infoShare ACADEMY

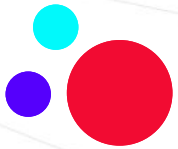
KUBERNETES – pt. 3

infoShare Academy

Piotr Ignatowski

infoShareAcademy.com

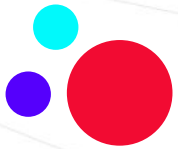
01. Co nowego dzisiaj?



Projekt na dziś

https://github.com/dockersamples/example-voting-app

Jest Redis, jest baza SQL, jest front. Będzie się działo :)

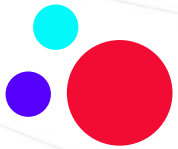


Narzędzie do testów - Apache Benchmark

https://httpd.apache.org/docs/2.4/programs/ab.html

Używać go będziemy do generowania ruchu testowego.

Na przykład:
ab -n 1000 -c 10 https://google.pl/



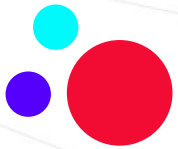
Helm

https://helm.sh/

Narzędzie do zarządzania instalacjami aplikacji. Daje możliwość budowania złożonych, wielostopniowych procesów instalacji/aktualizacji/usuwania.

“Przepisy” na instalację/aktualizację/usunięcie znajdują się w chartach. Te z kolei składowane są w repozytoriach (charts repositories).

helm repo add gitlab https://charts.gitlab.io/ - dodanie repozytorium do lokalnej bazy map (charts)



Operatory - idealne wykorzystanie kontrolerów

Operatory są kontrolerami bazującymi na obiektach definiowanych poza oficjalnym API k8s, CRD (Custom Resource Definition). Realizują logikę operacyjną związaną z zarządzaniem aplikacjami.

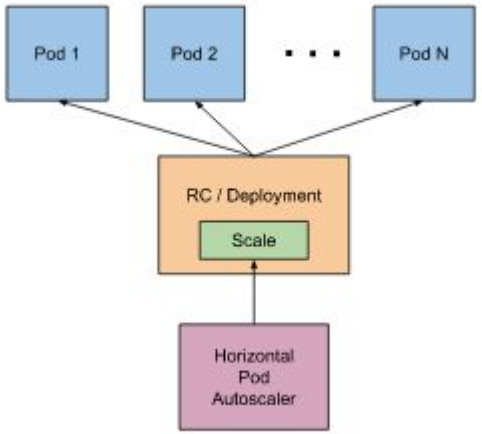
Zanim zaczniesz łamać głowę nad zarządzaniem jakimś znanym stackiem aplikacyjnym, zajrzyj na:

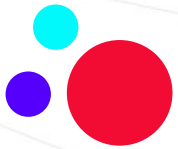
https://operatorhub.io/

Kilka nowych obiektów

HPA – moc skalowania

Horizontal Pod Autoscaler jest obiektem, który pozwala na podjęcie dynamicznej decyzji na temat liczebności replik deploymentu, czy statefulsetu biorąc pod uwagę zadane metryki (utylizację CPU/RAM lub custom metrics)



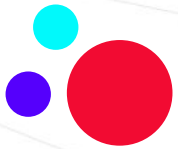


Kontenery nie powinny mieć nieograniczonych zasobów

W specyfikacji kontenerów umieścić można informacje o zasobach, które chcemy przydzielić uruchomionym procesom (spec.containers[].resources)

Przykład:

```
apiVersion: v1
kind: Pod
metadata:
  name: some-batch-job
spec:
  containers:
- name: busybox
   image: busybox
   command: [“/bin/bash”]
   args: [“sleep 999”]
   resources:
    requests:
      memory: "10Mi"
      cpu: "100m"
    limits:
      memory: "128Mi"
      cpu: "200m”
```



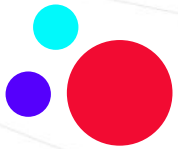
Request vs Limit

Zasoby z grupy request są gwarantowane - scheduler umieści danego poda tylko na workerze, który może zagwarantować zasoby zdefiniowane w sekcji request kontenerów.

Zasoby z grupy limit są ... no cóż ... limitem.

CPU jest kompresowalne, po osiągnięciu limitu proces jest kolejkowany przez kernel.

RAM nie jest z gumy - po osiągnięciu limitu kernel wysyła SIGTERM do kontenera.

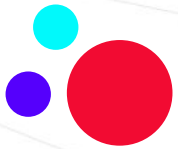


RBAC - tożsamość

W Kubernetesie są dwa rodzaje kont - normlani użytkownicy i konta usługowe (service accounts).

Nie ma obiektów reprezentujących normalnych użytkowników, tożsamości brane są z certyfikatów generowanych przez administratora (https://kubernetes.io/docs/reference/access-authn-authz/certificate-signing-requests/#normal-user) lub z pluginów uwierzytelniających.

Konta usługowe są zwykłymi obiektami przypiętymi do namespace



RBAC - czy mogę?

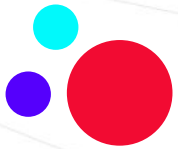
Usage:

kubectl auth can-i VERB [TYPE | TYPE/NAME | NONRESOURCEURL] [options]

Na przykład:

kubectl auth can-i list secrets -n kube-system

kubectl auth can-i list secrets --as=system:serviceaccount:vote:event-listener



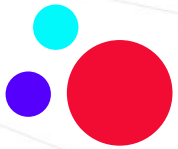
RBAC - tożsamość

Komenda: kubectl create serviceaccount event-listener

Tworzy obiekt:

apiVersion: v1
kind: ServiceAccount
metadata:
  name: event-listener
secrets:
- **name**: event-listener-token-4jmzf

W podanym sekrecie (event-listener-token-4jmzf) znajduje się token konta usługowego.

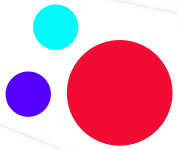


RBAC - definiowanie uprawnień

Uprawnienia o zasięgu namespace definiowane są w obiektach typu Role:

```
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
namespace: vote
metadata:
  name: event-listener
rules:
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - list
```

Uprawnienia na cały klaster definiowane są w ClusterRole.



RBAC - powiązanie tożsamości i uprawnień

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: list-events-sa
  namespace: vote
subjects:
- kind: ServiceAccount
  name: event-listener
  namespace: vote
roleRef:
  kind: Role
  name: event-listener
  apiGroup: rbac.authorization.k8s.io
```

RoleBinding jest przypisany do namespace. Można przypisać ClusterRole za pomocą RoleBinding tworząc lokalne uprawnienia.

ClusterRoleBinding daje uprawnienia na poziomie całego klastra

Dziękuję za uwagę!

infoShareAcademy.com