per exemple, d'`euca2ools`) o GUI (com a Eucalyptus *User Console*).

- *User-Facing Services* (UFS) serveixen com a punts finals per als serveis compatibles amb AWS oferts per Eucalyptus: EC2 (computació), AS (*AutoScaling*), CW (*CloudWatch*), ELB (*LoadBalancing*), IAM (Euare) i STS.

- *Object Storage Provider* (OSP) és el proveïdor d'emmagatzematge i pot ser Eucalyptus Walrus, Riak CS o Ceph-RGW. Walrus és l'equivalent a l'AWS-S3 i ofereix l'emmagatzematge de les MV i pot ser utilitzat com a servei d'emmagatzematge simple mitjançant *HTTP put/get*.

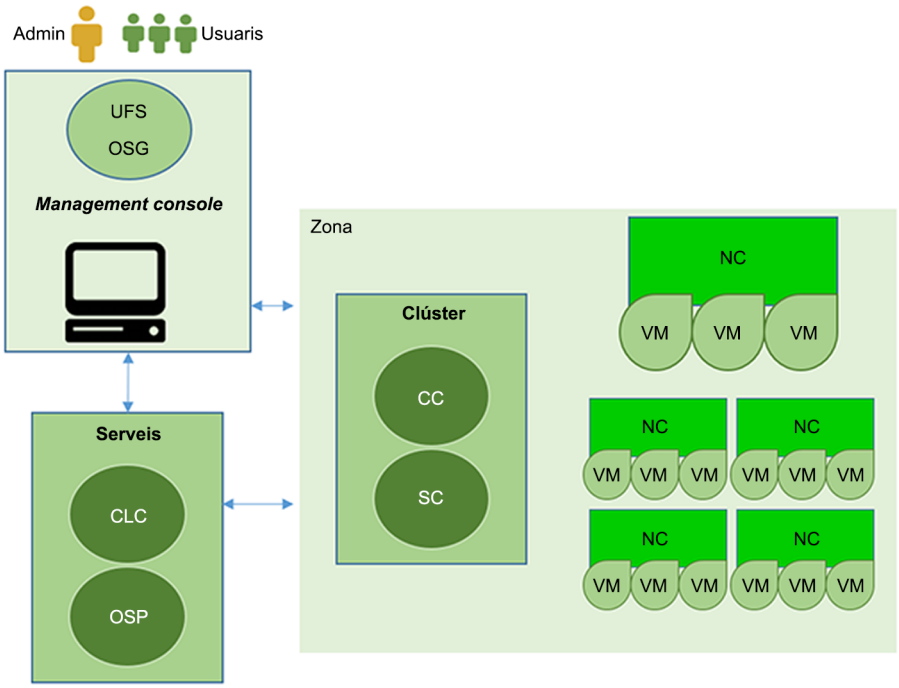
- *Management Console* és una interfície web fàcil d'usar que permet administrar el *cloud* d'Eucaliptus (1 o més), i generalment, està en la mateixa màquina d'UFS.

- *Cluster Controler* (CC) s'ha d'executar en una màquina *host* que tingui connectivitat de xarxa amb les màquines que executen els controladors de node (NC), ja que recopilen informació sobre aquests i planifiquen l'execució de les màquines virtuals (MV) a les NC específiques. Totes les NC associades amb un sol CC han d'estar a la mateixa subxarxa.

- *Storage Controller* (SC) és equivalent a Amazon *Elastic Block Store* (EBS) i pot interactuar amb diversos sistemes d'emmagatzematge exportant els volums d'emmagatzematge i que podran ser connectats a les màquines virtuals i muntats, o accedits com un dispositiu *raw*.

- *Node Controller* (NC) és el que s'executa en qualsevol màquina que allotja instàncies de VM. L'NC controla les activitats de VM, incloent l'execució, el monitoratge i la terminació d'instàncies de VM.

La plataforma també inclou Euca2ools, que són eines CLI per a interactuar amb els W que exporten una REST/*Query-based* API compatible amb els serveis EC2/S3 i que també funcionen amb Eucalyptus (són equivalents a les d'Amazon *api-tools* i *ami-tools*) i accepten les mateixes variables i paràmetres. La figura següent mostra l'arquitectura d'Eucalyptus.

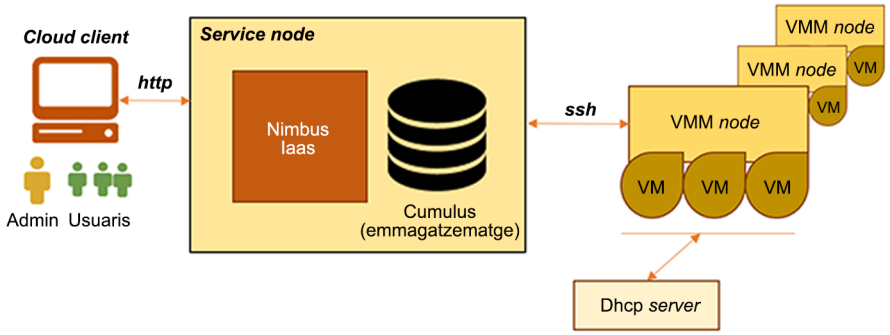


c) Nimbus: són un conjunt d'eines *open source* (ASL) enfocades a proveir capacitats d'IaaS, com ara *clouds* privats o comunitaris, incorporant recursos externs (MV) mitjançant el Nimbus *Workspace Service* i gestionant, utilitzant Cumulus, un emmagatzematge basat en quotes sobre el *cloud*. A més, utilitzant Nimbus *Context Broker*, es poden crear configuracions segures basades en contextos o utilitzar eines d'escalat per a adequar la infraestructura a la càrrega dinàmicament per mitjà dels proveïdors, treballar amb diferents hipervisors (Xen o KVM), incloure la gestió de recursos locals (per exemple, sistemes com PBS) i API (inclou compatibilitat amb EC2). Els requeriments per a la seva instal·lació són els habituals i solament cal complir amb les versions de determinats paquets i que els serveis de Nimbus estiguin en el mateix *host* que Cumulus (a partir de v2.6).

La plataforma es complementa amb altres eines com Phantom (monitoritza els recursos i automàticament en configura i desplega de nous per a mantenir diferents demandes o fallades), cloudinit.d (eina per llançar, controlar i monitoritzar entorns complexos en el núvol facilitant la gestió i desplegament

d'MV basades en «plans» prèviament configurats), i el *Context Broker* (un servei que permet als usuaris coordinar i desplegar grans clústers virtuals automàticament i en forma repetitiva).

La figura següent mostra l'arquitectura de Nimbus:



d) OpenNebula: plataforma *open source* (ASL) per a construir *clouds* IaaS tant privats com públics o híbrids 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 a desplegar serveis en múltiples nivells (per exemple, en 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 la interoperabilitat i portabilitat, i permet que els usuaris puguin realitzar interconnexions amb diferents interfícies (EC2 *Query*, OGF *Open Cloud Computing Interface* i vCloud) i interactuar amb diferents hipervisors (Xen, KVM i VMware).

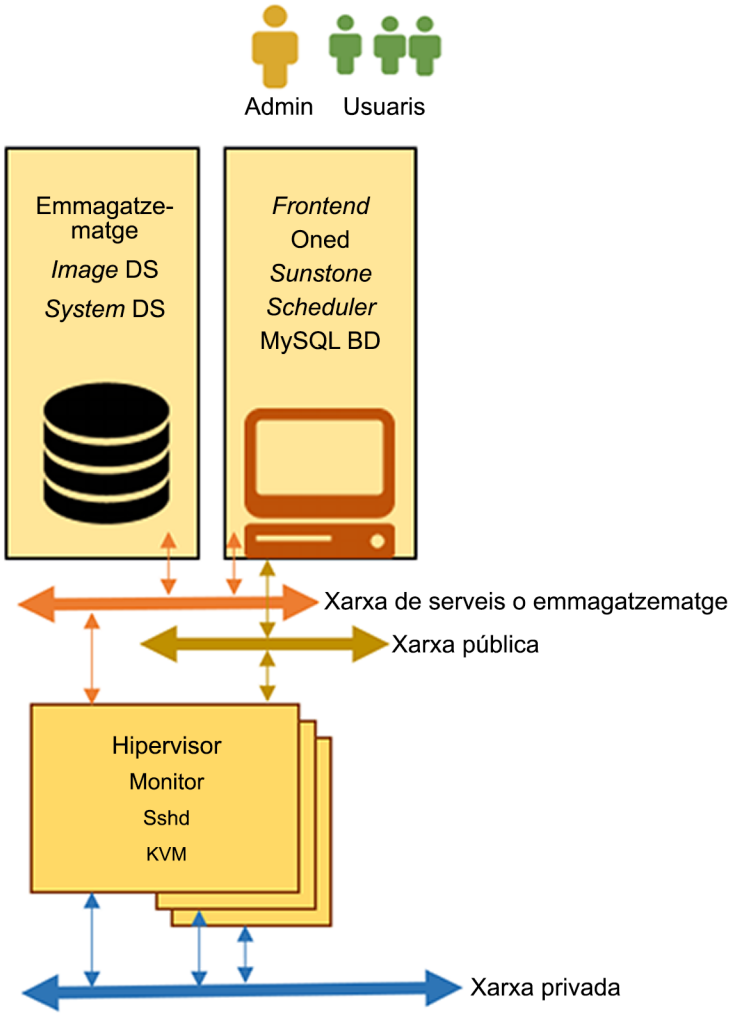
OpenNebula Systems (anteriorment C12G Labs) és la responsable del desenvolupament i del manteniment, i la plataforma compta amb un ecosistema anomenat AppMakarket com un catàleg en què els usuaris o les organitzacions poden ràpidament distribuir i desplegar *appliances ready-to-run* en *clouds* OpenNebula.

L'estabilitat de la plataforma i la qualitat dels serveis que proveeix l'han consolidat 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 TU, centres de supercomputació, laboratoris de recerca i projectes de recerca internacionals.

L'esquema d'interacció és mitjançant un portal basat en rols i autoservei intuïtiu, amb un catàleg de serveis automatitzats, interfícies d'administració i l'*appliance* AppMarket que permeten gestionar i administrar tots els recursos per als diferents nivells o rols des d'un únic punt. Els requisits per a instal·lar la plataforma es diferencien en instal·lacions bàsiques o mitjanes (desenes d'hipervisors) o avançades (centenes d'hipervisors). En relació amb les xarxes,

la de serveis entre el *frontend* i l'emmagatzematge poden ser compartides (atès el baix volum de comunicacions dels serveis) i les MV requeriran una xarxa pública i una altra privada (per a implementar *Vlans* aïllades).

Pel que fa als sistemes operatius pot ser Debian/Ubuntu o CentOS/RHEL (en totes les màquines) amb paquets específics per a cada distribució, amb KVM pels 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, i quant a l'autenticació es pot configurar el sistema natiu d'autenticació que disposa OpenNebula o Active Directory. Els requeriments de maquinari del *frontend* en instal·lacions bàsiques són: 1 CPU (2 *cores*), 2GB RAM, 100GB disc i 2 NIC, mentre que per a les avançades: 2 CPU (4 *cores*), 4Gb RAM, 500GB disc i 2 NIC. La figura següent mostra l'arquitectura d'OpenNebula [Ona15].



OpenNebula utilitza KVM com a hipervisor, però, sobre la base de la seva flexibilitat i interoperabilitat, algunes organitzacions o empreses l'utilitzen com a *cloud management* en VMware vCenter, anomenat vOneCloud, per a proporcionar una capa d'aprovisionament *multi-tenancy* per sobre de VMware vCenter. Aquests desplegaments busquen característiques de provisió, elasticitat i

multi-tenancy com a aprovisionament de centres de dades virtuals i federació de centres de dades, mentre que la infraestructura és administrada per eines ja conegudes com ara vSphere i vCenter Operations Manager.

e) OpenQRM (*community edition*): és una plataforma *open source* (GPL) de *cloud computing* per a manejar la infraestructura de centres de dades, permetent construir en aquests *clouds* privats, públics i híbrids gestionant la virtualització (per mitjà de KVM, Linux-VServer, OpenVZ, VMware ESX o Xen), l'emmagatzematge, la xarxa, el monitoratge i la seguretat. Amb això, permet desplegar serveis *multi-tier* en clústers de còmput com a màquines virtuals combinant tant els recursos locals com els remots i gestionant la seva assignació mitjançant les 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 pot ser substituït sense que calgui reconfigurar el programari i permet diferents models de migració d'MV (P2V –*physical to virtual*–, V2P –*virtual to physical*–, i V2V –*virtual to virtual*–), i també és possible fer una transició d'un hipervisor a uns altres amb la mateixa MV.

L'arquitectura d'aquesta plataforma és extensible mitjançant connectors (*pluglins*) que permeten inserir API o connectors cap al *cloud* públic (per exemple, AWS) o altres *clouds* (Openstack, Eucaliptus, etc.). Els requeriments per a fer una prova funcional són un servidor (per al servidor openQRM, rd virtualització i emmagatzematge) amb Intel o AMD de 64bit, *dual/quad core* i extensions VT-x/AMD-V, 2 Gb RAM, almenys 20 GB disc i un NIC (recomanat 1 Gbit/s) i accés a internet.

L'última versió (5.3) s'ha de descarregar des del desenvolupador (que té publicats una sèrie de *How-To*), però les versions anteriors s'han de descarregar directament des de SourceForge on també es pot accedir a la documentació de la comunitat (si bé està poc actualitzada).

g) OpenStack: és una plataforma *open source* (ASL) de *cloud computing* i que permet desplegar IaaS mitjançant un conjunt de components interrelacionats que controlen els recursos de maquinari (de diferents venedors) 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*) per mitjà de Web o CLI o utilitzant una API RESTful. OpenStack s'inicia el 2010 com un projecte conjunt de Rackspace Hosting i la NASA passant a ser gestionat el 2014 per l'OpenStack Foundation amb l'objectiu de promoure aquesta plataforma i la seva comunitat, de la qual avui en formen part més de cinccentes empreses, organitzacions i institucions.

La comunitat està organitzada en cicles de desenvolupament i actualitzacions semestrals que es reuneixen en sessions de treball (multitudinàries) en l'OpenStack Summit per a facilitar el desenvolupament i generar plans de fu-

tur (en la sessió d'abril de 2016, es van reunir 7.500 persones vinculades a la plataforma i l'última va tenir lloc a Barcelona, l'octubre de 2016. Es pot accedir a tots els vídeos de les sessions). Avui es considera que la comunitat OpenStack està formada per més de 62.000 desenvolupadors o usuaris, més de 640 companyies de 187 països i inclou més de 20 M línies de codi.

El primer codi d'OpenStack (2010) prové de la plataforma Nebula de la NASA (no confondre amb OpenNebula) i de la plataforma Rackspace Cloud. El 2011, Ubuntu adopta OpenStack per a la seva estratègia de *cloud* (reemplaçant Eucaliptus) i dona suport als *clouds* basats en Ubuntu 11.04 i a la versió OpenStack Cactus. Igualment passa amb Debian 7.0, que inclou la versió Essex, el mateix per SuSE, i, a partir de l'any 2012, totes les grans companyies de tecnologia el comencen a incloure en les seves estratègies i fer-lo disponible per als seus clients, integrat o adaptat juntament amb els seus productes (RHEL, Oracle, HP, etc.), arribant avui dia a tenir instal·lacions molt particulars com la del Cern (la més gran de més de 190.000 *cores*), SnapDeal (Índia, Markeplace més de 100.000 *cores* i 16 PBytes d'emmagatzematge), China Mobile (10.000 nodes amb 100.000 *cores*), Walmart (*e-commerce* a 30 regions amb més de 170k+ *cores*) entre d'altres.

L'arquitectura d'OpenStack és modular amb diversos components, entre els quals es poden enumerar [Osd16]:

- ***Compute* (Nova)**: és la part principal de l'IaaS i actua com a controlador del *cloud*. Aquest gestiona els conjunts de recursos (*pools*) i pot treballar amb diferents tecnologies de virtualització, maquinari-base (*bare metal*) i configuracions de *high-performance computing* (HPC) amb KVM, VMware, Xen, Hyper-V com a possibles hipervisors i amb LXC com a suport per a contenidors. Aquest mòdul no requereix cap maquinari propietari (està escrit amb Python), escala horitzontalment i utilitza diferents llibreries externes com ara Eventlet, Kombu o SQLAlchemy.

- ***Networking* (Neutron)**: gestiona les xarxes i els seus paràmetres perquè no representin un coll d'ampolla en una instal·lació per mitjà de l'autoservei. L'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 tràfic i connectar els servidors i els dispositius a una o més xarxes. Els administradors poden aprofitar les xarxes definides per a programari de tecnologia (SDN) com OpenFlow i permet annexar serveis de xarxa addicionals, com els sistemes de detecció d'intrusos (IDS), balanceig de càrrega, tallafocs o xarxes privades virtuals (VPN).

- ***Block Storage* (Cinder)**: aquest mòdul proporciona dispositius persistents d'emmagatzematge a nivell de bloc que poden ser utilitzats per les instàncies de Nova i gestiona la creació, el muntatge o desmuntatge dels dispositius de bloc als servidors, els quals s'integren plenament a Nova i al pa-

nell de control (*dashboard*) perquè els usuaris puguin gestionar les seves pròpies necessitats d'emmagatzematge. A més de l'emmagatzematge local de Linux, pot utilitzar les plataformes d'emmagatzematge Ceph, CloudByte, Coraid, EMC GlusterFS, Hitachi, IBM, LIO, NetApp, Nexenta, Scality, SolidFire i HP, entre d'altres. L'emmagatzematge de blocs és apropiat per a escenaris en què el rendiment és sensible, com ara bases de dades, sistemes d'arxius de creixement dinàmic o per a permetre al servidor accedir a nivell de bloc «en brut» (*raw*).

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

- ***Image* (Glance)**: proporciona la gestió de 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 les d'emmagatzemar i gestionar les imatges del disc i dels servidors en una varietat de *back-ends* (incloent OpenStack *Object Storage* Swift) i disposa d'una API REST per a gestionar les imatges (per exemple, amb Heat, per a tenir metadades de les imatges, o amb Nova, per a 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 calgui.

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

- ***Dashboard* (Horizon)**: interfície del panell de control que permet als administradors o usuaris accedir i gestionar la provisió i automatització dels recursos basats en el *cloud*. El seu disseny és obert i permet que es puguin incloure noves funcionalitats (facturació, monitoratge, etc.), transformant-se en un 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* per mitjà de l'OpenStack-*native* REST API o *CloudFormation-compatible Query* API.

- ***Workflow* (Mistral)**: és un servei que gestiona els fluxos de tasques (*workflows*) generats pels usuaris (escrits en YAML). Pot interactuar per mitjà de la REST API i iniciar l'execució del *workflow* mitjançant la mateixa API o programant el seu inici per mitjà d'un esdeveniment (*trigger*).

- ***Telemetry* (Ceilometer)**: proveeix un únic punt de contacte per a la facturació, unificant els comptadors necessaris per mitjà de tots els components de l'arquitectura i permetent que es puguin auditar i fer un seguiment dels mateixos amb l'objectiu de la transparència.

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

- ***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 d'MV utilitzant PXE i IPMI per a gestionar les màquines *hardware*.

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

- ***Shared File System* (Manila)**: proveeix una API per a gestionar compartició d'objectes (independents del proveïdor) permetent crear, esborrar, donar accés o denegar els objectes en diferents entorns d'emmagatzematge com ara EMC, NetApp, HP, IBM, Oracle, Quobyte i Hitachi o en sistemes d'arxius com, per exemple, RHEL GlusterFS.

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

- ***Search* (Searchlight)**: proveeix capacitat de cerca per mitjà dels serveis de *cloud* OpenStack mitjançant les API proveïdes per aquests i indexant les dades mitjançant ElasticSearch.

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

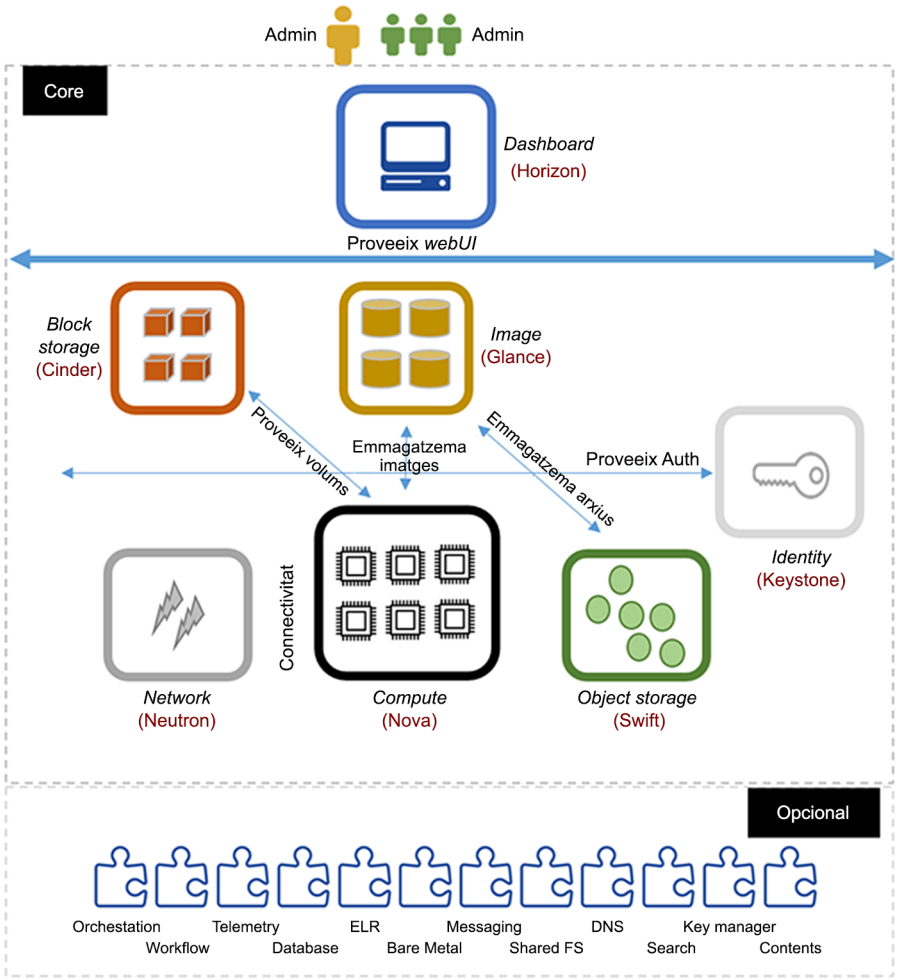
- ***Contents* (Magnum)**: proveeix un catàleg d'aplicacions d'OpenStack permetent que els desenvolupadors i administradors puguin publicar aplica-

cions *cloud-ready* i que aquestes estiguin disponibles per als usuaris i, així, puguin compondre entorns d'aplicació amb una simple selecció.

L'última versió, de novembre de 2016, és Newton i incorpora Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Heat, Ceilometer, Trove, Sahara, Ironic, Zaqar, Manila, Designate, Barbican, Searchlight, Magnum, considerant-se el nucli (*core services*) d'Openstack format per **Nova**, **Neutron**, **Swift**, **Cinder**, **Keystone**, **Glance** (+Horizon) i la resta de serveis opcionals. Consultant el MarketPlace podreu veure les distribucions o *appliances* que la suporten, els *clouds* públics i privats que la implementen, els proveïdors de formació o les consultories relacionades amb OpenStack o els controladors per a dispositius específics.

Com es pot observar en el *marketplace*, hi ha una gran quantitat d'*appliances* d'OpenStack amb diferents objectius entre les quals trobem Ubuntu Autopilot, Oracle, Mirantis, entre d'altres (s'ha de tenir en compte que algunes *appliances* d'alguns proveïdors són versions de prova per un temps limitat).

La figura següent mostra els components d'OpenStack Newton (versió d'octubre de 2016) i les seves interrelacions.



h) Ovirt: és una plataforma *open source* (ASL), creada per RH com a projecte de la comunitat i d'acord amb la seva distribució i tecnologia de gestió i provisió d'entorns virtualitzats (amb KVM) i formada per dos components: **oVirt engine** i **oVirt node**. Aquesta permet, per mitjà d'una interfície web molt senzilla, gestionar de forma centralitzada les màquines virtuals i els recursos de còmput, la xarxa i l'emmagatzematge de forma remota i sense accedir als recursos físics (és l'equivalent a l'*open source* a VMware™ vSphere™).

Els seus requeriments són molt bàsics, per exemple, per a l'oVirt Engine *dual-core*, 4GB RAM, 25 GB disc i NIC 1 GB, Fedora 19+, CentOS 6.5+, i pels *hosts* que desplegaran les MV (que poden ser oVirt Node, Fedora Host, CentOS Host) *dual-core* server, 10 GB RAM (considerar que cada VM s'endurà 1 GB aproximadament), 10 GB disc, NIC 1 GB amb extensions AMD-V, VT-x i emmagatzematge com NFS, iSCSI, FCP, Local, POSIX FS, o GlusterFS. A més, caldran les imatges dels discos d'instal·lació (Windows, RHEL, Ubuntu, Fedora, OpenSuSE, CentOS) de les quals després es crearan les MV.

El nucli d'oVirt engine està escrit en Java, la interfície en GWT *webtoolkit* i s'executa sobre el servidor d'aplicacions WildFly (abans JBoss). Els usuaris poden ser gestionats internament o connectats a un LDAP o AD. Per a l'emmagatzematge, oVirt utilitza PostgreSQL i proveeix RESTful API com a interfície a les funcionalitats de nucli. oVirt node és un servidor que executa RHEL, CentOS o Fedora (experimentalment Debian) amb hipervisor KVM i un daemon VDSM (*Virtual Desktop and Server Manager*). Tota la gestió i administració es realitza per mitjà del portal web d'oVirt engine, que es connecta amb VDSM per a realitzar les tasques indicades. oVirt engine pot ser instal·lat en un node separat o en un node del clúster com un MV (*self-hosted engine*), el qual es podrà instal·lar manualment o utilitzar una *appliance* llesta per a executar.

4.4. PaaS (*platform as a service*)

Les PaaS proporcionen als usuaris eines per a desenvolupar, executar i administrar aplicacions (generalment web), i atès que aquestes s'ofereixen com un servei per mitjà d'internet, els desenvolupadors poden treballar dins d'aquestes sense haver de gestionar la infraestructura subjacent, permetent-los concentrar-se únicament a desenvolupar la seva aplicació. Les ofertes de plataformes inclouen components de molts serveis utilitzats en el desenvolupament de programari, incloent bases de dades, servidors web, sistemes operatius i emmagatzematge, permetent que en algunes d'aquestes hi treballin múltiples usuaris, facilitant la col·laboració i compartició de codi i recursos.

Entre les més destacades (o que millor surten en alguns indicadors com a satisfacció, quota de mercat, suport i servei, per exemple, en G2 *Crowd categories/cloud-platform-as-a-service-paas*) són: **Heroku**, **Amazon EC2**, **Microsoft Azure**, **Force.com i OpenShift** (Heroku i Force.com sobre SalesForce).

Hi ha plataformes amb un alt grau de satisfacció pels seus usuaris però que encara no han aconseguit una alta quota de mercat, com poden ser **Morpheus** i **Dokku**. També hi ha productes que tenen presència i recursos, encara que amb valoracions per sota del primer grup com, per exemple, **Google App Engine**, **AWS Elastic Beanstalk** i **AWS Lambda**.

Entre les plataformes de «nínxol» o les molt especialitzades trobem **Cloud Foundry** i **Deis**. Hi ha diverses llistes de les PaaS més habituals i altres valoracions com la d'IDG , que inclouen a més de les esmentades **LongJump**, **IBMSmartCloud**, **CloudBees**, **EngineYard**, **HPE Helion Stackato** o d'altres no menys interessants: **Platform-sh**, **Zoho Creator**, **Mendix**, **Morpheus**.

Hi ha diverses raons per a seleccionar les plataformes PaaS, entre les quals es poden esmentar:

- Llenguatges de programació: atès que serà un projecte programari, les plataformes escollides seran aquelles que donin suport al llenguatge amb el qual es desenvoluparà el projecte i estiguin dins dels requeriments d'aquest.

- Públic enfront de privat: la PaaS pública representa beneficis de disponibilitat i immediatesa, ja que després del registre es comença a treballar. La PaaS privada requereix adequació i desplegament d'infraestructura i provisió del servei requerint recursos addicionals i un departament d'IT adequat als objectius, tenint com a contrapartida control, seguretat i privadesa.

- Temps de funcionament del servei: dependència total en el cas de la pública i controlat en el cas de la privada que poden afectar seriosament la planificació del projecte.

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

- Recursos: si la pròpia plataforma aporta els seus propis recursos o si aquests han de ser gestionats externament (per exemple, sobre EC2/S3).

- Integració: després el projecte s'haurà d'integrar a altres serveis, per la qual cosa s'ha d'analitzar el temps necessari per a realitzar-les i amb quina política per a evitar inconvenients una vegada iniciat el treball.

Sobre plataformes *open source* que poden ser instal·lades sobre infraestructura pròpia (o *cloud*) podem esmentar (cal tenir en compte que, si bé totes comparteixen objectius similars, la visió i missió de cadascuna d'aquestes és diferent i escapa al detall d'aquest apartat):

- **OpenShift origin**: Origin és el projecte comunitari que impulsa OpenShift i està construït sobre la base de contenidors Docker i gestió de clústers de

contenidors Kubernetes. Origin és una plataforma activa que redefineix el cicle de vida de l'aplicació i les eines de DevOps, proporcionant una plataforma completa d'aplicació de contenidors *open source*.

- **Apache Stratos** (abans W02): és un entorn PaaS extensible que permet executar aplicacions Apache Tomcat, PHP i MySQL i amb 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 a desenvolupar, provar i executar aplicacions escalables i, per als proveïdors de TU, obtenen benefici d'altes taxes d'utilització, gestió automatitzada de recursos i la visió general de la plataforma, inclosa la supervisió i facturació.

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

- **Tsuru**: PaaS *open source* desenvolupada per Globo.com i disponible en GitHub. Els desenvolupadors poden utilitzar `git, tsuru cli` o el propi panell de control (*dashboard*) per a desenvolupar aplicacions web en diferents llenguatges.

- **Cloud Foundry**: plataforma PaaS *open source* desenvolupada originalment per VMware i ara per Pivotal Software (*join venture* entre EMC, VMware i General Electric). Aquesta plataforma suporta tot el cicle de vida de les *apps* des del desenvolupament inicial, 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 que tingui permisos per accedir.

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

- **Deis Workflow**: PaaS que agrega una capa *developer-friendly* a qualsevol clúster Kubernetes, facilitant el desenvolupament i desplegament d'aplicacions.

- **AppScale**: plataforma *open source* que automàticament desplega i escala sobre *clouds* privats o públics aplicacions Google App Engine sense modificacions.

Si bé algunes d'aquestes **no** es poden qualificar com a plataformes PaaS, és útil esmentar les plataformes orientades als desenvolupadors (*cloud IDE*) com ara:

- **Cloud9.io**: ofereix quaranta llenguatges, control sobre l'aplicació d'espai de treball (*workspaces*) públic il·limitat gratuït, depuració d'aplicacions en diferents entorns (Django, Rails, WordPress...) en unes 300 combinacions, entre d'altres característiques.

- **Codenvy**: ofereix *cloud workspaces* per a grups de desenvolupadors basat en contenidors Docker i permet disposar d'una plataforma per a DevOps.

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

- **Codiad**: és una plataforma IDE basada en web amb uns requeriments molt reduïts, que permet disposar d'un entorn de desenvolupament en el *cloud* sense caure en potents recursos que necessiten els entorns IDE habituals. És per això que els usuaris d'eines com Eclipsi, 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 setanta-dos llenguatges i entorns preconfigurats.

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

També podem trobar entorns basats en el *cloud* d'objectius generalistes, per exemple, **Coding Ground**, que presenta 17 terminals en línia i més de 96 IDE interactius per a editar, compilar, executar i compartir programes en línia en una plataforma *cloud*.

4.5. SaaS (*software as a service*)

Les plataformes SaaS són un model de distribució de programari en què el suport lògic i les dades que maneja s'allotgen en servidors d'una companyia d'IT (no necessàriament el proveïdor del servei) i el client accedeix a aquests per mitjà d'un programari específic (programari client, encara que en desús) o per mitjà d'un servei web.

L'empresa proveïdora s'ocupa del manteniment, de l'operació diària i del suport del programari usat pel client i, en la majoria de casos, aquest pot accedir-hi 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 estan als servidors del proveïdor. El client no s'ha de preocupar per la seva disponibilitat, accés, manteniment i 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, per exemple:

- el client no necessita ni inversions en infraestructura ni personal tècnic especialitzat en el servei, reduint els costos i riscos d'inversió,
- l'operació recau en l'empresa proveïdora (disponibilitat, funcionalitat, seguretat, etc.) 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 cal tenir-lo en compte perquè continuï pagant,
- no cal preocupar-se per les llicències de les màquines (aquelles que es paguen, s'utilitzin o no), ja que solament es paga per l'ús,
- no cal modificar la màquina del client perquè accedeixi al servei ni adequar la potencia o les prestacions, ja que tot s'executarà en la màquina del proveïdor.

Així mateix, es poden enumerar un conjunt de **desavantatges** com ara:

- no es té accés a les dades (excepte que el servei prevegi un mètode de descàrrega),
- si no es compta amb mecanismes de xifrat i control disminueix l'índex de privadesa, el control i la seguretat,
- l'usuari accedeix a un servei i no el pot modificar ni configurar, més enllà de les opcions que li permet el proveïdor,
- es pot produir el *data lock-in* (en cas de tancament de l'activitat de l'empresa proveïdora) o *vendor lock-in* (costos més alts que la competència, ja que els de la migració encara són majors),
- pèrdua del servei o reducció de la qualitat, si la connexió és deficient.

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

a) Es pot accedir al servei per mitjà d'un navegador o de protocols normalitzats com ara REST?

b) Es pot accedir a un model de pagament com pot ser *pay-as-you-go*?

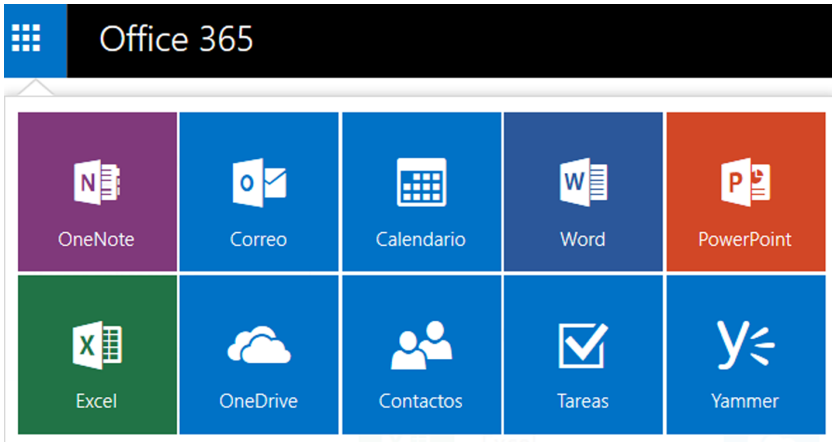
c) El programari escala sota demanda?

d) El proveïdor assumeix la responsabilitat de la infraestructura, la gestió, el manteniment, la millora i la supervisió del programari, deixant al client només utilitzar el programari?

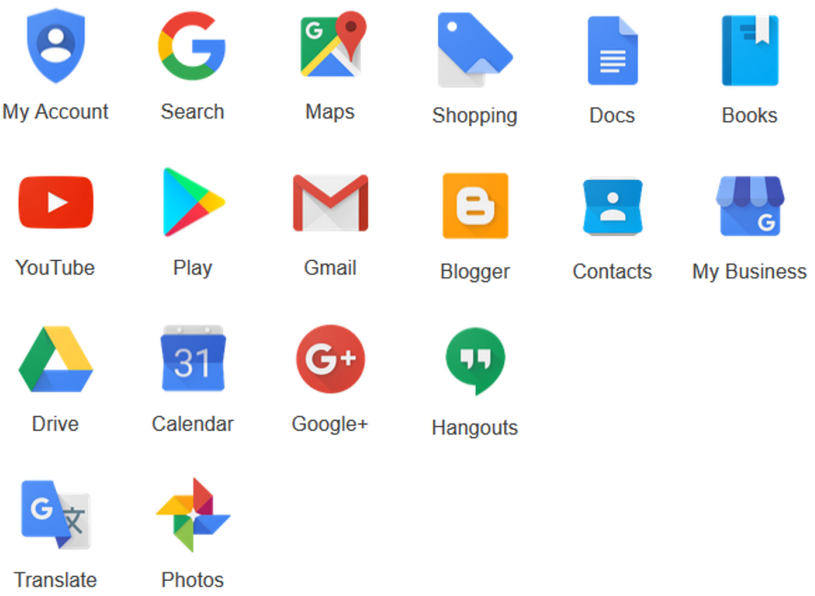
Si la resposta és un sí rotund, podem assegurar de ben segur que estem davant un SaaS.

Google i Microsoft

Dos dels exemples que es mostren habitualment de SaaS són dues de les grans companyies d'IT: Google i Microsoft. Les imatges, que hi ha a continuació, mostren els serveis oferts dins de les plataformes corresponents:



Amb 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, Yamer), missatges instantanis (Lync), correu (Exchange), emmagatzematge (OneDrive) però també CRM, comunicació (Skype) i BMI, entre els més destacats. S'ha de tenir en compte que alguns d'aquests són gratuïts (no *open source*) com, per exemple, el correu, l'emmagatzematge (fins a una determinada grandària) i d'altres són amb llicència amb diferents models (per exemple, per a centres educatius Office365 és gratuïta).



Google també ofereix els seus recursos, encara que amb un altre model de negoci (més gratuït):

A més dels mostrats 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 d'eines són d'ús gratuït i el model de negoci (encara que no es coneix totalment) està en les eines com Adwords i Adsense, que funcionen sobre les dades que es recullen des de les diferents eines (concepte anomenat dades massives o *big data*).

Entre d'altres exemples (solament algun d'aquests) de plataformes SaaS d'èxit podem esmentar (ordre alfabètic):

Categoria	Lloc
Emmagatzematge	Dropbox MediaFire Mega
Col·laboració	Asana Todoist Trello Evernote
Documents	Pixlr, Polar (edició fotos) SiteBuilder (creació web) SlideRocket, Prezzi (creació presentacions) Slideshare (presentacions o documents) StackEdit (edició a la web)
Multimèdia	NetFlix, Flickr (fotos) OutBrain (contingut) Spotify (música), Wikipedia (enciclopèdia)
Serveis	bit.ly (URL) Flood (test) Ionic (aplicacions) Mailchimp (correu electrònic) Panda (antivirus)

Categoria	Lloc
Social	BuzzStream Facebook Instagram LinkedIn LuckyOrange Twitter

Alguns autors consideren que el SaaS és una evolució d'un model de negoci àmpliament conegut, anomenat **ASP** (*application service provider*), que proporciona serveis als clients per mitjà d'una xarxa i permet l'accés a una aplicació de programari en particular. Aquest model ha evolucionat a partir dels creixents costos de programari especialitzat i de la complexitat pel que fa a la instal·lació i manteniment, que es redueixen i el fan accessible quan funcionen en mode ASP.

En principi, podem dir que són conceptes semblants als del 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), que es basa en una relació de tu-a-tu, mentre que SaaS és un model global, flexible i que, en principi, es basa en la ubiqüitat i en l'àmplia distribució (encara que la mateixa empresa o tercers puguin fer adaptacions i configuracions especialitzades).

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

- Les ASP (la majoria d'aquestes) s'executen mitjançant models client-servidor, per la qual cosa es requereix un programari específic amb un SO determinat. Es perd la ubiqüitat i la independència de l'SO (amb SaaS solament es necessita un navegador) i, com que és tecnologia d'un tercer, les actualitzacions i les versions depenen de la instal·lació realitzada. Això no passa amb el SaaS, ja que tots els clients sempre tenen l'última versió i el suport és unificat amb un servei uniforme, amb la conseqüent reducció dels costos.

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

- L'ASP pot integrar diferents serveis com ara ofimàtica + directori + CRM en un sol servei en funció dels acords amb els diferents proveïdors, mentre

que el SaaS està orientat a un únic producte (encara que alguns proveïdors ofereixen SaaS multiproductes).

- Finalment, pel que fa a la relació amb el desenvolupador o proveïdor amb 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, amb l'abans descrit, que un programari complex, com un ERP, solament pot funcionar en mode ASP per la gran adaptació i personalització que requereix, però hi ha casos com, per exemple, OpenBravo (ERP, TPV, POS Retail) que presten aquest servei com a SaaS o en un IaaS extern (AWS) per a pimes.

Quant als models de **monetització de SaaS**, són de diferents tipus i força més difusos que en altres segments del *cloud*.

Els habituals són ***pay-per-user*** (model similar al de les llicències en instal·lacions locals) o ***pay-as-you-go*** (per quantitat de recursos –emmagatzematge, CPU, etc.– que han estat consumits en un període, per exemple, per hora) o el seu equivalent ***pay-what-you-want***.

Pel que fa als models ***freemium*** i ***premium***, el *freemium* representa un percentatge molt alt d'usuaris en relació amb el *premium*, té limitacions en les prestacions o opcions i és la font que alimenta el *premium*. Els usuaris que realment provin i «usin» l'aplicació s'acabaran convertint en *premium*. Pagaran per això i això sustentarà el model de negoci. El proveïdor s'hi mirarà a conrear-ho amb el sentit de pertinença a un grup amb privilegis. Els *freemium* tindran incentius per a passar-se a *premium* i, en determinats moments, reduccions dinàmiques de les prestacions per a afavorir aquest pas.

Altres models de monetització són:

- *long tail* (catàleg de molts productes que es venen en poques unitats: model Amazon),

- *bait & hook*, esquer i ham (similar al *freemium*, accés a uns recursos, però limitats per a incentivar a passar-se a pagament. Per exemple, *pay-as-you-go*),

- subscripció o *paywall* (distribució dels costos en una gran quantitat d'usuaris i distribució d'impacte inicial per una llarga subscripció),

- anuncis (mètode indirecte que la publicitat pròpia o de tercers monetitza el servei),

- microtransaccions (generalment aplicat a l'oci, en què s'adquireixen recursos o serveis amb inversions de molt baix valor però que en el seu conjunt representa un gran volum), o

- *tiered service* (variant del *pay-as-you-go*, incrementa el preu de forma progressiva a mesura que es requereixen més recursos).

En relació amb l'**estructura del SaaS**, la majoria es basa en arquitectures *multitenant* en què una única versió de l'aplicació, amb una única configuració (maquinari, xarxa, sistema operatiu), s'utilitza per a tots els clients (*tenants*) i per a suportar l'escalabilitat, l'aplicació s'instal·la en diverses màquines, permetent l'escalat horitzontal.

No obstant això, algunes SaaS no utilitzen *multitenancy* sinó models més aïllats com els contenidors (model de Google) o màquines virtuals, ja que permeten una millor privadesa, funcionament i baix ús de recursos. Tanmateix, és un tema de gran discussió en fòrums especialitzats i, actualment, cada proveïdor té la seva visió particular entre *multitenancy* o *containers*.

Pel que a les plataformes per a construir infraestructures SaaS, podem esmentar:

- Innomatic: entorn *open source* per a construir plataformes *SaaS multitenant*, productes i aplicacions empresarials en PHP. Amb aquesta, ISV i empreses poden construir aplicacions *multitenancy* preparades per a ser executades en el *cloud* i basades en els entorns Composer i Symfony2.

- Appserver.io: *multithreaded application server for PHP* per a crear aplicacions *multitenancy* basades en PHP.

- Drupal: CMS SaaS (openSaaS).

En models amb llicència Multihoster, SurPaaS /SaaS OpenBox, SaaS Maker (gratuïta per a desenvolupadors) i Lithium entre d'altres.

4.6. Altres serveis *cloud*

Ja hem esmentat anteriorment que hi ha una gran quantitat de XaaS i, en aquest apartat, revisarem els dos que tenen un alt impacte sobre els anteriors, o que poden ser comercialitzats per separat o juntament amb els anteriors:

Storage as a service

En aquest apartat, han sorgit diferents tipus de negoci, com ja s'ha esmentat anteriorment, com Amazon S3 o Dropbox/Drive (alguns autors engloben tot això com a *cloud storage*) que, malgrat que tenen com a premissa l'emmagatzematge, els models d'explotació són diferents. El primer ofereix

els serveis d'emmagatzematge a internet amb una interfície *web services* (REST, SOAP i BitTorrent), mentre que el segon o tercer són bàsicament explotats mitjançant SaaS.

No obstant això, i més enllà dels serveis i models de negoci utilitzats, una part important per a tot tipus de *cloud* és el programari necessari per a aconseguir aquest servei. En aquest sentit, descriurem els paquets *open source* que permeten generar un servei d'emmagatzematge distribuït, fiable i segur.

a) GlusterFS: utilitza FUSE (sistema d'arxius a l'espai de l'usuari) per a generar un VFS (*virtual file system*) i permet crear un sistema d'arxius de xarxa en un clúster a l'espai de l'usuari utilitzant sistemes d'arxius existents com ext3, ext4, xfs, etc. per a emmagatzemar les 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, distribuint els arxius per mitjà d'una col·lecció de subvolums, fent unitats majors que la unió d'espais petits i distribuïts, permetent la replicació. És per això que té redundància i una alta disponibilitat.

b) Ceph: la base d'aquest sistema d'emmagatzematge és *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 biblioteques que s'ofereixen a les aplicacions client, els usuaris poden aprofitar RADOS *Block Device* (RBD), RADOS *Gateway*, i també el sistema d'arxius Ceph. El 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 els sistemes de fitxers compatibles amb aquest estàndard poden utilitzar fàcilment el sistema d'emmagatzematge d'objectes de Ceph. Aquest sistema d'arxiu d'altes prestacions permet la lectura i escriptura parcial o completa, instantànies, assignacions de valor-clau de nivell de l'objecte i transaccions atòmiques.

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

d) Sheepdog: permet l'emmagatzematge d'objectes distribuïts i es caracteritza per la seva petita grandària, simplicitat i facilitat d'ús. Aquest sistema gestiona volums i serveis que, de forma intel·ligent, maneja els discos i nodes amb un escalat òptim que pot arribar a centenars o milers de dispositius. Sheepdog es pot annexar a MV de QEMU i els *targets* SCSI de Linux, té suport per Libvirt

i OpenStack, i pot interactuar amb HTTP Simple Storage. A nivell del sistema d'arxius permet fer instantànies, clonació i aprovisionament delegat i es pot annexar a MV i SO perquè s'executin en *bare-metal* (amb interfície iSCSI).

Network as a service

Freqüentment, s'associa aquest terme a altres serveis amb el *cloud* o també amb *communication-as-a-service* (CaaS), però en un sentit ampli aquest inclou la provisió d'un servei de xarxa virtual des de la xarxa específica cap a tercers i s'utilitzen protocols com OpenFlow. Entre el programari *open source* utilitzat per a proveir aquest servei en el *cloud* podem enumerar:

a) Floodlight: és un controlador OpenFlow d'altes prestacions en Java i amb llicència d'Apache. És un controlador SDN (*software defined networks*) obert que funciona amb dispositius (*switches*) físics i virtuals que interactuen per mitjà del protocol OpenFlow. Es pot escollir el protocol per a interactuar amb els altres dispositius remots de la xarxa (*switches*, *routers*, *virtualswitches* o *accesspoints*). L'OpenFlow permet explotar tota la funcionalitat d'aquest i controlar-lo remotament (taules de reexpedició de paquets, regles de flux, reexpedició o bloqueig de tràfic) i aprofitar les interfícies personalitzades i els llenguatges de seqüències d'ordres. Un dels promotors del projecte és Big Switch Networks, que ofereix el seu SDN versió *community* sense cost.

b) 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 una part del nucli d'OpenStack, Neutron pot ser considerat per la seva grandària 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. També es poden afegir serveis avançats mitjançant l'ús de connectors (*plugins*).

c) Open vSwitch: és un *switch* virtual multicapes *open source* (ASL) d'altes prestacions i funcionalitat estesa amb una àmplia gamma de característiques incloent 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, reenviament per mitjà del *kernel* de Linux i una base de dades de configuració transaccional. L'Open vSwitch pot funcionar completament a l'espai de l'usuari amb un mòdul de *kernel* o com un commutador basat en *kernel* que suporta múltiples tecnologies de virtualització, incloent Xen o XenServer, KVM i VirtualBox.

Resum

Com s'ha pogut observar en aquest mòdul, el món *cloud* gaudeix d'una perspectiva encoratjadora de futur i les tendències (informe de RighScale) són cap a estratègies *multi-clouds*, centrant-se en una integració (*clouds* híbrids) en què els entorns DevOps prenen força dins de l'ecosistema *cloud*, com ara PaaS, i amb actors molt definits dins de cada àmbit. A Eurostat es poden observar les tendències i com aquesta tecnologia està sent usada per les empreses i es poden extreure les tendències del mercat en el futur proper.

També cal tenir en compte les recomanacions de l'EC (*Opinion 05/2012 on Cloud Computing, Opinion 02/2015 on C-SIG Code of Conduct on Cloud Computing*) en relació amb la protecció de dades, i també l'Estratègia *Cloud* a Europa i el finançament d'aquesta tecnologia en les propostes en curs o que encara estan obertes.

Activitats

1. Analitzar dos hipervisors diferents i extreure conclusions dels seus avantatges i desavantatges.

2. Utilitzar dues plataformes IaaS públiques analitzant les seves prestacions, avantatges i desavantatges.

3. El mateix per PaaS. El mateix per SaaS (si pot ser dins del mateix àmbit de negoci o funcionalitat)

4. Comparar i extreure conclusions sobre els avantatges d'una plataforma PaaS orientada al desenvolupament d'aplicacions i un *cloud* IDE.

5. Fer un quadre comparatiu sobre les diferents opcions del *cloud*, avantatges i desavantatges. Proposar un model de negoci basat en una d'aquestes tenint en compte els condicionants, el model de monetització, la infraestructura, el «nínxol» i tots els elements que considereu necessaris i que poden condicionar un negoci basat en aquesta tecnologia.

Glossari

AJAX Asynchronous JavaScript And XML (cat. JavaScript asíncron i XML) Tècnica de desenvolupament web per a crear aplicacions interactives o RIA.

appliances Entorn que inclou tot el programari preconfigurat i llest per utilitzar, inclòs l'SO que es pot executar en un *bare metal* o un hipervisor.

bare-metal Expressió utilitzada per a referir-se a una màquina sense SO instal·lat.

CAPEX Cost d'inversió.

cloud computing (cat. computació/serveis/informàtica en el núvol) Proposta tecnològica que permet accedir a aplicacions o serveis remots en forma ubiqua per mitjà d'un navegador i que poden ser aprovisionats/alliberats sota demanda i en un temps reduït.

containers, jails (cat. contenidors, gàbies) Tècnica de virtualització del sistema operatiu que permet altes prestacions (exemple: Docker, LXC/LXD).

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

high performance computing (cat. còmput d'altes prestacions) Recursos dedicats a l'execució d'aplicacions complexes amb grans necessitats de potència de còmput.

hipervisor Capa de programari que s'ha de posar en l'SO de la màquina sobre la qual es desitgen virtualitzar els recursos.

IaaS (cat. infraestructura com a servei) Està a la capa inferior i és un mitjà per a proporcionar còmput, emmagatzematge i xarxa com a serveis estandarditzats a la xarxa.

infiniband Xarxa de comunicació d'altes prestacions (per exemple, 40 Gbits).

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

llicència *open source* Llicència que dona suport al codi *open source*. Exemples d'aquestes són GPL (GNU General Public License), ASL (Apache Software License), CC (Creative Commons), BSDL (Berkeley Software Distribution License).

multitenency (cat. multitinença) Mètode que permet executar una sola instància del programari en la infraestructura del proveïdor i serveix a múltiples clients, mantenint l'aïllament de les dades de cadascun d'aquests.

software open source (cat. programari de codi obert) Programari que disposa d'una llicència de manera que els usuaris poden estudiar, modificar i millorar el seu disseny mitjançant la disponibilitat del seu codi font.

OpenMPI, OpenMP Llibreries per a la programació d'aplicacions distribuïdes i de memòria compartida respectivament.

OPEX Costos d'operació.

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

pay-as-they-go, pay-as-they-grow, pay-per-user, pay-as-xx Models de monetització dels recursos i serveis en el *cloud*.

SaaS (cat. programari com a servei) Aplicació completa oferta com un servei, sota demanda, accessible per mitjà d'un navegador, en què l'usuari no té control sobre aquesta, eliminant la necessitat d'instal·lar qualsevol programari i sense preocupar-se del manteniment de l'aplicació.

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

SLA (*service level agreement*; cat. acord de nivell de servei) Acord escrit entre un proveïdor de servei i el seu client a fi de fixar el nivell acordat per a la qualitat d'aquest servei.

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

standby services Serveis preparats llestos per a posar-se en execució i que estaran disponibles en un temps mínim.

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

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

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

web services (cat. serveis web) Tecnologia que utilitza un conjunt de protocols i estàndards que serveixen per intercanviar dades entre les aplicacions, que són desenvolupades en llenguatges de programació diferents.

Bibliografia

S'han visitat tots els enllaços l'octubre de 2016.

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

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

[Cca] 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, núm. 3, 2011). <http://www.cloudbook.net/resources/stories/considering-cloud-appliances-for-private-cloud-deployments>

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

[Cis] Magic Quadrant for Cloud Infrastructure as a Service. L. Leong, G. Petri, B. Gill, M. Dorosh. 2016. ID: G00278620. Gartner. <https://www.gartner.com/doc/reprints?id=1-2g2o5fc&ct=150519>

[Cmh] Magic Quadrant for Cloud-Enabled Managed Hosting. D. Toombs, B. Gill, M. Dorosh. 2015. ID: G00269143. Gartner. <https://www.gartner.com/doc/reprints?id=1-2k50b8g&ct=150729&st=sb>

[Cvh] 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>

[Dcc] The NIST Definition of Cloud Computing. P. Mell and T. Grance. National Institute of Standards & Technology. Special Publication (pàg. 145-800). 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

[Iar] Iconos con licencia de uso libre. <http://www.iconarchive.com> <http://www.customicondesign.com> <http://icons8.com>

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

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

[Ona] OpenNebula. Open Cloud Reference Architecture. 2015. OpenNebulaSystems. <https://support.opennebula.pro/hc/en-us/article_attachments/202208405/opennebula-open_cloud_reference_architecture_rev1.0_20150421.pdf>

[Osc] A Guide to Open Source Cloud Computing Software. C. BryantJune. 2014. <http://www.tomsitpro.com/articles/open-source-cloud-computing-software,2-754.html>

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

[Ovf] Open Virtualization Format Specification. DSP0243. 2015. <http://www.dmtf.org/sites/default/files/standards/documents/dsp0243_2.1.1.pdf>

[SaO] SaaS. Apprenda. 2016. <https://apprenda.com/library/software-on-demand/>

[Saa] SaaS. J. Moreno Martín. 2016. Saasmanía. <http://www.saasmania.com>

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

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

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

[Vcc] 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, núm. 4, pàg. 50-58). https://doi.org/10.1145/1721654.1721672