infoShare ACADEMY

KUBERNETES – one, to rule them all

infoShare Academy

Michał Wójtowicz

infoShareAcademy.com

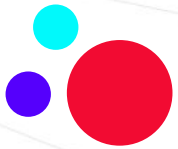
01. O procesach, kontenerach...

Małe przypomnienie i dyskusja wstępna

Metody izolacji procesów

te same blachy, oddzielne kernele (np. KVM w AWS Lambda - https://firecracker-microvm.github.io/

ten sam kernel, izolacja na poziomie kernela (BSD jails https://docs.freebsd.org/en/books/handbook/jails/, Linux containers https://linuxcontainers.org/)



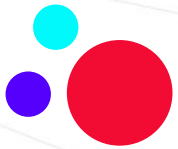
Jak to było z kontenerami?

- Linux kernel izoluje procesy w przestrzeni nazw:
  - Cgroups
  - IPC
  - Network
  - Mount
  - PID
  - Time
  - User
  - UTS

  https://man7.org/linux/man-pages/man7/namespaces.7.html

- Cgroups pozwalają na hierarchiczne zarządzanie utylizacją zasobów

  https://man7.org/linux/man-pages/man7/cgroups.7.html



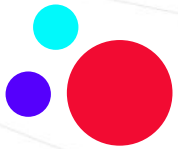
Zabawa z przestrzeniami nazw

- lsns - narzędzie do listowania przestrzeni nazw

  https://man7.org/linux/man-pages/man8/lsns.8.html

- nsenter - narzędzie do uruchamiania procesu w wybranej/wybranych przestrzeniach nazw

  https://man7.org/linux/man-pages/man1/nsenter.1.html



Co uruchamia kontenery (API vs Platforma)?

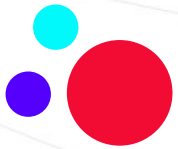
- Docker - https://www.docker.com/
- Podman - https://podman.io/
- RunC - https://github.com/opencontainers/runc
- LXC/LXD - https://linuxcontainers.org/
- CRI-O - https://cri-o.io/
- ContainerD - https://containerd.io/

02.

A gdyby tak zarządzać nimi hurtowo...

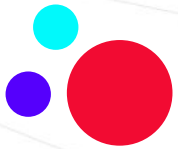
Czyli jak zrobić, żeby się nie narobić

infoShare ACADEMY



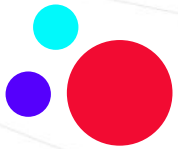
Orkiestratory – zarządzanie procesami na dużą skalę

- Docker Swarm - https://docs.docker.com/engine/swarm/
- Hashicorp Nomad - https://www.nomadproject.io/
- Apache Mesos - https://mesos.apache.org/
- Google Borg - https://research.google/pubs/pub43438/
- Kubernetes - https://kubernetes.io/



Co one potrafią?

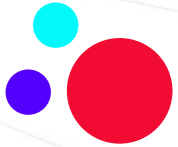
- Zarządzają uruchamianiem procesów na dużej ilości maszyn
- Tworzą “darmowe” abstrakcje, które pozwalają:
  - zapominać o infrastrukturze pod spodem
  - granularnie kontrolować środowisko uruchomieniowe
  - łączyć procesy w grupy logiczne
  - zarządzać sieciami w zupełnie nowy sposób
- Dają użytkownikom API umożliwiające dynamiczne zarządzanie konfiguracją
- Zarządzają cyklem życia procesów



Kubernetes – jest w nim coś wyjątkowego?

- Rozproszone kontrolery (polecam lekturę świetnej analizy zagadnienia: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41684.pdf) gwarantujące wysokie HA
- Wysoka zgodność środowisk on-prem ze środowiskami chmurowymi (z pewnymi wyjątkami)
- Przenaszalność środowisk
- Świetnie zaprojektowane API
- Gigantyczne wsparcie społeczności OpenSource
- … i nie mniej gigantyczne wsparcie finansowe ze strony graczy pokroju Google :)

03. W takim razie kto to ten Kubernetes?



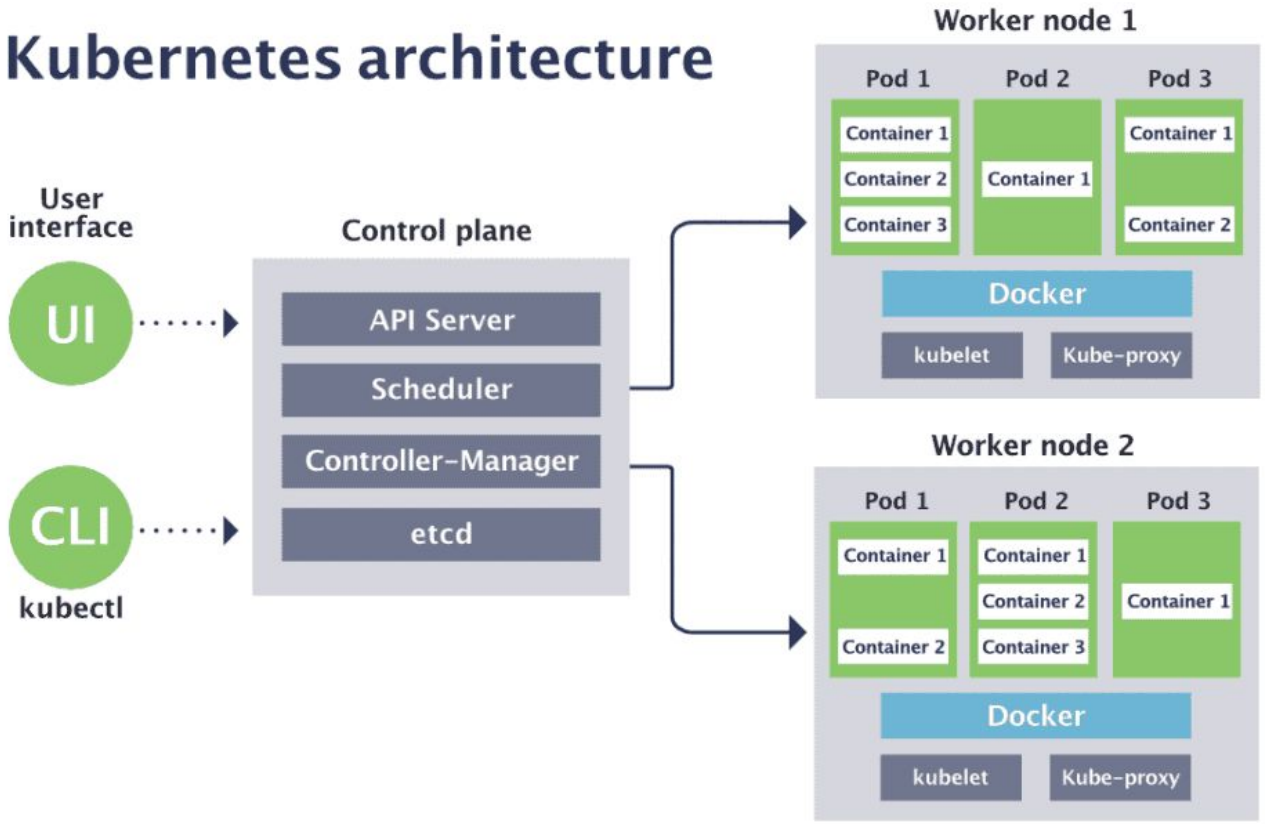
Kubernetes to przede wszystkim bardzo popularny projekt OpenSource

Znaleźć go można tutaj:

https://github.com/kubernetes/kubernetes

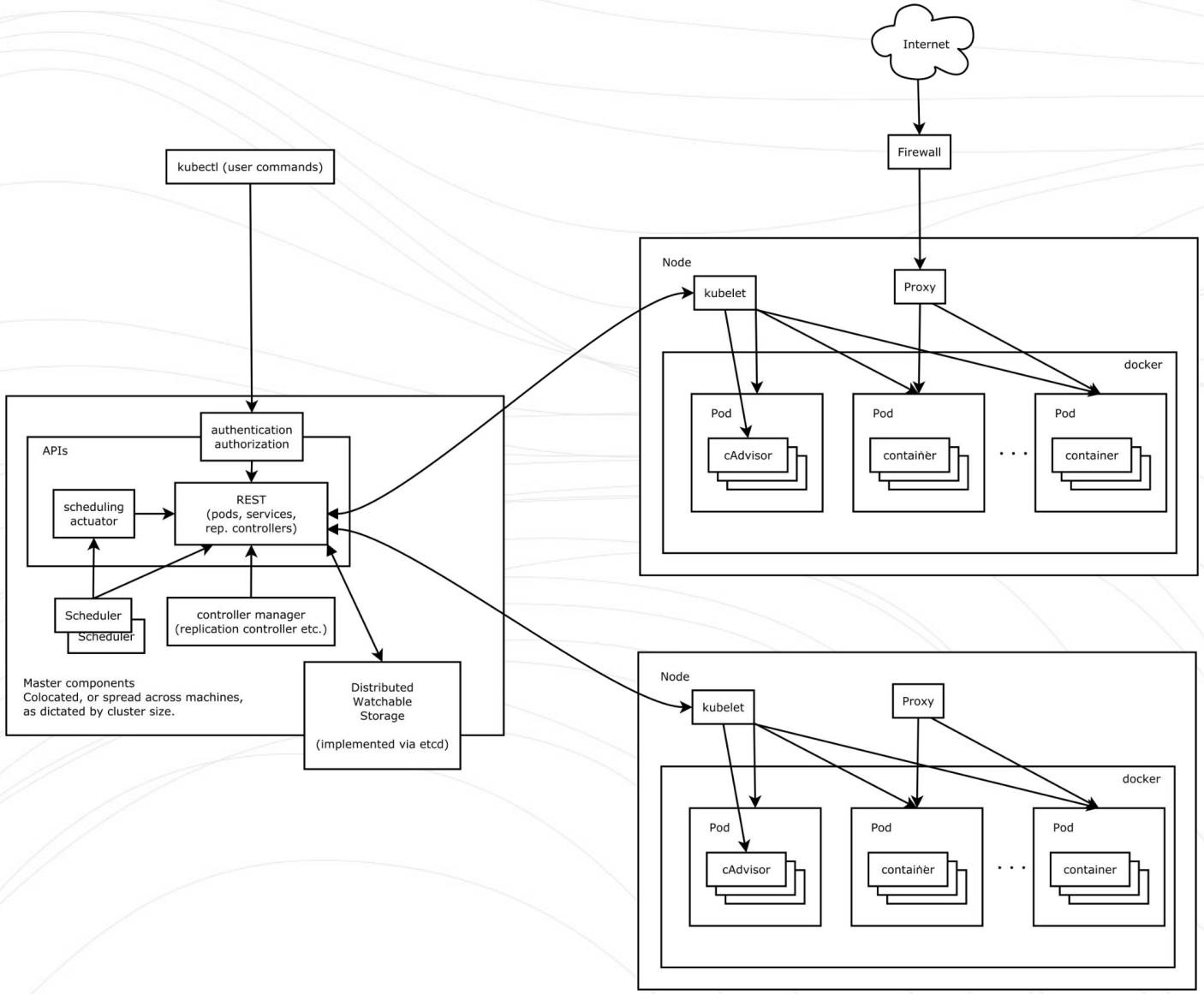
Każdy może zrobić git clone, skompilować projekt i rozpocząć zabawę.

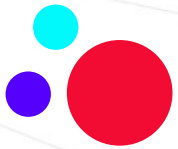
A tak wygląda jego architektura



https://www.cncf.io/blog/2019/08/19/how-kubernetes-works/

A tak wygląda jego architektura



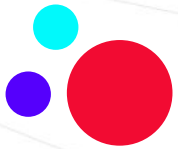


Jak się tym pobawić lokalnie?

Polecam Minikube - pozwala na uruchomienie Kubernetesa używając Dockera, KVM, VirtualBoxa i wielu innych.

https://github.com/kubernetes/minikube

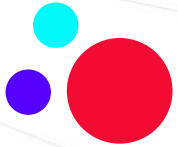
‘minikube start’ i jesteście w domu :)



Na każdym węźle klastra uruchomiony jest kubelet

czyli program będący agentem wykonawczym - rejestruje węzeł w klastrze, monitoruje i raportuje jego stan, monitoruje API szukając zadań związanych z uruchamianiem i zabijaniem procesów, pełni rolę lokalnego kontrolera monitorującego stan procesów.

A propos procesów - w Kubernetesie nie traktujemy każdego kontenera jako zupełnie oddzielny byt. Łączymy je w logiczne twory, nazywane podami (kapsułki).

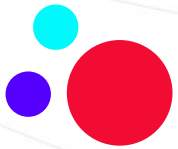


POD - zupełnie jak rodzina w społeczeństwie :)

Pod jest co najmniej jednym kontenerem uruchomionym we współdzielonych przestrzeniach nazw IPC oraz Network (istnieje też możliwość współdzielenia PID:

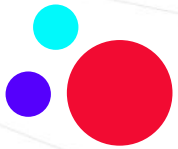
https://kubernetes.io/docs/tasks/configure-pod-container/share-process-namespace/

Czyli kontenery w ramach jednego poda zawsze uruchomione są na tej samej maszynie, widzą się po localhost i mogą korzystać ze wspieranych przez OS IPC (Inter Process Communication).



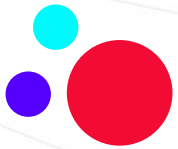
kube-proxy

Lokalny proces zajmujący się implementacją routingu pomiędzy podami, węzłami i abstrakcjami używanymi do opisywania komunikacji sieciowej - czyli endpointami i serwisami.



Ekipa rządząca - czyli control plane

- API serwer - stanowi fasadę w komunikacji z warstwą persystencji - czyli bazą ETCD. Każdy kontroler w klastrze (w tym kubelet) używa uniwersalnej i dobrze przez każdego rozumianej reprezentacji obiektów. API serwer zapewnia operacje CRUD na obiektach reprezentujących zasoby.
- kube-controller-manager - jest demonem, w ramach którego uruchomione są control loops (naprawdę brakuje mi polskiego odpowiednika) podstawowych kontrolerów
- ETCD - wspomniana wcześniej baza danych, warstwa persystencji
- kube-scheduler - kontroler odpowiedzialny za podejmowanie decyzji o tym, gdzie uruchamiać pody
- cloud-controller-manager (jak nazwa wskazuje, tylko w chmurach) - kontroler zajmujący się zasobami od dostawcy chmury

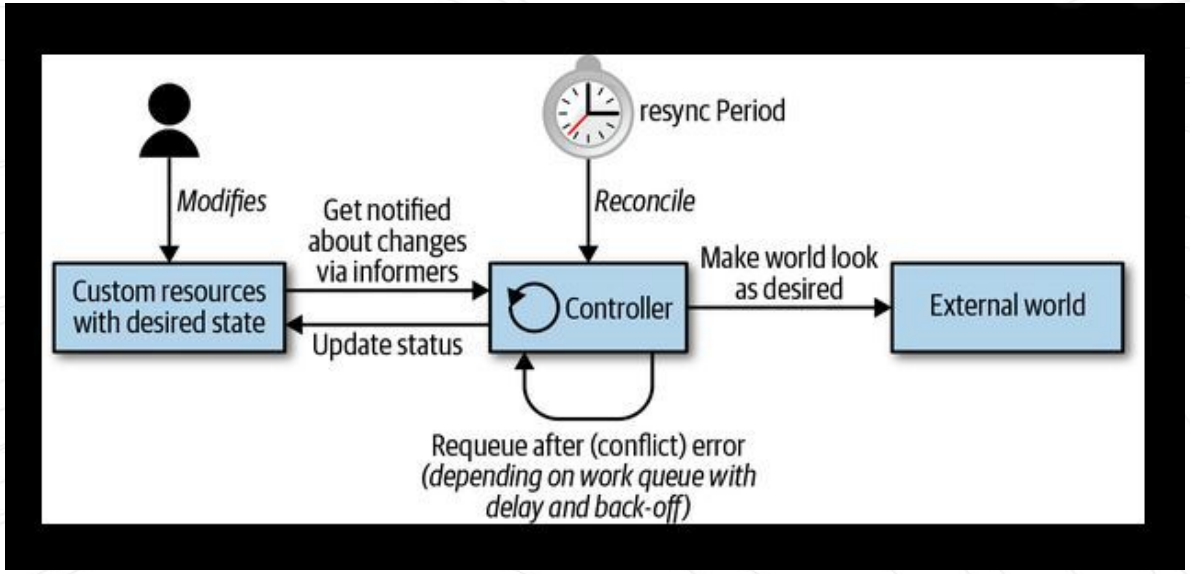


Klienci - kubectl oraz dashboard

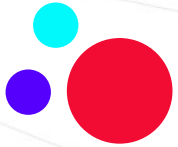
- API serwer przyjmuje zapytania protokołem HTTP
- Można z niego korzystać ręcznie (na przykład via cURL albo jeszcze mocniej - netcat :))
- Można też użyć klienta CLI - kubectl
- Albo portalu webowego, kubernetes-dashboard

Ważna dygresja: control loop, czyli wyjątkowość Kubernetesa

- Kubernetes nie jest systemem deklaratywnym - to grupa rozproszonych kontrolerów, które małymi krokami osiągają opisany stan

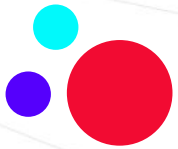


API



kubectl api-resources

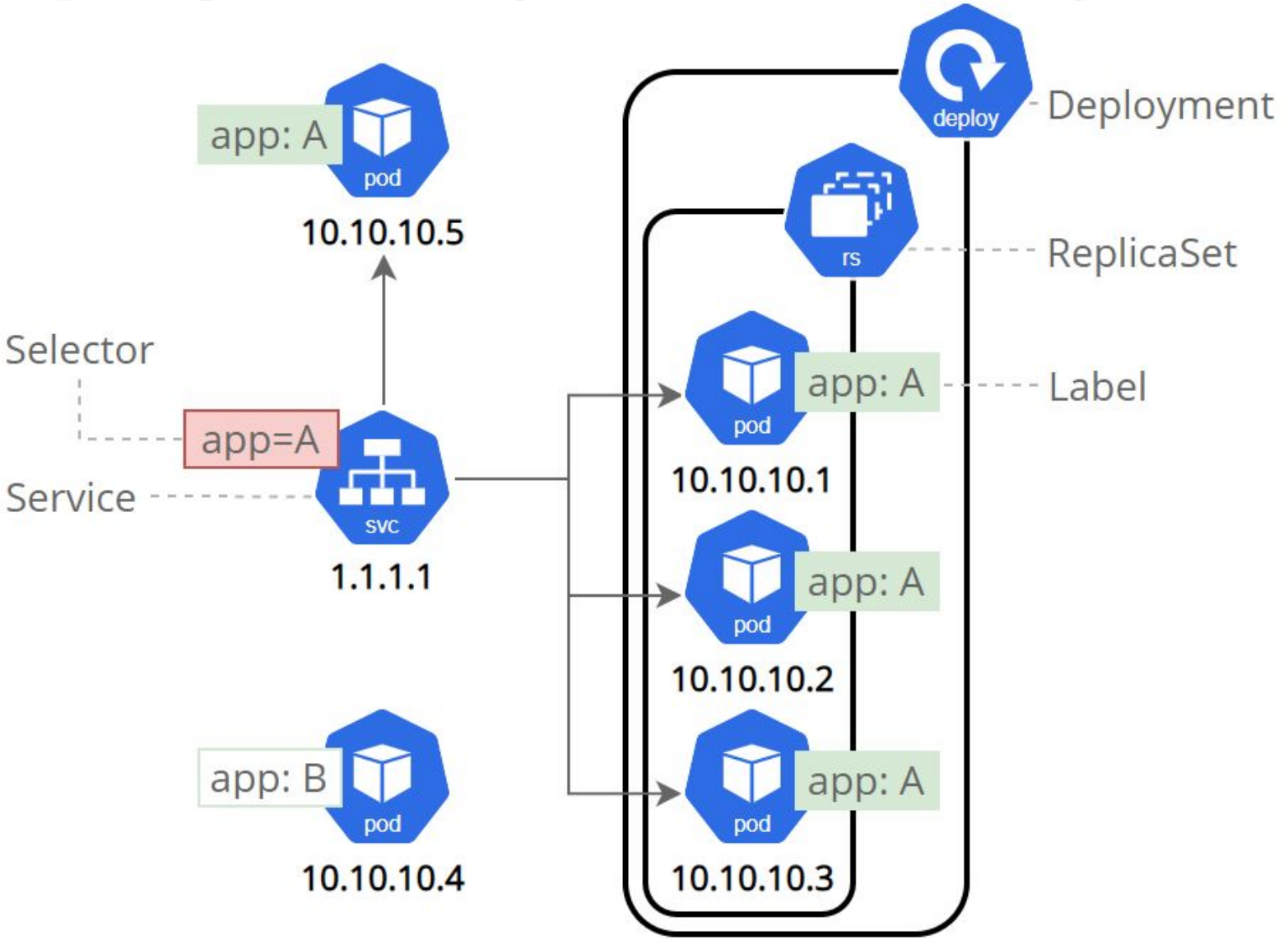
```
~ kubectl api-resources
NAME                              SHORTNAMES   APIVERSION                             NAMESPACED   KIND
bindings                                       v1                                     true         Binding
componentstatuses                 cs           v1                                     false        ComponentStatus
configmaps                        cm           v1                                     true         ConfigMap
endpoints                         ep           v1                                     true         Endpoints
events                            ev           v1                                     true         Event
limitranges                       limits       v1                                     true         LimitRange
namespaces                        ns           v1                                     false        Namespace
nodes                             no           v1                                     false        Node
persistentvolumeclaims            pvc          v1                                     true         PersistentVolumeClaim
persistentvolumes                 pv           v1                                     false        PersistentVolume
pods                              po           v1                                     true         Pod
podtemplates                                   v1                                     true         PodTemplate
replicationcontrollers            rc           v1                                     true         ReplicationController
resourcequotas                    quota        v1                                     true         ResourceQuota
secrets                                        v1                                     true         Secret
serviceaccounts                   sa           v1                                     true         ServiceAccount
services                          svc          v1                                     true         Service
mutatingwebhookconfigurations                  admissionregistration.k8s.io/v1        false        MutatingWebhookConfiguration
validatingwebhookconfigurations                admissionregistration.k8s.io/v1        false        ValidatingWebhookConfiguration
customresourcedefinitions         crd,crds     apiextensions.k8s.io/v1                false        CustomResourceDefinition
apiservices                                    apiregistration.k8s.io/v1              false        APIService
controllerrevisions                            apps/v1                                true         ControllerRevision
daemonsets                        ds           apps/v1                                true         DaemonSet
deployments                       deploy       apps/v1                                true         Deployment
replicasets                       rs           apps/v1                                true         ReplicaSet
statefulsets                      sts          apps/v1                                true         StatefulSet
tokenreviews                                   authentication.k8s.io/v1               false        TokenReview
localsubjectaccessreviews                      authorization.k8s.io/v1                true         LocalSubjectAccessReview
selfsubjectaccessreviews                       authorization.k8s.io/v1                false        SelfSubjectAccessReview
selfsubjectrulesreviews                        authorization.k8s.io/v1                false        SelfSubjectRulesReview
subjectaccessreviews                           authorization.k8s.io/v1                false        SubjectAccessReview
horizontalpodautoscalers          hpa          autoscaling/v1                         true         HorizontalPodAutoscaler
cronjobs                          cj           batch/v1beta1                          true         CronJob
jobs                                           batch/v1                               true         Job
certificatesigningrequests        csr          certificates.k8s.io/v1                 false        CertificateSigningRequest
leases                                         coordination.k8s.io/v1                 true         Lease
endpointslices                                 discovery.k8s.io/v1beta1               true         EndpointSlice
events                            ev           events.k8s.io/v1                       true         Event
ingresses                         ing          extensions/v1beta1                     true         Ingress
```



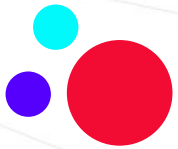
Z czego będziemy korzystali?

- Deployment (https://kubernetes.io/docs/concepts/workloads/controllers/deployment/)
- Service (https://kubernetes.io/docs/concepts/services-networking/service/)
- InitContainer (https://kubernetes.io/docs/concepts/workloads/pods/init-containers/)
- StatefulSet (https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/)
- ConfigMap (https://kubernetes.io/docs/concepts/configuration/configmap/)
- Secret (https://kubernetes.io/docs/concepts/configuration/secret/)
- Ingress (https://kubernetes.io/docs/concepts/services-networking/ingress/)

Struktura zasobów z wykorzystaniem serwisów



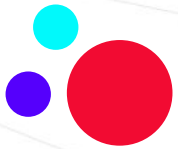
Miejsce Ingress w klastrze



Projekt - v1

Napiszmy system składający się z:

- bazy danych PostgreSQL
- cache'a typu Redis
- API, które łączy się z wyżej wymienionymi
- frontendu, który pobiera dane z API



Projekt - v2

Rozszerzymy projekt o:

- drugą wersję API
- drugi frontend, który uruchamia się w określonych warunkach, jak domena lub ścieżka http

Dziękuję za uwagę!

infoShareAcademy.com