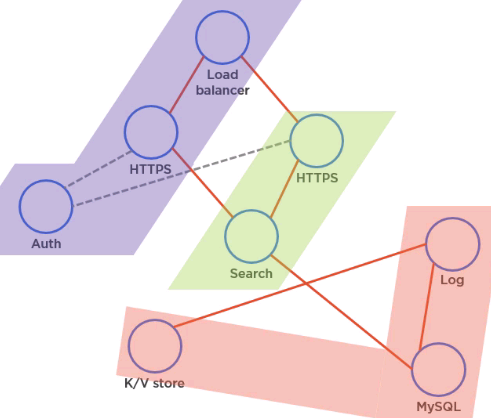
Kubernetes=K8S

- kontenery daja nam skalowalnosc, bardzo duzo moze byc, jak bardzo duzo to wyzwanie, k8s pozwala nimi zarzadzac jakby ich zbior to byl 1 komputer

- klaster manager

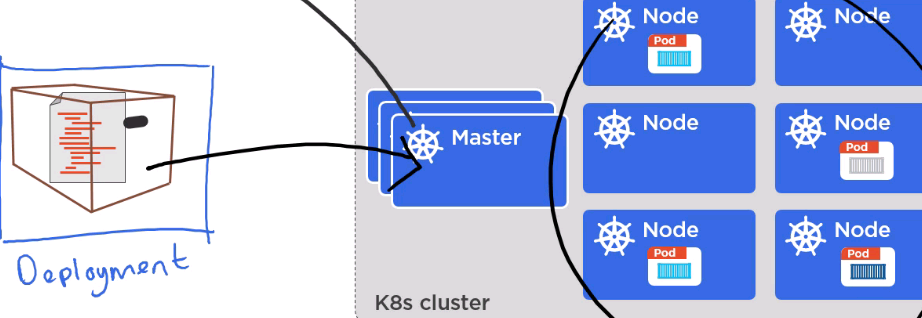
Mamy osobne aplikacje, a K8 organizuje je by dzialaly jak druzyna, powiazane, inne funkcje



- orchestration by dzialaly razem

- master zarzadza, monitoruj, odbiera zadania, wprowadza zmiany, kolejkuje zadania

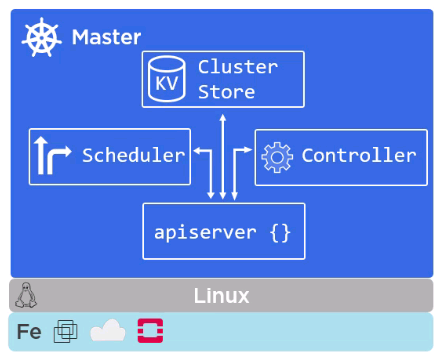
- nodes – do the work



- deployment ma w sobie yml, mowi jak apka wyglada, jakich images uzyc, jakie porty, siec, ile replica

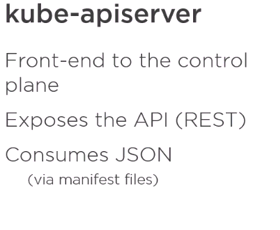
- po wrzuceniu deployment pody odpalane na nodach – replica

MASTER

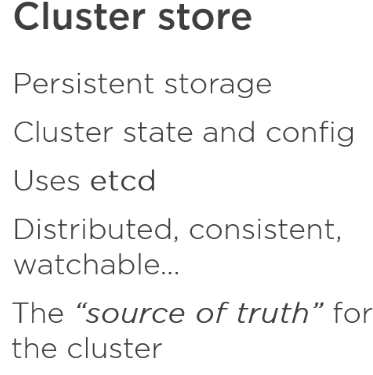


Stoi na linux, pod spodem moze byc chmura

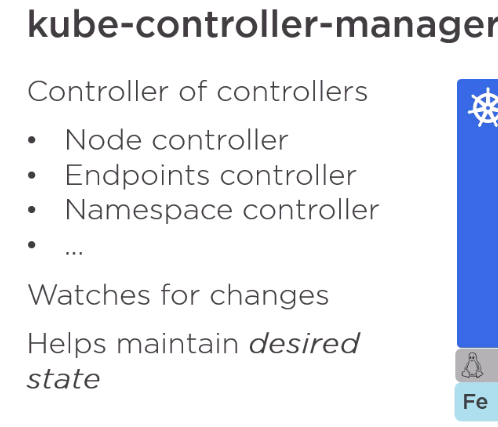
Z kiilku czesci sie sklada

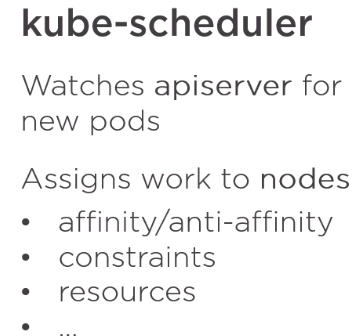


Jest wewnetrzna baza noSQL – etcd, trzyma stan, konfiguracje,..., to jednyna czesc statefull



- mamy wiele kontenerow do kontroli poszczegolnych zasobow





- z master mamy interakcje, przez api serwer komendy, autentykacja, odbiera json, updatuje pozostale elementy i robi refresh nodes

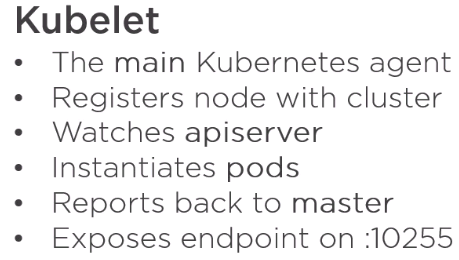
----------

NODES

- k8s jest jak stado

- node z 3 czesci **Kublet**, container runtime i proxy

- to prawie node, raportuje stan, bledy do mastera

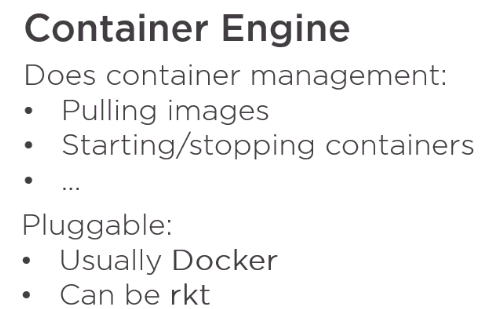




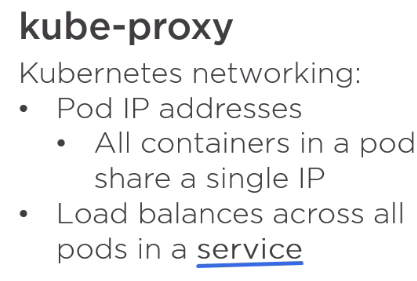
Specyfikacja, zdrowie i running pods

- pod do 1 lub kilka kontenerow spakowanych razem i odpalonych jak jedna jednostka - unit

- **container engine** – do zarzadzania kontenerami



- uzywa API dockera jak pod spodem, czyli ten engine to dla nas docker



- serwis daje wiele nodow za jednym adresem sieciowym

DECLARATIVE MODEL & DESIRED STATE

-declarative model, wszystko jak ma wygladac w manifest dostarczonym (yml lub json)

- mowimy jak ma wygladac, co chcemy, a kubernetis ogarnia by to osiagnac

- mówimy co nie jak, np. Jaki image chcemy uzyc

- k8 sicaga obrazy, startuje kontenery, buduje siec,

- w manifest dostarczamy nasz desired state

- za kazdym razem jak stan klastra odbiega od desired, kubernetis dziala by przywrocic , np. Jak node pada to stawia pod na innym

- w petli spawdza czy spoko stan

-----

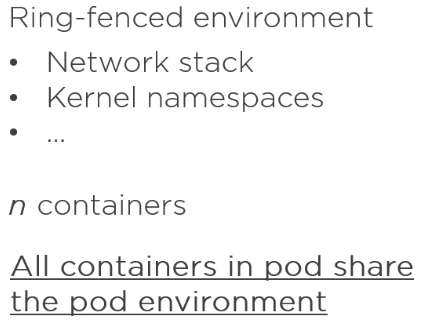
PODS

- k8 odpala kontenery ale zawsze w pod, to najmniejsza jednostka tu

- moze miec kilka kontenerow, zaawansowane

- pod nic nie odpala, to po prostu srodowicko ktore wspoldzila kontenery

- jak 2 kontenery w pod, to wspolne IP, moga gadac po localhost



- thight coupling – kontenery wspoldziela pamiec i system plikow, to 1 POD

- jak loose coupling to lepiej osobne, gadaja po sieci

- skalowanie przez replica poda

- nie czekampy na wstanie pod, nie ma wstawania, jest albo go nie ma

- jest atomowy, na jednym node

- pod w stanie pending/born running success/failed – jak umiera to nowy sie rodzi, nie ma przywracania

- de deploymentu uzywany ReplicationController

--

SERVICES

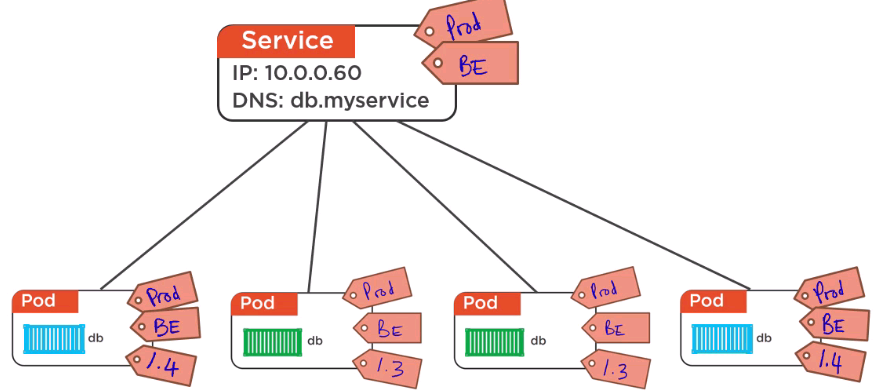
- POD MA SWOJE IP/PORT, jak pod umiera na jego miejsce nowy , moze na innym node z innym IP/PORT

- bezposrednie polaczenie bylby balagan, jak pod nowy IP sie pojawia, wiec mamy warstwe serwisow

- klienci do serwisu, ma swoje IP, DNS, robi load balancing pod zarejestrowane pods

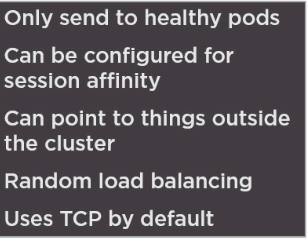
- serwis robi load balance na pody z taka LABELS jak ok, musie wszystkie np. PROD, BE, 1.3

- moze serwice miec mniej labels, i laczy wszystkie co maja wszystkie jego



- jak dodamy 1.4 do serwice, to stare odcina , juz do nich nie uderza

- serwisy tylko sla ruch do zdrowych pods



- UDP tez mozliwe

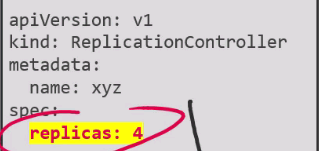
- moze byc sesja i uderzac poza cluster

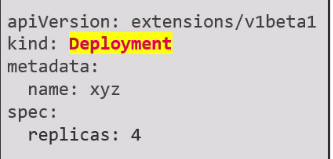
-----

DEPLOYMENT

- rozmieszczenie nodes przez replication controller

Manifesty :

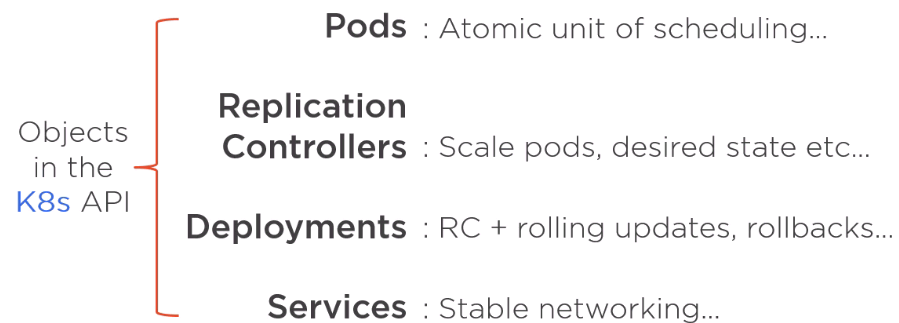




-Deklaratywnie co chcemy, jest samo dokumentujacy, wersjonowany, raz specyfikowany dla wielu deploy, proste update i rollback

- czyli mamy obiekt REST, ktowy przesyla yml do apiserwer

- mozna dodac features do ReplicationControllers – replica sets



INTALACJA K8S

-mozna postawic MINICUBE na wirtualce, daje nam **kubectl - klienta**

- mozna latwo zainstalowac miniqube, dziala jak k8

- na win sciagam kubectl.exe i minicube.exe

- moge minicybe start i potem kubectl get pods

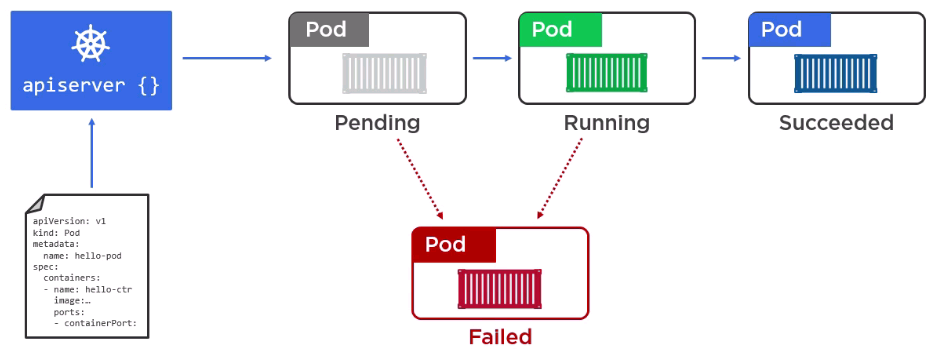
PODS

- kazdy pod ma 1 IP

- kontenery w srodku maja swoje porty

W pod kontenery po localhost, miedzy pody po ip

- ponizej cykl zycia, pending az wszystkie kontenery w srodku wystartuja

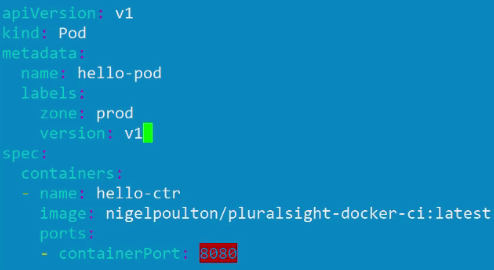


--

Deploy POD

- operacje przez **kubectl**

Dla deploy pod potrzebny manifest



Wersja, co deployowane, nazwa i labelki, w spec jakie kontenery w podach , ich nazwa z czego i jaki port

Start:



Teraz pobieram pody, mam ich stan



**- kubectl describe pods** – wypisuje szczegolowe dane o podach

- widzimy ze pod status jest pending, a container waiting, czyli image sie buduje

Moge pobrac konkretny pod



Moge ze wszystkich namespace



- replication controller co obtacza pody na podstawie replica z manifestu tworzy odpowiednia ilosc

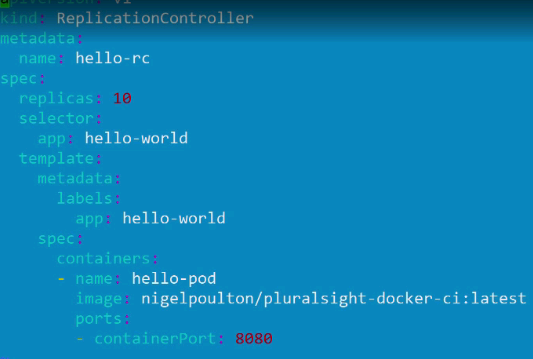
- do tego chodzi loop, co sprawdza czy mamy odpowiedni stan

- moge wyrzucic pod



- w rc.yml moge zdefiniowac replication controller

- ma kind ReplicationController, nie pod jak wczesniej



- majac replication controller mozemy podac replicas i stworzy ich odpowiednia ilosc

- jak po create chcemy zmiany , to zmieniamy yml i odpalam apply



Ponizsze da nam status podow



SERVICES

- SERVICE SIE NIE ZMIENIA, WIEC KLIENT MA STALY URL

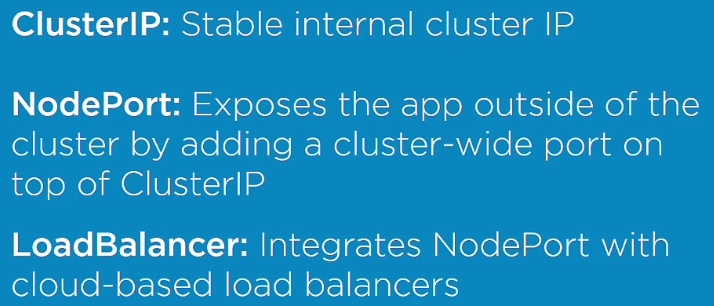
- service na porcie – na zewnatrz

- w kazdym node w klastrze kazdy pod polaczony z serwisem dziala na tym samym porcie

- przy tworzeniu serwisu tworzony endpoint, co ma liste pod i ich IP

- servicde powiazany z podami przez LABELS

- jest kilka typow serwisow, my mamy NodePort



--

Tworzenie serwisu interaktywnie

- po deploy nie mozemy wejsc na port 8080, bo to nie serwisu, tylko kontenera, potrzebny expose portu uzywajac rc



Teraz czytamy detale



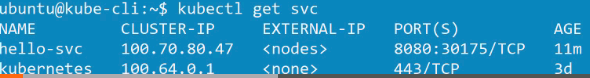
I widimy port naszego node jako 30175, to port node, mapowany na 8080 kontenera

Teraz kazdy node w klaster ma ten port, mapuje na 8080 jego poda z nasza apka

--

Tworzenie serwisu deklaratywnie – z yml

Serwisy moge pobrac



Moge wyrzucic serwis



Tworze nowy yml dla service



Kind : Service, nazwa i jaka apka

Port mowi na jaki port w kontenerze, a nodePort jaki na zewnatrz wystawiony dla calego klastra

Tworzymy service na podtawie yml, apki juz stoja



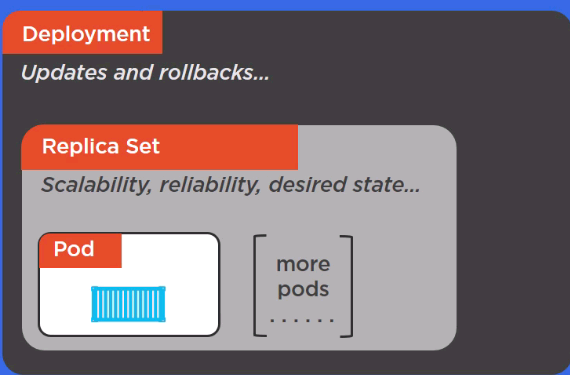
- kubectl get ep – endpointy pobiera

- kubectl describe ep svc-name – poda nam wszystkie endpointy I wiecej info

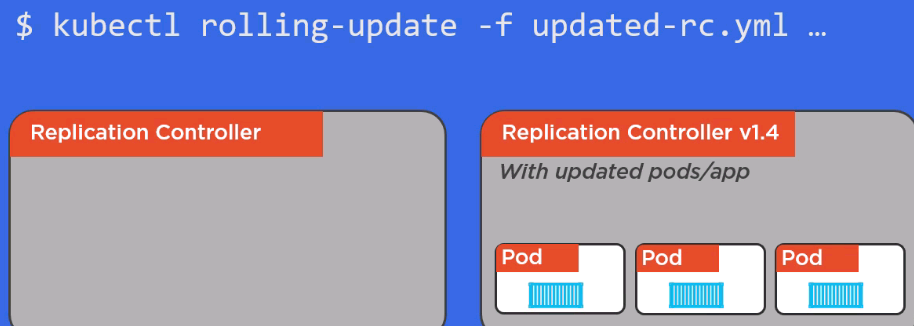
- latwo laczyc serwis i pods przez labelki, zmiana wersji , odcinanie

DEPLOYMENTS

Deployment tez przed yml, jest ponad replication controller



* Do tej pory update to nowy yml I nowy replication controller



* Teraz update dajemy w deployment.yml , przeniesienie podow do nowego replica gladko
* Stare replica Set sa trzymane jak by byl potrzebny rollback
* Wyrzucam stary replication controller, by wgrac replikacje przez deploy



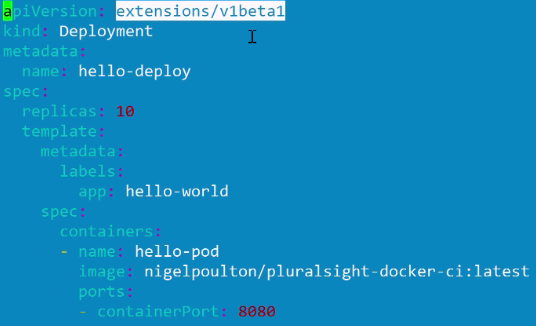
Teraz kubectl get pods – maja status terminating

Sprawdzam service



* Mam apke I chce ja teraz w inny sposob deploy

Przykladowy deploy.yml



- na gorze nowy apiVersion, nowy kind tez

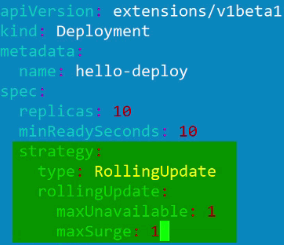
- tera bez deploy, bo w yml ,ze deployment



- kubectl get rs – pobieram replica set

Kuberctl describe rs – wiecej info

* By nowy deploy, zmieniamy image z ktorego brane
* Dodaje minReadySeconds, by kolejny 10s po wczesniejszym



* Nowa czesc to strategy ze update bedzie, kolejne zeby bralo po 1nym i po 1nym do pod wkladalo
* Przy aplikowaniu nowej wersji daje –record, by pamietalo zmiane, mozna bylo wrocic
* 
* Sprawdzam status danego deploy
* , mowi ile podmienione
* Mozemy probrac historie deploymentow dla danego
* 
* Kubecrl gt rs – pokaze nam liste deploy, nowy ma 10 pods, stary 0
* Jak zrobie descibe deployment, to widze, ze z nowego image
* 
* By wrocic komenda powrotu do poprzedniego
* 
* Jak zrobie get deploy to ciagle 2, wiec musze jeszcze status wycofac
* 
* Pamietac, ze po create jest chcek in to source control