LUNTRY

Kubernetes Resource Model (KRM): Everything-as-Code

Дмитрий Евдокимов

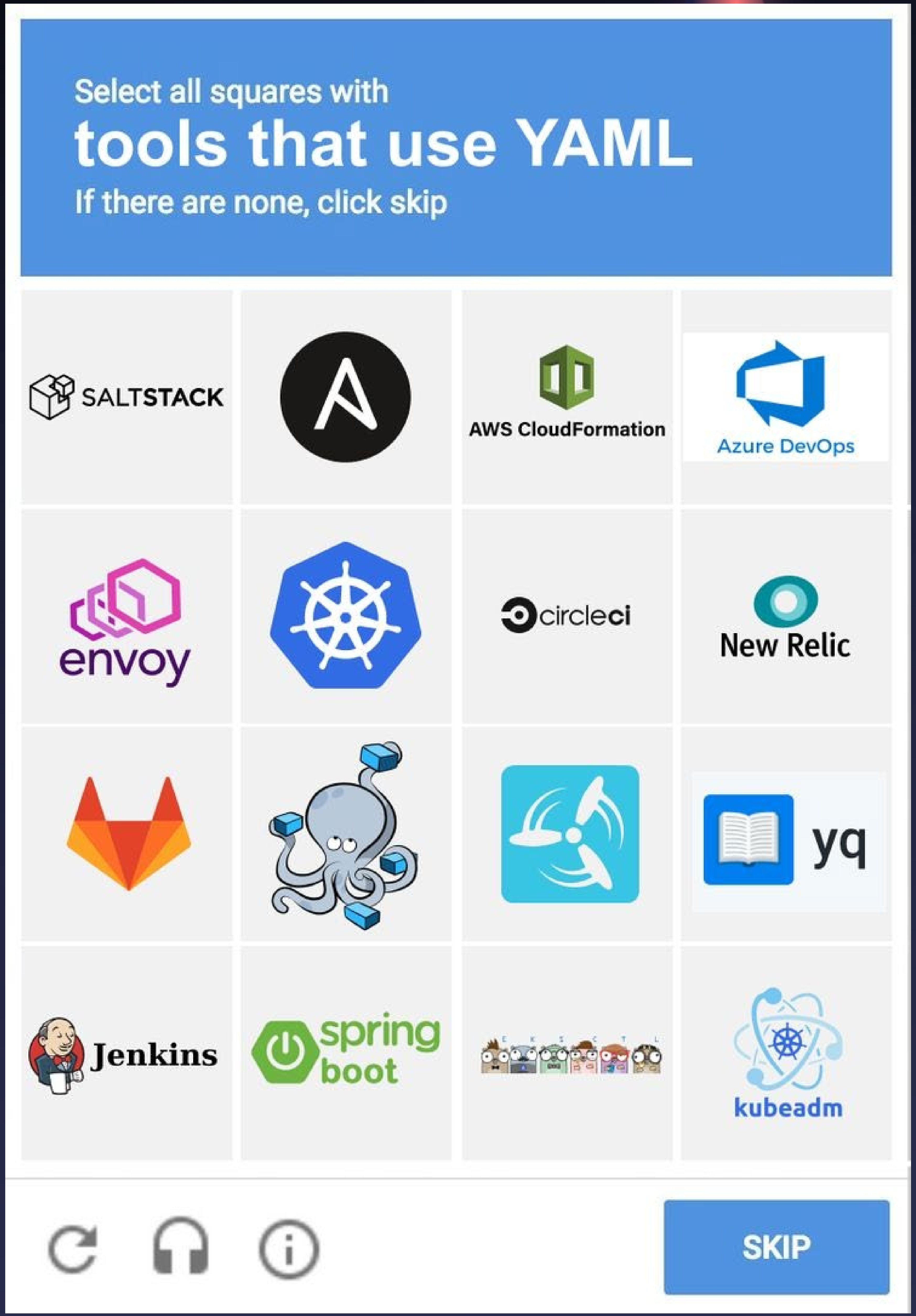
Founder, CTO Luntry

Whoami

- Основатель, технический директор [Luntry]
- Соорганизатор конференций ZeroNights, DEFCON Russia (#7812)
- В прошлом — автор статей журнала «ХАКЕР»
- Автор Telegram-канала [k8s (in)security]
- Автор тренинга «Cloud-Native безопасность в Kubernetes»
- Спикер: BlackHat (USA, UAE), HITB (Singapore), ZeroNights (Russia), HackInParis (France), Confidence (Poland), SAS (Mexico, Spain), PHDays (Russia), DevOpsConf (Russia), Kuber Conf (Russia), HighLoad (Russia) и др.

План

1. YAML
2. YAML
3. YAML
4. YAML
5. YAML
6. YAML
7. YAML
8. YAML



Давайте разберемся с названием доклада

YOU DINOSAURS HAVE PROBABLY NEVER SEEN YAML
THIS MAKES ME SO EFFICIENT I CAN SAVE HOURS EVERY DAY.
WHAT DO YOU DO WITH ALL THE SPARE TIME?
I WRITE MORE CONFIGURATION IN YAML
WOW! THEN YOU CAN SAVE EVEN MORE TIME!
© 1990 United Feature Syndicate, Inc.
4-25

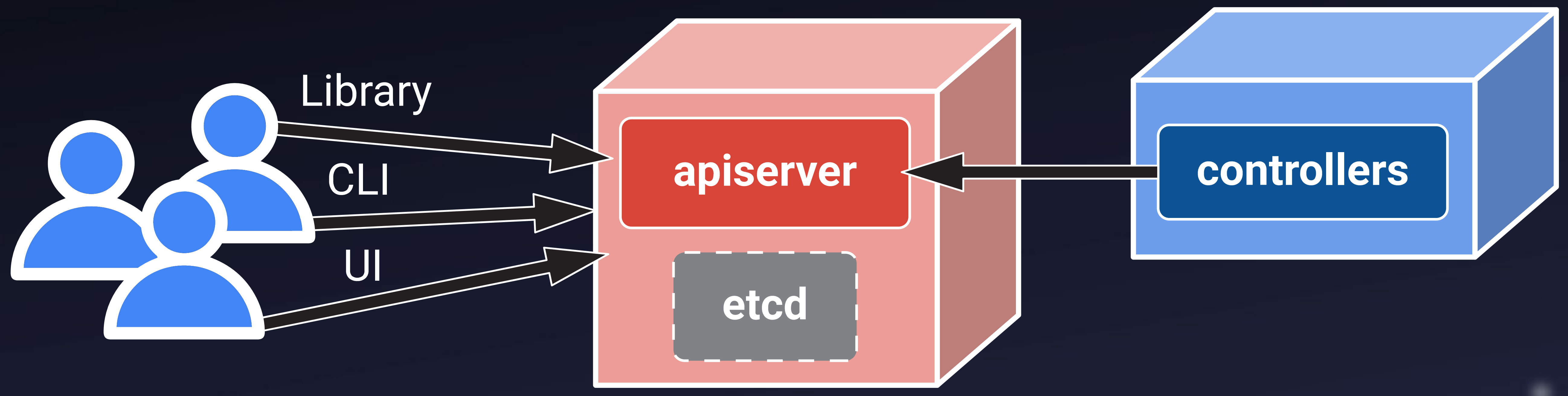
Everything-as-Code (EaC)

EaC переносит приоритет с ручных, повторяющихся задач на рабочие процессы, основанные на конечных целях и требуемых состояниях.

Это приближает процесс управления инфраструктурой к отработанным процессам создания программного обеспечения.

- Infrastructure as Code
- Configuration as Code
- Documentation as Code
- Observability as Code
- Pipeline as Code
- Security as Code
- Policy as Code
- Compliance as Code
- ...

Kubernetes Resource Model (KRM)



“Kubernetes Resource Model (KRM) – is the declarative format you use to talk to the Kubernetes API. Often, KRM is expressed as YAML.”

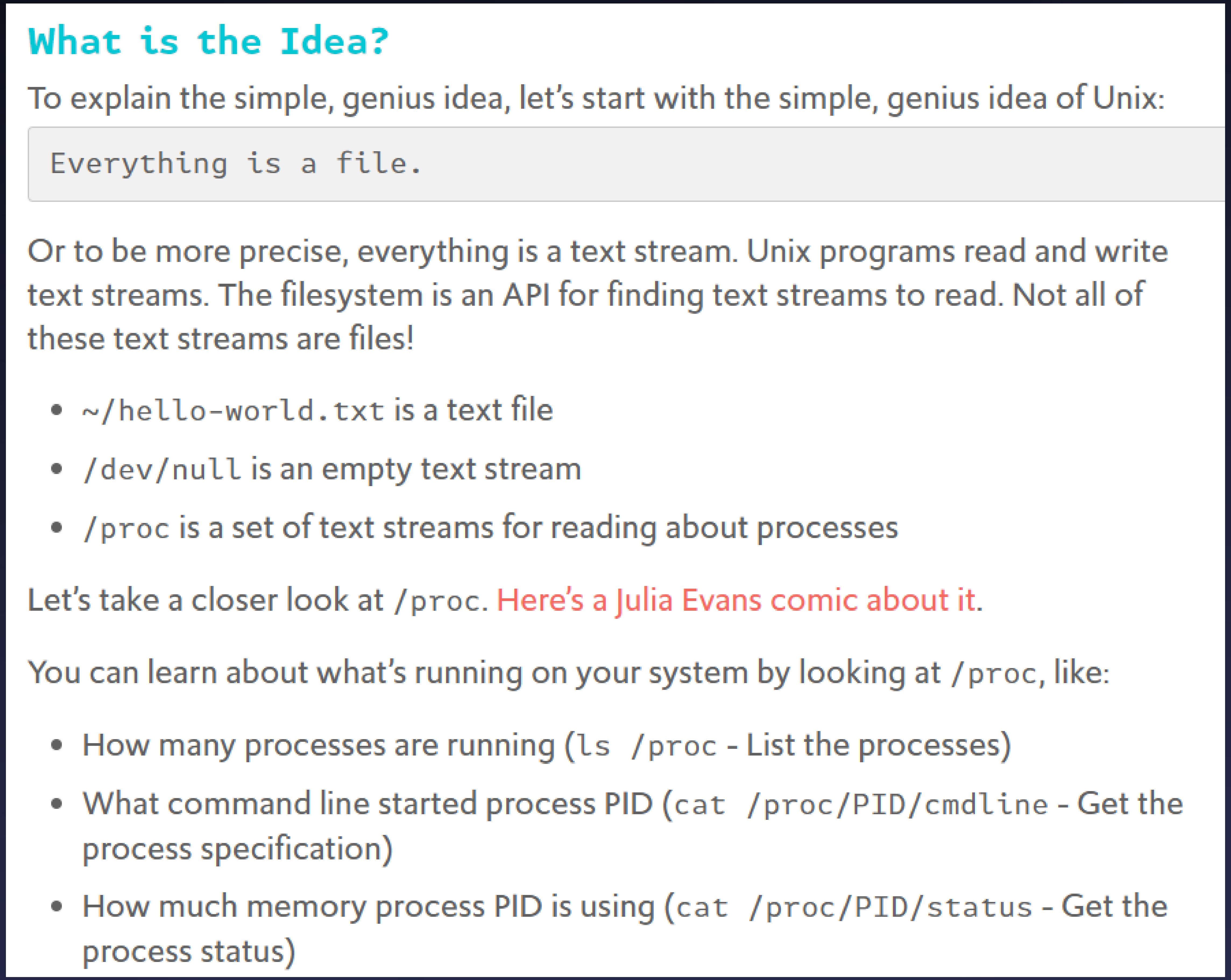
"Build a platform with KRM", Google Cloud

"Kubernetes API resource specifications are designed for humans to directly author and read as declarative configuration data, as well as to enable composable configuration tools and automated systems to manipulate them programmatically.“

“Kubernetes design proposals”

YAML – всему голова

Kubernetes – это ядро Linux XXI-го века



What is the Idea?

To explain the simple, genius idea, let's start with the simple, genius idea of Unix:

```
Everything is a file.
```

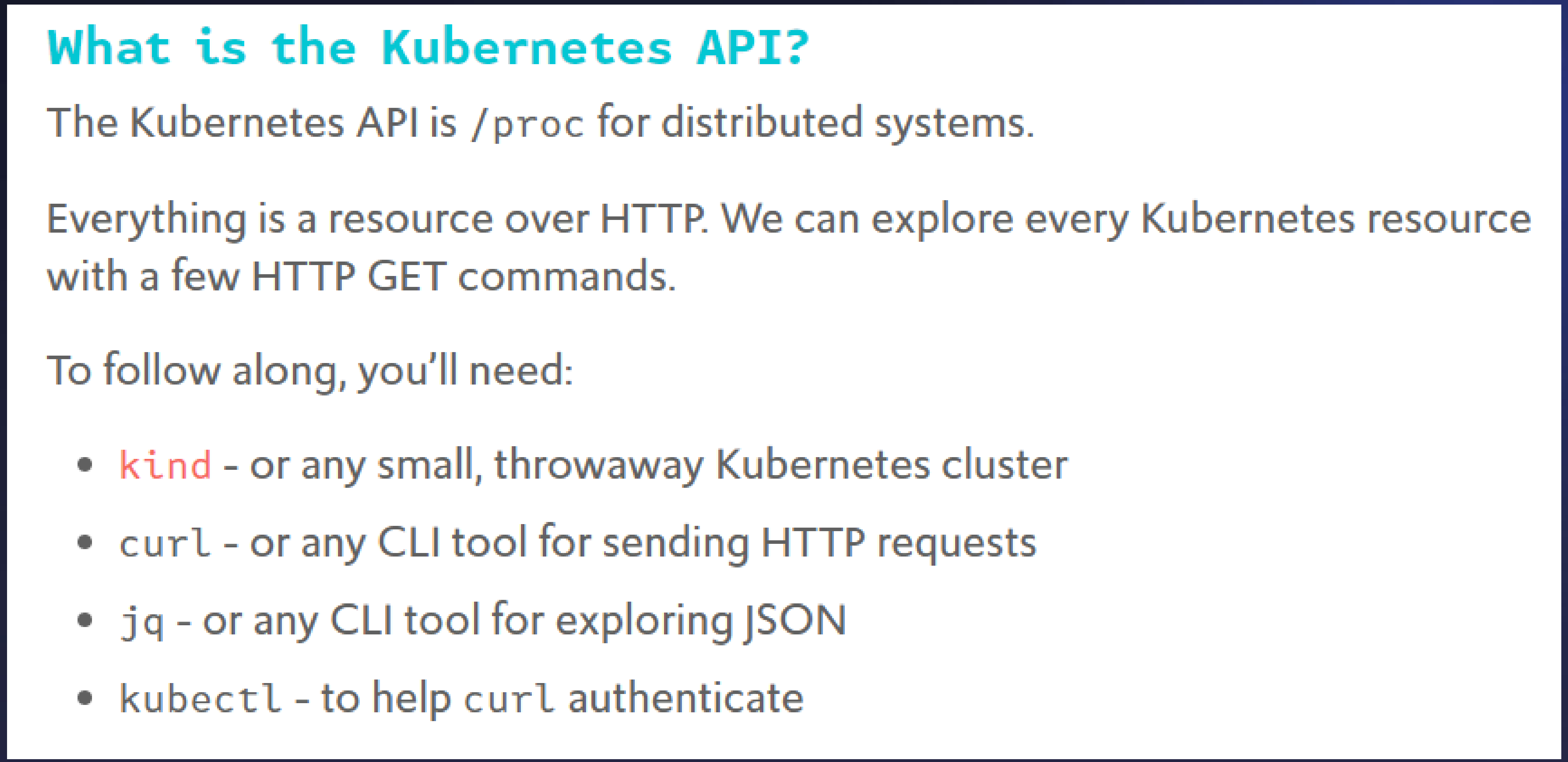
Or to be more precise, everything is a text stream. Unix programs read and write text streams. The filesystem is an API for finding text streams to read. Not all of these text streams are files!

- `~/hello-world.txt` is a text file
- `/dev/null` is an empty text stream
- `/proc` is a set of text streams for reading about processes

Let's take a closer look at `/proc`. Here's a Julia Evans comic about it.

You can learn about what's running on your system by looking at `/proc`, like:

- How many processes are running (`ls /proc` - List the processes)
- What command line started process PID (`cat /proc/PID/cmdline` - Get the process specification)
- How much memory process PID is using (`cat /proc/PID/status` - Get the process status)



What is the Kubernetes API?

The Kubernetes API is `/proc` for distributed systems.

Everything is a resource over HTTP. We can explore every Kubernetes resource with a few HTTP GET commands.

To follow along, you'll need:

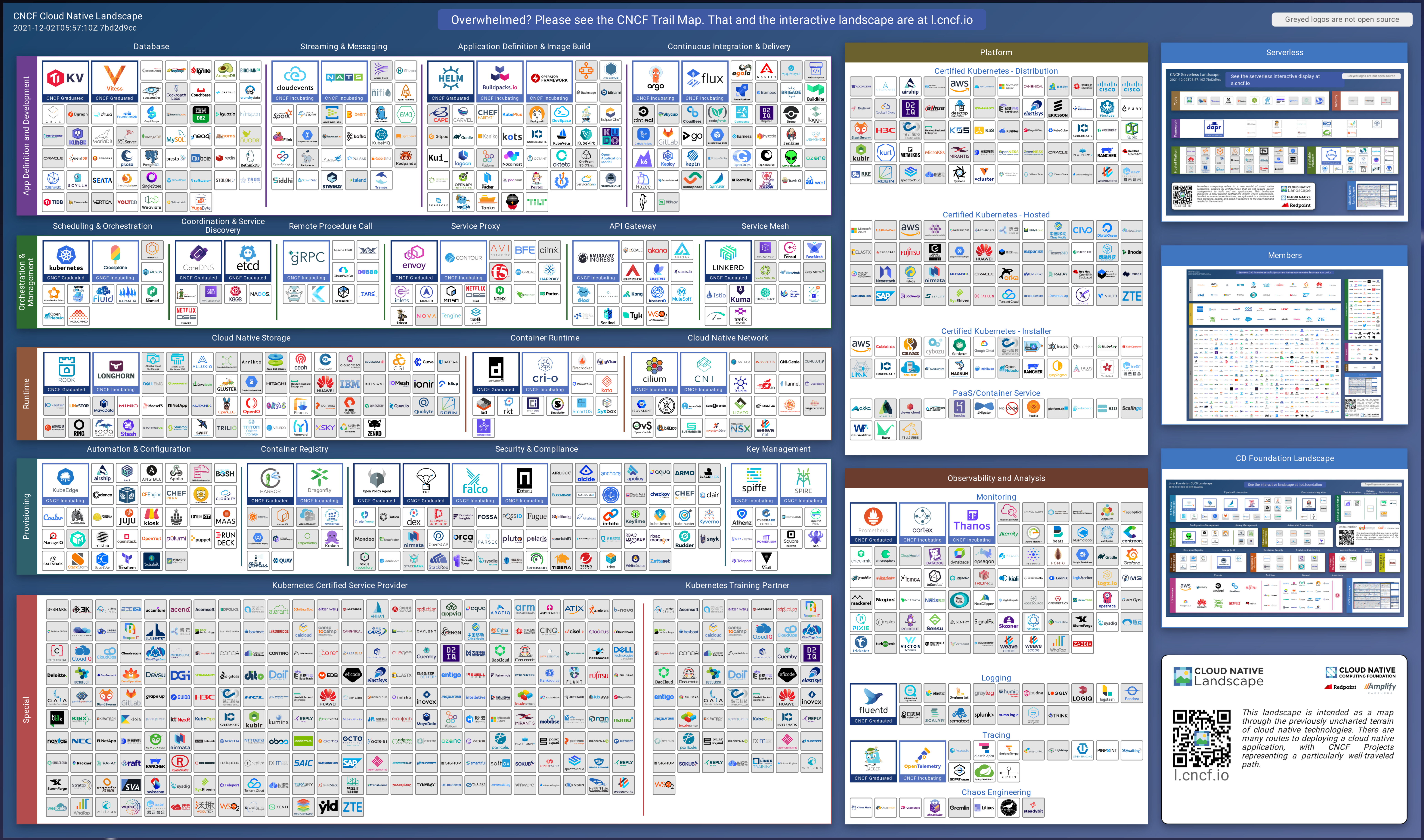
- `kind` - or any small, throwaway Kubernetes cluster
- `curl` - or any CLI tool for sending HTTP requests
- `jq` - or any CLI tool for exploring JSON
- `kubectl` - to help `curl` authenticate

"Kubernetes is so Simple You Can Explore it with Curl"

Kubernetes operators и Custom resources

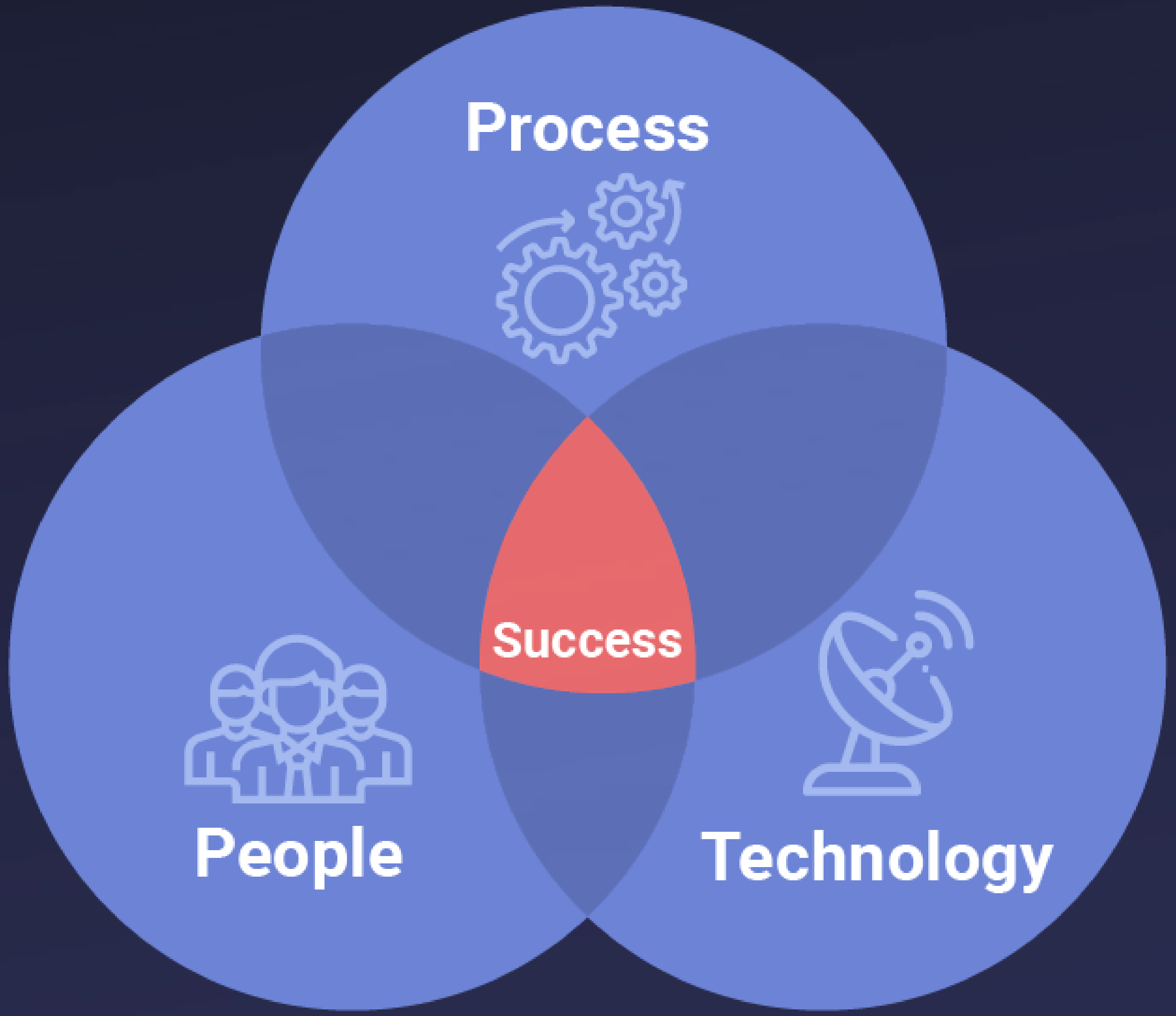
- **CNI:**
  - Cilium (CiliumEndpoint, CiliumClusterwideNetworkPolicy, CiliumNetworkPolicy, ...)
  - Calico (NetworkSet, HostEndpoint, IPPool, GlobalNetworkPolicy, ...)
- **ServiceMeshs:**
  - Istio (AuthorizationPolicy, Gateway, ServiceEntry, Sidecar, ...)
  - linkerd2 (Servers, ServerAuthorizations, ...)
- **API gateway:**
  - Gloo (Gateway, Route, Ingress, Certificate, Proxy, ...)
- **Policy Engines:**
  - Kyverno (PolicyReport, ClusterPolicyReport)
  - Gatekeeper OPA (K8sAllowedRepos, AssignMetadata, ConstraintPodStatus, ...)
- **Database:**
  - Redis Operator (RedisCluster, Redis)
  - ClickHouse (ClickHouseInstallation, ClickHouseInstallationTemplate, ClickHouseOperatorConfiguration)
- **Monitoring:**
  - Prometheus Operator (ServiceMonitor, PodMonitor, PrometheusRule, ...)
- **Pipeline:**
  - Tekton (Task, Run, Pipeline,...)
- **GitOps:**
- ArgoCD (AppProject, Application, ...)
- Flux2 (HelmRelease, ...)
- **Serverless:**
  - Knative (Broker, Trigger, EventType, ...)
- **Security:**
  - Starboard (CISKubeBenchReport, ConfigAuditReport, KubeHunterReport, VulnerabilityReport, ...)
  - Kubernetes Security Profiles Operator (AppArmorProfile, SelinuxProfile, SeccompProfile, ProfileBinding, ..)
- **Infrastructure:**
  - ClusterAPI (KubeadmControlPlane, Cluster, Machine, MachineSet, MachineDeployment, ...)
  - Crossplane (Provider, Configuration, ControllerConfig,Composition, ...)

Cloud Native Landscape


CNCF Cloud Native Landscape
2021-12-02T05:57:10Z 7bd2d9cc
Overwhelmed? Please see the CNCF Trail Map. That and the interactive landscape are at l.cncf.io
Greyed logos are not open source
App Definition and Development
Database
Streaming & Messaging
Application Definition & Image Build
Continuous Integration & Delivery
Orchestration & Management
Scheduling & Orchestration
Coordination & Service Discovery
Remote Procedure Call
Service Proxy
API Gateway
Service Mesh
Runtime
Cloud Native Storage
Container Runtime
Cloud Native Network
Provisioning
Automation & Configuration
Container Registry
Security & Compliance
Key Management
Special
Kubernetes Certified Service Provider
Kubernetes Training Partner
Platform
Certified Kubernetes - Distribution
Certified Kubernetes - Hosted
Certified Kubernetes - Installer
PaaS/Container Service
Observability and Analysis
Monitoring
Logging
Tracing
Chaos Engineering
Serverless
Members
CD Foundation Landscape
CLOUD NATIVE Landscape
CLOUD NATIVE COMPUTING FOUNDATION
This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.
l.cncf.io

Эра DevSecOps

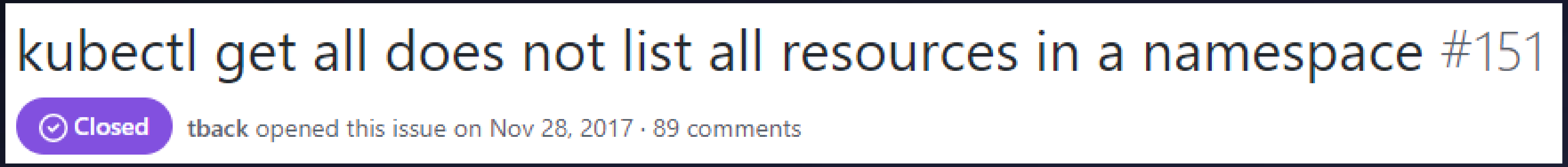
- С кластером работает много разных команд
- Команды должны видеть общую картину и разговаривать друг с другом на одном языке
- Команды должны помогать друг другу, а не мешать



Пример: Кто сломал ?!

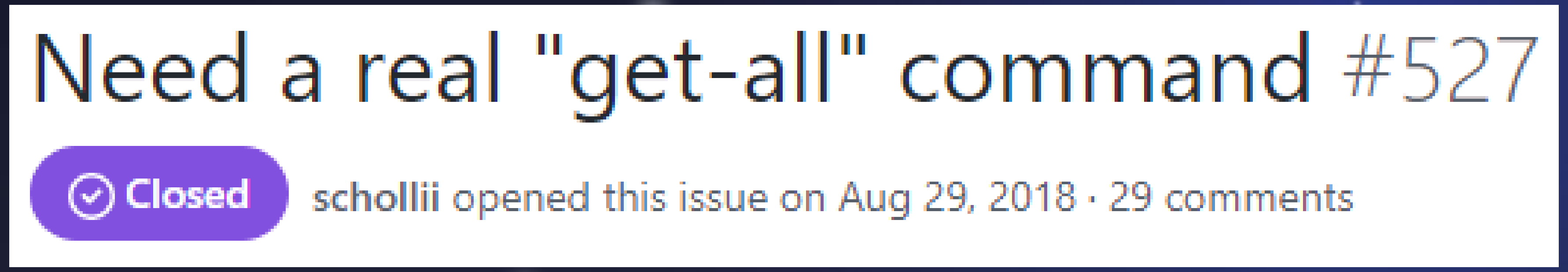
Занимательные факты

kubeclt -n namespace get all - не работает!



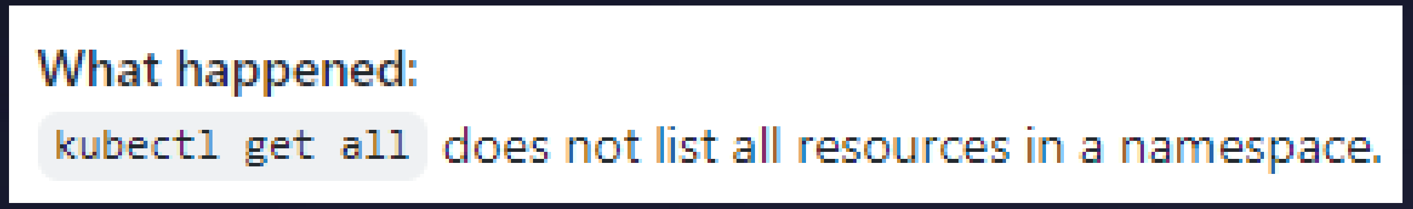
kubectl get all does not list all resources in a namespace #151

Closed tback opened this issue on Nov 28, 2017 · 89 comments



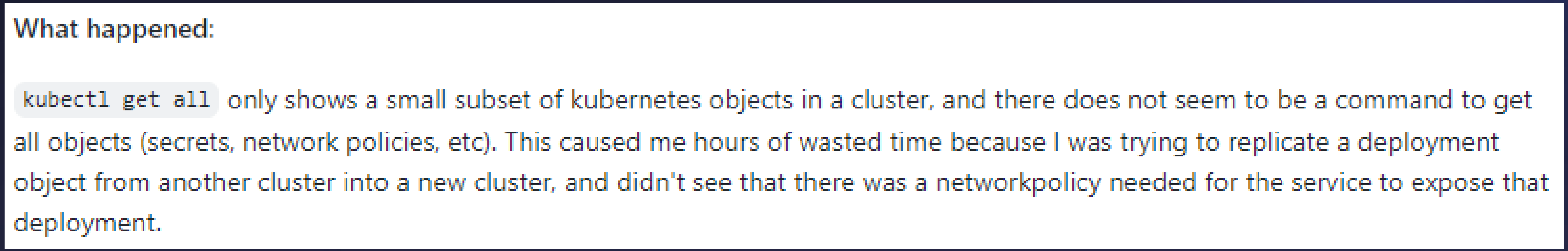
Need a real "get-all" command #527

Closed schollii opened this issue on Aug 29, 2018 · 29 comments



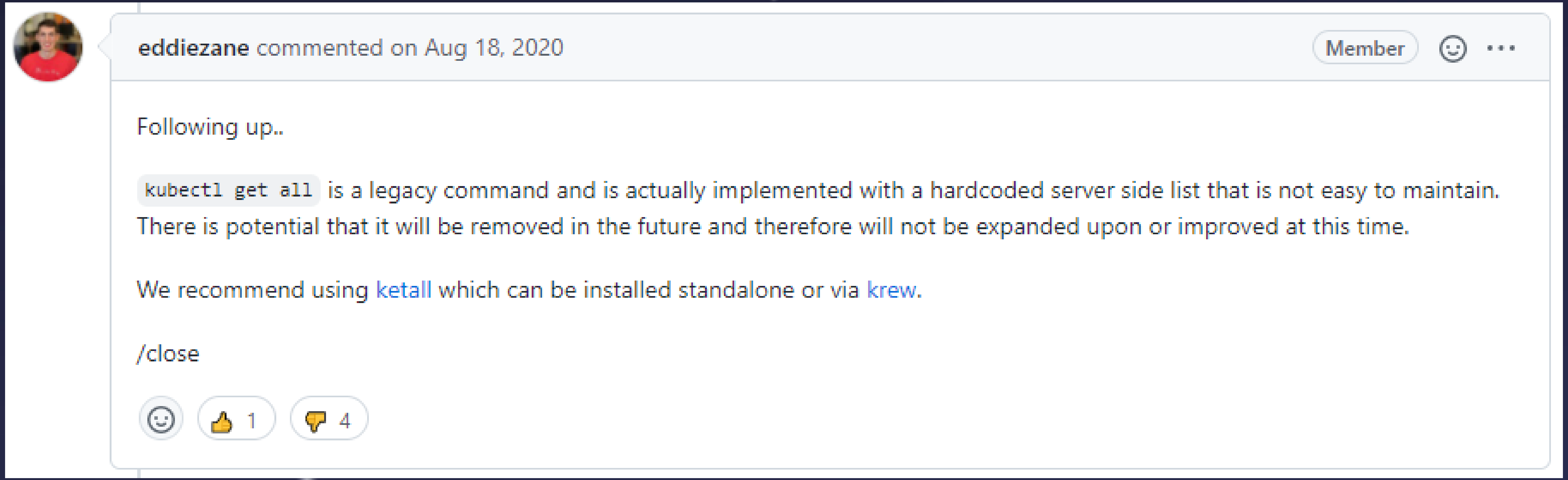
What happened:

`kubectl get all` does not list all resources in a namespace.



What happened:

`kubectl get all` only shows a small subset of kubernetes objects in a cluster, and there does not seem to be a command to get all objects (secrets, network policies, etc). This caused me hours of wasted time because I was trying to replicate a deployment object from another cluster into a new cluster, and didn't see that there was a networkpolicy needed for the service to expose that deployment.



eddiezane commented on Aug 18, 2020 Member

Following up..

`kubectl get all` is a legacy command and is actually implemented with a hardcoded server side list that is not easy to maintain. There is potential that it will be removed in the future and therefore will not be expanded upon or improved at this time.

We recommend using ketall which can be installed standalone or via krew.

/close

👍 1 👎 4

Мысли #1

- Kubernetes это фреймворк
- Есть Kubernetes operators и Custom resources на любой случай жизни
- Все команды в одной системе могут определять, наблюдать, контролировать, управлять всеми аспектами системы и приложений.

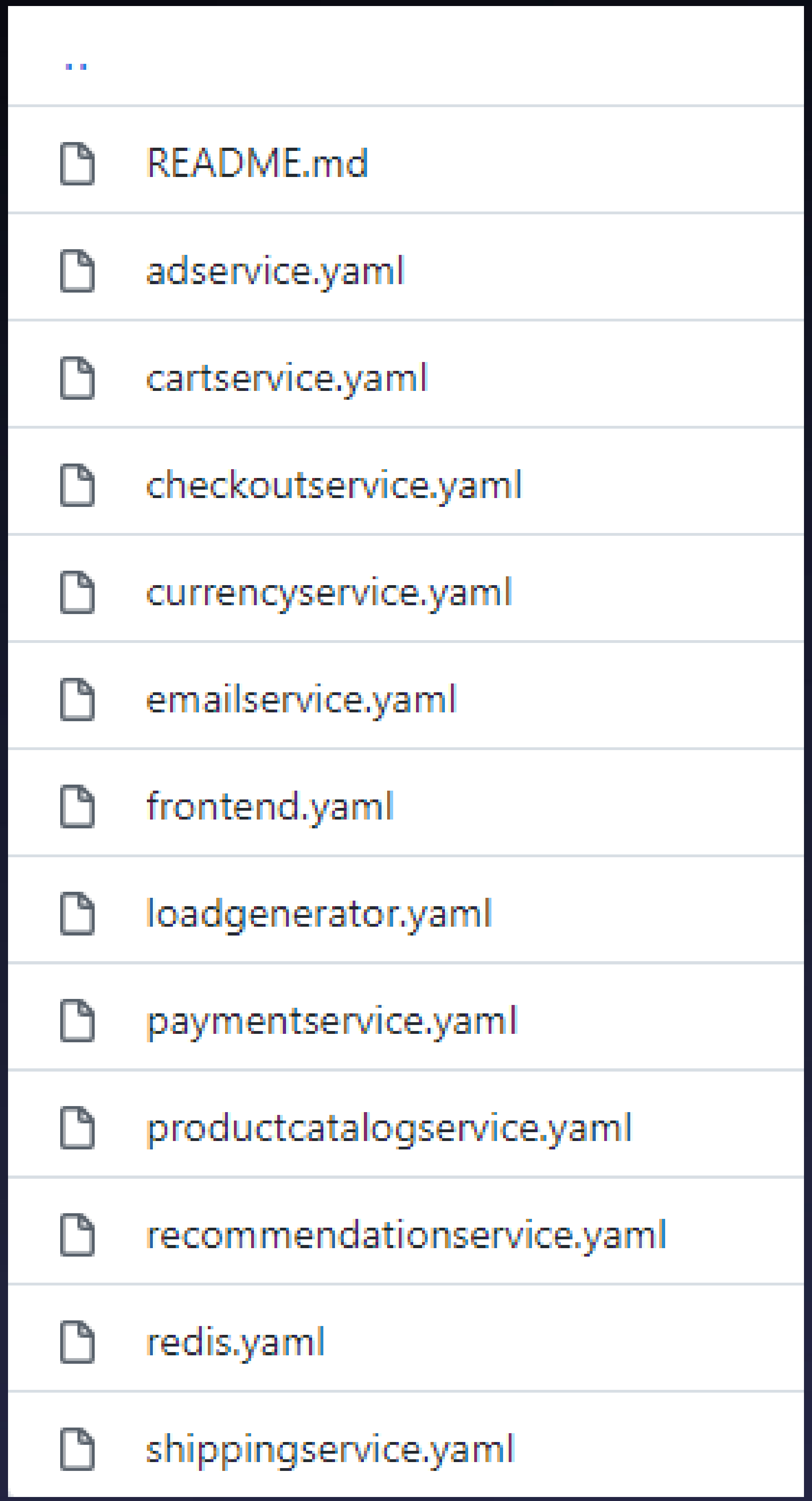
"Extensible. Kubernetes enables you to integrate it into your environment and to add the additional capabilities you need, by exposing the same interfaces used by built-in functionality."

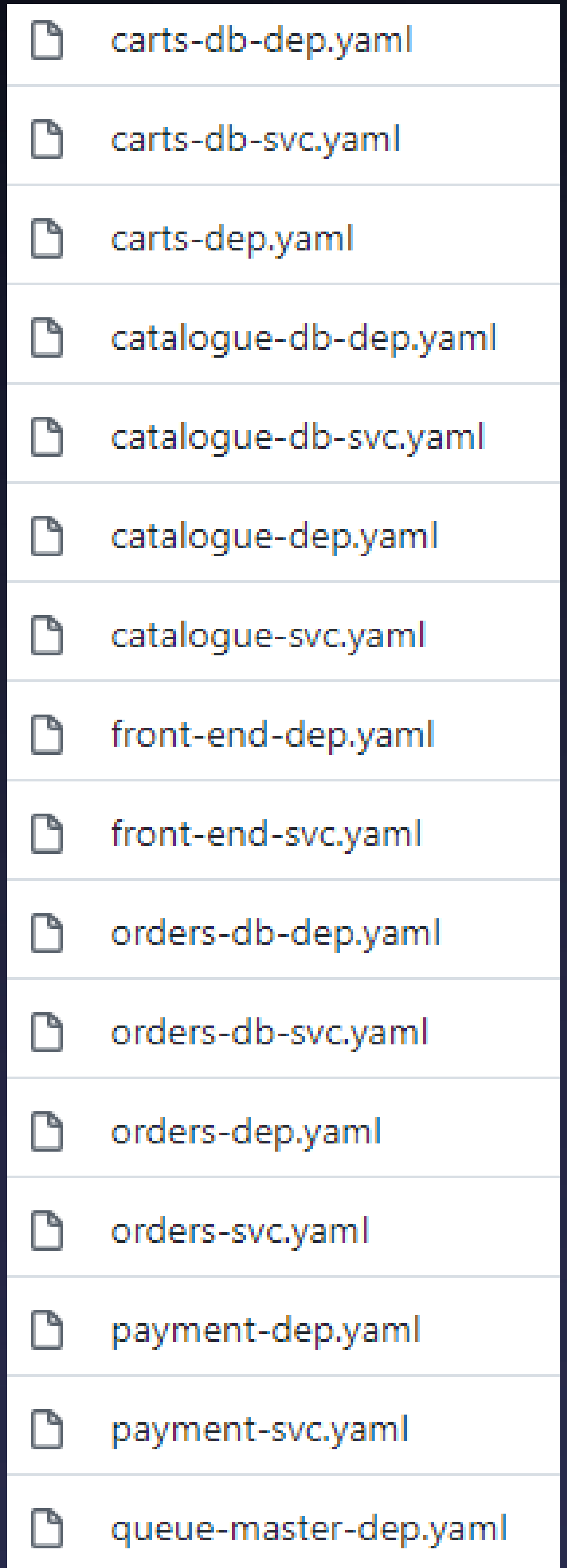
“Kubernetes design proposals”

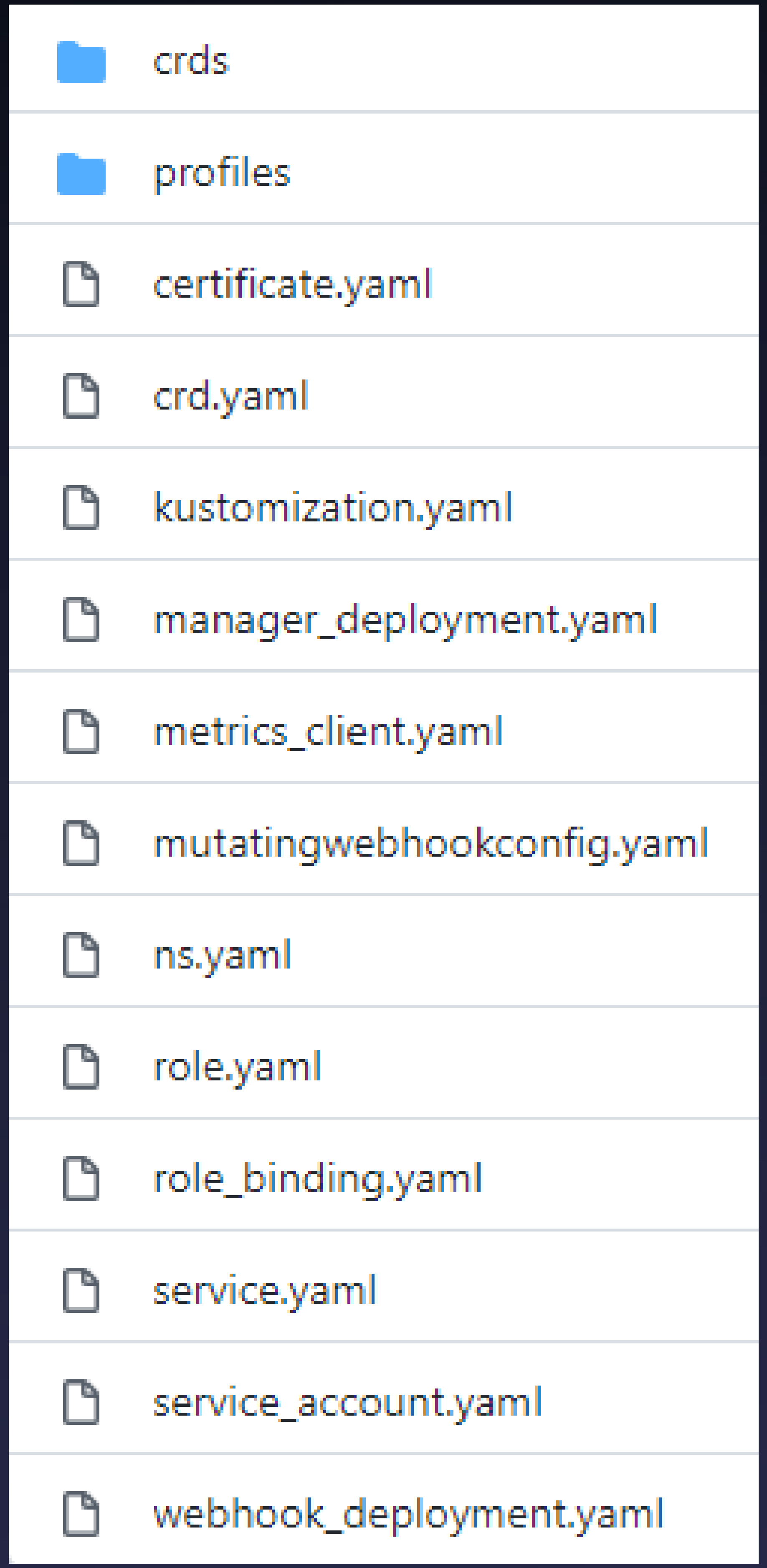
YAML не так прост, как кажется

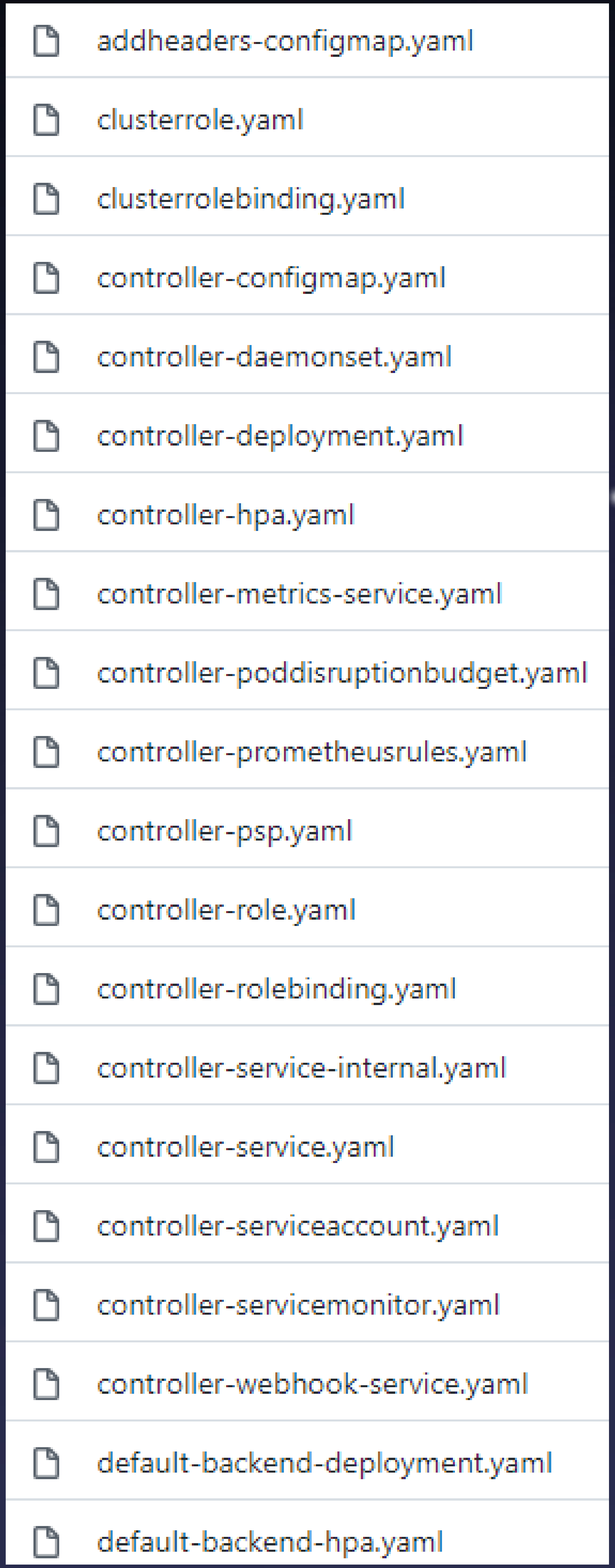
WHEN YOU HEAR THE INFRA TEAM TALKING ABOUT K8S:
"K8S IS SO EASY, IT'S JUST A BUNCH OF YAML TEMPLATING"
imgflip.com

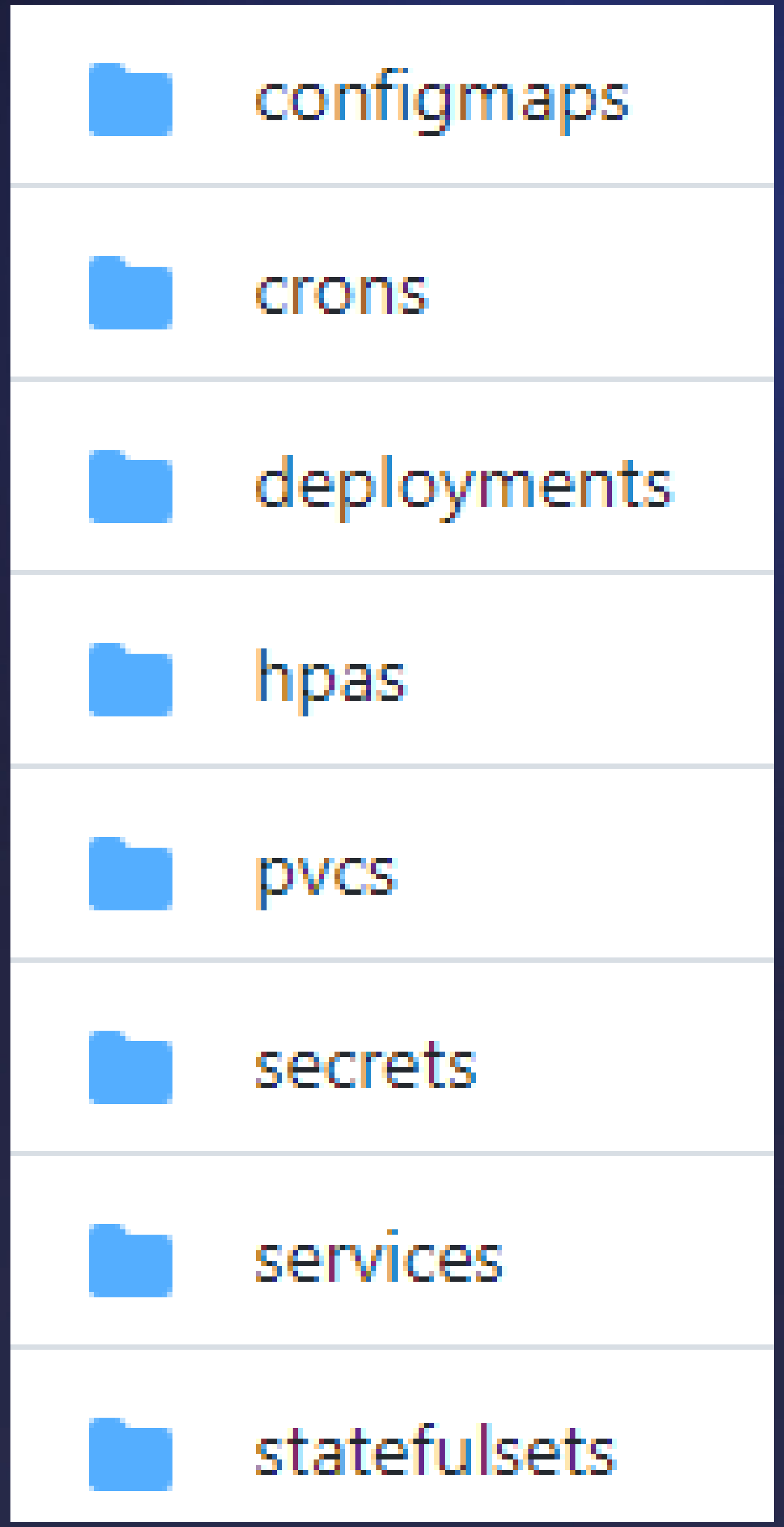
Все думаете списками ?


..
README.md
adservice.yaml
cartservice.yaml
checkoutservice.yaml
currencyservice.yaml
emailservice.yaml
frontend.yaml
loadgenerator.yaml
paymentservice.yaml
productcatalogservice.yaml
recommendationservice.yaml
redis.yaml
shippingservice.yaml

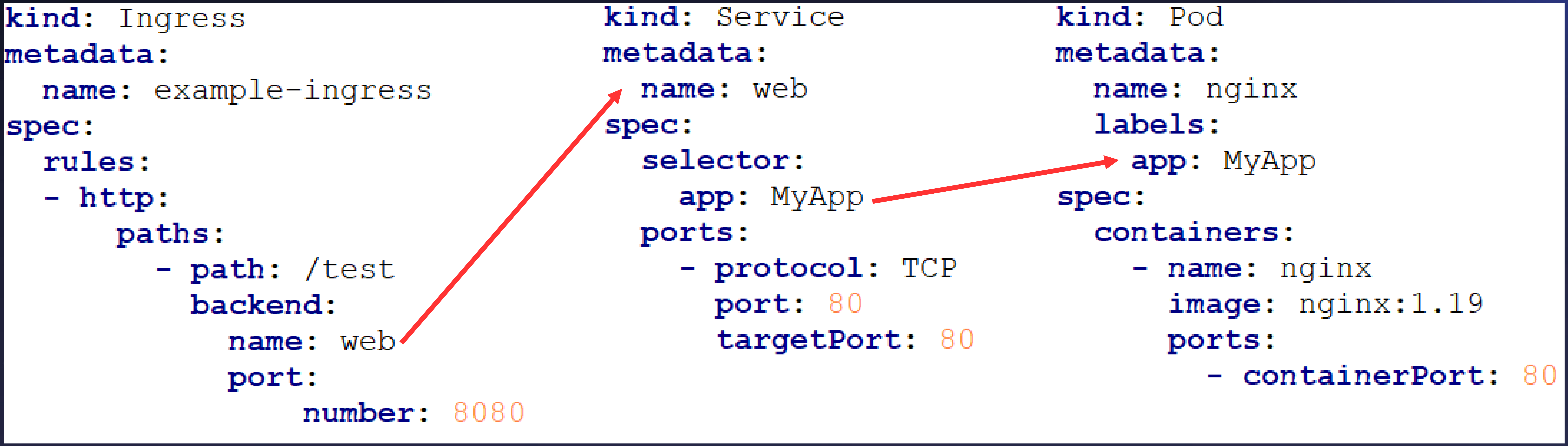

carts-db-dep.yaml
carts-db-svc.yaml
carts-dep.yaml
catalogue-db-dep.yaml
catalogue-db-svc.yaml
catalogue-dep.yaml
catalogue-svc.yaml
front-end-dep.yaml
front-end-svc.yaml
orders-db-dep.yaml
orders-db-svc.yaml
orders-dep.yaml
orders-svc.yaml
payment-dep.yaml
payment-svc.yaml
queue-master-dep.yaml


crds
profiles
certificate.yaml
crd.yaml
kustomization.yaml
manager_deployment.yaml
metrics_client.yaml
mutatingwebhookconfig.yaml
ns.yaml
role.yaml
role_binding.yaml
service.yaml
service_account.yaml
webhook_deployment.yaml


addheaders-configmap.yaml
clusterrole.yaml
clusterrolebinding.yaml
controller-configmap.yaml
controller-daemonset.yaml
controller-deployment.yaml
controller-hpa.yaml
controller-metrics-service.yaml
controller-poddisruptionbudget.yaml
controller-prometheusrules.yaml
controller-psp.yaml
controller-role.yaml
controller-rolebinding.yaml
controller-service-internal.yaml
controller-service.yaml
controller-serviceaccount.yaml
controller-servicemonitor.yaml
controller-webhook-service.yaml
default-backend-deployment.yaml
default-backend-hpa.yaml


configmaps
crons
deployments
hpas
pvcs
secrets
services
statefulsets

Связи k8s-ресурсов



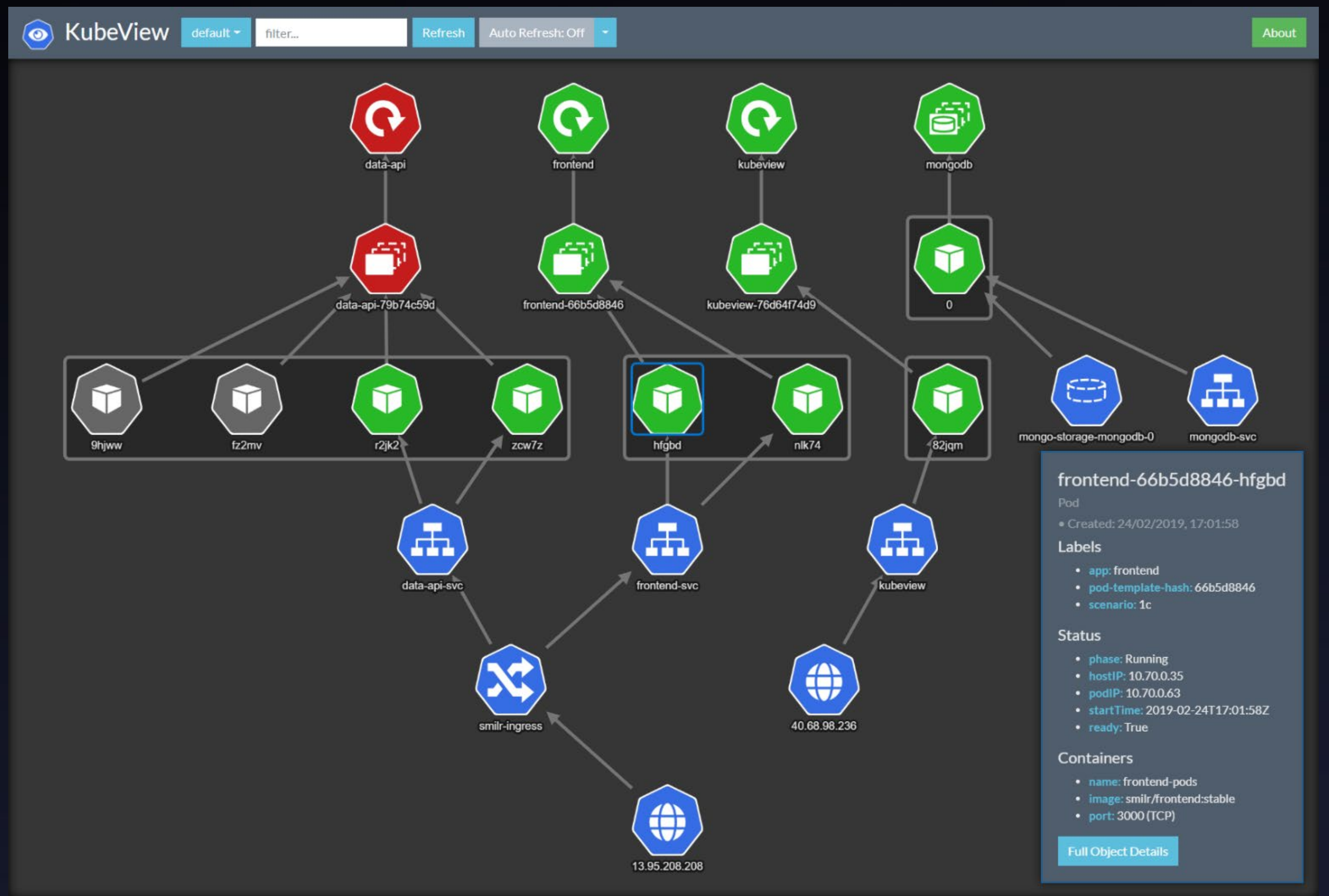
```yaml
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - http:
      paths:
        - path: /test
          backend:
            name: web
            port:
                number: 8080
```

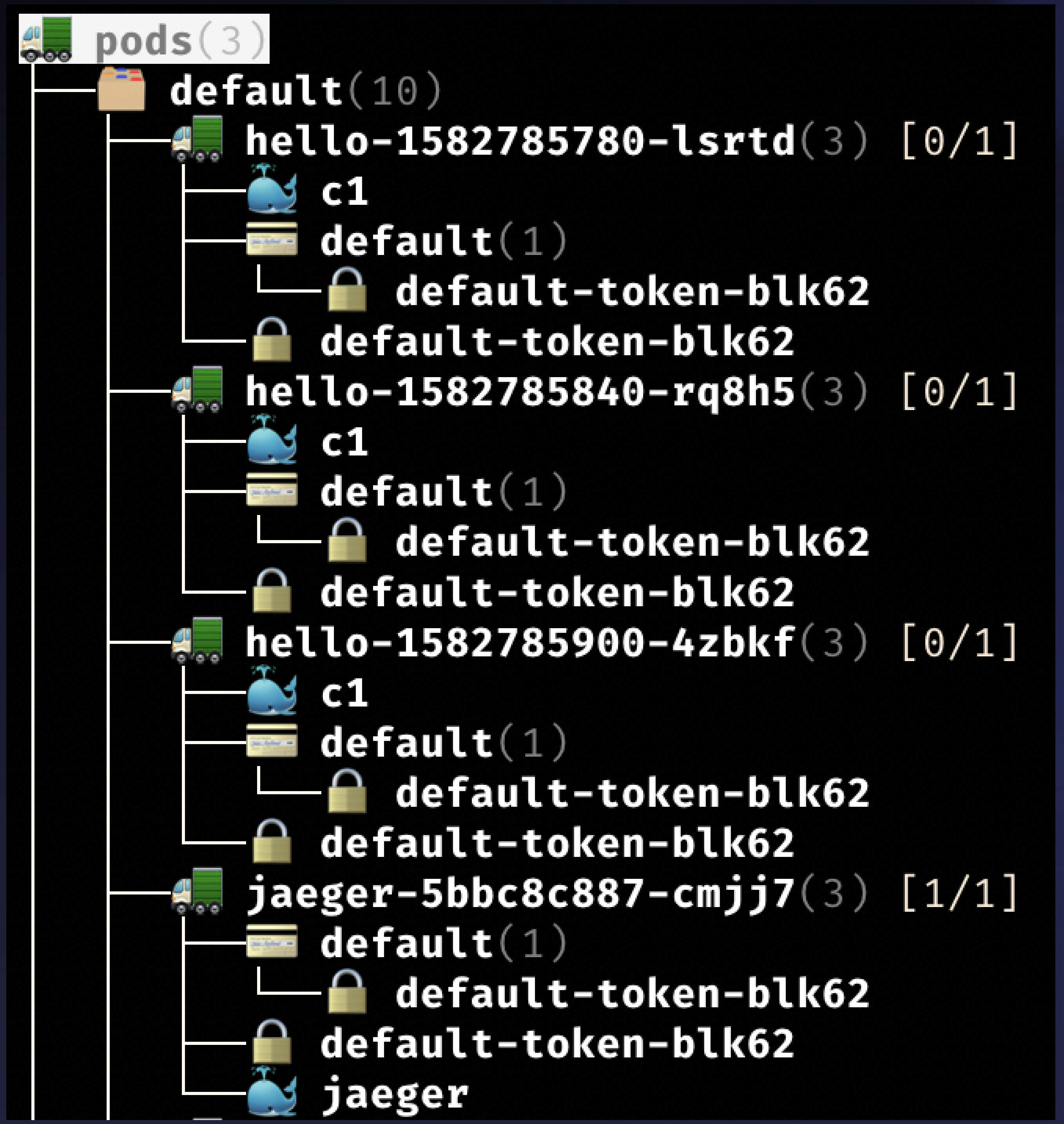
```yaml
kind: Service
metadata:
  name: web
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
```

```yaml
kind: Pod
metadata:
  name: nginx
  labels:
    app: MyApp
spec:
  containers:
    - name: nginx
      image: nginx:1.19
      ports:
        - containerPort: 80
```

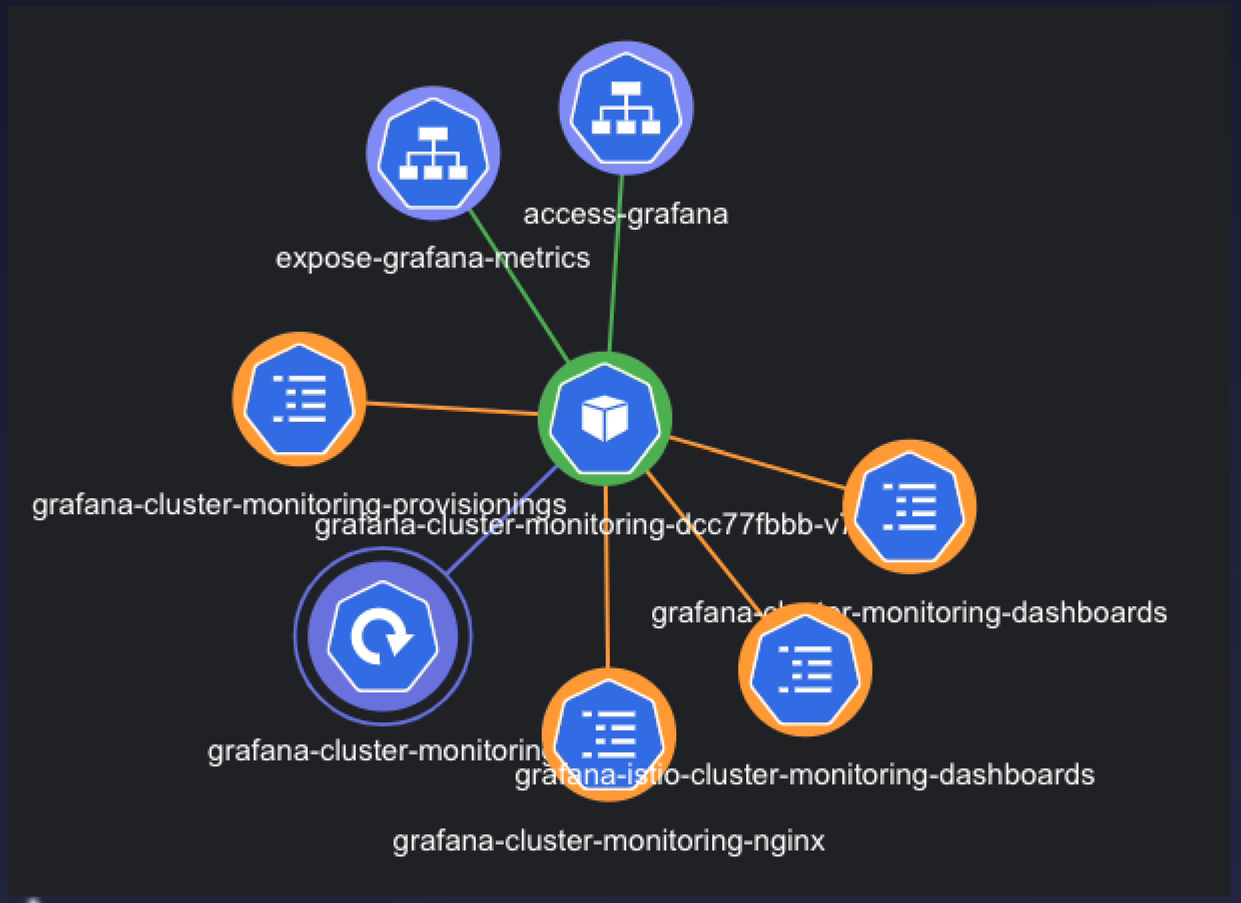
Tools

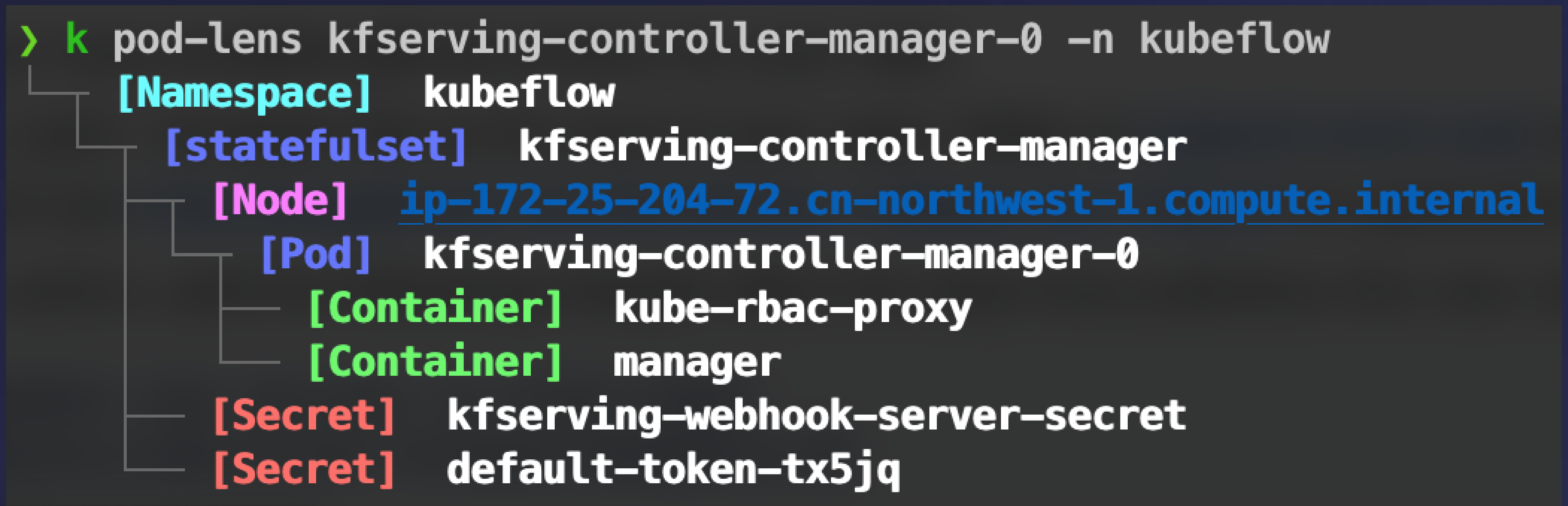
- kubectl-graph
- kubectl tree
- kube-lineage
- Kubectl Pod Lens
- Kubesurveyor
- kubectl service-tree
- Lens Resource Map
- KubeView
- k9s (XRay)
- ...



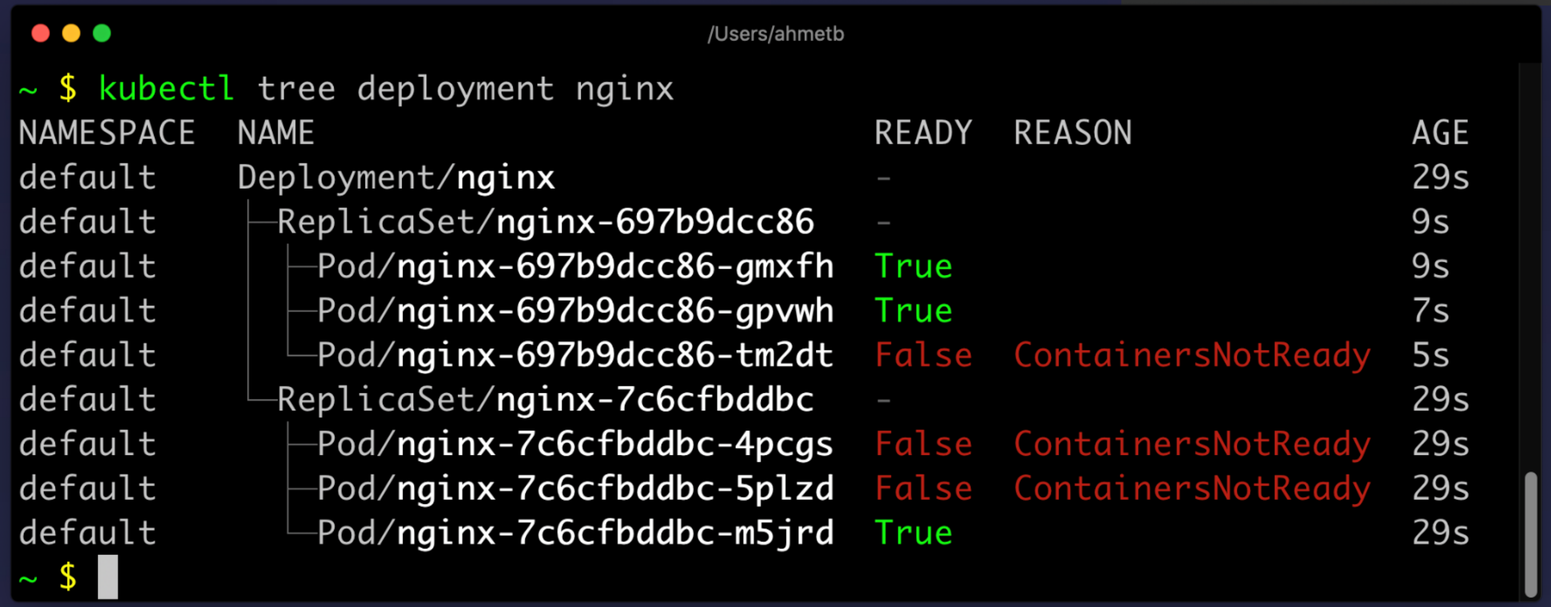


```
pods(3)
└─ default(10)
   ├─ hello-1582785780-lsrtd(3) [0/1]
   │  ├─ c1
   │  ├─ default(1)
   │  │  └─ default-token-blk62
   │  └─ default-token-blk62
   ├─ hello-1582785840-rq8h5(3) [0/1]
   │  ├─ c1
   │  ├─ default(1)
   │  │  └─ default-token-blk62
   │  └─ default-token-blk62
   ├─ hello-1582785900-4zbkf(3) [0/1]
   │  ├─ c1
   │  ├─ default(1)
   │  │  └─ default-token-blk62
   │  └─ default-token-blk62
   ├─ jaeger-5bbc8c887-cmjj7(3) [1/1]
   │  ├─ default(1)
   │  │  └─ default-token-blk62
   │  ├─ default-token-blk62
   │  └─ jaeger
```


access-grafana
expose-grafana-metrics
grafana-cluster-monitoring-provisionings
grafana-cluster-monitoring-dcc77fbbb-v
grafana-cluster-monitoring-dashboards
grafana-cluster-monitoring
grafana-istio-cluster-monitoring-dashboards
grafana-cluster-monitoring-nginx



```
❯ k pod-lens kfserving-controller-manager-0 -n kubeflow
└─ [Namespace]   kubeflow
   └─ [statefulset]   kfserving-controller-manager
      ├─ [Node]   ip-172-25-204-72.cn-northwest-1.compute.internal
      │  └─ [Pod]   kfserving-controller-manager-0
      │     ├─ [Container]   kube-rbac-proxy
      │     └─ [Container]   manager
      ├─ [Secret]   kfserving-webhook-server-secret
      └─ [Secret]   default-token-tx5jq
```



```
/Users/ahmetb
~ $ kubectl tree deployment nginx
NAMESPACE  NAME                              READY  REASON              AGE
default    Deployment/nginx                  -                          29s
default    ├─ReplicaSet/nginx-697b9dcc86     -                          9s
default    │ ├─Pod/nginx-697b9dcc86-gmxfh    True                       9s
default    │ ├─Pod/nginx-697b9dcc86-gpvwh    True                       7s
default    │ └─Pod/nginx-697b9dcc86-tm2dt    False  ContainersNotReady  5s
default    └─ReplicaSet/nginx-7c6cfbddbc     -                          29s
default      ├─Pod/nginx-7c6cfbddbc-4pcgs    False  ContainersNotReady  29s
default      ├─Pod/nginx-7c6cfbddbc-5plzd    False  ContainersNotReady  29s
default      └─Pod/nginx-7c6cfbddbc-m5jrd    True                       29s
~ $
```

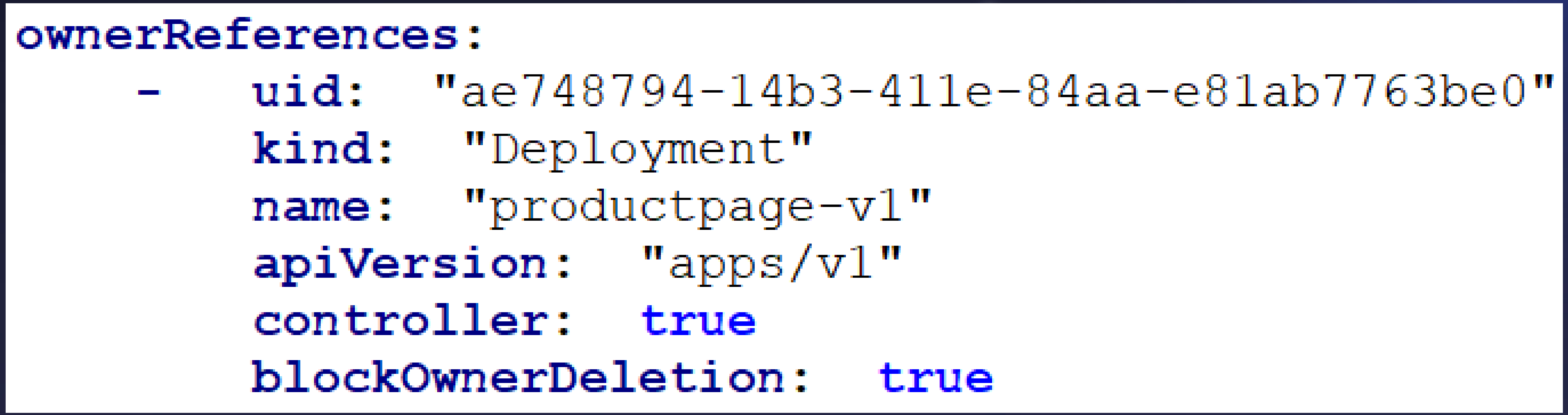
ВИДЯТ И ЗНАЮТ НЕ ВСЕ!

Разные связи, очень разные связи

- ownerReference (UID)
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- targetRef
- name
- annotations
- Магия …

Разные связи, очень разные связи

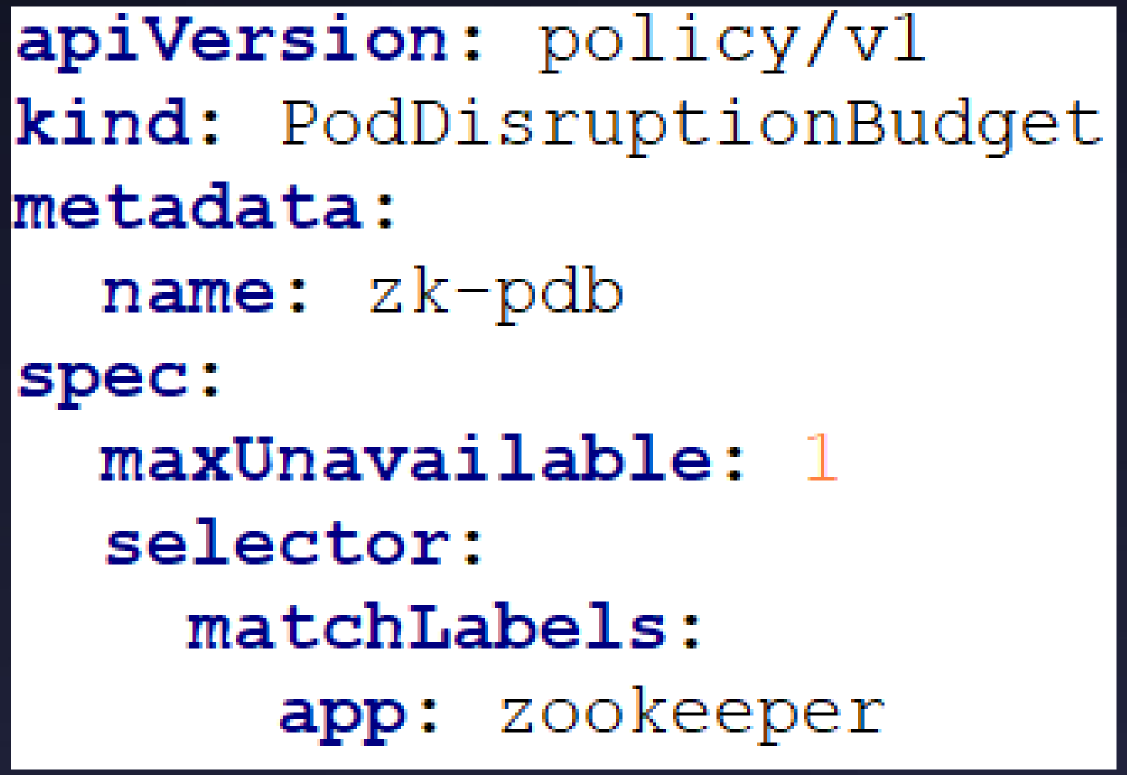
- **ownerReference (UID)**
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- targetRef
- name
- annotations
- Магия …



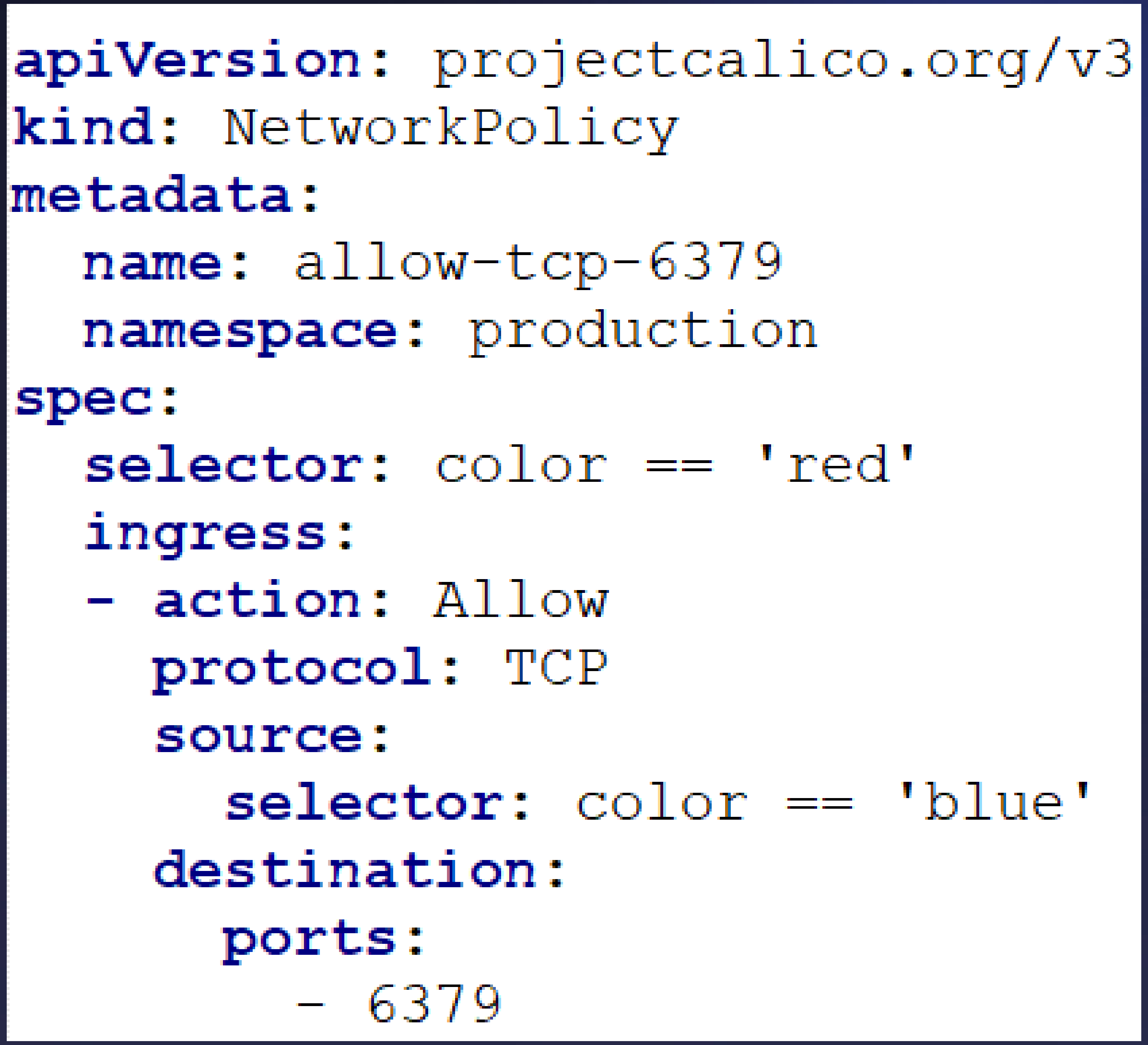
```
ownerReferences:
    -   uid:  "ae748794-14b3-411e-84aa-e81ab7763be0"
        kind:  "Deployment"
        name:  "productpage-v1"
        apiVersion:  "apps/v1"
        controller:  true
        blockOwnerDeletion:  true
```

Разные связи, очень разные связи

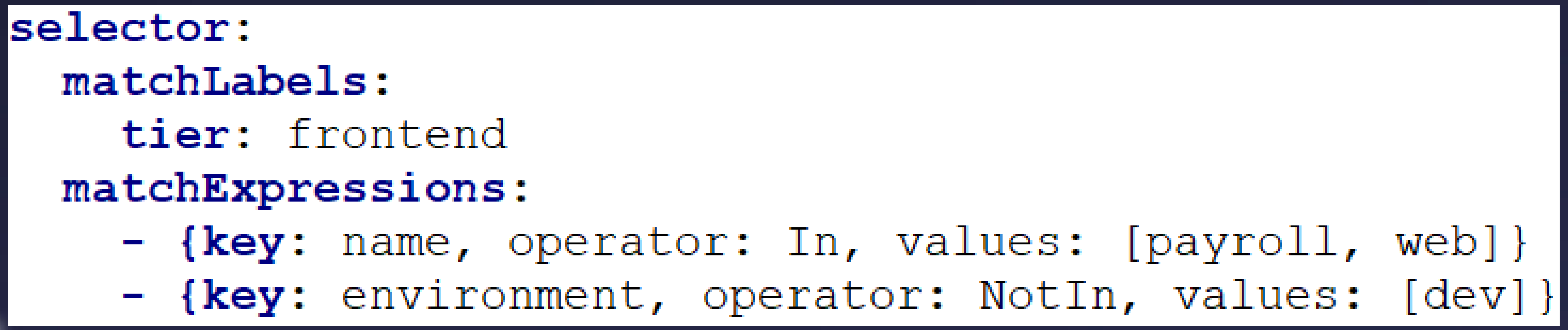
- ownerReference (UID)
  - GarbageCollector
- **Selector**
  - **matchLabels**
  - **matchExpression**
- targetRef
- name
- annotations
- Магия …



```yaml
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: zk-pdb
spec:
  maxUnavailable: 1
  selector:
    matchLabels:
      app: zookeeper
```



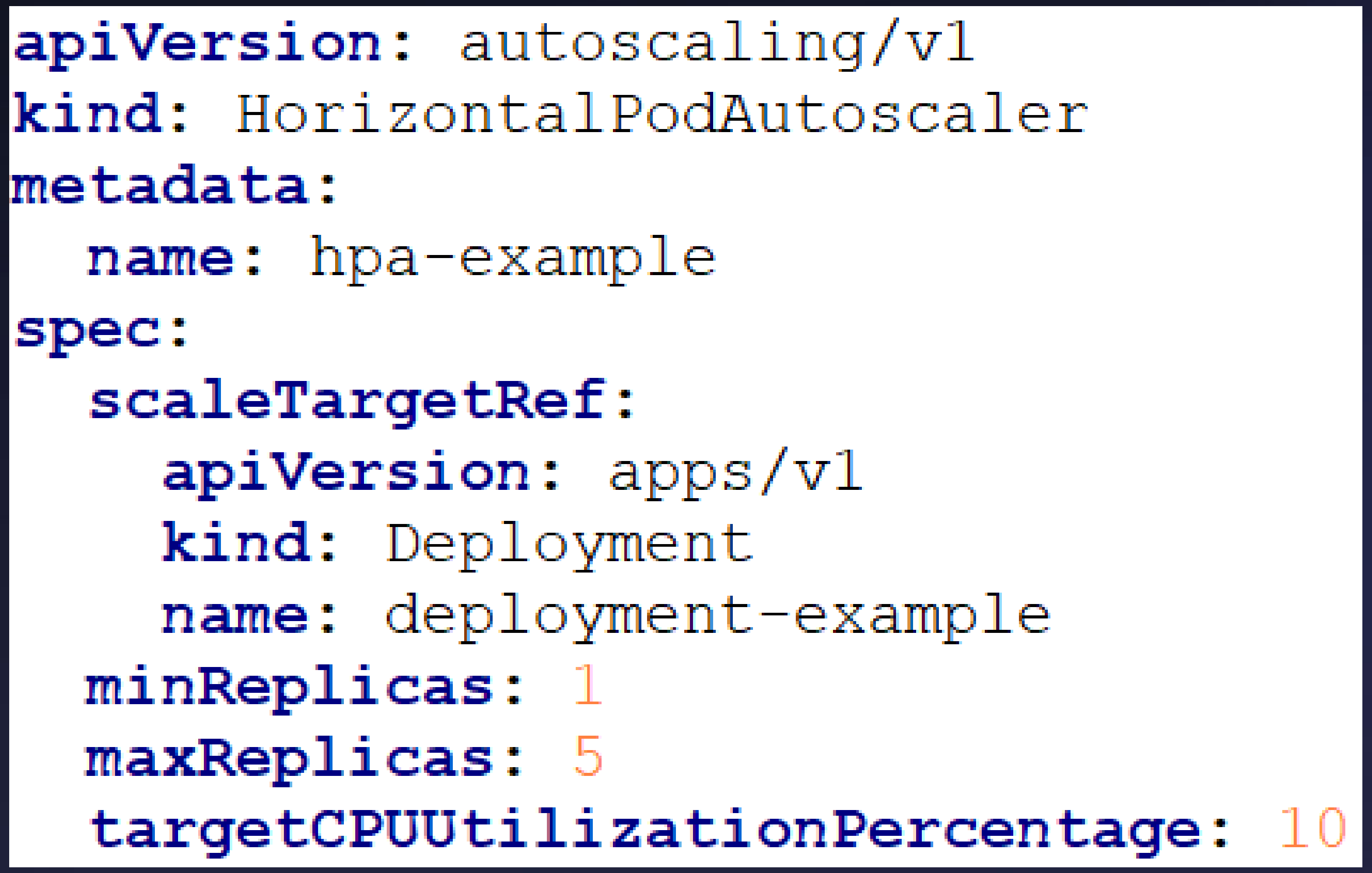
```yaml
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: allow-tcp-6379
  namespace: production
spec:
  selector: color == 'red'
  ingress:
  - action: Allow
    protocol: TCP
    source:
      selector: color == 'blue'
    destination:
      ports:
        - 6379
```



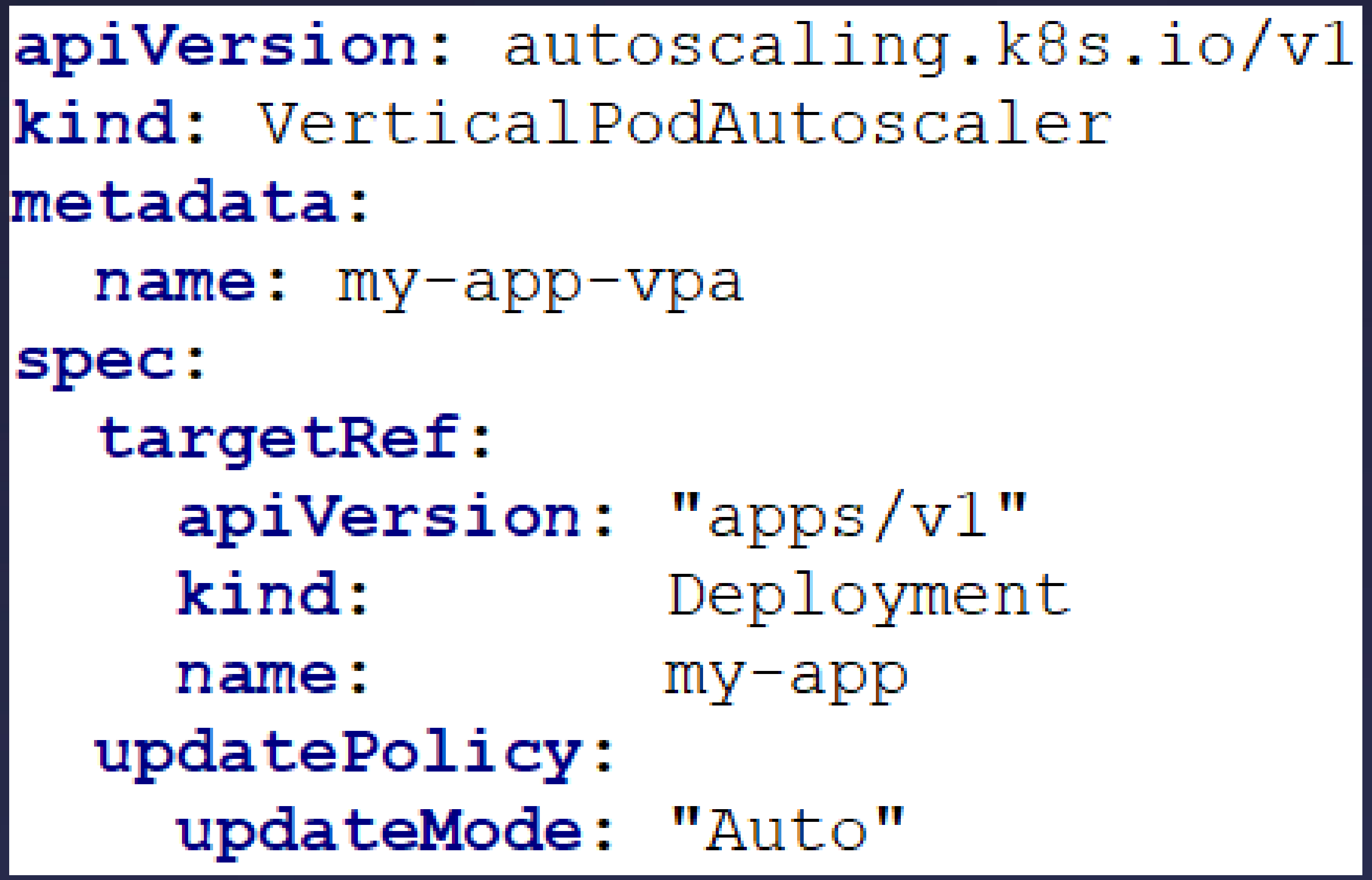
```yaml
selector:
  matchLabels:
    tier: frontend
  matchExpressions:
    - {key: name, operator: In, values: [payroll, web]}
    - {key: environment, operator: NotIn, values: [dev]}
```

Разные связи, очень разные связи

- ownerReference (UID)
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- **targetRef**
- name
- annotations
- Магия …



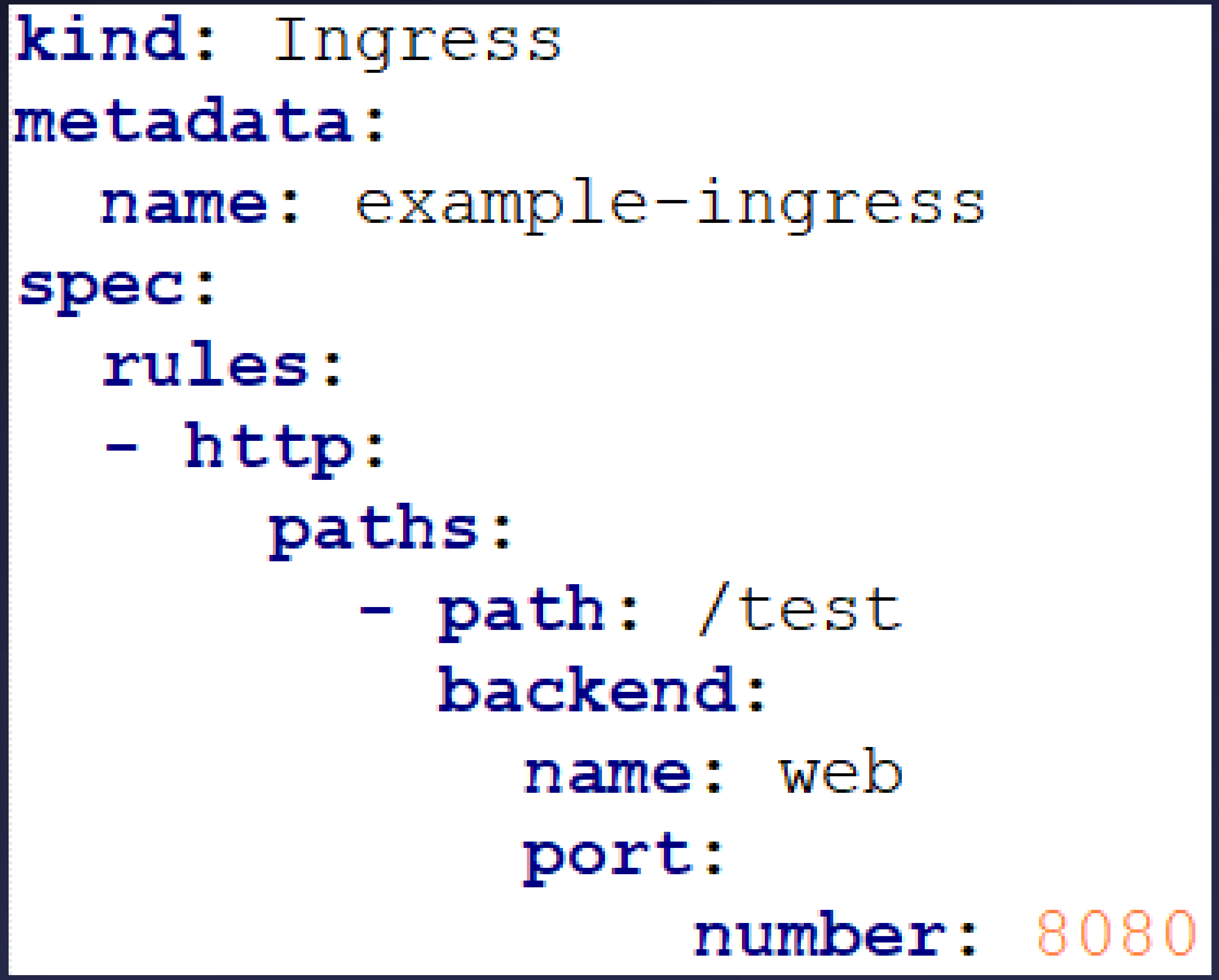
```yaml
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: hpa-example
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: deployment-example
  minReplicas: 1
  maxReplicas: 5
  targetCPUUtilizationPercentage: 10
```



```yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: my-app-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind:       Deployment
    name:       my-app
  updatePolicy:
    updateMode: "Auto"
```

Разные связи, очень разные связи

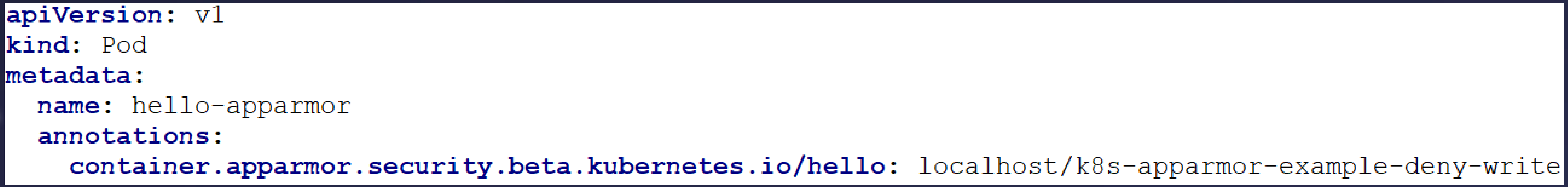
- ownerReference (UID)
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- targetRef
- **name**
- annotations
- Магия …



```
kind: Ingress
metadata:
  name: example-ingress
spec:
  rules:
  - http:
      paths:
        - path: /test
          backend:
            name: web
            port:
              number: 8080
```

Разные связи, очень разные связи

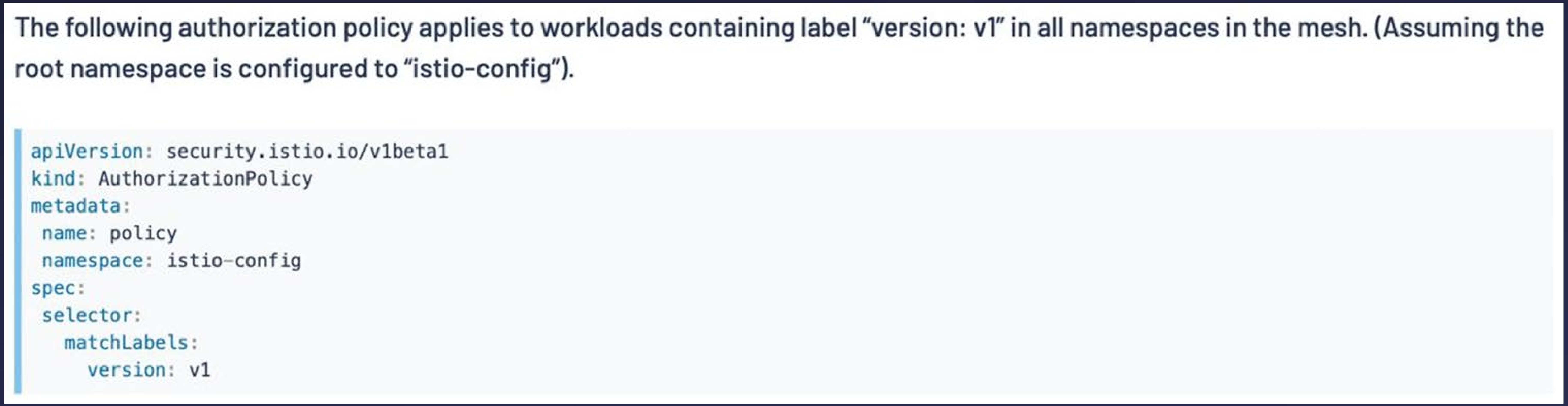
- ownerReference (UID)
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- targetRef
- name
- **annotations**
- Магия …



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-apparmor
  annotations:
    container.apparmor.security.beta.kubernetes.io/hello: localhost/k8s-apparmor-example-deny-write
```

Разные связи, очень разные связи

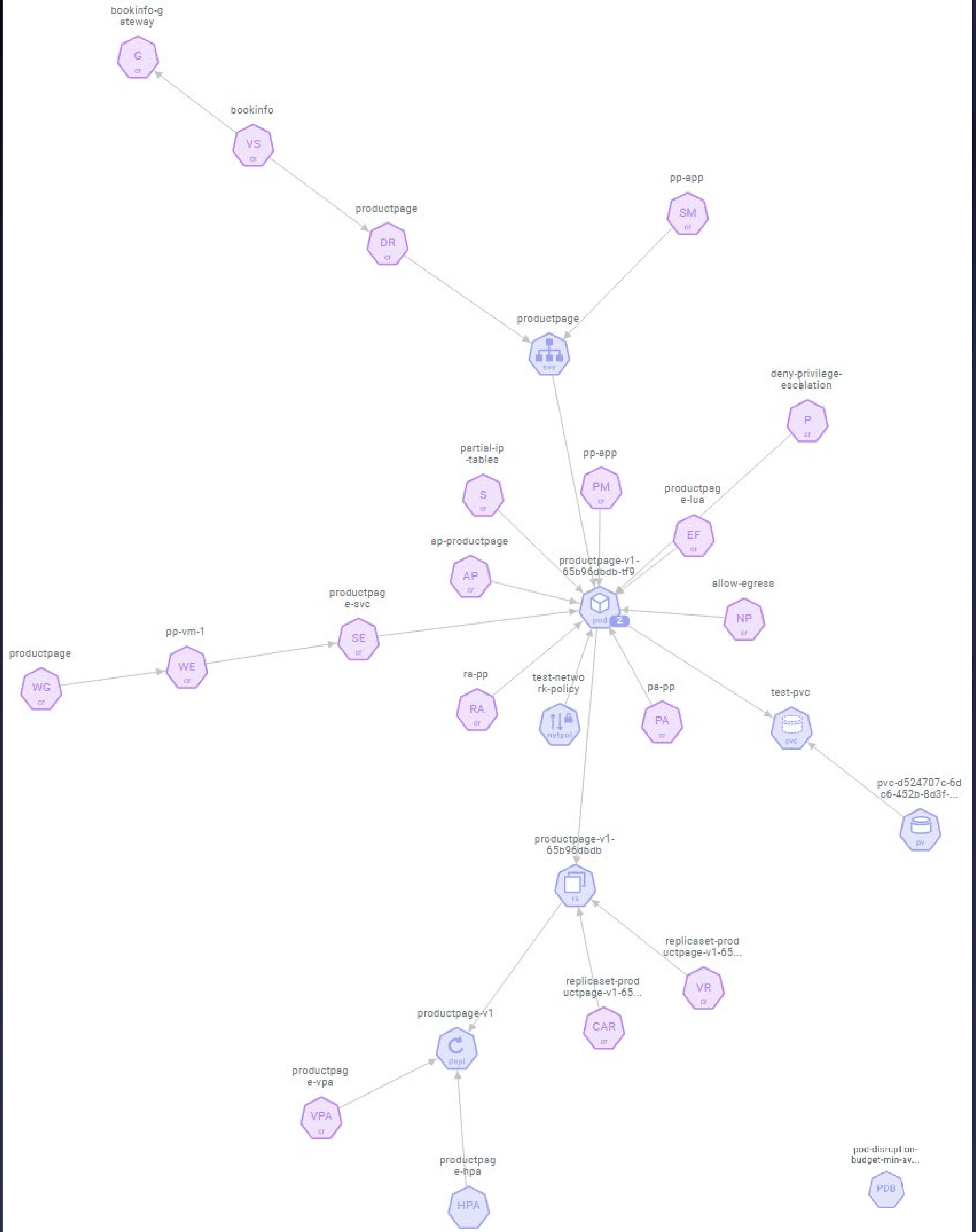
- ownerReference (UID)
  - GarbageCollector
- Selector
  - matchLabels
  - matchExpression
- targetRef
- name
- annotations
- **Магия …**


The following authorization policy applies to workloads containing label "version: v1" in all namespaces in the mesh. (Assuming the root namespace is configured to "istio-config").

```yaml
apiVersion: security.istio.io/v1beta1
kind: AuthorizationPolicy
metadata:
 name: policy
 namespace: istio-config
spec:
 selector:
   matchLabels:
     version: v1
```

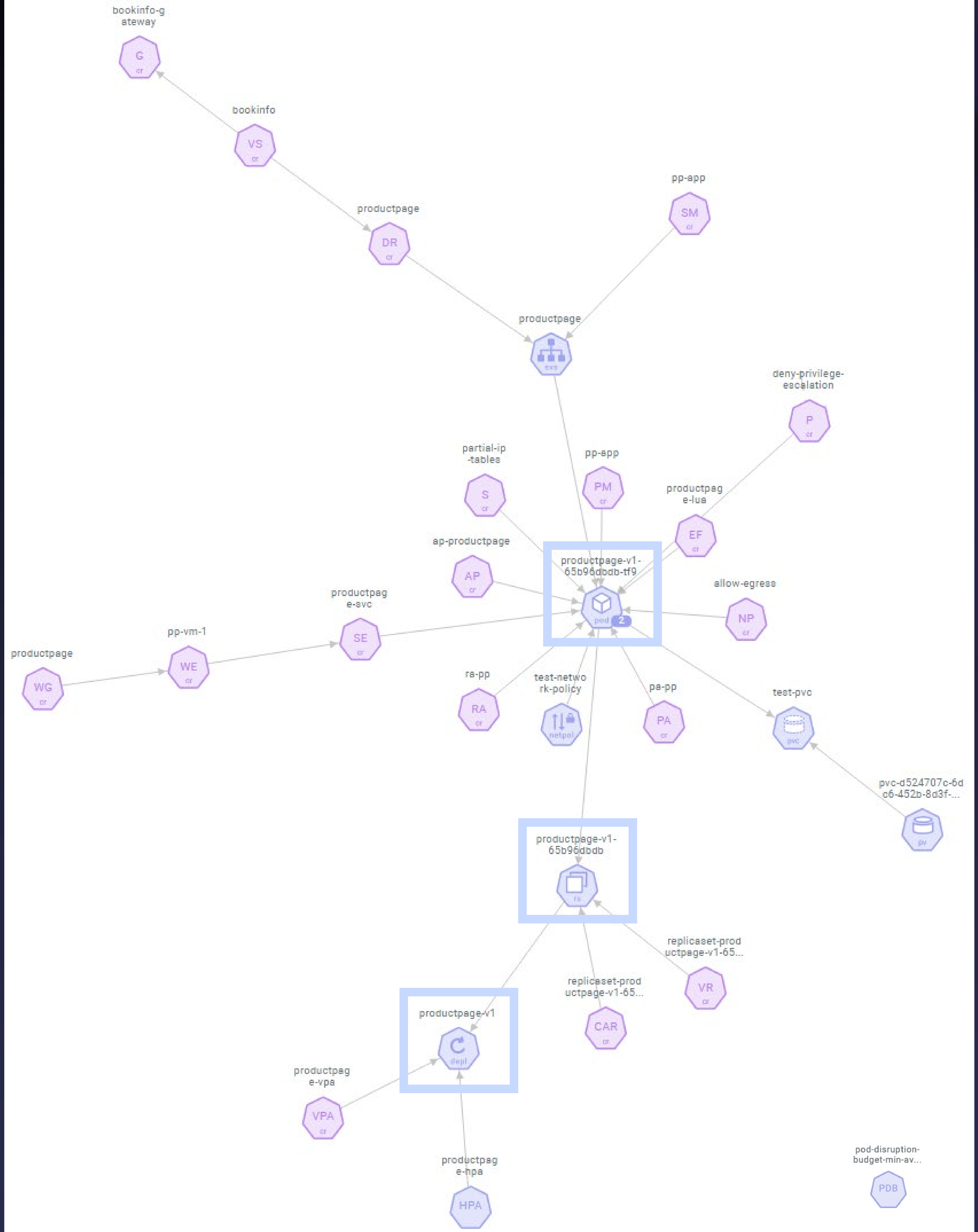
Пример

- Deployment
- Service
- NetworkPolicy
- PV/PVC
- HPA/VPA/PDB
- Service Mesh (Istio)
- Policy Engine (Kyverno)
- Prometheus operator
- Starboard operator



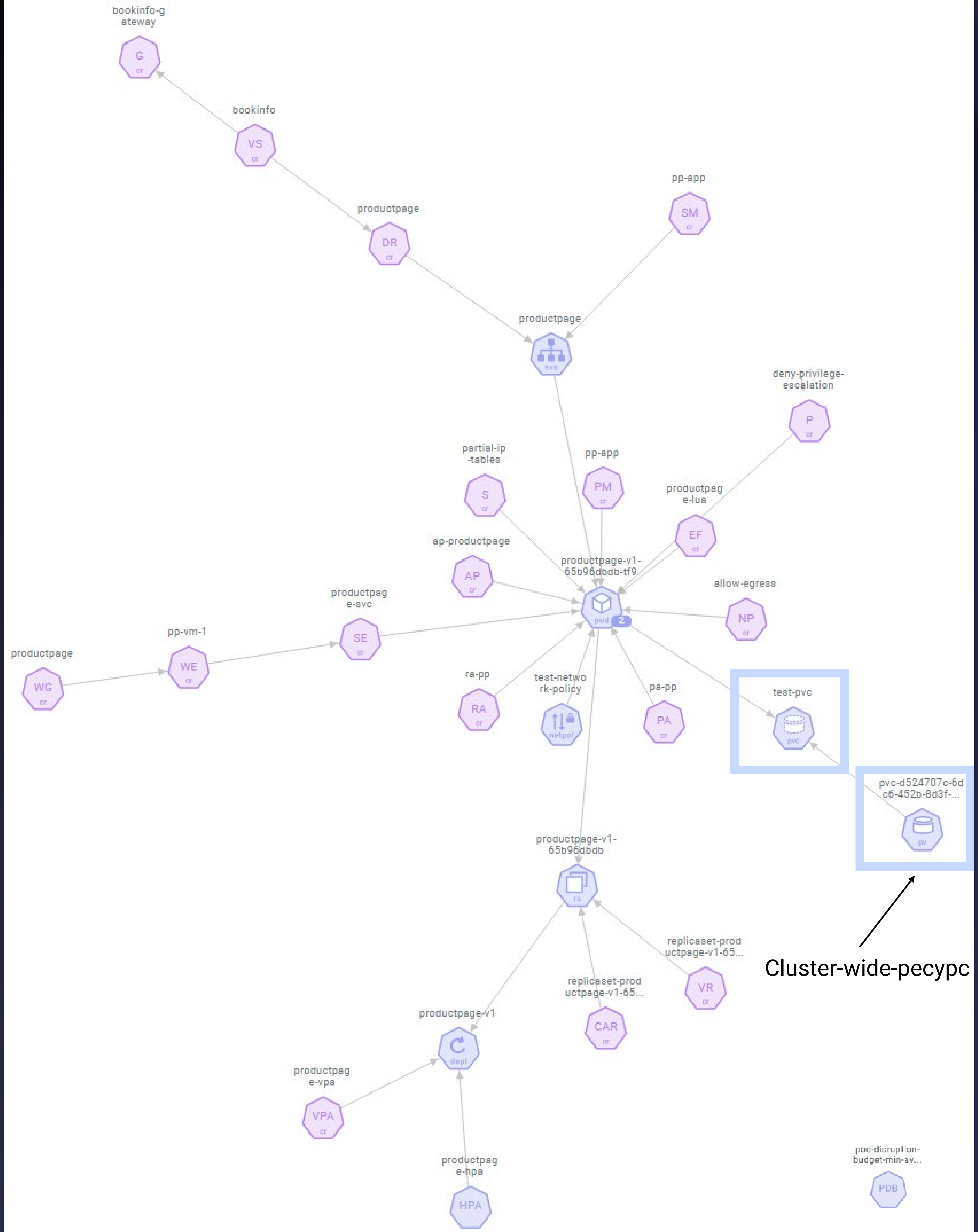
Пример

- Workload:
  - Deployment
  - ReplicaSet
  - Pods



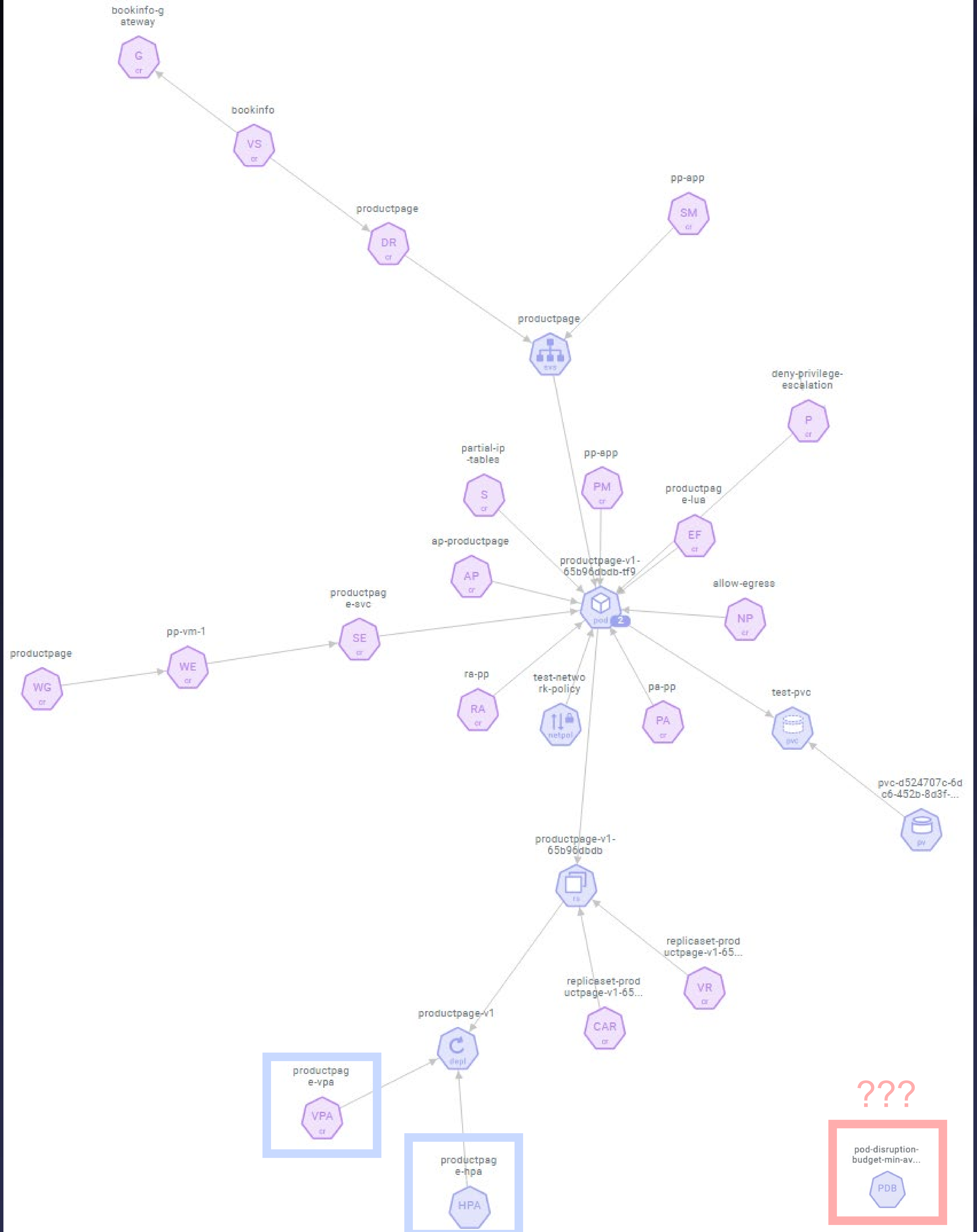
Пример

- Storage:
  - PersistentVolume
  - PersistentVolumeClaim



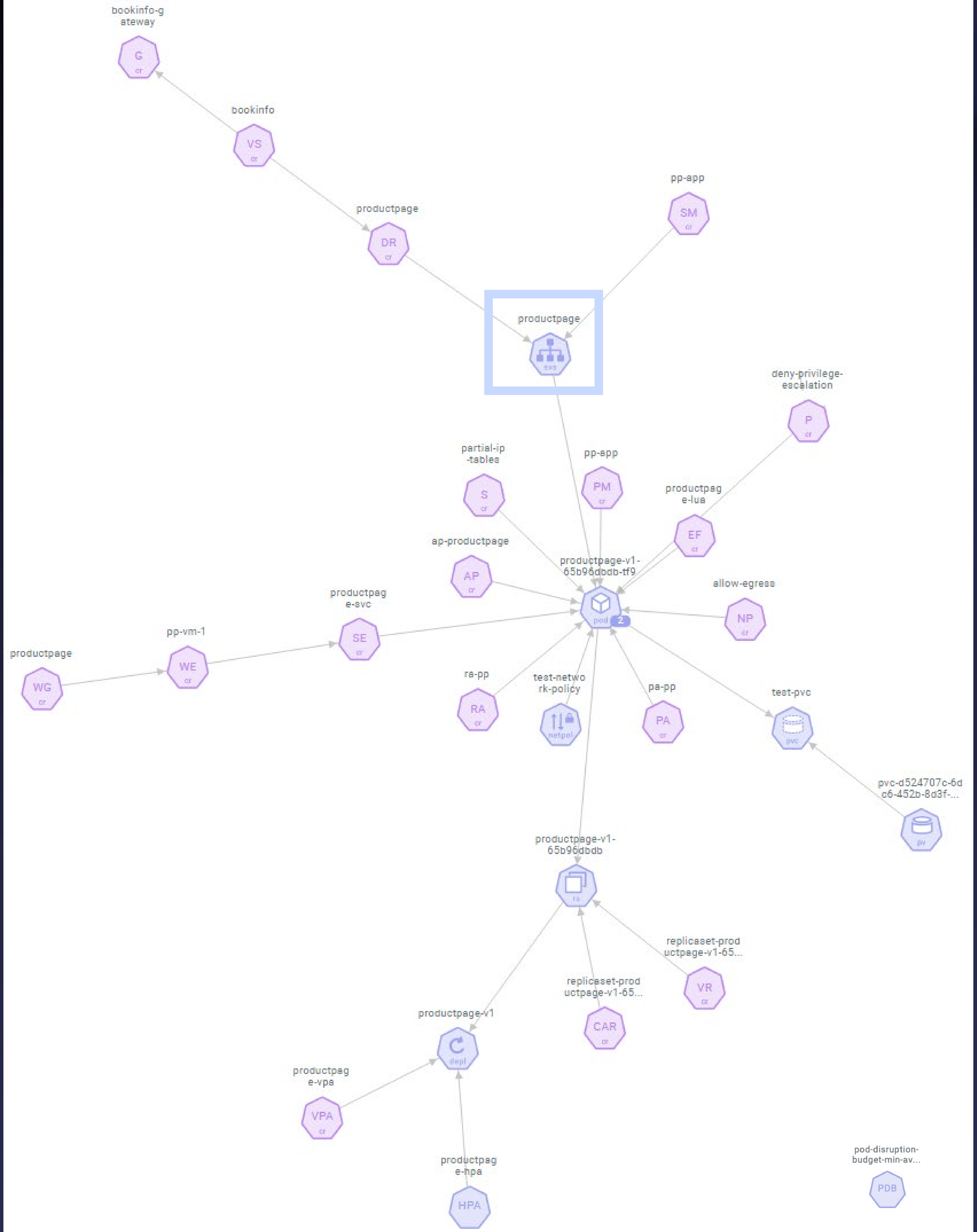
Пример

- High availability:
  - HPA
  - VPA
  - PDB



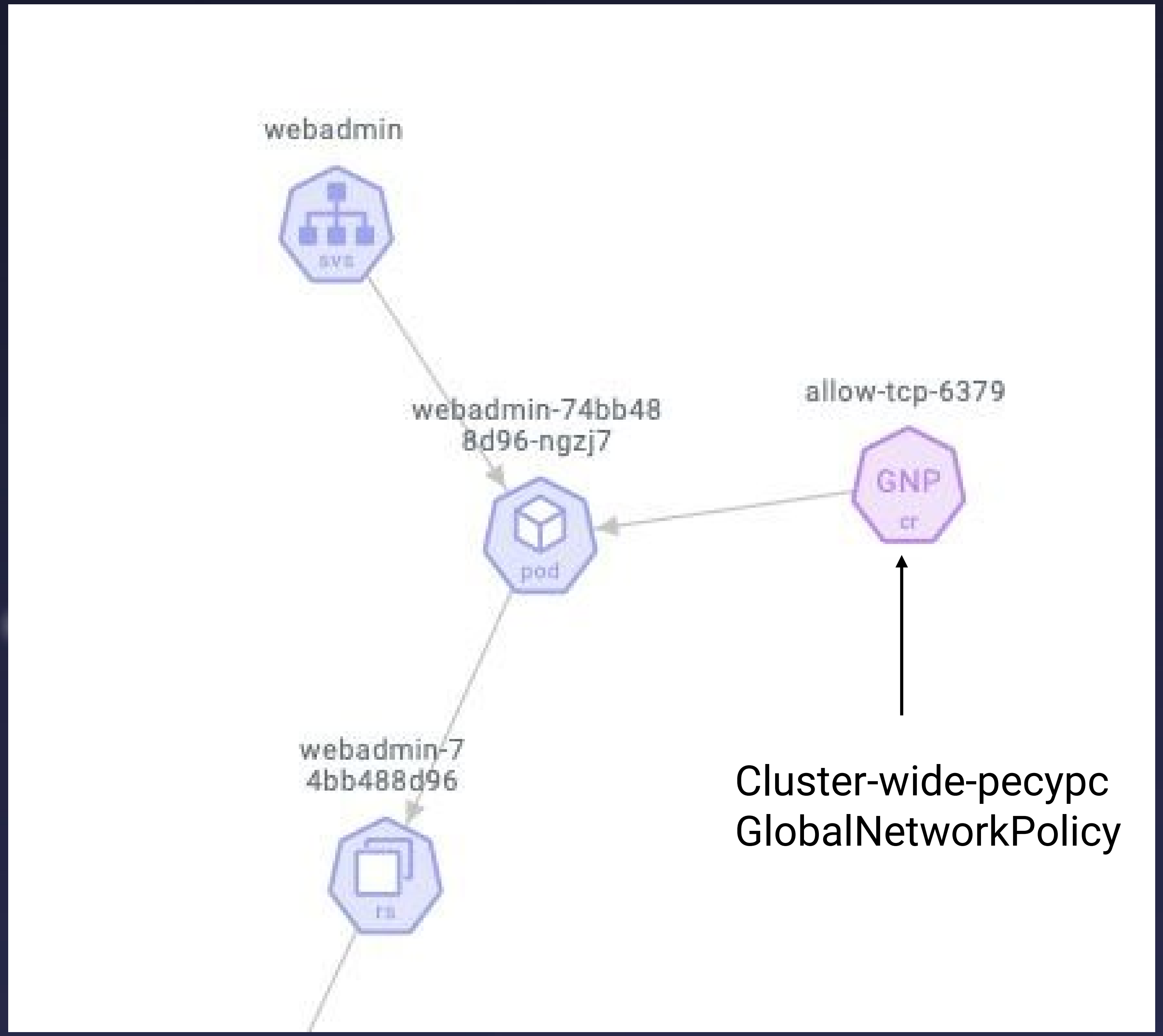
Пример

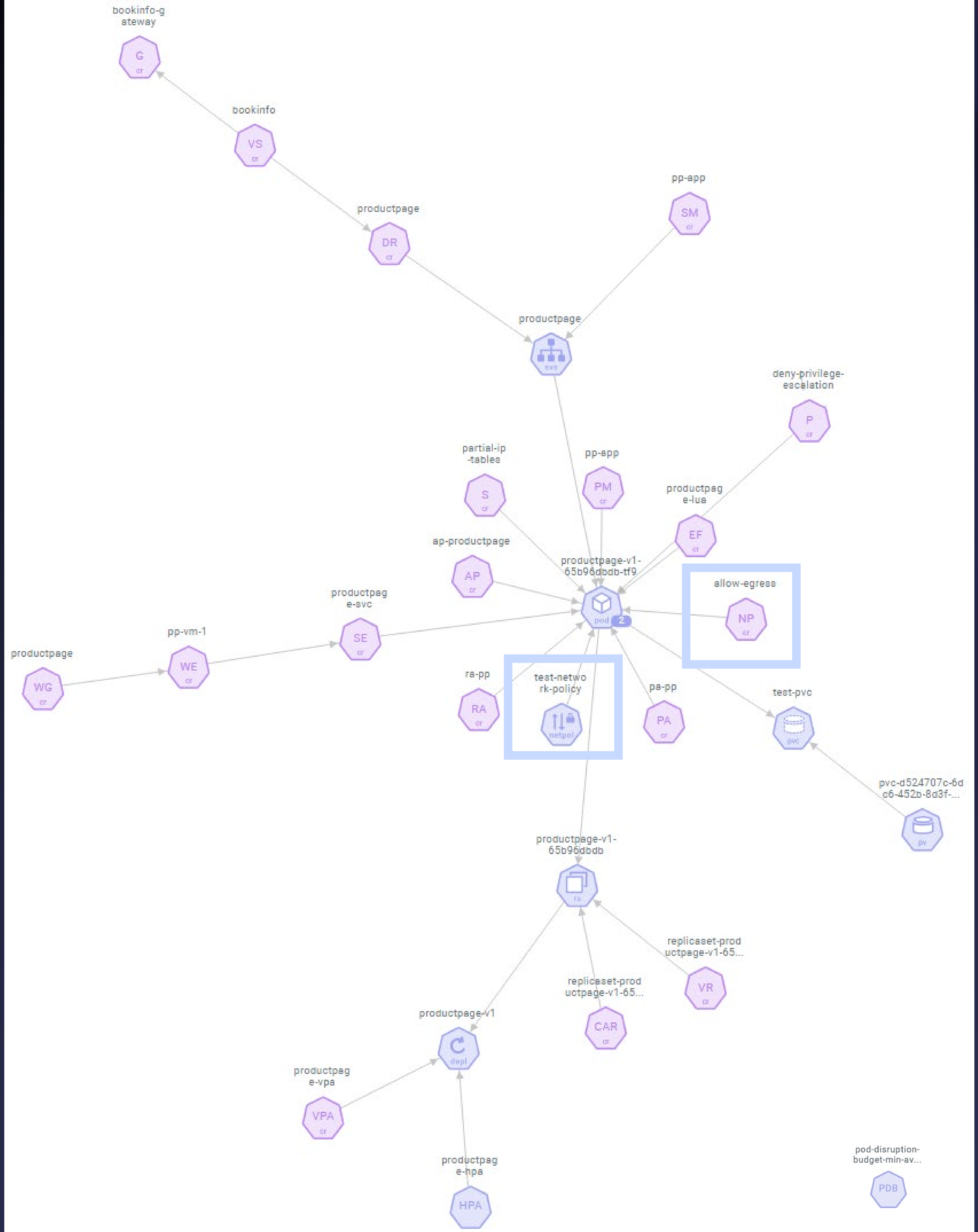
- Service



Пример

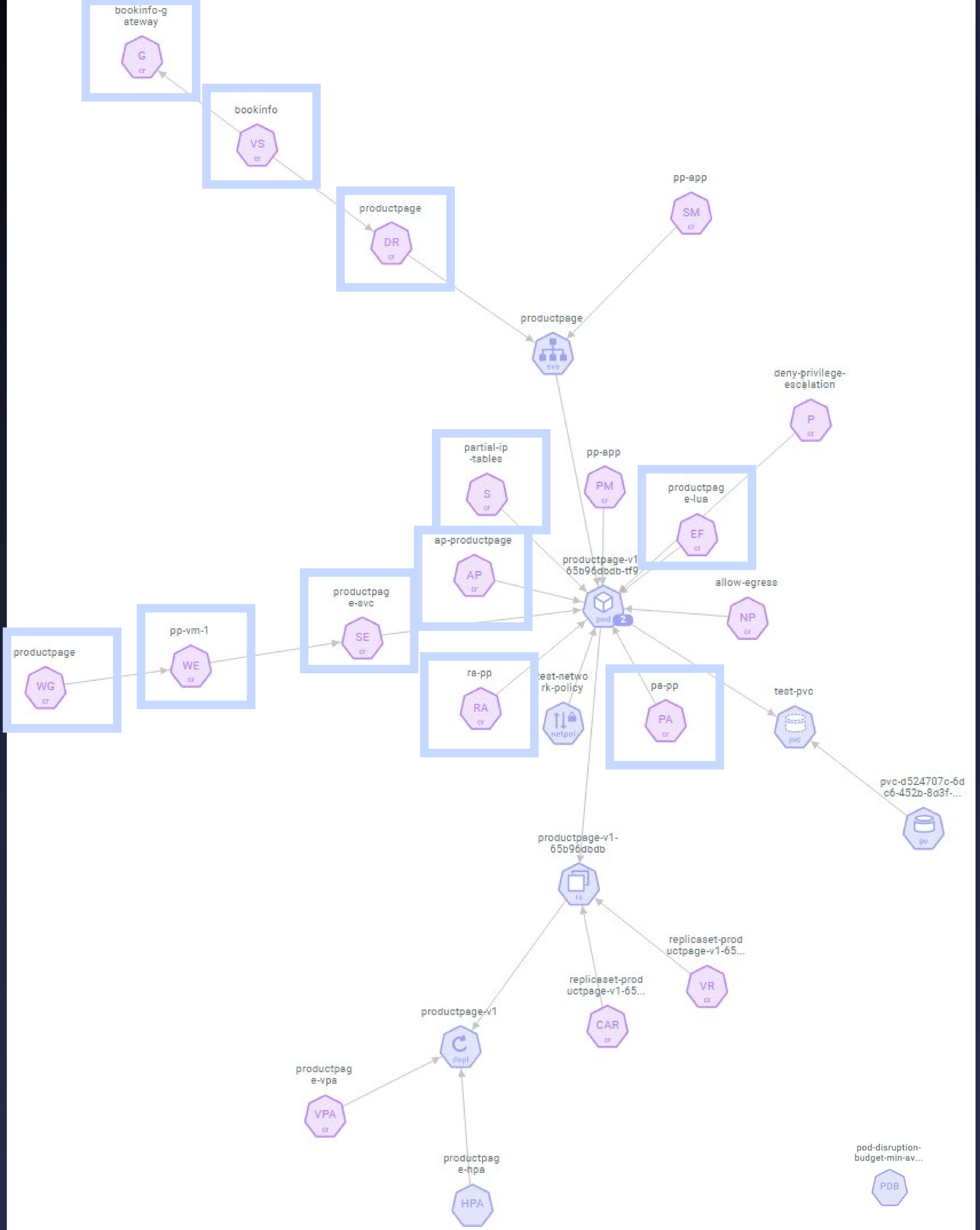
- NetworkPolicy:
  - Calico
  - Native





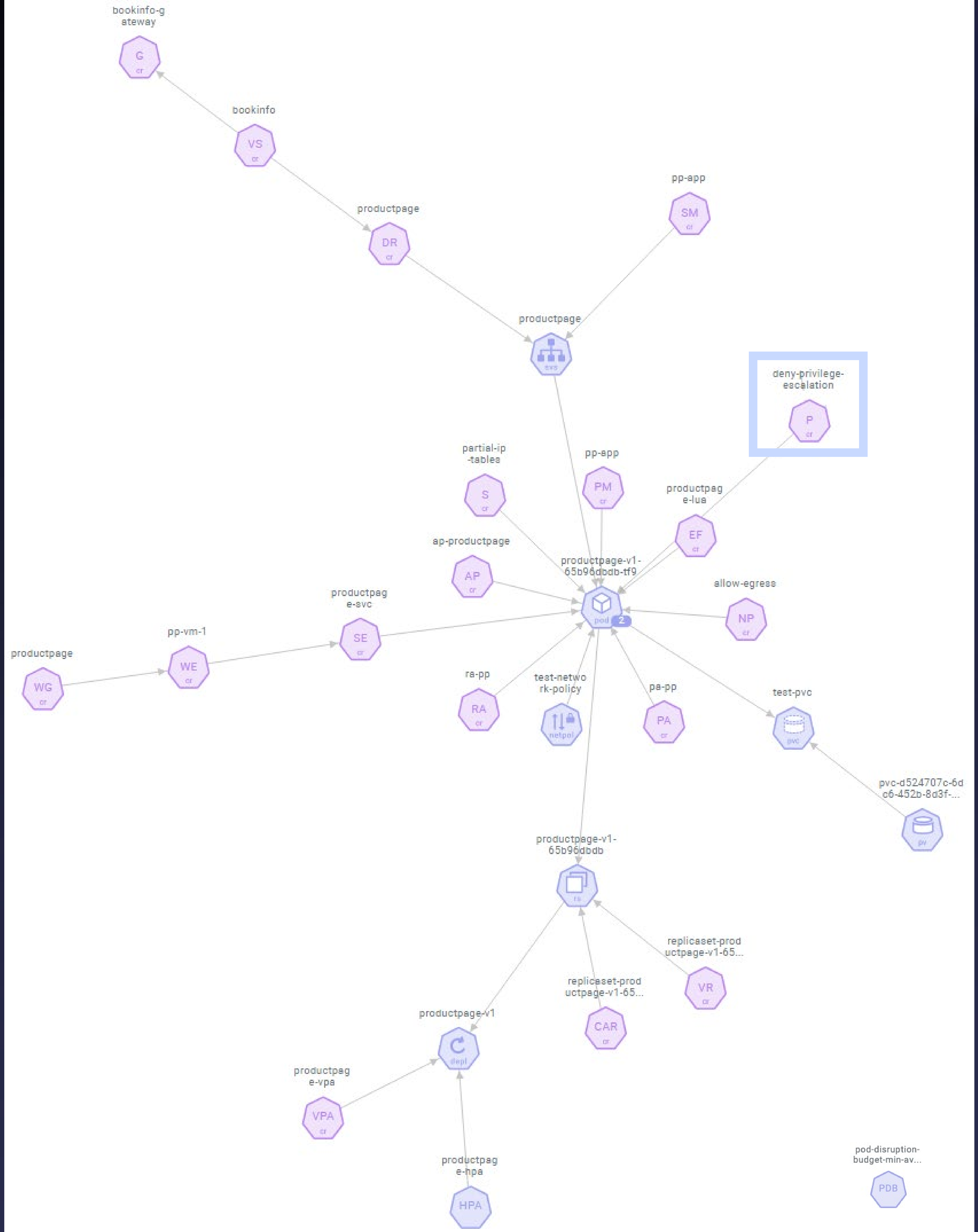
Пример

- Istio:
  - Gateway
  - VirtualService
  - DestinationRule
  - WorkloadGroup
  - WorkloadEntry
  - ServiceEntry
  - Sidecar
  - AuthorizationPolicy
  - RequestAuthentication
  - PeerAuthentication
  - EnvoyFilter



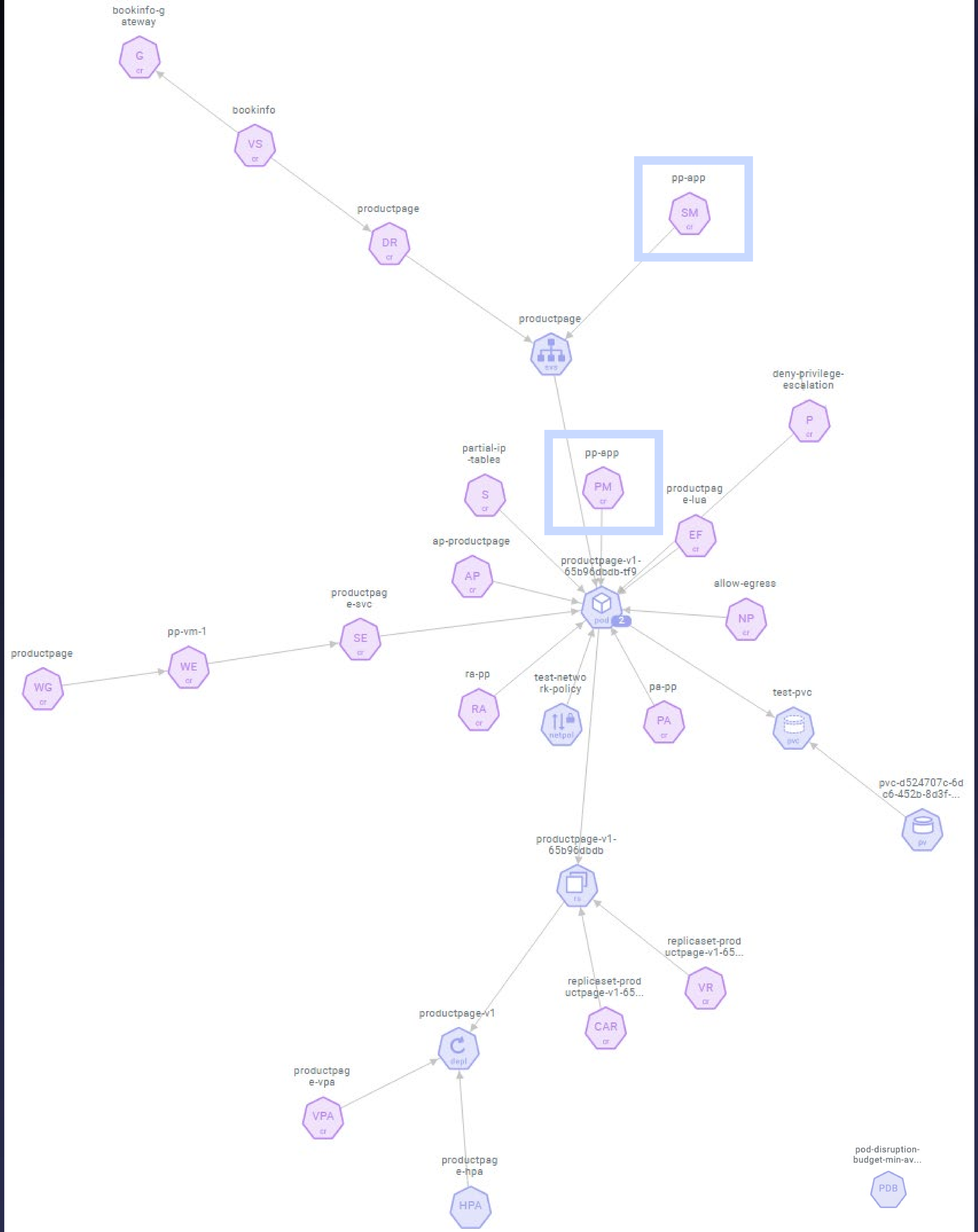
Пример

- Policy Engine:
  - Kyverno Policy



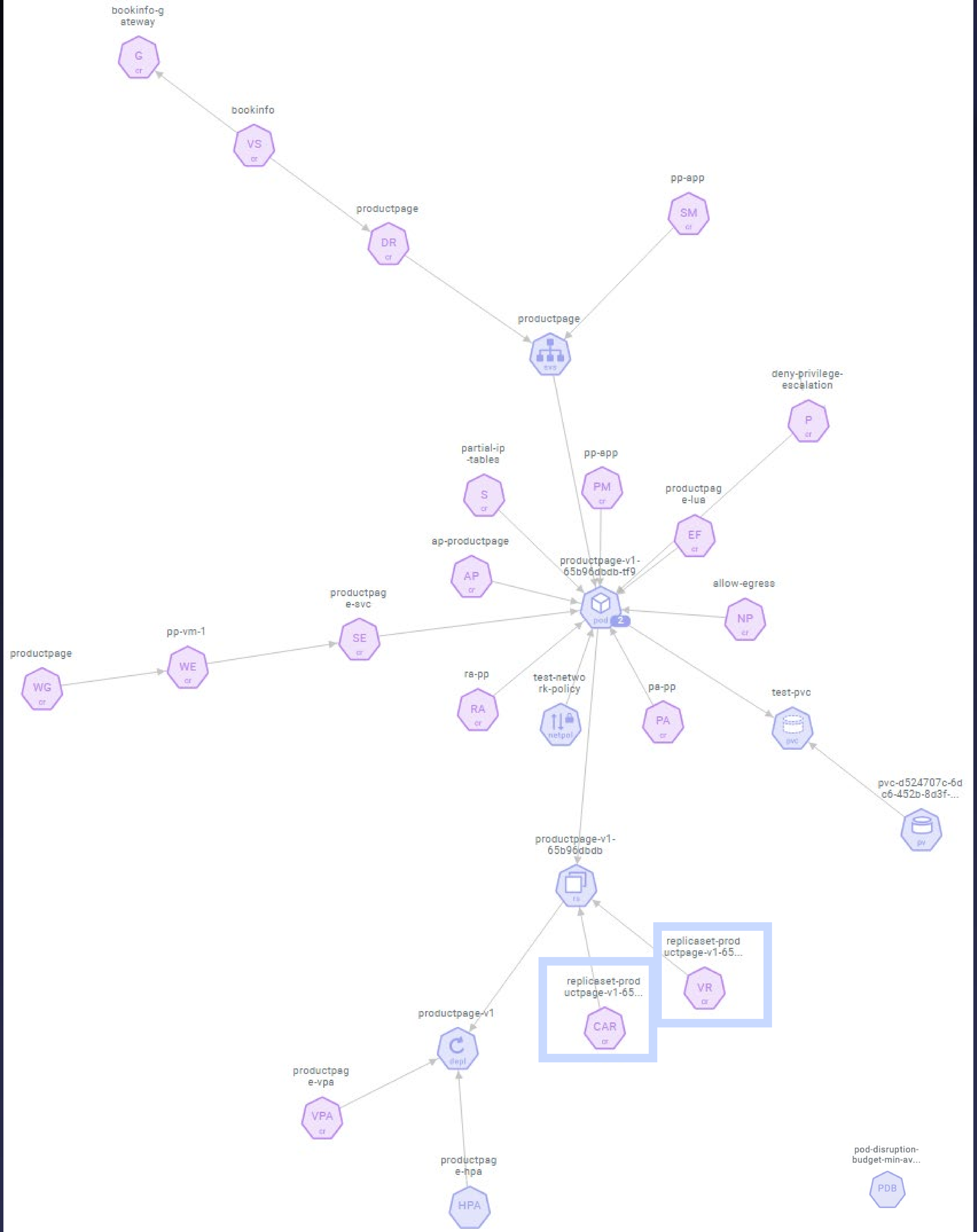
Пример

- Prometheus operator:
  - PodMonitor
  - ServiceMonitor



Пример

- Starboard operator:
  - VulnerabilityReport
  - ConfigReport



Занимательные факты

- Ресурс может указывать на сущность, которой нет

  Пример: Почти любой ресурс =)

- Ресурс может указывать на сущность за пределами кластера

  Пример: Endpoint на IP за пределами кластера

- Namespaced-ресурсы могут ссылаться на Cluster-wide-ресурсы и наоборот

  Пример: PVC на PV или Cluster NetworkPolicy на Pod

- Git не источник правды – он может врать:

  Пример: MutatingAdmissionWebhook или generate rule from Policy Engine

Мысли #2

- Приложение в Kubernetes это не список ресурсов, а дерево ресурсов
- Наличие ресурса в системе не говорит о том, что он что-то делает или на что-то влияет.
- Связи между Kubernetes ресурсами создают контекст для той или ной сущности

Заключение

Выводы

1. KRM-подход поможет выжать максимум из Kubernetes
2. Kubernetes – это система для всех, а не для одного департамента
3. Everything-as-Code ближе, чем может казаться ;)

Q&A

Спасибо за внимание!

Дмитрий Евдокимов

Founder, CTO

Email: de@luntry.com

Twitter: @evdokimovds

Telegram: @Qu3b3c

LUNTRY

luntry.ru