



Università degli Studi di Salerno
Dipartimento di Informatica

Tesi di Laurea di I livello in
Informatica

Kubernetes in Fly

Relatore

Prof. Vittorio Scarano

Correlatore

Dott. Giuseppe D'Ambrosio

Dott. Carmine Spagnuolo


Candidato

Luigi Barbato

Anno Accademico 2020-2021

Abstract

Kubernetes è una piattaforma portatile, estensibile e open-source per la gestione e l'orchestrazione di applicativi Cloud-Native. Esso consente di orchestrare applicazioni containerizzate all'interno di cluster di nodi su cui vengono eseguiti, grazie agli strumenti ed alle API fornite, operazioni per la gestione del carico di lavoro, del traffico di rete e dell'archiviazione. Nonostante i vantaggi, Kubernetes rimane un sistema estremamente complesso ed articolato nella sua gestione architetturale ed implementativa. Fly è un linguaggio di programmazione domain-specific incentrato sul calcolo scientifico il cui obiettivo è quello di semplificare lo sviluppo di applicazioni su Cloud introducendo un livello di astrazione che permetta all'utente di utilizzare le funzionalità e le potenzialità del cloud provider in modo semplice ed efficiente. Questo lavoro di tesi si prefigge l'obiettivo dell'integrazione di un ambiente di esecuzione, basato su Kubernetes, all'interno del linguaggio di programmazione Fly, rendendo così possibile l'esecuzione del cluster in locale come su Cloud. L'intero processo di esecuzione può così essere automatizzato grazie a costrutti Fly appositamente sviluppati.

Questa tesi è stata sviluppata in  **ISISLab**

Indice

1	Introduzione	1
1.1	Cloud Computing	1
1.1.1	Vantaggi	1
1.1.2	Modelli di servizio	2
1.1.3	Modelli di distribuzione	3
1.2	Multi-cloud	4
1.3	Containerizzazione	5
1.3.1	Vantaggi	5
1.3.2	Macchina Virtuale e Containerizzazione	6
1.4	Kubernetes	6
1.4.1	Architettura	7
1.4.2	Componenti del Control Plane	7
1.4.3	Componenti del Worker Node	8
1.4.4	Oggetti Kubernetes	8

Capitolo 1

Introduzione

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

1.1 Cloud Computing

Il Cloud Computing è una particolare ed innovativa forma di erogazione di risorse da un fornitore ad un cliente attraverso la rete internet. Esso segue un paradigma architetturale di tipo distribuito ed i servizi offerti possono essere vari come l'archiviazione, la computazione oppure una semplice trasmissione dati. Le risorse e le configurazioni a cui fa fede, non sono predefinite ma, con l'utilizzo di processi automatizzati, ogni Cloud Provider è in grado di personalizzare e scalare le risorse opportune per quel determinato servizio a quel determinato cliente. Una volta che quest'ultimo termina l'utilizzo delle risorse assegnate, queste tornano nuovamente disponibili, pronte per una nuova assegnazione.

1.1.1 Vantaggi

E' possibile individuare i maggiori vantaggi che motivano le aziende e gli sviluppatori a utilizzare il Cloud Computing per i propri applicativi o per le proprie infrastrutture.

Costi

E' chiaro che in quest'ottica, il modello economico Pay-as-you-go è diventato parte integrante della strategia di fruizione dei servizi, in quanto porta innumerevoli vantaggi ai fornitori ma soprattutto ai clienti stessi, tra cui:

- **Ottimizzazione del capitale** - la libertà di pagare solo ciò di cui si ha realmente bisogno, l'eliminazione dei costi di investimento e dei costi di mantenimento ed il miglioramento dell'efficienza energetica permettono un notevole abbattimento dei costi ed un'ottimizzazione del capitale.
- **Flessibilità** - I clienti, in ogni momento ed in maniera quasi istantanea, hanno la massima libertà di poter annullare, sostituire o fermare qualsiasi servizio in corso.
- **Scalabilità** - Il sistema adottato, accompagna senza problemi la crescita di piccole infrastrutture. In effetti, per una Start-up agli inizi, è pratico dotarsi di uno strumento adeguato in termini di prezzi e funzionalità, per poi farlo crescere in parallelo con l'evoluzione e le ambizioni dell'azienda.

Performance e scalabilità

La tecnologia alla base del Cloud Computing è la virtualizzazione la quale, date le sue potenzialità, permette di ridimensionare agilmente le risorse assegnate. La potenza computazionale, la larghezza di banda e lo spazio di archiviazione sono esempi di risorse che possono essere scalate in base al carico di richieste, andando quindi ad allocarne una quantità maggiore in caso di maggiore attività, una quantità minore in caso di minore attività e a deallocare quelle inutilizzate in caso d'inattività prolungata.

1.1.2 Modelli di servizio

Il Cloud Computing è un mondo in continua crescita ed evoluzione, nuovi prodotti e servizi cloud arrivano quasi ogni giorno, spinti dalle costanti innovazioni tecnologiche. Nonostante la maturità del mercato, molte organizzazioni non sono ancora a conoscenza dei servizi e dei modelli d'implementazione che i diversi Cloud Provider forniscono. I tre modelli principali a oggi maggiormente diffusi sono così definiti:

Software as a Service (SaaS)

Software as a Service (SaaS) è il modello di servizio cloud che fornisce l'accesso a un prodotto software completo, eseguito e gestito dal fornitore del servizio. In questo particolare modello, il fornitore ha la piena responsabilità sull'intero ciclo di vita del software e sulla manutenzione dell'infrastruttura sottostante. Questo consente al cliente di concentrarsi esclusivamente su come utilizzare al meglio le specifiche e le funzionalità del software offerto.

Platform as a Service (PaaS)

Platform as a Service (PaaS) è il modello usato più comunemente per lo sviluppo di applicazioni. Il fornitore del servizio fornisce l'accesso alle proprie risorse infrastrutturali come: basi di dati, sistemi operativi e server, senza la complessità di gestione che ne deriva. Questo permette al cliente di dedicare le proprie risorse all'ottimizzazione dell'applicativo invece che all'installazione e alla configurazione dell'infrastruttura.

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) è il modello che permette al cliente d'implementare la propria infrastruttura cloud. Il Cloud Provider fornisce l'accesso on-demand delle proprie risorse (sia fisiche che virtuali) volte alla computazione, archiviazione e alla rete. Il cliente ottiene così l'enorme vantaggio di poter distribuire e gestire i propri applicativi avendo la sicurezza di avere risorse sempre disponibili, affidabili e scalabili.

1.1.3 Modelli di distribuzione

Ogni modello d'implementazione del cloud ha una propria configurazione unica con una gamma di requisiti diversi e vantaggi associati.

Public Cloud

Il Public Cloud è un modello di distribuzione in cui i servizi cloud sono di proprietà di fornitori esterni. Scegliere questa metodologia garantisce una maggiore agilità operativa e una scalabilità pressoché illimitata, perché sfrutta le funzionalità e le risorse di grandi fornitori come Google, Microsoft e Amazon. Trattandosi inoltre di ambienti gestiti, il cliente si solleva dalle responsabilità di manutenzione e controllo delle risorse utilizzate.

Private Cloud

Nel modello di distribuzione Private Cloud si sceglie di sviluppare, mantenere e gestire la propria infrastruttura cloud, fornendo l'accesso solo alla propria rete interna. Con il Private Cloud le organizzazioni mantengono un controllo completo dell'infrastruttura. Questo si traduce in un controllo completo delle proprie risorse e ad una maggior libertà di personalizzazione dei servizi, che sono consumati all'interno del cloud.

Hybrid Cloud

L'Hybrid Cloud è un modello di distribuzione ibrido in quanto si sceglie di unire i vantaggi del cloud pubblico e di quello privato. Ad esempio, si possono utilizzare le risorse del Public Cloud per attività di computazione, e tenere al sicuro i dati sensibili e le applicazioni critiche nel Private Cloud. Oppure, di fronte a picchi improvvisi e temporanei della domanda di risorse, le organizzazioni possono scegliere di spostare i carichi di lavoro dal Private Cloud al Public Cloud. Utilizzato in questo modo, l'Hybrid Cloud consente di ottenere il meglio da entrambe le infrastrutture.

1.2 Multi-cloud

Il multi-cloud è una forma architetturale in cui si utilizzano risorse e servizi di Cloud Provider differenti convergendoli in una singola architettura eterogenea. Le motivazioni che muovono un'organizzazione ad adottare questo tipo di strategia sono varie:

- **Necessità differenti** - Ogni Cloud Provider propone la propria gamma di servizi che non sempre rispecchia a pieno le esigenze del cliente. La possibilità e quindi la libertà di poter scegliere e selezionare distintamente i singoli servizi concedono all'utente il vantaggio di poter coprire totalmente le proprie esigenze.
- **Localizzazione** - Ogni Cloud Provider detiene le proprie infrastrutture dislocate in diverse aree geografiche, l'utente può decidere di utilizzarle in base alle proprie esigenze logistiche e legali.
- **Prestazioni** - L'utilizzo di risorse provenienti da fornitori distinti, permette una distribuzione più ampia dei servizi dell'applicativo.

1.3 Containerizzazione

Le prime applicazioni realizzate in informatica venivano eseguite ognuna su una macchina fisica dedicata, con conseguenti costi durante le fasi di acquisto, configurazione e manutenzione e con il rischio di non sfruttare mai completamente le risorse allocate. La *virtualizzazione hardware* ha risolto tale problema, permettendo di allocare e condensare i servizi su poche macchine server fisiche, portando ad una riduzione dei costi e migliorando soprattutto la manutenibilità di tali architetture. un software particolare, noto come hypervisor è responsabile del collegamento con l'hardware e permette di condividere un singolo sistema host fisico tra diversi ambienti, separati e ben distinti. Con il termine containerizzazione si intende una para-virtualizzazione dell'ambiente applicativo che consente di eseguire software, librerie, dipendenze e tutte le componenti necessarie, in un processo isolato che prende il nome di contenitore. Esso può essere considerato un vero e proprio eseguibile che non dipende da alcuna sorgente esterna. Questo lo rende estremamente portatile e affidabile in quanto può essere eseguito e trasferito in ogni tipo di ambiente e d'infrastruttura. L'idea alla base del paradigma non è in realtà nuova in quanto sfrutta una funzionalità del kernel Linux presente già dalla versione 2.6.24 che permette l'isolamento e la gestione delle risorse di uno o più processi.

1.3.1 Vantaggi

L'utilizzo di questa tecnologia ha rivoluzionato il modus operandi del settore IT, dello sviluppo software e della loro distribuzione su infrastrutture Cloud. Questo grazie a tre principali vantaggi:

- **Portabilità** - Poichè un container contiene tutte le componenti necessarie all'esecuzione dell'applicazione ed è un processo isolato del sistema ospitante,
- **Efficienza** - I contenitori hanno la capacità di condividere il kernel della macchina ospitante evitando così di dover disporre di risorse hardware e software dedicate.
- **Ottimizzazione dello spazio** - L'immagine dell'applicazione che viene eseguita all'interno del contenitore, incapsula solo ed unicamente le componenti e le informazioni necessarie alla sua esecuzione.
- **Ambienti di sviluppo innovativi** - Una nuova strategia di sviluppo che si sta affermando negli ultimi anni è quella del DevOps e la con-

tainerizzazione è un punto cardine di questo nuovo paradigma. Per la natura stessa dei contenitori, gli sviluppatori possono condividere facilmente il loro software, eliminando così problemi di compatibilità dettati da ambienti di sviluppo e di esecuzione differenti.

1.3.2 Macchina Virtuale e Containerizzazione

Una macchina virtuale è una rappresentazione virtuale delle risorse hardware e software di un sistema informatico. Essa ha il compito dunque di ricreare CPU, memoria, interfaccia di rete e storage e di eseguire un intero sistema operativo. Un contenitore si basa sul kernel del sistema operativo host e contiene solo le componenti necessarie all'esecuzione dell'applicativo containerizzato. La somiglianza tra containerizzazione e virtualizzazione sta nel fatto che entrambe consentono l'isolamento completo delle applicazioni, che le rende operative su più ambienti. La differenza sta invece nelle dimensioni, nella portabilità e nell'efficienza.

1.4 Kubernetes

Kubernetes è una piattaforma portatile, estensibile e open-source per la gestione e l'orchestrazione di applicativi Cloud-Native. La piattaforma è scritta in linguaggio Go ed è stato inizialmente sviluppato da Google per migliorare la gestione dei propri applicativi. Il progetto è attualmente parte della Cloud Native Computing Foundation¹, organizzazione che promuove e mantiene progetti open-source volti all'approccio Cloud Native. Kubernetes utilizza un insieme di oggetti fruibili tramite API per descrivere lo stato desiderato del cluster, indicando, ad esempio, quali applicazioni eseguire, quali immagini utilizzare, il numero di repliche da istanziare, quali risorse di rete e di spazio su disco rendere disponibili. L'interazione viene resa possibile grazie a kube-apiserver, il server HTTP che implementa l'API Kubernetes. L'utente ha dunque due possibilità per manipolare le configurazioni del cluster, la prima è quella di eseguire esplicitamente richieste di tipo RESTful al server API, la seconda e' quella di utilizzare kubectl, un'interfaccia da linea di comando costituita da una serie di comandi e sotto-comandi che astraggono le chiamate API.

¹Cloud Native Computing Foundation - [link](#)

1.4.1 Architettura

Un cluster Kubernetes è un insieme di macchine, chiamate nodi, che eseguono carichi applicativi. Il cluster deve avere almeno un Worker Node ed un Master Node. Il Worker Node è un tipo di nodo che ospita i Pod, particolari componenti di Kubernetes che contengono uno o più container. Il Master Node è il nodo su cui viene eseguito il Control Plane, una componente fondamentale di Kubernetes che svolge le attività operative del cluster.

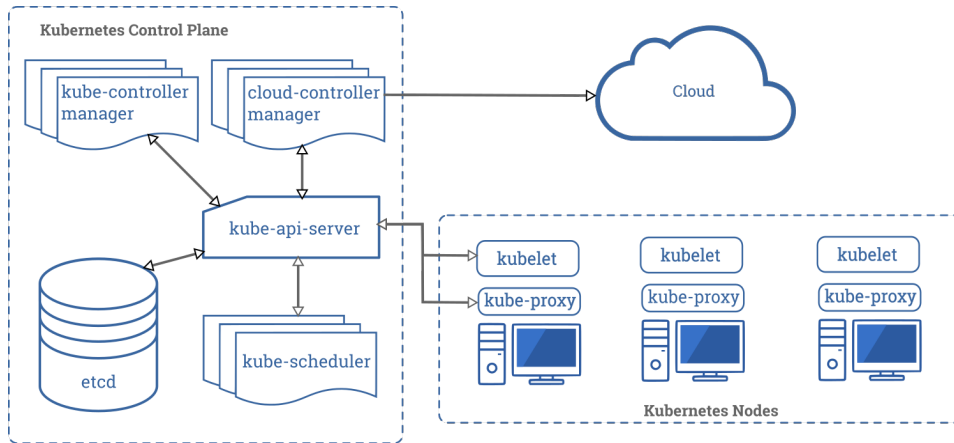


Figura 1.1: Architettura di Kubernetes

1.4.2 Componenti del Control Plane

Il Kubernetes Control Plane è il centro nevralgico delle operazioni del cluster. È costituito da una serie di componenti, in esecuzione all'interno del cluster, che hanno il compito di garantire l'integrità esecutiva del cluster come ad esempio lo scheduling dei Worker Node e la gestione dei possibili errori dei Pod.

- **kube-apiserver** kube-apiserver è il server API per il cluster Kubernetes. È il punto di contatto centrale a cui accedono tutti gli utenti, l'automazione e i componenti nel cluster Kubernetes. Il server API implementa un'API RESTful su HTTP, esegue tutte le operazioni API ed è responsabile dell'archiviazione degli oggetti API in un backend di archiviazione persistente.
- **etcd** etcd è la componente di archiviazione principale di Kubernetes, etcd memorizza e replica tutti gli stati dei cluster Kubernetes. Il

meccanismo di memorizzazione che adotta è di tipo chiave-valore ovvero ad ogni valore memorizzato viene associato una chiave che rappresenta il suo identificatore univoco. In questo modo il Control Plane è in grado di storicizzare gli stati del Cluster confrontando lo stato attuale con quello desiderato in modo di intervenire tempestivamente in caso di necessità.

- **kube-scheduler** kube-scheduler è la componente che consente al Control Plane lo scheduling dei Pod sui nodi. il kube-scheduler controlla i pod appena creati che non hanno un nodo assegnato, e dopo averlo identificato glielo assegna.

1.4.3 Componenti del Worker Node

Il Worker Node è il nodo del cluster su cui viene eseguito effettivamente l'applicazione. Su ogni nodo di tipo Worker Node vengono eseguiti tre componenti di Kubernetes.

- **kubelet** La componente kubelet è un agente che è sempre in esecuzione su ogni Worker Node e ha come compito quello di controllare il ciclo di vita dei container all'interno dei Pod. La kubelet riceve un set di specifiche definite in fase di deployment dall'utente e si assicura che i container rispettino tali specifiche.
- **kube-proxy** La componente kube-proxy è un proxy eseguito su ogni Nodo, ossia un particolare componente che consente di identificare i nodi all'interno della rete del cluster e permette che essi possano comunicare sia all'interno che all'esterno del cluster.
- **Container Runtime** il Container Runtime è il software responsabile dell'esecuzione dei container. Kubernetes supporta tutte le implementazioni di Kubernetes CRI (Container Runtime Interface) come ad esempio Docker e containerd.

1.4.4 Oggetti Kubernetes

Kubernetes offre una varietà di oggetti per definire le specifiche del proprio sistema. Questi oggetti costituiscono entità persistenti per rappresentare lo stato del cluster, quali applicazioni sono in esecuzione e su quali nodi, le risorse da allocare e le politiche da adottare. Per istanziare un oggetto con le direttive e le informazioni desiderate, si utilizza spesso un file di tipo YAML oppure JSON. Ciascun oggetto è caratterizzato da due campi che ne

descrivono la configurazione:

Specifiche Le specifiche descrivono lo stato desiderato dell'oggetto, cioè le caratteristiche che deve assumere.

Stato Lo stato descrive lo stato attuale dell'oggetto e viene automaticamente aggiornato da Kubernetes.

- **Pod** Un Pod costituisce la più semplice, piccola e basilare unità di esecuzione in Kubernetes. La sua natura funzionale è quella di incapsulare uno o più container e di dividerne le risorse come l'archiviazione dei dati e le risorse di rete. I Pod generalmente non vengono creati direttamente in Kubernetes perchè questi sono considerati oggetti effimeri. Solitamente i Pod vengono creati e gestiti da controller di più alto livello come Deployment, StatefulSet o DaemonSet.
- **Deployment** L'oggetto Deployment si occupa di creare e gestire il ciclo di vita di uno o più Pod e dei ReplicaSet. Un Deployment fornisce un approccio dichiarativo per la creazione e la modifica di Pod e ReplicaSet, con esso è possibile dunque descrivere lo stato che il Pod deve assumere ed il Deployment Controller si occuperà di aggiornare lo stato attuale con quello desiderato.
- **ReplicaSet** Un ReplicaSet ha il compito specifico di mantenere attive una o più copie di un Pod. Questo meccanismo garantisce la piena disponibilità del Pod a cui il ReplicaSet è agganciato in quanto nel momento in cui un Pod, per un qualsiasi motivo, cessa il suo funzionamento, verrà sostituito da una sua esatta copia.
- **Service** I Pod in Kubernetes sono entità effimere, essi vengono creati e distrutti in base allo stato desiderato del cluster. Per natura quindi non hanno un'identificazione statica all'interno della rete del cluster ma ad ogni Pod viene assegnato un indirizzo IP dinamico che varia nel tempo. Un Service in Kubernetes costituisce un'astrazione che definisce un insieme logico di Pod e una politica con cui accedervi. Kubernetes offre diversi tipi di Service:

ClusterIP Espone il Service con un indirizzo IP interno al cluster, rendendo il cluster raggiungibile solo dall'interno del cluster stesso.

NodePort Espone il Service su ciascun indirizzo IP corrispondente ad un nodo del cluster, su una porta scelta automaticamente, uguale per tutti i nodi.

LoadBalancer Permette di esporre le applicazioni all'esterno del cluster.

- **Job** Un oggetto di tipo Job permette di eseguire e gestire task complessi all'interno del cluster. Un Job infatti permette di eseguire un insieme di Pod in maniera sequenziale o parallela considerandoli come un'unica attività di lavoro da portare a termine. Il Job terrà in esecuzione l'attività fino a quando un numero specifico di Pod non verrà terminato correttamente. Quando viene raggiunto un numero specifico di completamenti riusciti, l'attività risulterà completata ed il Job cesserà la sua esecuzione. L'eliminazione del Job distruggerà dunque tutti i Pod precedentemente assegnati.