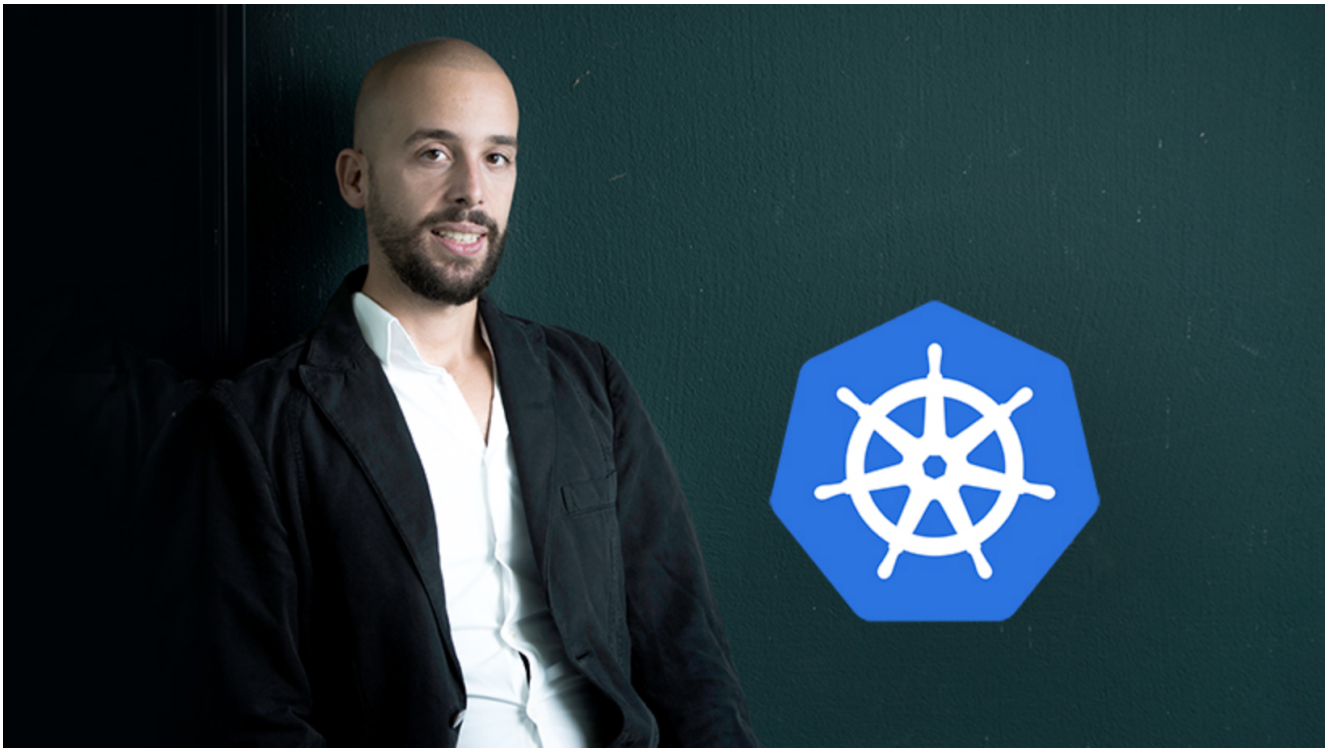
I 5 Errori più comuni dei principianti con Kubernetes



Dopo alcuni anni di esperienza con Kubernetes e diverse consulenze per svariati clienti ho notato alcuni errori comuni commessi dagli sviluppatori. Ho pensato quindi di selezionarne 5 abbastanza frequenti, andando a spiegare cosa c'è di errato.

Per chi avesse bisogno della basi per capire meglio i punti seguenti, ecco il link al primo corso Kubernetes in italiano: Corso Kubernetes pratico e veloce per Principianti

Il selector delle labels su un Service non ha un match con i Pod

Per poter funzionare correttamente come bilanciatore di rete, un Service specifica in genere dei selectors, i quali permettono di trovare i Pod che fanno parte del pool di bilanciamento. Nel caso non ci sia un match, il Service non avrà Endpoints ai quali inoltrare il traffico. Teniamo presente che il loadbalancing verso i Pod è di tipo casuale .

Porta sbagliata del Container mappata sul Service

Ogni Service ha due parametri fondamentali “*targetPort*” e “*port*” i quali vengono spesso confusi e usati in modo scorretto. Questo risulta quindi in messaggi di errore come “Connection Refused” o la mancanza di risposta ad una richiesta.

In modo da evitare questo errore, ricordiamo che “*targetPort*” è la porta destinazione nei Pod, quella a cui un Service va ad inoltrare il traffico. Il parametro “*port*” invece è riferito alla porta esposta dal Service verso i client.

Esse possono essere uguali ovviamente, importante è conoscerne il significato!

Liveness e readiness probes

Ci sono diversi errori che vengono commessi a riguardo.

Il primo è il non definire alcun tipo di health check per il nostro applicativo, il quale non verrà mai riavviato in caso di problemi e rimarrà sempre all’interno del pool di loadbalancing di un Service.

Il secondo tipo di errore riguarda il definire liveness e readiness probes uguali, ad esempio andando a contattare lo stesso identico endpoint HTTP. Questo può essere dovuto ad una comprensione sbagliata di questi tipi di test. La liveness probe è collegata al concetto di applicazione in salute, quindi nel caso fallisca, verrà riavviato il Pod. La readiness probe è relativa all’inserimento o meno del Pod nel pool di loadbalancing durante tutto il ciclo di vita del Pod. Definirle uguali potrebbe far rischiare ad esempio il riavvio di un applicativo che è temporaneamente lento per via di molto traffico.

Un altro aspetto importante, è il non far fallire le probes nel caso una delle dipendenze del servizio in esame sia down. Questo infatti sarebbe solamente il propagare il problema, che invece deve essere rilevato nel servizio opportuno.

Risorse – Requests e Limits

In genere i principianti dimenticano di settare requests e limits per CPU e memoria. Questo errore porta ad avere dei nodi worker sui quali sono schedulati troppi applicativi, diventando "overcommitted". Il risultato finale sono applicativi poco performanti.

Altre volte invece vengono settati dei limits troppo bassi, che portano nel caso della CPU a basse performance e nel caso della memoria al "kill" del Pod. Se hai già visto il messaggio di errore OOMkill si tratta proprio di questo problema. La strategia ideale è assegnare la giusta quantità di CPU e memoria, possibilmente settando le requests pari ai limits (Guaranteed QoS) per applicazioni critiche.

Il modo per capire le risorse necessarie è sostanzialmente effettuare dei test e monitorare con tools come Prometheus o simili.

Troppi Services di tipo LoadBalancer

Capita spesso che i principianti espongano troppi Service all'esterno utilizzando il tipo Loadbalancer. In questo caso, un controller specifico, fornito dal cloud provider, andrà a creare un Loadbalancer esterno al cluster, all'interno della VPC. Queste risorse possono diventare abbastanza costose, soprattutto se sono associate ad indirizzi IP statici.

La soluzione migliore è esporre il cluster con un singolo Loadbalancer che andrà ad inoltrare il traffico ai nodi del cluster, i quali in genere hanno un ingress controller per effettuare il routing utilizzando risorse di tipo Ingress.

Bonus: Eseguire comandi senza specificare un namespace

Durante i workshop ho notato che in genere gli studenti dimenticano di posizionarsi sul namespace corretto, o comunque dimenticano il flag -n per specificare un namespace. È importante essere sicuri di utilizzare o specificare il namespace corretto per eseguire un comando!

Conclusioni

Spero che questa breve lista vi possa essere d’aiuto per interagire nel modo corretto con Kubernetes sin dall’inzio.

Non esitate a contattarmi sui social per eventuali chiarimenti, collaborazioni o per aver più informazioni sul corso su Udemy!

Paolo

https://italiancoders.it/i-5-errori-piu-comuni-dei-principianti-con-kubernetes/