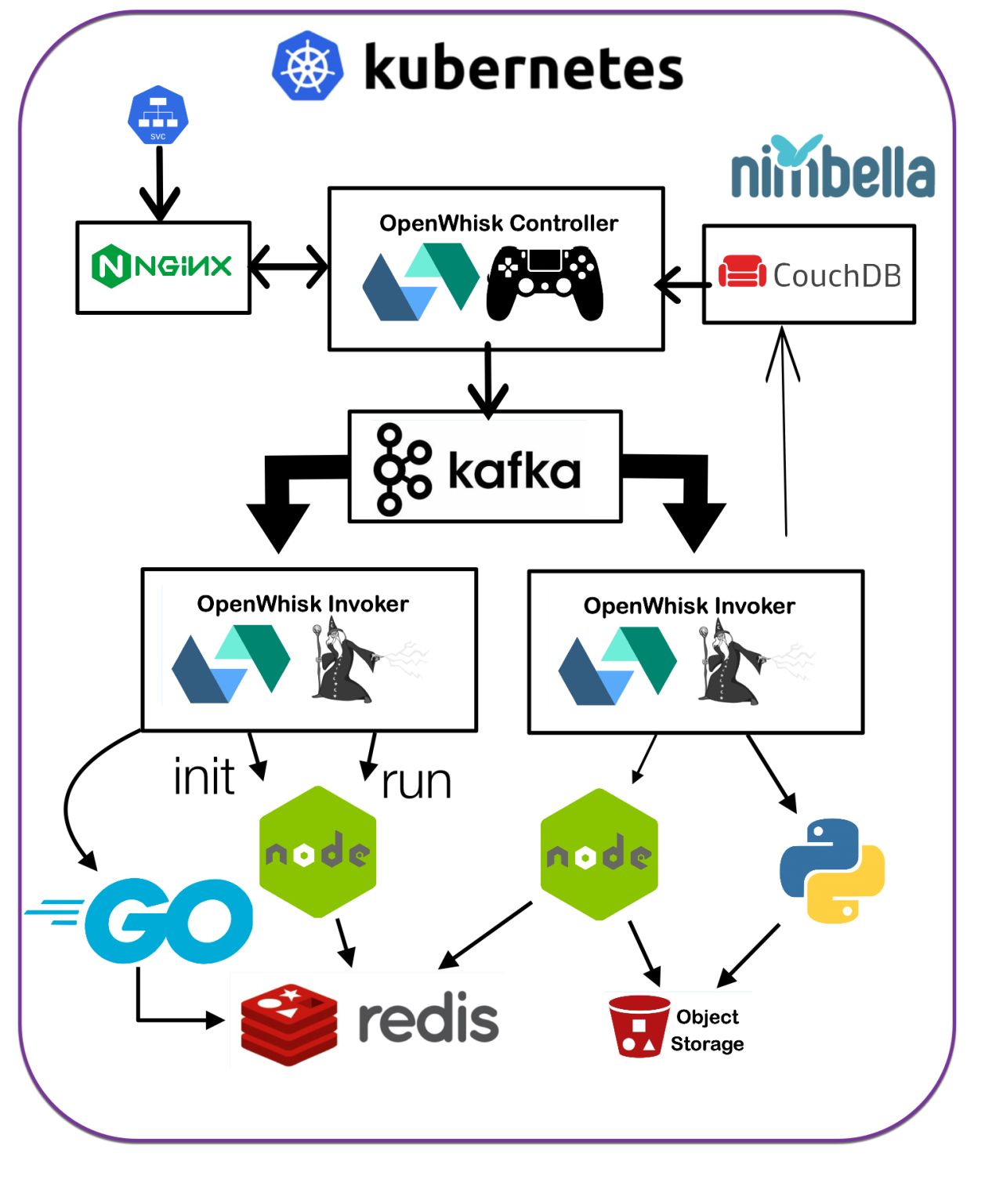
Dockerizziamo OpenWhisk: compilazione Part 1/3

Apache OpenWhisk è un sistema Serverless Open Source molto flessibile, sviluppato come progetto Open Source alla Apache Software Foundation da una grande comunità di oltre 200 persone. È una tipica applicazione distribuita e può quindi essere installato in Kubernetes come sistema distribuito. Può Utilizzare Kafka come coda di messaggi e CouchDB come database NoSQL per memorizzare configurazioni e operazioni eseguite. In tutti il suo “fulgore” ha questa architettura:



Tutti i componenti di OpenWhisk sono replicabili e scalabili e si può creare un ambiente con centinaia di nodi. Per questo motivo è un ambiente ideale da installare per apprendere come funziona Kubernetes in ambienti realisitici. Ho deciso pertanto di scrivere una serie di articoli su come si installa OpenWhisk in Kubernetes allo scopo di presentare un tutorial su come installarlo, impacchettamento in una immagine Docker e deploy su Kubernetes. Cominciamo proprio dalla installazione e impachettamento in Docker.

Non parlerò in questa serie invece di come funziona OpenWhisk stesso come sistema serverless, anche se è indubbiamente un argomento anch'esso interessante. Delego a questo proposito la lettura del libro in inglese "Learning Apache OpenWhisk", pubblicato da O'Reilly, di cui sono l'autore. Cortesemente Google ne pubblica un ampio estratto di ben 94 pagine che copre tutte le basi, lo trovate facilmente su Google stesso con la chiave"Learning Apache OpenWhisk". Inoltre potete trovare su YouTube in italiano un corso gratuito che copre Nimbella, che è un servizio commerciale Serveless basato su OpenWhisk. Tutto quanto si applica a Nimbella è generalmente valido anche per OpenWhisk (fatta eccezione per alcune estensioni fornite da Nimbella). La playlist ha URl *bit.ly/intronim*.

# Compilazione in locale

Iniziamo le operazioni di installazione partendo dalla modalità più semplice, installazione in locale in modalità singola ("standalone"). Questo passo è propedeutico per poi creare una immagine Docker da usare poi in Kubernetes.

Quindi si può cominciare ad usarlo in maniera molto semplice ed immediata. Tutto quello che vi serve è Docker stesso, un Java Development Kit per compilare il backend, e Node.js per poter compilare il front-end. Se siete uno sviluppatore molto probabilmente avete già tutto questo nella vostra macchina.

Se avete i prerequisiti sulla vostra macchina potete cimentarvi nell'impresa, tutto sommato piuttosto semplice, di installarlo e farlo partire in locale. Potete semplicemente scaricare i sorgenti da qui:

<https://github.com/apache/openwhisk>

Nelle release troverete la versione 1.0, in formato zip e tar.gz. Vi conviene comunque prendere il master con un bel *git clone*, in quando è stabile e di solito più aggiornato delle releases, e quindi contiene le ultime novità e bug fix.

Una volta scaricato e spacchettato, o clonato il sorgente, andate nella directory principale e potete compilare OpenWhisk standalone con:

./gradlew :core:standalone:build

Dovete avere un attimo di pazienza (ci mette un po' a compilare) ma poi avrete OpenWhisk versione locale pronto all'uso. Mentre compila vi consiglio di andare su:

<https://github.com/apache/openwhisk-cli>

per scaricare la command line interface per il vostro sistema operativo. Una volta scaricata, dezippate il pacchetto e mettete il comandu **wsk** nel *PATH* da qualche parte nel vostro sistema (per esempio tipicamente in  */usr/local/bin* su Mac e Linux, o *C:\Windows\System32* su Windows).

Tutto è pronto per lanciarvi nel mondo del serverless! Potete lanciare OpenWhisk in locale con il comando:

java –jar bin/openwhisk-standalone.jar

Questo dovrebbe normalmente aprire il browser con il playground. Si tratta di una semplice applicazione javascript che vi permette di scrivere una funzione Serverless nei vari linguaggi direttamente nel browser e provarla.

# Sperimentare con il serverless

Una volta che avete fatto hello world nel playground, la cosa migliore è scrivere delle applicazioni con un editor, visto che il playground non è un completo ambiente di sviluppo. E qui viene utile la utility a riga di comando **wsk**.

La CLI la dovete innanzitutto configurare. Se leggete accuratamente l'output di OpenWhisk quando lo lanciate (fate attenzione, è verso l'inizio, e viene immediatamente 'scrollato via' dagli altri messaggi che vengono mostrati) troverete il comando per configurare. Qualcosa come:

wsk property set … (omissis)

A questo punto copiate uno degli esempi che vi presenta il playground, per esempio *hello.js* in un file, e scrivete:

wks action update hello hello.js

per creare l'azione hello. A questo punto la potete invocare e vedere l'output con

wsk action invoke hello –p name Michele -r

per vedere l'output. Dovreste vedere un messaggio di benvenuto in formato JSON.

# Conclusioni

Abbiamo fatto i primi passi con OpenWhisk, imparato come prendere i sorgenti e compilarlo e anche come utilizzarlo in locale. Questo dovrebbe dare l’idea del suo potenziale, che si estrinseca pienamente quando viene messo in Docker e deployto in Kubernetes. Il prossimo passo sarà appunto quello di impacchettarlo in una immagine Docker, e ne approfitteremo per spiegare un po’ di Docker. Alla seconda parte.