Facultat d'Informàtica de Barcelona
Departament d'Arquitectura de Computadors
Centres de Processament de Dades

Activitat AiP

16. Eines d'orquestació

Gil Vázquez, Álvaro Antonio
Pirau, Calin-Constantin

Data: 08/04/2021

1 Orquestració

1.1 Definició d'orquestració

Una de les definicions d'orquestració que podem trobar al diccionari Cambridge ens diu el següent:

A careful arrangement of something to achieve a particular result, often in a way that is unfair or wrong.

1.2 Relació amb les TI

Tal i com posa a la definició anterior, es fa un arranjament cuidadós per tal d'obtenir cert resultat.

Fins fa uns anys, els enginyers informàtics ho tenien molt més cru per fer desplegaments de codi i serveis, a més d'haver de mantenir els mateixos, amb actualitzacions, manteniments periòdics, etc., per culpa de que ho havien de fer tot manualment. Tot això s'ha anat arreglant gràcies a les eines d'orquestració.

Actualment, aquestes eines fan que tots aquests processos puguin ser automatitzats, facilitant-nos la vida i fent que sigui més complicat que es puguin produir errors humans. Òbviament, encara podem equivocar-nos i configurar aquestes eines de manera incorrecta, però, un cop s'han configurat bé, podem estar segurs de que seran més fiables que deixar les tasques al factor humà.

Així doncs, podem dir que a les TI, l'orquestració està relacionada amb l'automatització de tasques, per tal d'assegurar-nos que arribem al resultat desitjat.

2 Eines d'orquestració

Com hem comentat al punt anterior, hi ha moltes eines que fan la feina d'automatitzar processos relacionats amb les TI. Nosaltres hem seleccionat les següents eines, ja que són conegudes i àmpliament utilitzades a les empreses actuals.

2.1 Ansible

Ansible és una eina orientada a la automatització de les TI. Gràcies a ella, es poden configurar sistemes, desplegar software, configurar desplegament continu i actualitzacions sense temps de downtime, entre d'altres tasques més avançades.

Els objectius d'Ansible són la simplicitat i la facilitat d'ús, a més de ser una eina segura, ja que utilitza OpenSSH per a fer el transport d'informació (encara que es permeten altres protocols com a alternativa).

Un dels avantatges d'Ansible respecte altres eines és que és capaç de gestionar les màquines sense necessitat d'haver d'instal·lar software adicional en aquestes, és a dir, no es necessiten els típics "agents" o daemons per tal de fer la connexió. Un altre dels avantatges és la fàcil connexió amb sistemes centralitzats d'autentificació com Kerberos o LDAP.

Objectes d'Ansible

Hi ha una sèrie de conceptes i objectes bàsics que s'han d'entendre per tal d'utilitzar Ansible. Són els següents:

Control node

Qualsevol màquina que tingui Ansible instal·lat és un control node. Per tal d'executar una comanda o un playbook d'Ansible, només cal invocar `ansible` o `ansible-playbook` desde un control node. Es poden utilitzar laptops, ordinadors d'escriptori, servidors, etc com a control nodes (podem tenir més d'un), sempre i quan no executin Windows.

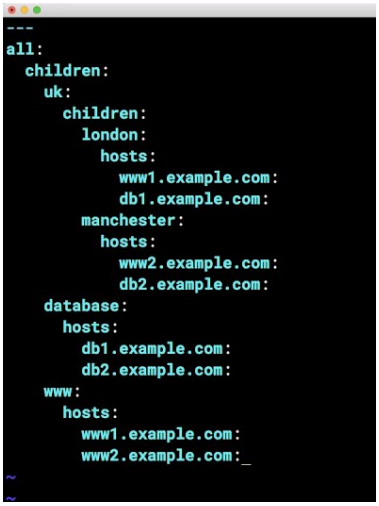
Managed nodes

Són els dispositius de xarxa (o servidors) que gestionem amb Ansible. Ansible no s'instal·la en aquestes màquines, ja que la connexió a elles es fa mitjançant SSH.

Inventory

És una llista de managed nodes. Normalment es guarden a un fitxer anomenat "hostfile". Dins de l'inventari es poden especificar les adreces IP, hostname, etc, dels hosts. També es poden organitzar grups i subgrups, de manera que sigui més fàcil escalar.

Un exemple de hostfile seria el següent:



```
---
all:
  children:
    uk:
      children:
        london:
          hosts:
            www1.example.com:
            db1.example.com:
        manchester:
          hosts:
            www2.example.com:
            db2.example.com:
    database:
      hosts:
        db1.example.com:
        db2.example.com:
    www:
      hosts:
        www1.example.com:
        www2.example.com:_
~
~
```

Collections

Les col·leccions són el format en el que es distribueix contingut d'Ansible, com podrien ser playbooks, rols, mòduls, plugins, etc. Per exemple, podem veure una de les col·leccions de AWS.

Amazon.Aws

Collection version 1.4.1

Plugin Index

These are the plugins in the amazon.aws collection

Callback Plugins

- aws_resource_actions – summarizes all "resource:actions" completed

Inventory Plugins

- aws_ec2 – EC2 inventory source
- aws_rds – rds instance source

Modules

Són les unitats de codi que Ansible executa. Cada mòdul té un ús particular, des de administrar usuaris en un tipus de base de dades, gestionar interfaces de xarxa en un tipus de dispositius de xarxa específic, etc. Podem veure com la col·lecció de AWS anterior també conté alguns mòduls:

Modules

- aws_az_info – Gather information about availability zones in AWS.
- aws_caller_info – Get information about the user and account being used to make AWS calls.
- aws_s3 – manage objects in S3.
- cloudformation – Create or delete an AWS CloudFormation stack
- cloudformation_info – Obtain information about an AWS CloudFormation stack
- ec2 – create, terminate, start or stop an instance in ec2
- ec2_ami – Create or destroy an image (AMI) in ec2

Tasks

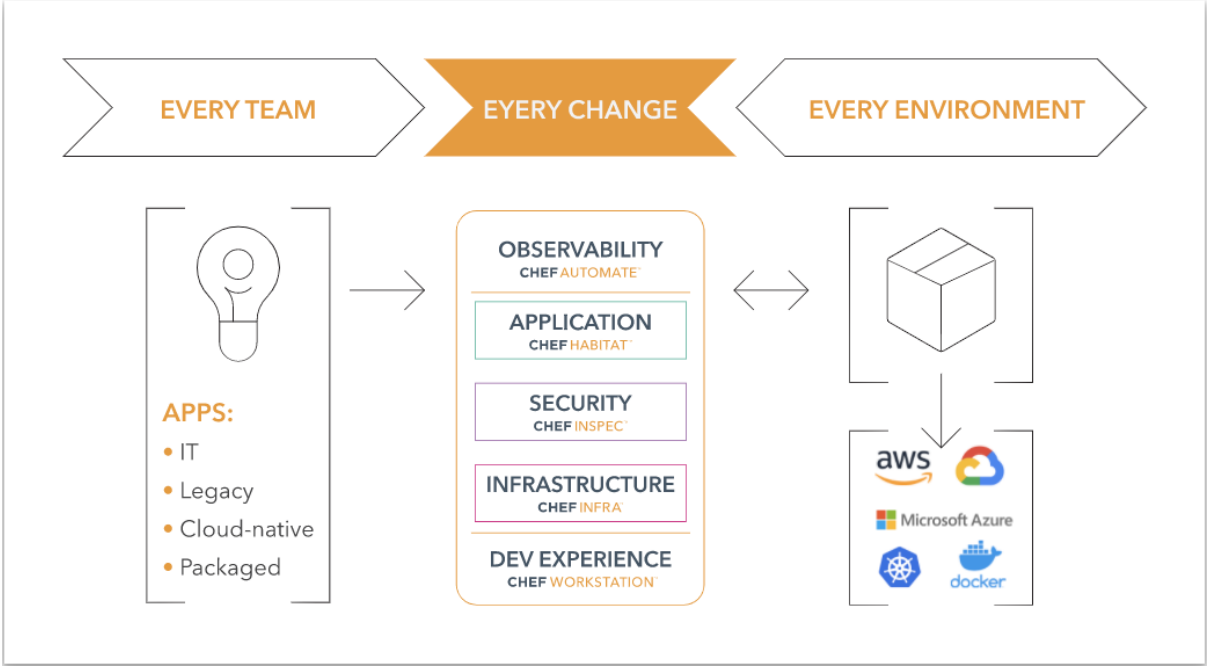
Són les unitats més petites d'una acció en Ansible. Es pot utilitzar una tasca amb una comanda ad-hoc.

Playbooks

Són llistes ordenades de tasques, de manera que que es poden utilitzar repetidament. S'escriuen en YAML, així que són fàcils de llegir, escriure, etc.

2.2 Chef

Chef és una empresa d'automatització. Inicialment, la seva idea era juntar els desenvolupadors amb els administradors de sistemes gràcies al seu producte estrella, Chef Infra. Amb els anys, la seva idea d'automatització s'ha anat expandint, de manera que ara tenen diferents productes, tots orientats a l'automatització.



Chef Infra

És una plataforma que transforma l'infrastructura a codi (IaC). S'utilitza per automatitzar com es configura l'infrastructura, com es desplega i es gestiona a través de la xarxa, independentment del tamany i de si estem operant al cloud, en un entorn on-premises o en un entorn híbrid.

En el següent diagrama es veu com es desenvolupa, testeja i desplega el codi de Chef Infra:

Chef workstation

Com hem vist al diagrama, el primer que s'utilitza es Chew Workstation, per tal de crear els "cookbooks" i administrar l'infrastructura.

Quan s'escriu el codi, s'utilitzen resources per descriure l'infrastructura. Un resource correspon a una peça d'infrastructura, que pot estar definida en un fitxer, un template, etc. Al igual que en Kubernetes (que veurem més endavant), el que es fem es definir l'estat desitjat al que volem arribar, però no com ho ha de fer Chef per aconseguir-ho, ja que això ho gestiona ell mateix. A més, Chef Infra ja proveeix de resources llestos per utilitzar, hi ha una àmplia comunitat, etc.

Dos dels objectes més importants són les recipes i els cookbooks. Una recipe consisteix en un fitxer que agrupa diferents resources, com per exemple, els diferents elements que es necessiten per montar un servidor web, un servidor de dades o un balancejador de càrrega. Els cookbooks el que fan és organitzar les recipes, aportant estructura a aquestes.

Una de les diferències respecte a Ansible, per exemple, és que els hosts de Chef, és a dir, els clients, han d'instal·lar un software específic per a poder gestionar-los, el Chef Infra Client. Aquest software va demanant al Chef Infra Server si hi ha hagut algun canvi a les configuracions i, en cas positiu, les demana i executa els processos necessaris per arribar a l'estat desitjat.

2.3 Puppet

Puppet és una altre eina d'automatització i orquestració. Una de les diferències que té respecte a les eines anteriors és que es pot configurar el sistema seguit una arquitectura client-servidor (mitjançant els Puppet agents i Puppet master, semblant a Chef) o seguint una arquitectura stand-alone (mitjançant Puppet apply).

Catalogs

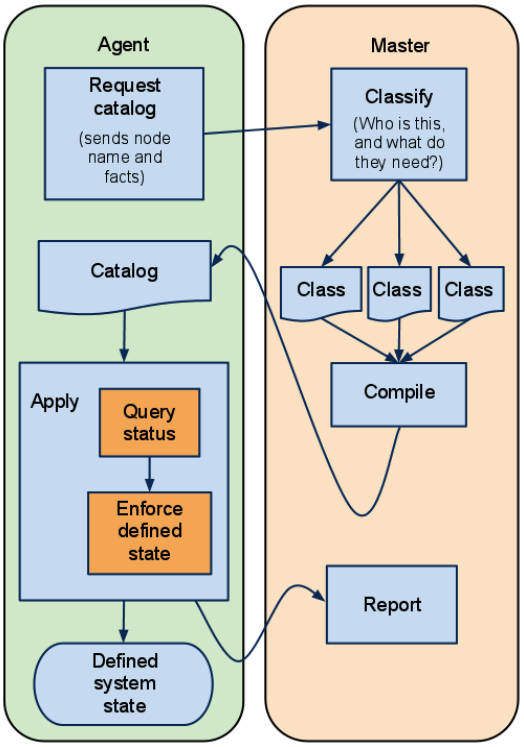
Un catalog és un document que descriu l'estat desitjat del sistema per a una màquina específica (semblant a Chef i Kubernetes, però per una màquina). Dins del catàleg, estan llistats tots els recursos que s'han de gestionar, a més de les dependències entre aquests recursos. Per tal de configurar un sistema, Puppet executa dos passos, compilar el catàleg i aplicar-lo.

Arquitectura client-servidor

Quan s'utilitza aquesta arquitectura, tenim un esquema similar al de Chef, on hi ha un servidor que controla la informació de les configuracions, i cada node agent demana els seus catalogs al master.

Els nodes agents tenen instal·lada l'aplicació Puppet agent, normalment en background. A més, hi ha un o més servidors amb l'aplicació Puppet Server, que és l'aplicació dels Puppet masters.

De manera periòdica, els agents demanen els catàlegs al master, aquest els compila i els retorna al agents. Un cop l'agent rep el catàleg, l'agent l'aplica comprovant cada recurs que hi ha descrit. Si troba algun recurs que no té l'estat desitjat, fa els canvis necessaris per a corregir-ho o reportar-ho (en cas de que hi hagi una política de no canvis). Un cop realitzat el procés, es genera un informe que s'envia al master.



Arquitectura stand-alone

De manera alternativa, Puppet es pot executar de manera stand-alone, és a dir, cada node té una còpia de la seva informació de configuració i compila el seu propi catàleg.

En aquesta arquitectura, els nodes executen l'aplicació Puppet apply, normalment de manera periòdica amb cron jobs, encara que també es pot executar a demanda per tal de tenir una configuració inicial.

El procés d'aplicar el catàleg és igual que el de l'arquitectura client-servidor, amb la diferència de que cada node compila el seu catàleg, l'aplica i un cop ha generat el report, el guarda en disc.

2.4 Saltstack

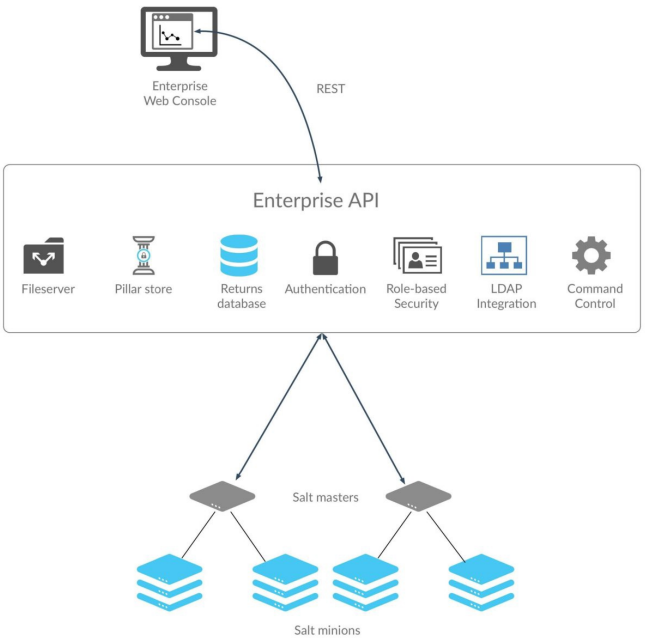
Saltstack és una altre eina d'automatització i orquestració. És semblant a Ansible, ja que la seva funció core és l'execució de comandes en sistemes remots.

Execució remota

Al igual que amb Chef i amb Puppet, hi ha un servidor master (Salt master) i diferents clients (Salt minions). Les instruccions de Salt s'executen a través de la línia de comandes al Salt master, indicant quins minions hauran de fer la tasca.

Gestió de configuracions

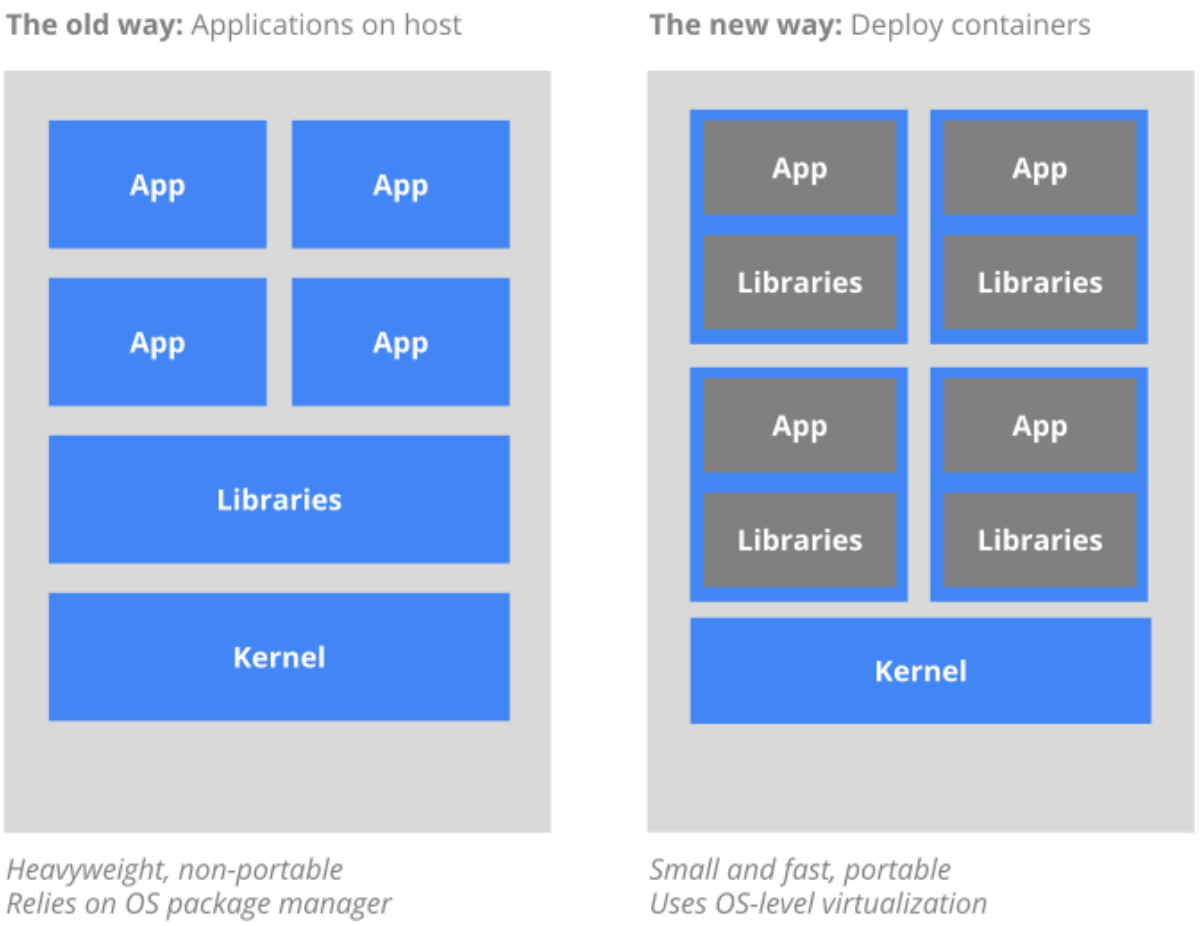
Semblant a Chef i a Puppet, Salt utilitza un framework en el qual es defineixen els estats desitjats, de manera que, en cas de que no es compleixi aquest estat, es fan els canvis necessaris per arribar-hi.



2.5 Kubernetes

A nivell de contenidors, Kubernetes és l'eina d'orquestració preferida per els usuaris. Creat per Google utilitzant Golang, s'utilitza especialment per

Per què Kubernetes utilitza contenidors? L'explicació pot ser tant senzilla com complexa, però es pot resumir en que els contenidors són aïllats del servidor en el qual resideixen a nivell de llibreries, fent així que el programari que contenen sigui 100% funcional sense dependre del que contingui el servidor. La següent imatge il·lustra aquesta explicació:



Podem també resumir els beneficis d'utilitzar contenidors de la següent manera:

- **Àgil creació i desplegament d'aplicacions:** Major facilitat i eficiència en crear imatges de contenidor en comptes de màquines virtuals.
- **Desenvolupament, integració i desplegament continu:** Permet que la imatge de contenidor es construeixi i desplegui de forma freqüent i fiable, facilitant els rollbacks ja que la imatge és immutable.
- **Separació de tasques entre Dev i Ops:** Es poden crear imatges de contenidor a l'hora de compilar i no a l'hora de desplegar, desacoblant l'aplicació de la infraestructura.
- **Observabilitat:** No solament es presenta la informació i mètriques de sistema operatiu, sinó la salut de l'aplicació i altres senyals.
- **Consistència entre els entorns de desenvolupament, proves i producció:** L'aplicació funciona igual en un laptop i en el núvol.

- **Portabilitat entre núvols i distribucions:** Funciona amb Ubuntu, RHEL, CoreOS, un datacenter físic, Google Kubernetes Engine i més.
- **Administració centrada en l'aplicació:** Eleva el nivell d'abstracció de sistema operatiu i el maquinari virtualitzat a l'aplicació que funciona en un sistema amb recursos lògics
- **Microserveis distribuïts, elàstics, alliberats i dèbilment acoblats:** Les aplicacions es separen en peces petites i independents que poden ser desplegades i administrades de forma dinàmica, i no com una aplicació monolítica que opera en una sola màquina de gran capacitat.

Objectes de Kubernetes

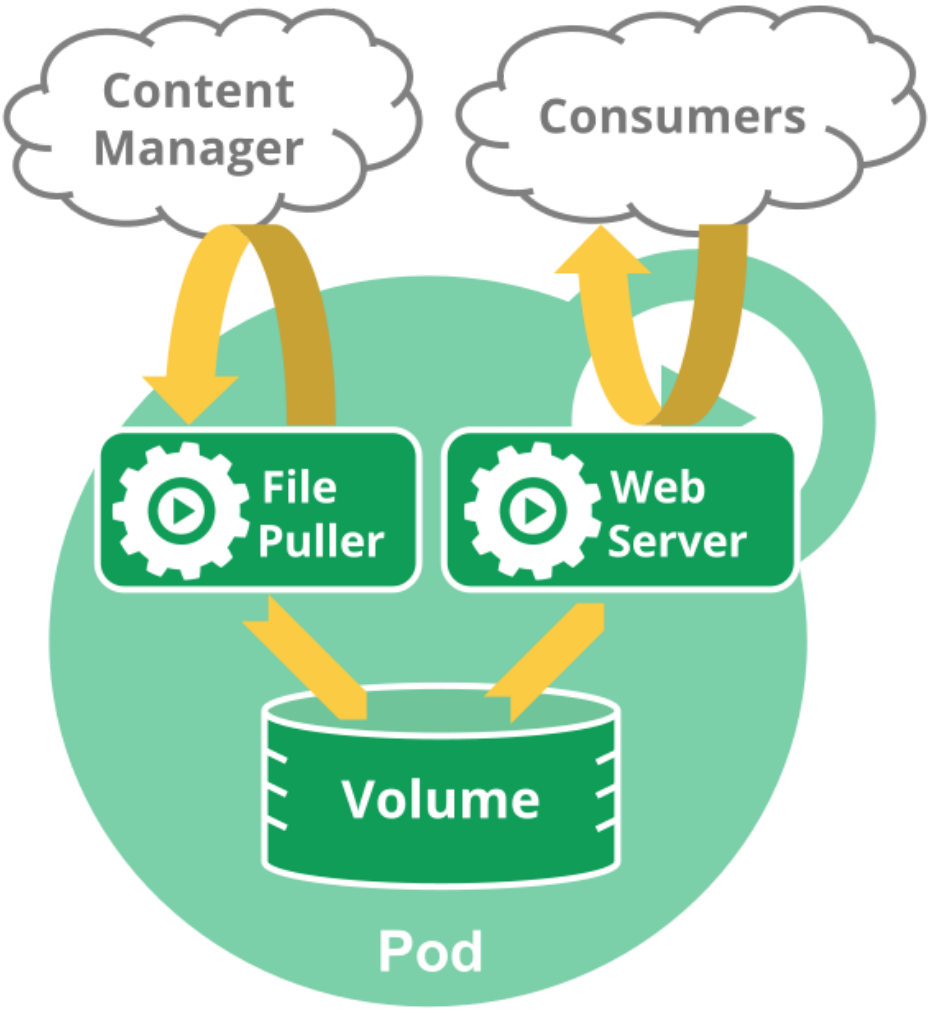
Kubernetes en sí està composat de diferent objectes, els quals representen l'estat del sistema. Poden haver-hi des de aplicacions conteneritzades desplegades i càrregues de treball, els seus recursos de xarxa i emmagatzematge associats fins a informació addicional sobre el que el clúster està fent en un moment donat. Detalls sobre els objectes es mostren a continuació:

Pods

Un pod es compon per un grup d'un o més contenidors. També conté emmagatzemament i recursos de xarxa, a més d'una especificació de com executar els containers.

Els pods estan dissenyats per donar suport a múltiples processos que cooperen entre ells (com a contenidors) que formen una unitat de servei cohesionada. Els contenidors d'un Pod es co-localitzen i programen automàticament a la mateixa màquina física o virtual del clúster. Els contenidors poden compartir recursos i dependències, comunicar-se entre ells i coordinar quan i com s'acaben.

Per exemple, és possible que es tingui un contenidor que actua com a servidor web per a fitxers en un volum compartit i un altre contenidor separat que actualitzi aquests fitxers des d'una font remota, tal com es mostra en el diagrama següent:



Services

Els Services de Kubernetes són una abstracció lògica per a un grup desplegat de pods en un clúster (els quals fan la mateixa funció).

Com que els pods són efímers, un servei permet assignar un nom i una adreça IP única (clusterIP) a un grup de pods, que proporcionen funcions específiques (serveis web, processament d'imatges, etc.). Mentre el servei executi aquesta adreça IP, no canviarà. Els serveis també defineixen polítiques per al seu accés.

Aquests Servers connecten un conjunt de pods a un nom i adreça IP de servei abstracte. Els serveis ofereixen descobriment i enrutament entre pods. Per exemple, els serveis connecten un frontend al seu backend, cadascun dels quals s'executa en desplegaments separats en un clúster. Els serveis utilitzen etiquetes i selectors per fer coincidir els pods amb altres aplicacions. Els atributs bàsics d'un servei de Kubernetes són:

- Un selector d'etiquetes que localitza els pods
- L'adreça IP del clusterIP i el número de port assignat
- Definicions de ports
- Assignació opcional de ports entrants a un targetPort

També podem dir que hi ha diferents tipus de Services:
- ClusterIP. Exposa un servei que només és accessible des del clúster.
- NodePort. Exposa un servei mitjançant un port estàtic a la IP de cada node.
- LoadBalancer. Exposa el servei mitjançant el Load Balancer del proveïdor de Cloud.
- ExternalName. Assigna un servei a un camp de nom extern predefinit retornant un valor per al registre CNAME.

Volumes

A Kubernetes, es pot considerar un Volume com un directori accessible als contenidors d'un pod. Tenim diferents tipus de Volumes a Kubernetes i el tipus defineix com es crea i el seu contingut.

Hi ha molts tipus de Volumes, dels quals se'n poden destacar:
- **emptyDir:** és un tipus de volum que es crea quan un Pod s'assigna per primera vegada a un node. Es manté actiu mentre el Pod funcioni en aquest node. El volum inicialment està buit i els contenidors del pod poden llegir i escriure els fitxers al volum emptyDir. Un cop eliminat el Pod del node, les dades del directori buit s'esborren.
- **hostPath:** aquest tipus de volum munta un fitxer o directori des del sistema de fitxers del node host al vostre pod.
- **persistentVolumeClaim:** s'utilitza el persistentVolumeClaim per muntar un objecte PersistentVolume en un pod. Els PersistentVolume són una manera per als usuaris de "reclamar" emmagatzematge persistent provinent d'entorns externs (com ara un GCE PersistentDisk o un volum iSCSI) sense conèixer els detalls de l'entorn concret.

Persistent Volume i Persistent Volume Claim

Persistent Volume (PV): és una peça d'emmagatzematge connectat a xarxa que l'administrador ha subministrat. És un recurs del clúster que és independent de qualsevol pod individual que utilitzi el PV.

Persistent Volume Claim (PVC): l'emmagatzematge sol·licitat en Kubernetes per a pods concrets es coneix com a PVC. L'usuari no necessita conèixer el subministrament subjacent. Les PVC s'han de crear al mateix Namespace on es crea el pod.

Namespaces

Els Namespaces són una manera d'organitzar clústers en sub-clusters virtuals; poden ser útils quan diferents equips o projectes comparteixen un clúster de Kubernetes. Es poden admetre qualsevol nombre d'espais de noms dins d'un clúster, separats lògicament dels altres però amb la capacitat de comunicar-se entre ells. Els espais de noms no es poden anidar entre si.

Kubernetes inclou tres Namespaces per defecte. Ells són:
- **default**: com el seu nom indica, aquest és el Namespace al qual es fa referència per defecte per a cada ordre de Kubernetes i on es troben tots els recursos de Kubernetes. Fins que no es creen espais de noms nous, tot el clúster resideix a "default".
- **kube-system**: s'utilitza per als components de Kubernetes i no s'ha de modificar.
- **kube-public**: s'utilitza per a recursos públics. No és recomanable per als usuaris.

ReplicaSets

El propòsit d'un ReplicaSet és mantenir un conjunt estable de pods replicats que s'executen en un moment donat. Com a tal, sovint s'utilitza per garantir la disponibilitat d'un nombre especificat de pods idèntics.

Es defineix un ReplicaSet amb camps, inclòs un selector que especifica com identificar els Pods que pot adquirir, una sèrie de rèpliques que indiquen quants Pods hauria de mantenir i una plantilla de Pod que especifica les dades dels nous Pods que hauria de crear per satisfer el nombre de rèpliques. Un ReplicaSet compleix el seu propòsit creant i eliminant Pods segons sigui necessari per arribar al nombre desitjat. Quan un ReplicaSet necessita crear nous Pods, fa servir la seva plantilla de Pod.

No obstant això, un Deployment és un concepte de nivell superior que gestiona ReplicaSets i proporciona actualitzacions declaratives a Pods, juntament amb moltes altres funcions útils. Per tant, es recomana que s'utilitzin Deployments en lloc d'utilitzar directament ReplicaSets, tret que es necessitaria una orquestració d'actualitzacions personalitzada.

Deployments

Un Deployment s'utilitza per dir a Kubernetes com crear o modificar instàncies dels pods que contenen una aplicació en concret. Els Deployments poden escalar el nombre de pods de rèplica, permetre el llançament del codi actualitzat de manera controlada o tornar a una versió de desplegament anterior, si cal.

Gràcies als Deployments, Kubernetes automatitza les funcions de treball i repetitives que intervenen en els desplegaments, les escalacions i les actualitzacions d'aplicacions en producció.

Com que el Kubernetes deployment controller sempre controla la salut de pods i nodes, pot substituir un pod fallit o ignorar els nodes, substituint aquests pods per garantir la continuïtat de les aplicacions crítiques.

Els Deployments automatitzen el llançament d'instàncies de pod i asseguren que s'executin tal com es defineix a tots els nodes del clúster. Més automatització es tradueix en desplegaments més ràpids i amb menys errors.

ConfigMaps i Secrets

Un ConfigMap és un objecte de Kubernetes utilitzat per emmagatzemar dades no confidencials en el format clau-valor. Els Pods poden utilitzar els ConfigMaps com a variables d'entorn, arguments de la línia de comandaments o com a fitxers de configuració en un Volumen.

Un Secret és similar al ConfigMap, però sí permet encriptar les dades que es guarden, per tant es pot utilitzar per coses confidencials.

StatefulSets

StatefulSet és l'objecte de Kubernetes que s'utilitza per gestionar aplicacions “stateful”.

Aquest objecte gestiona el desplegament i l’escala d’un conjunt de pods i proporciona garanties sobre l’ordenació i la singularitat d’aquests pods.

De la mateixa manera que un Deployment, un StatefulSet gestiona pods que es basen en una especificació de contenidor idèntica. Tot i així té una diferencia, i és que un StatefulSet manté una identitat “sticky” per a cadascun dels seus pods. Aquests pods es creen a partir de la mateixa especificació, però no són intercanviables: cadascun té un identificador persistent que manté a través de qualsevol actualització que pugui tenir.

DaemonSets

Un DaemonSet garanteix que tots (o alguns) nodes executin la còpia d’un Pod que s’hagi especificat. A mesura que s’afegeixen nodes al clúster, s’hi afegeixen els Pods dels DaemonSets. De la mateixa manera, a mesura que s'eliminen els nodes del clúster, aquests pods són eliminats. Si s’esborra un DaemonSet, els Pods creats per el mateix es netejaran.

Jobs

Un Job crea un o més pods i continua amb l'execució dels pods fins que finalitzi amb èxit un nombre especificat dels mateixos. A mesura que els pods es completen correctament, el Job fa un seguiment de les finalitzacions correctes. Quan s'arriba a un nombre especificat de finalitzacions bones, la feina (Job) està completada. Si s’elimina un Job, s’esborren també els pods que el mateix va crear.

Un cas senzill d’exemple seria crear un objecte Job per executar un Pod de manera fiable fins a la seva finalització. L'objecte Job iniciaria un nou Pod si el primer Pod falla o s'elimina (per exemple, a causa d'un error de maquinaria del node o d'un reinici del node).

També es pot utilitzar un Job per executar diversos pods en paral·lel.

2.6 Openshift

Openshift és un sistema PaaS que utilitza Kubernetes com a plataforma per el desplegaments dels seus serveis. Es pot dir que és un orquestrador de contenidors, però en aquest cas amb la comoditat que comporta ser PaaS.

Aquesta comoditat ve de la mà del Openshift Catalog, que conté un abanic molt gran de serveis “prefabricats” que estan preparats per ser instal·lats automàticament en el clúster de Kubernetes d’Openshift. L’usuari en aquest cas no s’ha de preocupar de res de l’infrastructura de Kubernetes, ja que Openshift ja sap què ha de fer per a que el servei que l’usuari vol instalar funcioni.

També cal destacar que Openshift inclou coses com una interfície d'usuari per veure tots els objectes de Kubernetes que l'usuari tingui a la seva disposició, a més de la integració completa amb les comandes "oc" i "kubectl".

Finalment, gràcies a Red Hat, Openshift proporciona una capa extra de seguretat per a l'usuari o organització que vulgui contractar el servei, cosa que li dona un extra comparat amb els seus competidors.

2.7 Helm

Helm és un administrador de paquets per Kubernetes que facilita prendre aplicacions i serveis que són altament repetibles i implementar-se en un clúster típic de Kubernetes. En resum, es pot dir que Helm és un orquestrador de les configuracions dels objectes de Kubernetes.

Helm és una eina per gestionar uns paquets Kubernetes anomenats charts. Amb això, podem dir que a Helm hi ha 3 conceptes importants a tenir en compte:

- El *chart* és un conjunt d'informació necessària per crear una instància d'una aplicació de Kubernetes.
- La *config* conté informació de configuració que es pot combinar en un gràfic empaquetat per crear un objecte alliberable.
- Una versió és una instància en execució d'un gràfic, combinada amb una configuració específica.

Hi ha dos parts principals de Helm:

- El Helm Client és un client de línia d'ordres per als usuaris finals. El Client és responsable del següent:
  - Desenvolupament de charts locals
  - Gestió de repositoris
  - Gestió de versions
  - Interfície amb la Helm Library
  - L'enviament de *charts* a instal·lar
  - Sol·licitud d'actualització o desinstal·lació de versions existents

- La Biblioteca Helm proporciona la lògica per executar totes les operacions Helm. S'interface amb el servidor de l'API Kubernetes i proporciona la capacitat següent:

Combinant un gràfic i una configuració per crear una versió
Instal·lant gràfics a Kubernetes i proporcionant l'objecte de llançament posterior
Actualitzant i desinstal·lant gràfics interactuant amb Kubernetes
La biblioteca independent Helm encapsula la lògica Helm perquè els clients puguin aprofitar-la.

El Client i la Library de Helm s'han escrit en el llenguatge de programació Go.

La Helm Library utilitza la llibreria client de Kubernetes per comunicar-se amb ell. Actualment, aquesta llibreria utilitza REST + JSON. Emmagatzema informació a Secrets ubicats a l'interior de Kubernetes. No necessita la seva pròpia base de dades.

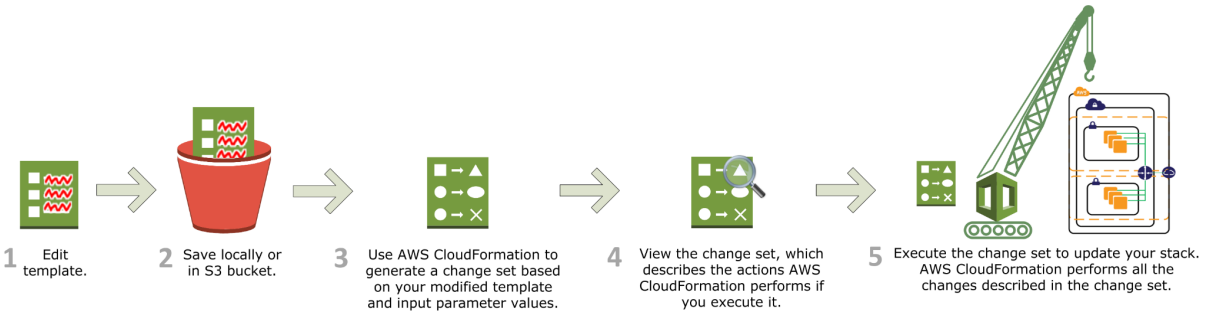
Els fitxers de configuració, sempre que sigui possible, s'escriuen en YAML.

2.8 CloudFormation

CloudFormation és una eina IaC (Infrastructure as Code) que ajuda a automatitzar i gestionar els desplegaments d'AWS. Dit això, CloudFormation està dissenyada específicament per Amazon Web Services, ja que és un dels serveis que la companyia proporciona a l'usuari.

CloudFormation adopta un enfocament declaratiu de la configuració, és a dir, l'usuari li explica com vol que es vegi l'entorn i i l'eina hi troba el camí per fer-ho possible.

Durant el procés de configuració, CloudFormation gestiona automàticament les dependències entre els recursos. Per tant, no cal que s'especifiqui l'ordre en què es creen, actualitzen o se suprimeixen els recursos. CloudFormation determina automàticament la seqüència correcta d'accions per crear el vostre entorn. A continuació s'ilustra un exemple de funcionament de CloudFormation:



La combinació d'CloudFormation amb eines de "configuration management" permet automatitzar la configuració, desplegament i gestió dels recursos de Cloud i el programari que s'executa en ells. Això millora considerablement l'eficiència i la coherència de les implementacions d'infraestructura. Chef i Puppet són els programes de "configuration management" més populars que s'utilitzen amb CloudFormation, i AWS té un producte anomenat OpsWorks que proporciona instàncies gestionades d'aquestes eines.

2.9 Jenkins

Jenkins ofereix una manera senzilla de configurar un entorn CI/CD per a gairebé qualsevol combinació d'idiomes i repositoris de codi font. Aquest és un sistema open-source que té un gran background de la comunitat, o sigui, casi tot el que és capaç de fer ha sigut gràcies als usuaris que li han donat suport. Podem dir que Jenkins es composa de 3 parts fonamentals, a part de moltes altres, que són les següents:

Plugins

Sense els plugins Jenkins no seria res avui en dia. Els plugins són característiques extra que s'instal·len dins la instància de Jenkins, per poder fer més coses amb aquesta eina. Des de plugins que permeten separar usuaris per carpetes, afegir configuracions de Active Directory fins a conectar amb un clúster de Kubernetes. Uns dels més utilitzats serien:

- **Blue Ocean:** Aquest plugin dona al usuari una nova interfície gràfica, que és molt més bonica que la original de Jenkins a més de ser molt més intuitiva i fàcil per al UI/UX.
- **Kubernetes:** Amb el plugin Kubernetes el que s'aconsegueix és connectar amb un clúster Kubernetes per tal de poder crear agents dinàmics amb ell, els quals seran explicats en l'apartat d'Agents.
- **Active Directory:** Si es necessita que hi hagi els usuaris d'un grup d'Active Directory això es fa possible gràcies a aquest plugin. Funciona utilitzant el protocol LDAP principalment.
- **Role-based Authorization Strategy:** Per poder distinguir els usuaris de Jenkins com a administradors, editors o "viewers", aquest plugin és l'idoni per la tasca.
- **Folder-based Authorization Strategy:** Si a més de distinguir usuaris per el seu rol volem separar els mateixos per grups o "equpis", el Folder-based Authorization Strategy és el plugin a utilitzar.
- **GitHub:** Plugin que permet en les Pipelines que es pugui fer "git clone" de repositoris de GitHub. També cal destacar que hi ha plugins que serveixen per altres programaris per l'estil, com Bitbucket o GitLab.

Pipelines

Les pipelines són els objectes estrella de Jenkins. Fàcilment extensibles gràcies als plugins, aquests poden fer qualsevol tipus de tasca que l'usuari vulgui per el development de la seva aplicació. Aquestes pipelines utilitzen els anomenats Jenkinsfile, que són fitxers basats en Groovy que contenen les accions que l'usuari vol que la pipeline faci. Aquest fitxer pot estar dins del repositori on s'ubica l'aplicació o es pot escriure directament a la pipeline quan es crea en Jenkins.

També cal dir que s'ha d'indicar en quin agent es volen executar les tasques adients. Es poden escollir més d'un tant estàtic com dinàmic, a més de poder-se crear i destruir els dinàmics en el temps de l'execució de la pipeline.

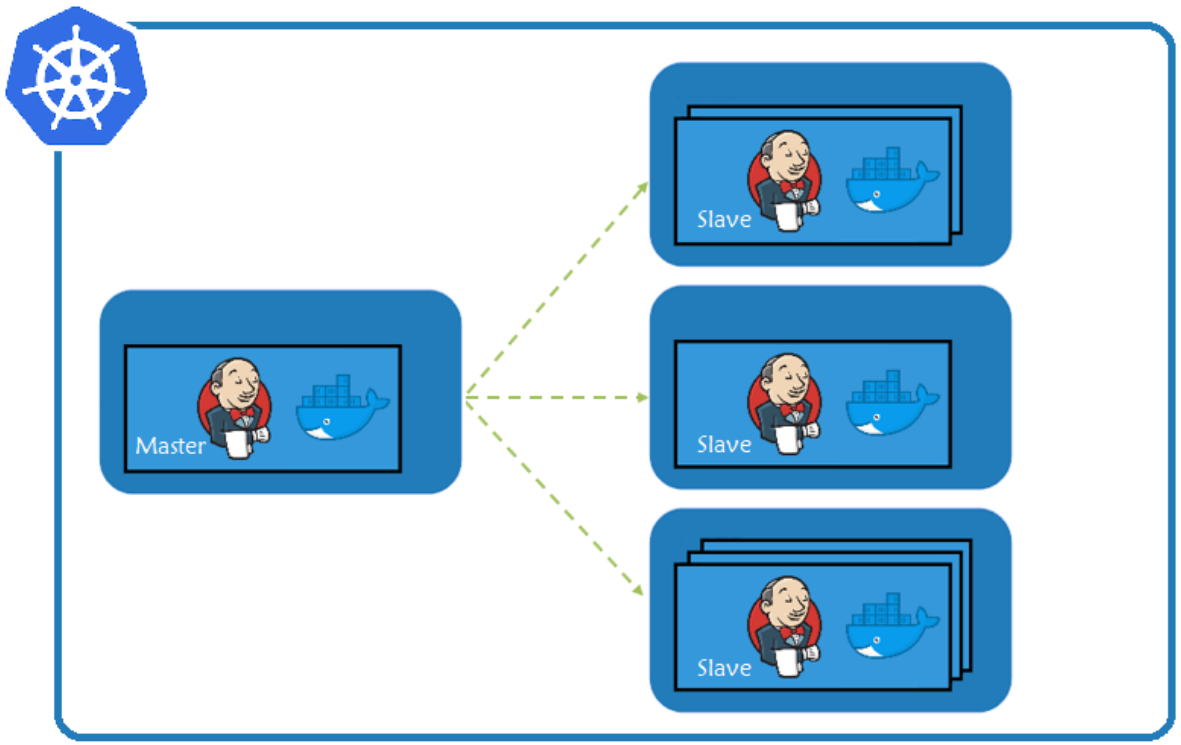
Blue Ocean dóna la possibilitat de poder "escriure" el Jenkinsfile amb una llista de possibles accions que es puguin fer mostrada en la seva interfície gràfica. En aquest cas la pipeline és automàticament escrita per Blue Ocean.

Agents

Els agents de Jenkins són els responsables d'executar les tasques proposades en les pipelines. L'usuari pot ser o no responsable de en quin o quins agents s'executen les pipelines. Això vol dir que els agents poden tenir "labels", les quals diuen quines característiques té l'agent, com podrien ser el Sistema Operatiu, el compilador que pugui tenir o les seu hardware en concret, entre d'altres.

També s'ha de mencionar que hi ha dos tipus d'agents en Jenkins:

- **Agents Estàtics:** aquest tipus d'agent està disponible 24/7, menys quan està sent utilitzat per un nombre limitat d'usuaris al mateix temps.Té varies funcionalitats disponibles, com compiladors, comandes com per exemple "git", "curl", entre d'altres. Podem dir que són els agents més complets dels dos. El problema que tenen és que al estar engegats tot el temps són una opció més cara, perquè s'ha de pagar un servidor sencer o parcial, el qual no es paga per el que s'utilitzi, si no per el temps que s'estigui "alquilant".
- **Agents Dinàmics:** aquests agents són totalment oposats als estàtics. Tenen les funcions molt més limitades, però tenen l'avantatge de que poden ser executats en paral·lel amb altres agents, per tal de compensar la falta de característiques. Tenen la peculiaritat de que no es paguen mensualitats per ells, sinó que es paguen només per els segons o minuts que existeixen, és a dir, per el temps en el que la pipeline hagi necessitat per executar les seves tasques, ja que n'és la responsable de fer que l'agent es creï a més de que es destrueixi quan les tasques s'acaben. Això li dona un clar avantatge en quant a preu comparat amb els estàtics, ja que es paga per utilització, no per posessió. Dos clars plugins que poden fer possible els agents dinàmics són els de Docker i Kubernetes. A continuació es pot veure un diagrama de com funcionaria el plugin de Kubernetes en aquesta situació, en el que el Master (instància de Jenkins) crea els Slaves (agents dinàmics):



4 Conclusions

Amb aquest document podem veure clarament que hi ha molts tipus d'orquestració en el món de les TI. Des d'orquestració de contenidors, configuracions d'infrastructura, CI/CD, automatitzacions, entre molts altres.

Sabent tot això, per a que un equip de sistemes o DevOps sapigui quines eines utilitzar per a cada cas hauria de fer un estudi exhaustiu de cadascuna per poder escollir la millor, ja que cada cas és diferent, i una eina podria servir més per un cas que per un altre. En el cas de CI/CD, és possible que a un equip li vingui millor Jenkins que Bamboo, ja que Jenkins és open source, té bastant suport gràcies a la comunitat i és un servei gratuït, i per les poques pipelines que necessita fer l'equip no els hi fa falta més. També podria ser que hi hagi un altre equip que valori més tenir suport directe de l'empresa que hagi creat l'eina, que seria el cas de Bamboo amb Atlassian, i així poder estar segur de que no hi haurien problemes greus mai, gràcies a l'ajuda que proporcionaria l'empresa.

Tot això es pot aplicar amb totes les subcategories d'orquestració. També podem dir amb bastanta seguretat de que l'orquestració és el futur, ja que ajuda molt a estalviar accions repetitives i complexes que podrien donar lloc a molts errors humans i que podrien ser assignades a un ordinador per aconseguir així una millora en l'estabilitat dels productes i/o serveis que proporciona un equip o empresa.

5 Bibliografia

Dictionary.cambridge.org. n.d. *orchestration*. [online] Available at: <https://dictionary.cambridge.org/es/diccionario/ingles/orchestration>.

n.d. *What is orchestration?*. [online] Available at: <https://www.redhat.com/en/topics/automation/what-is-orchestration>.

Docs.ansible.com. 2021. *Ansible concepts — Ansible Documentation*. [online] Available at: <https://docs.ansible.com/ansible/latest/user_guide/basic_concepts.html#basic-concepts>.

Docs.ansible.com. 2021. *Getting Started — Ansible Documentation*. [online] Available at: <https://docs.ansible.com/ansible/latest/user_guide/intro_getting_started.html#intro-getting-started>.

Docs.ansible.com. 2021. *Amazon.Aws — Ansible Documentation*. [online] Available at: <https://docs.ansible.com/ansible/latest/collections/amazon/aws/index.html#plugins-in-amazon-aws>.

Docs.chef.io. 2021. *Platform Overview*. [online] Available at: <https://docs.chef.io/platform_overview/>.

Puppet.com. 2021. [online] Available at: <https://puppet.com/docs/puppet/5.5/architecture.html>.

Docs.saltproject.io. 2021. *SaltStack Documentation*. [online] Available at: <https://docs.saltproject.io/en/latest/>.

Docs.saltproject.io. 2021. *Running Commands on Salt Minions*. [online] Available at: <https://docs.saltproject.io/en/latest/topics/execution/remote_execution.html>.

Docs.saltproject.io. 2021. *Configuration Management*. [online] Available at: <https://docs.saltproject.io/en/latest/topics/states/>.

Kubernetes. n.d. *¿Qué es Kubernetes?*. [online] Available at: <https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/>.

Kubernetes. n.d. *Conceptos*. [online] Available at: <https://kubernetes.io/es/docs/concepts/>.

Kubernetes. n.d. *Pods*. [online] Available at: <https://kubernetes.io/docs/concepts/workloads/pods/>.

VMware. n.d. *Kubernetes Services*. [online] Available at: <https://www.vmware.com/topics/glossary/content/kubernetes-services>.

Tutorialspoint.com. n.d. *Kubernetes - Volumes*. [online] Available at: <https://www.tutorialspoint.com/kubernetes/kubernetes_volumes.htm>.

VMware. n.d. *Kubernetes Namespace*. [online] Available at: <https://www.vmware.com/topics/glossary/content/kubernetes-namespace>.

Kubernetes. n.d. *ReplicaSet*. [online] Available at: <https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/>.

VMware. n.d. *Kubernetes Deployment*. [online] Available at: <https://www.vmware.com/topics/glossary/content/kubernetes-deployment>.

Kubernetes. n.d. *ConfigMaps*. [online] Available at: <https://kubernetes.io/es/docs/concepts/configuration/configmap/>.

Kubernetes. n.d. *ConfigMaps*. [online] Available at: <https://kubernetes.io/es/docs/concepts/configuration/configmap/>.

Kubernetes. n.d. *DaemonSet*. [online] Available at: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>.

Openshift.com. n.d. *What is OpenShift - Red Hat OpenShift*. [online] Available at: <https://www.openshift.com/learn/what-is-openshift>.

Helm.sh. n.d. *Helm Architecture*. [online] Available at: <https://helm.sh/docs/topics/architecture/>.

Docs.aws.amazon.com. n.d. *What is AWS CloudFormation? - AWS CloudFormation*. [online] Available at: <https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html>.

Chan, M., n.d. *What Is AWS Cloudformation And How Can It Help Your IaC Efforts?*. [online] Thorn Technologies. Available at: <https://www.thorntech.com/whatisawscloudformation/>.

Heller, M., n.d. *What is Jenkins? The CI server explained*. [online] InfoWorld. Available at: <https://www.infoworld.com/article/3239666/what-is-jenkins-the-ci-server-explained.html>.