Ural Digital Weekend 2023

Создание NetworkPolicy в Kubernetes: ключевые аспекты и рекомендации для разработчиков

Сергей Канибор
Security Researcher, Luntry

whoami

- R&D/Container Security в Luntry
- Специализируюсь на безопасности контейнеров и Kubernetes
- Спикер PHDays, VolgaCTF, HackConf, CyberCamp, БЕКОН
- Редактор телеграм канала @k8security

Agenda

- WTF Kubernetes ?!?

Agenda

- WTF Kubernetes ?!?
- NetworkPolicy

Agenda

- WTF Kubernetes ?!?
- NetworkPolicy
- Создание сетевых политик

Agenda

- WTF Kubernetes ?!?
- NetworkPolicy
- Создание сетевых политик
- Контроль ресурсов

Зачем разработчикам знать про NetworkPolicy ?!

Зачем контролировать?

- Неправильно написанная политика может сломать логику работы вашего сервиса

Зачем контролировать?

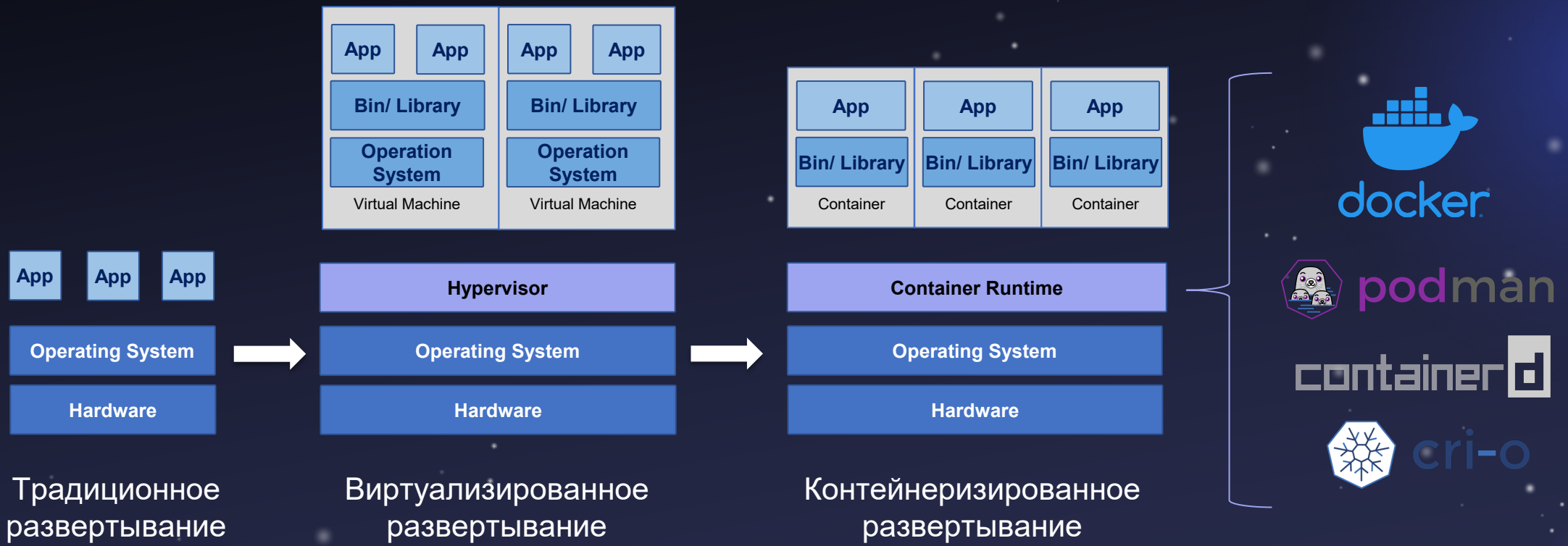
- Неправильно написанная политика может сломать логику работы вашего сервиса
- Контроль умешает поверхность атаки

Зачем контролировать?

- Неправильно написанная политика может сломать логику работы вашего сервиса
- Контроль умешает поверхность атаки
- Даже если ваше приложение уязвимо, злоумышленник не сможет проэксплуатировать уязвимость

WTF Kubernetes ?!?

Как мы до этого докатились



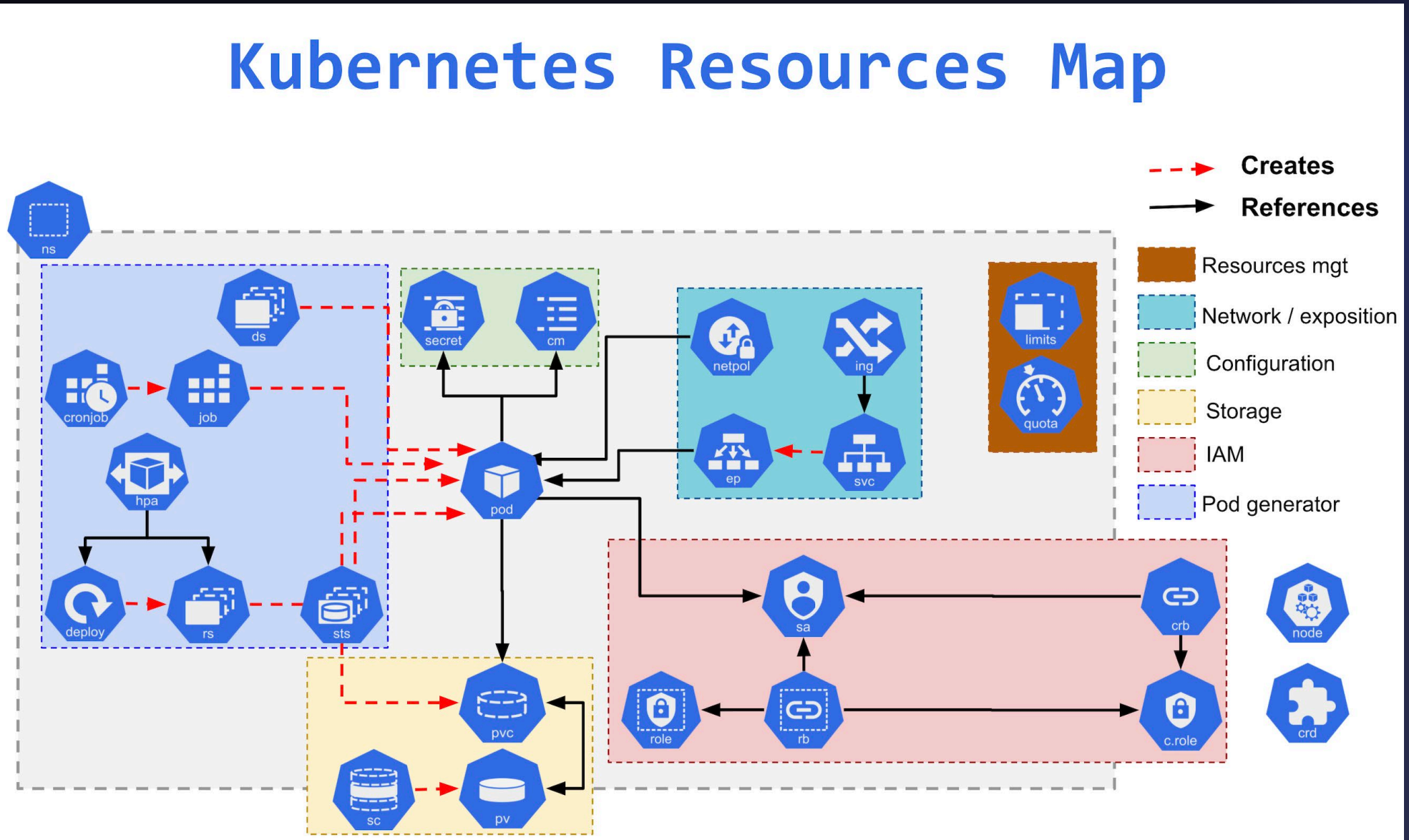
Kubernetes – ядро Linux 21 века

58%

компаний использует Kubernetes

Исследование
Express 42 – State of DevOps Russia 2023

Kubernetes == куча сущностей



Начнем с нюансов

Есть нюансы

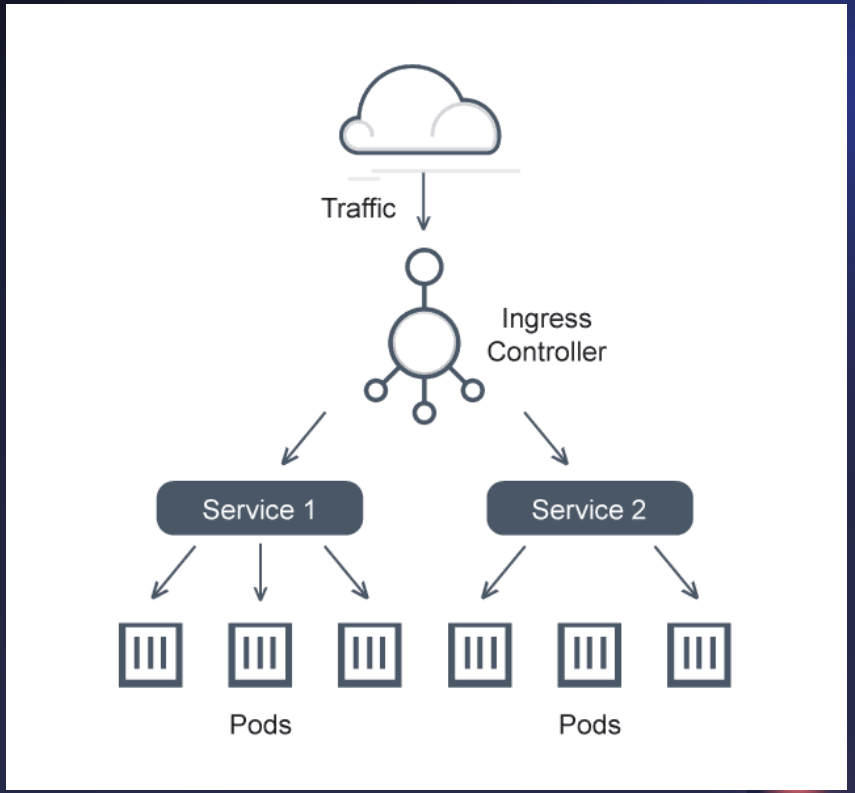
Условия:

- GitOps

Есть нюансы

Условия:

- GitOps

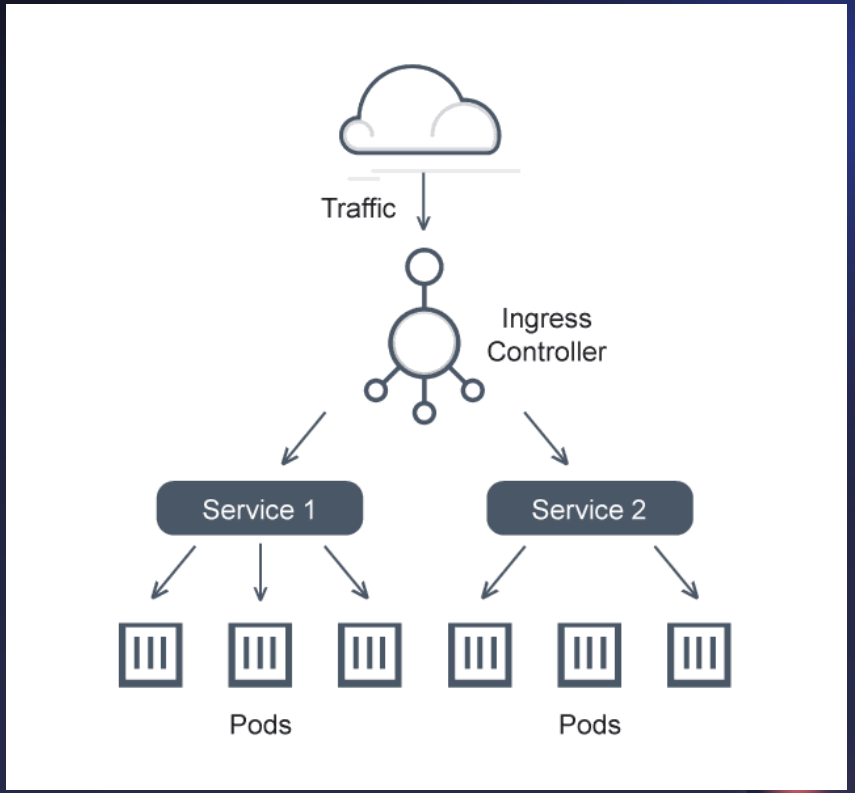


Есть нюансы

Условия:

- GitOps

Приложение:



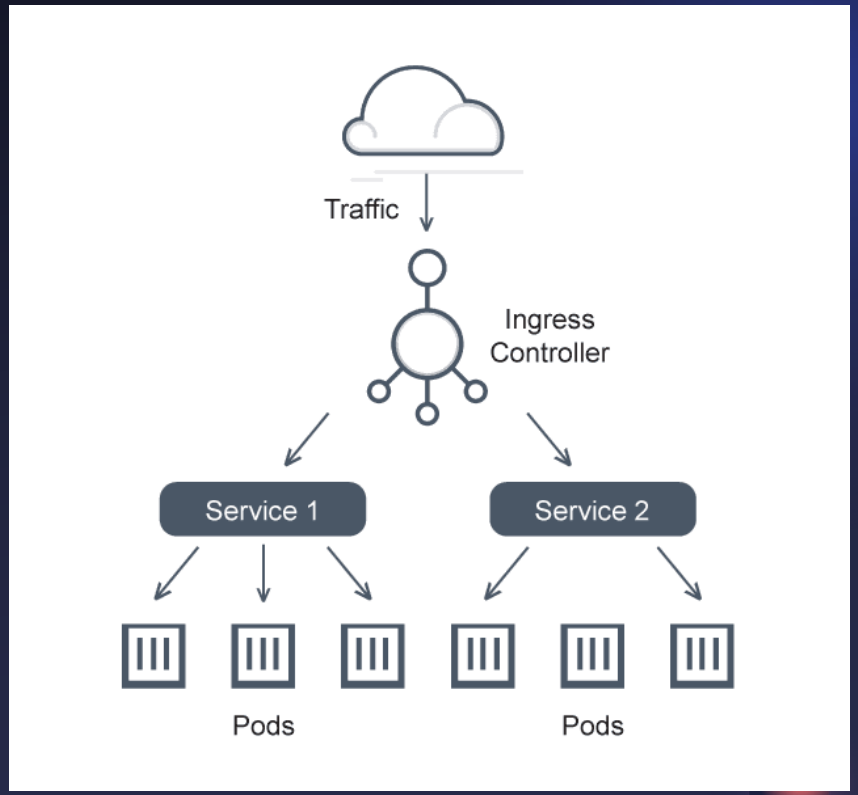
Есть нюансы

Условия:

- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress



Есть нюансы

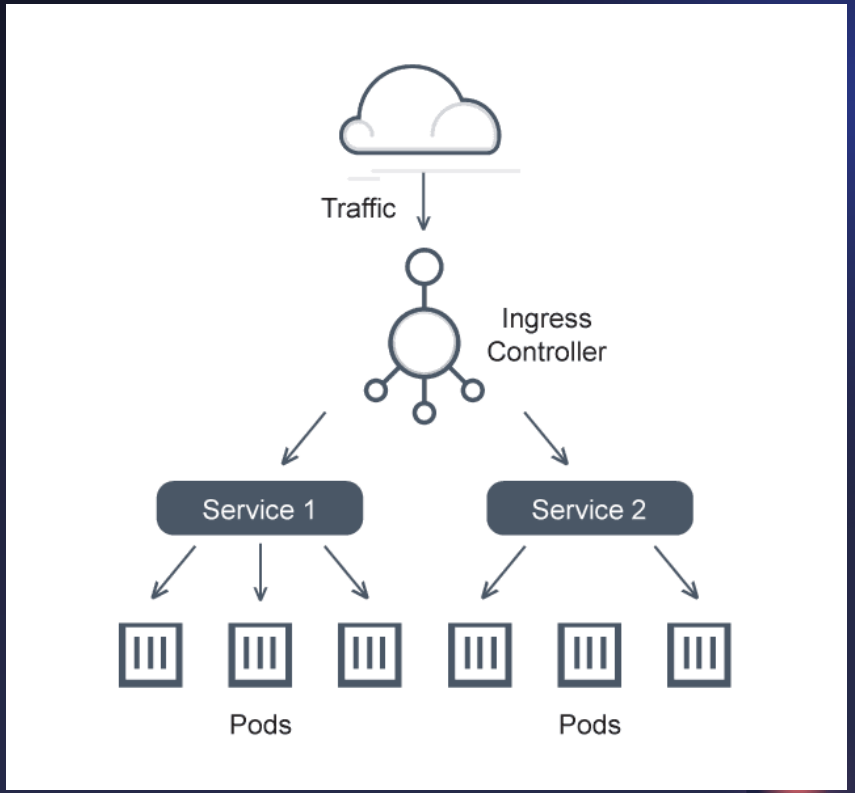
Условия:

- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress

Что нужно знать и учитывать:



Есть нюансы

Условия:

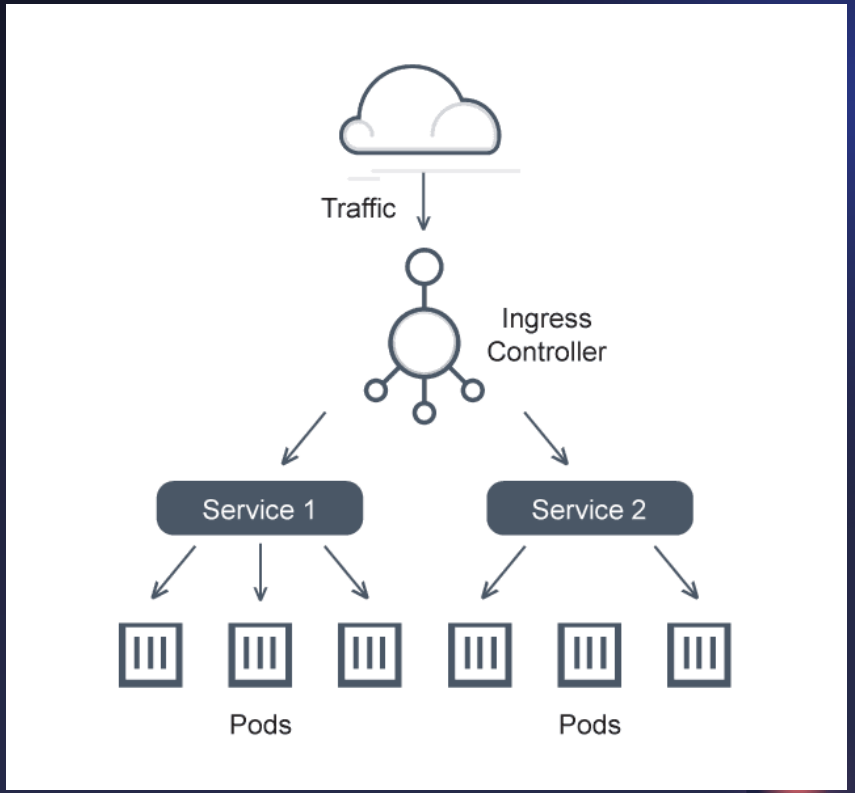
- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress

Что нужно знать и учитывать:

- Знание labels у взаимодействующих namespaces



Есть нюансы

Условия:

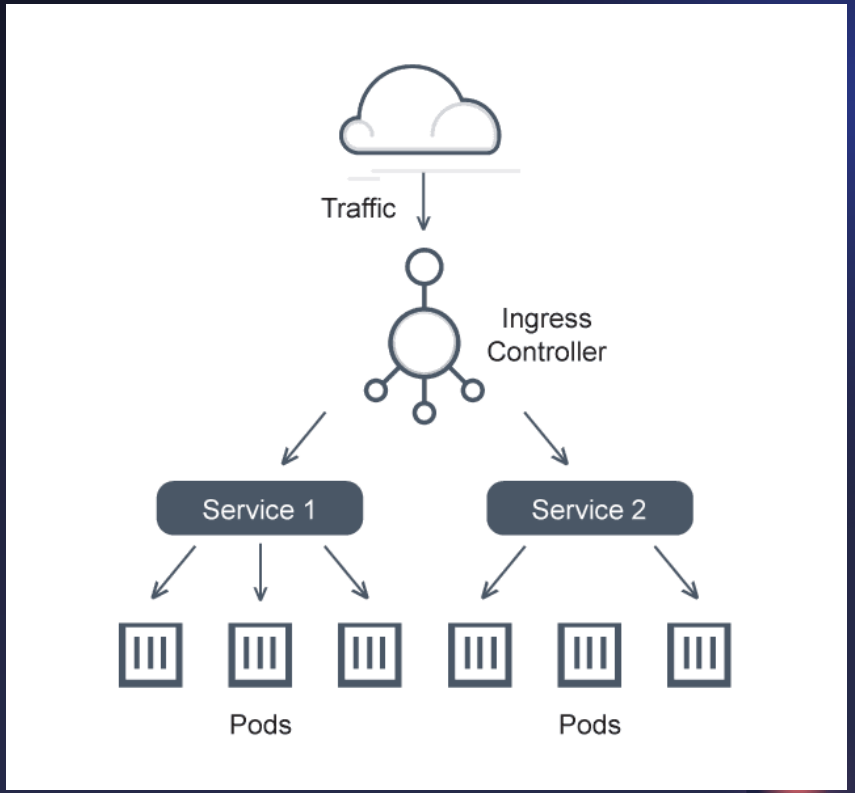
- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress

Что нужно знать и учитывать:

- Знание labels у взаимодействующих namespaces
- Используется ли ServiceMesh?



Есть нюансы

Условия:

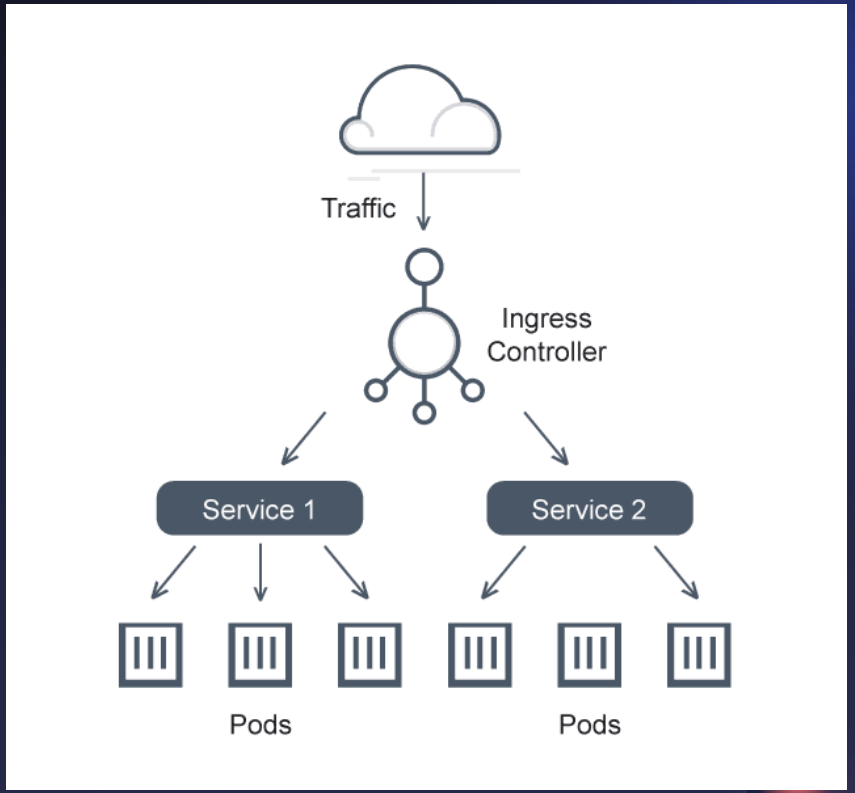
- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress

Что нужно знать и учитывать:

- Знание labels у взаимодействующих namespaces
- Используется ли ServiceMesh?
- Какая реализация Ingress?



Есть нюансы

Условия:

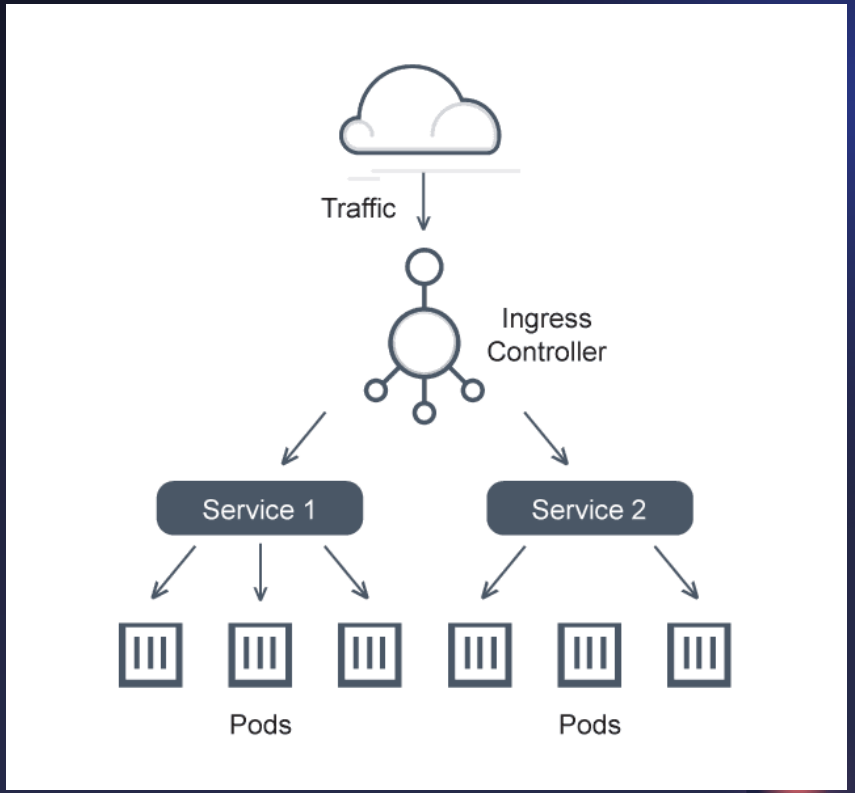
- GitOps

Приложение:

- 2 Deployments, 2 Service, 1 Ingress

Что нужно знать и учитывать:

- Знание labels у взаимодействующих namespaces
- Используется ли ServiceMesh?
- Какая реализация Ingress?
- Куда отдаются метрики, логи и трейсы?



Работа с нюансами

Работа с нюансами

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса

Работа с нюансами

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые NetworkPolicy – труд нескольких департаментов

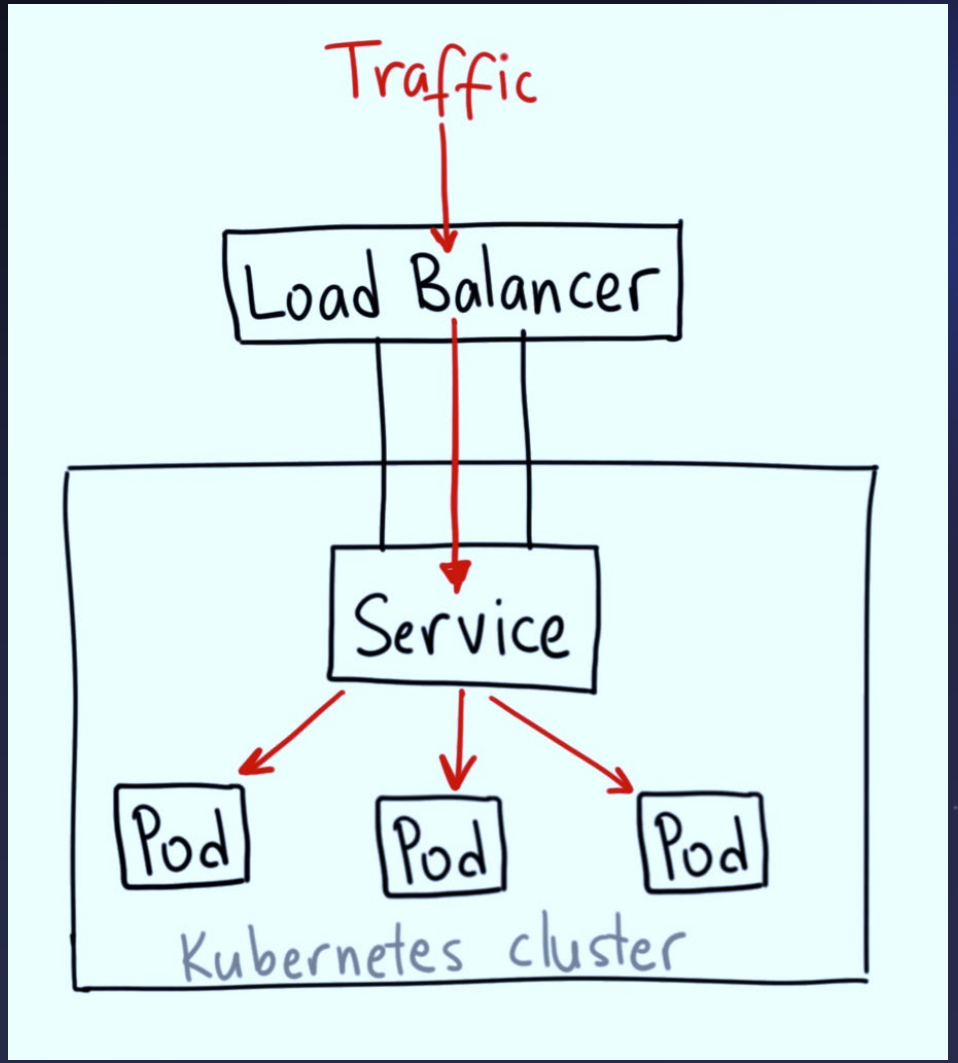
Работа с нюансами

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые NetworkPolicy – труд нескольких департаментов
- Разработчик лучше всех (должен) понимает бизнес-логику микросервиса

Kubernetes Network 101

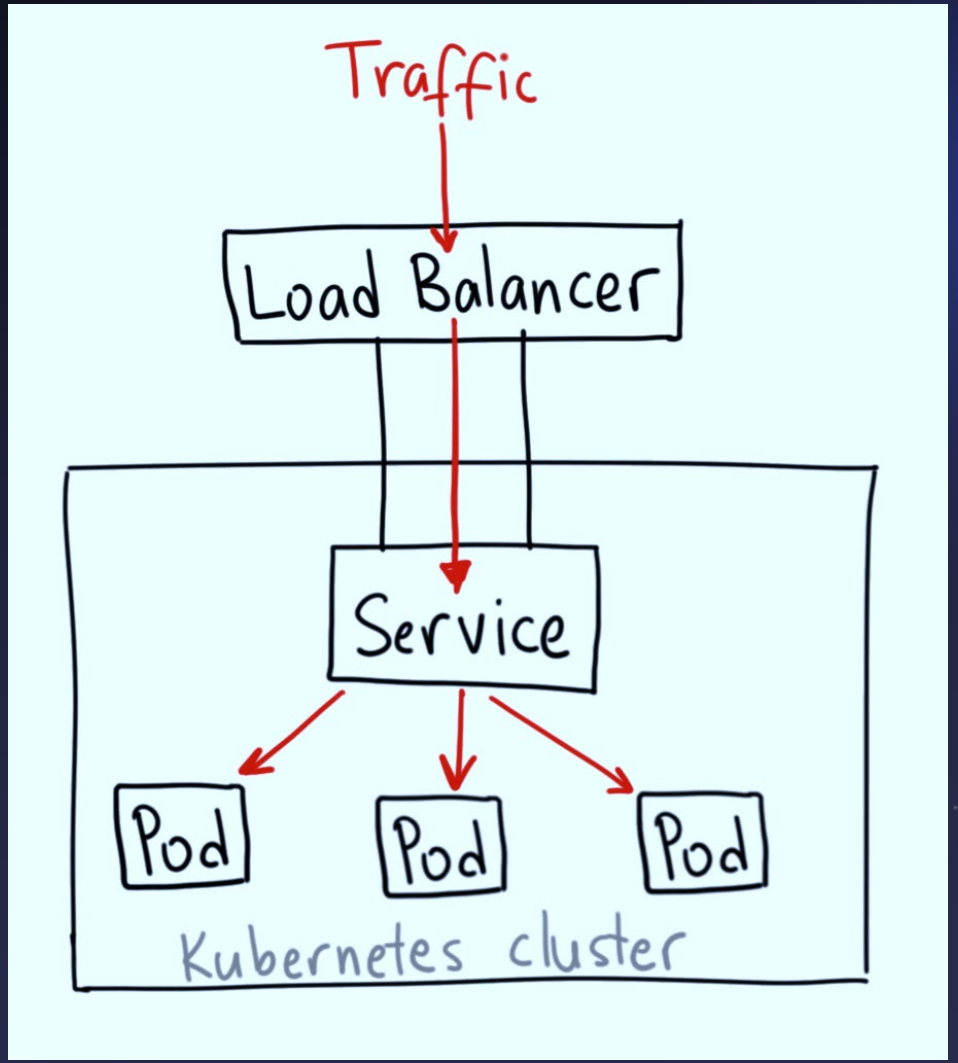
Kubernetes Network 101

- У всех подов есть IP



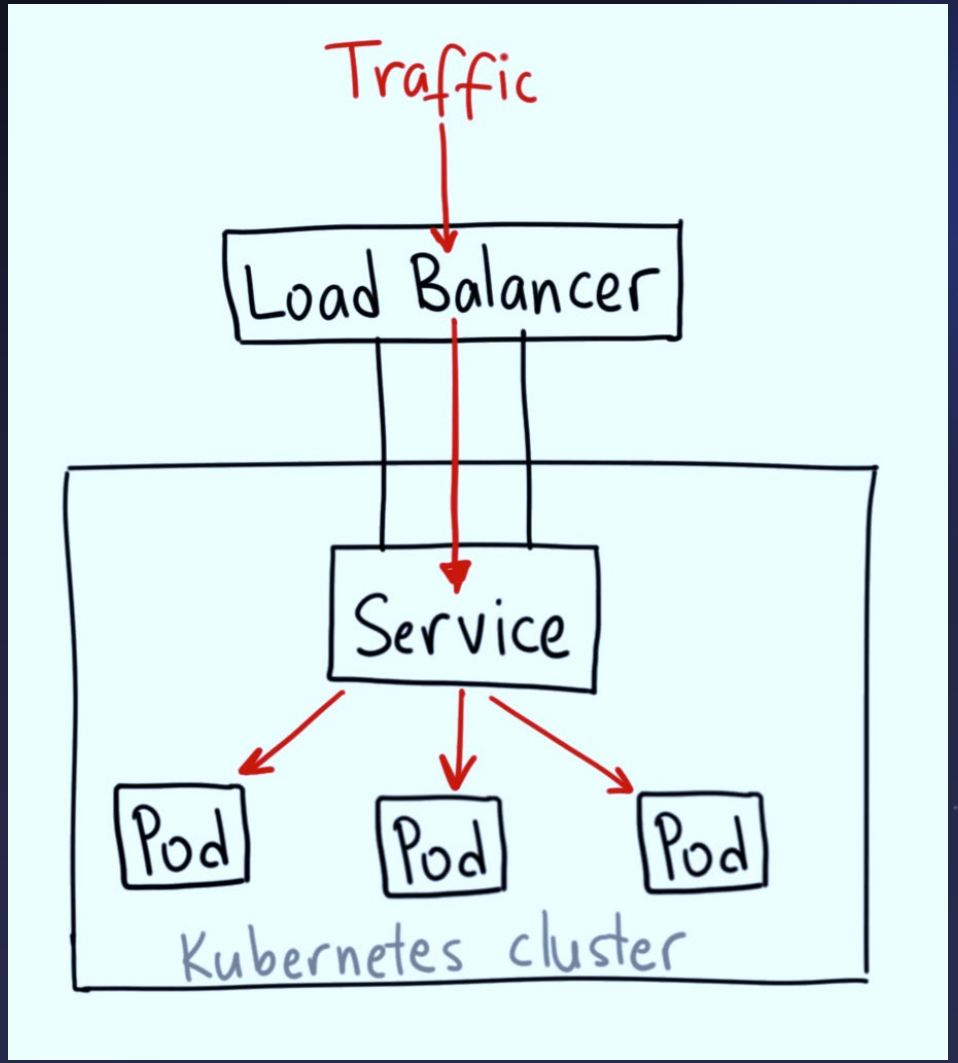
Kubernetes Network 101

- У всех подов есть IP
- PodCIDR на каждой* Node



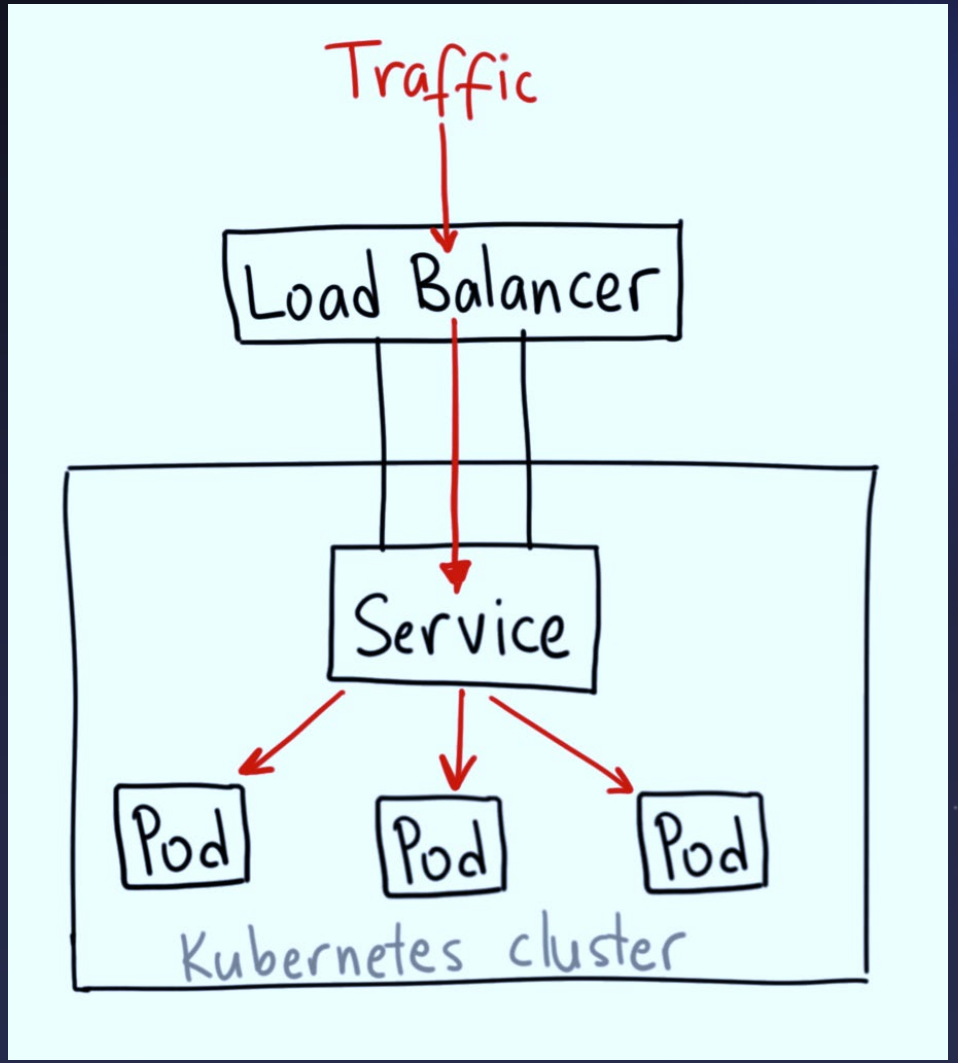
Kubernetes Network 101

- У всех подов есть IP
- PodCIDR на каждой* Node
- Service для load-balancing



Kubernetes Network 101

- У всех подов есть IP
- PodCIDR на каждой* Node
- Service для load-balancing
- DNS для service-discovery



Сетевое поведение Pod’ов

По умолчанию Pods в Kubernetes могут коммуницировать со всеми Pods без ограничений:

Сетевое поведение Pod’ов

По умолчанию Pods в Kubernetes могут коммуницировать со всеми Pods без ограничений:

- North-south traffic – с ресурсами за пределами Kubernetes

Сетевое поведение Pod’ов

По умолчанию Pods в Kubernetes могут коммуницировать со всеми Pods без ограничений:

- North-south traffic – с ресурсами за пределами Kubernetes
- East-west traffic – с другими приложениями в Kubernetes

Сетевое поведение Pod’ов

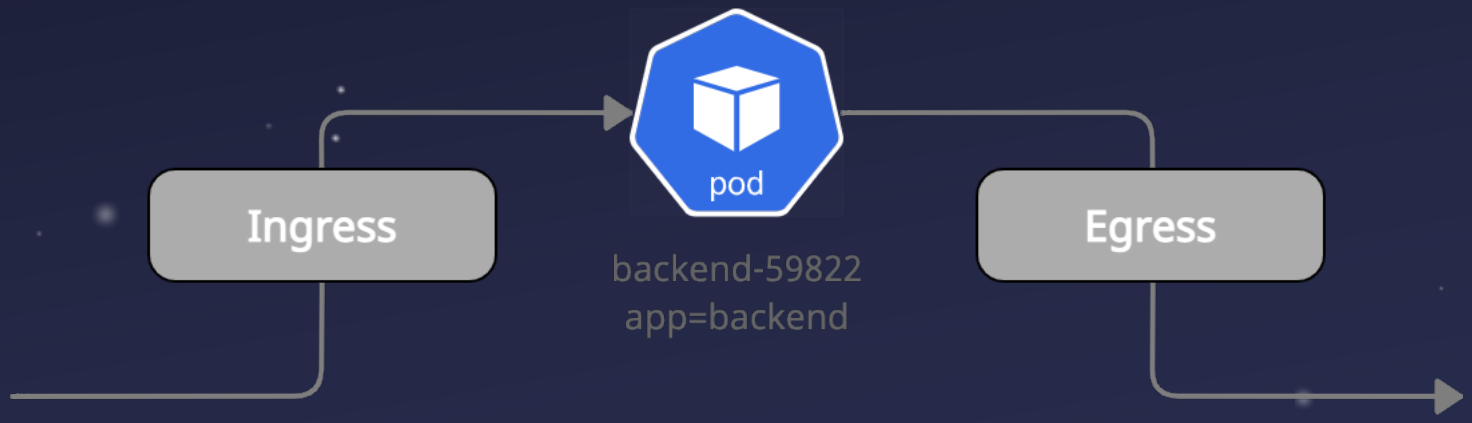
По умолчанию Pods в Kubernetes могут коммуницировать со всеми Pods без ограничений:

- North-south traffic – с ресурсами за пределами Kubernetes
- East-west traffic – с другими приложениями в Kubernetes
- Трафик не шифруется

Сетевое поведение Pod’ов

По умолчанию Pods в Kubernetes могут коммуницировать со всеми Pods без ограничений*:

- North-south traffic – с ресурсами за пределами Kubernetes
- East-west traffic – с другими приложениями в Kubernetes
- Трафик не шифруется



* Александр Кожемякин – «Как внедрить Default Deny на живом кластере и выжить»

Container Networking Interface (CNI)

kubectl get all -n kube-system | egrep -i "ACI|Calico|Canal|Cilium|CNI-Genie|Contiv|Contrail|Flannel|Knitter|Multus|OVN-Kubernetes|OVN4NFV-K8S-Plugin|NSX-T|Nuage|Romana|Weave"

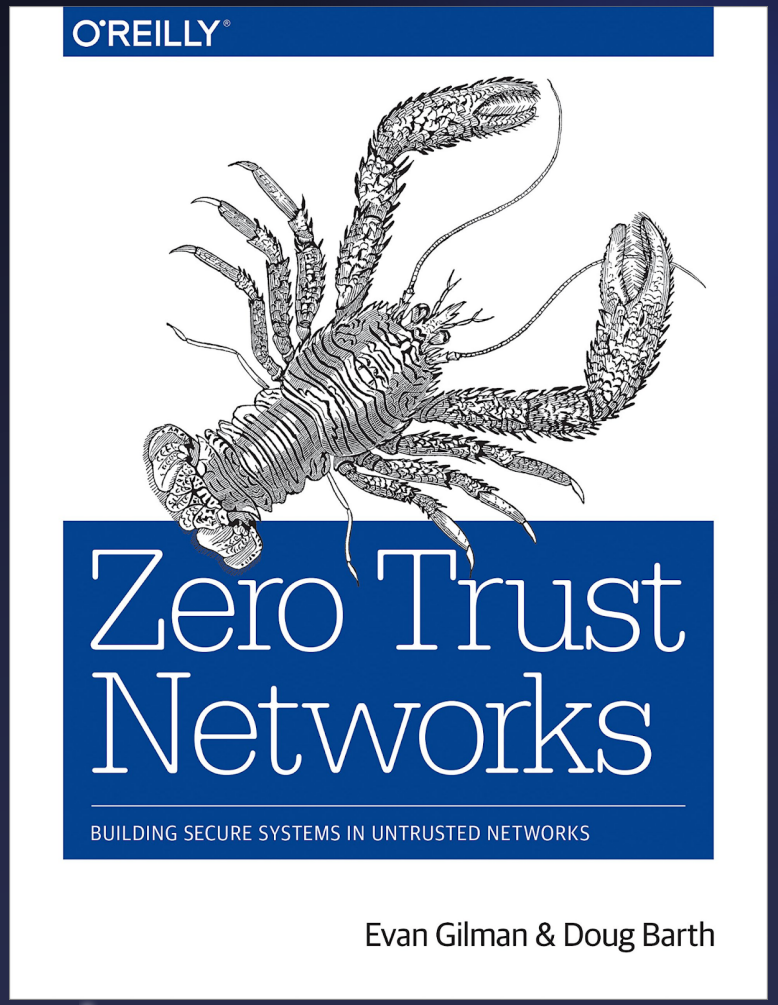
Контроль сети, концепция микросегментации

Концепция Zero Trust

“Zero Trust Networks are resilient even when attackers manage to breach applications or infrastructure. They make it hard for attackers to move laterally, and reconnaissance activities easier to spot.

Organizations that embrace the change control model in this How-To will be able to tightly secure their network without imposing a drag on innovation in their applications. Security teams can be enablers of business value, not roadblocks.”

Документация Calico



Контроль сетевого взаимодействия в Kubernetes

- На уровне приложений
  - Через библиотеку, добавленную при сборке, или через инжект
  - Реализовано в user space
- Дополнительные оверхеды
- Так делать не надо

Контроль сетевого взаимодействия в Kubernetes

- На уровне Service Mesh (Istio, Linkerd)
  - С помощью sidecar-контейнера
  - Обычно, это ресурс AuthorizationPolicy
  - Уровень L7
  - Реализовано в user space
- Можно рассмотреть как альтернативу
- Все ещё оверхеды

Контроль сетевого взаимодействия в Kubernetes

- На уровне CNI
  - Native или Custom NetworkPolicy
  - Уровень L3-L4 (иногда и L7)
  - Реализовано в kernel space

YAML NetworkPolicy

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

имя политики

в каком namespace действует

то к каким Pods будет применена политика

типы правил в политике. По умолчанию, Ingress.

блок ingress правил

Откуда трафик

1 источник

2 источник

3 источник

На какой порт

блок egress правил

Куда трафик

На какой порт

policyTypes – Native

Native, Calico: если никакой policyTypes не указан, то по умолчанию будет выставлен Ingress.

Cilium: если никакой policyTypes не указан, то работать не будет

Ingress rule present?	Egress rule present?	Value
No	No	Ingress (not working, Cilium)
Yes	No	Ingress
No	Yes	Egress
Yes	Yes	Ingress, Egress

Custom network policy

- Позволяет ограничивать доступ не только к Pod

Custom NetworkPolicy

- Позволяет ограничивать доступ не только к Pod
- L3,L4,L7 политики

Custom network policy

- Позволяет ограничивать доступ не только к Pod
- L3,L4,L7 политики
- DNS-based политики

Custom network policy

- Позволяет ограничивать доступ не только к Pod
- L3,L4,L7 политики
- DNS-based политики
- Clusterwide политики

Custom network policy

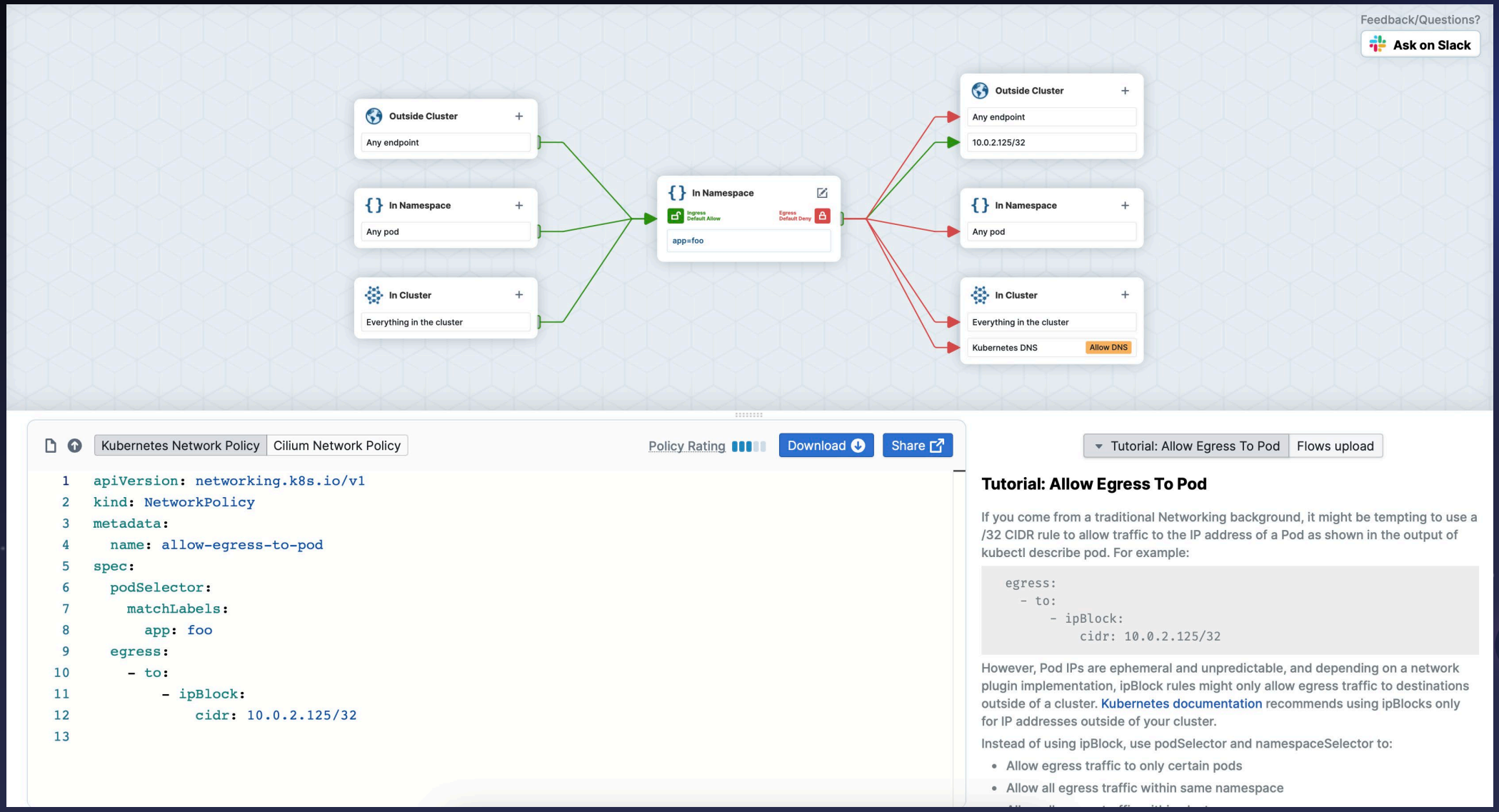
- Позволяет ограничивать доступ не только к Pod
- L3,L4,L7 политики
- DNS-based политики
- Clusterwide политики
- Расширенный функционал работы селекторов

Custom network policy

- Позволяет ограничивать доступ не только к Pod
- L3,L4,L7 политики
- DNS-based политики
- Clusterwide политики
- Расширенный функционал работы селекторов
- Привязка к Service, ServiceAccount, ...

Создание Сетевых Политик

Cilium Editor



Cilium Editor

- Можно легко накликивать нужные политики

Cilium Editor

- Можно легко накликивать нужные политики
- Есть возможность автоматической генерации по логам

Cilium Editor

- Можно легко накликивать нужные политики
- Есть возможность автоматической генерации по логам
- `hubble observe --output jsonpb --last 1000 --follow --namespace your-namespace > your-namespace-flows.json`

Cilium Editor

- Можно легко накликивать нужные политики
- Есть возможность автоматической генерации по логам
- `hubble observe --output jsonpb --last 1000 --follow --namespace your-namespace > your-namespace-flows.json`
- Сервис висит в Интернете

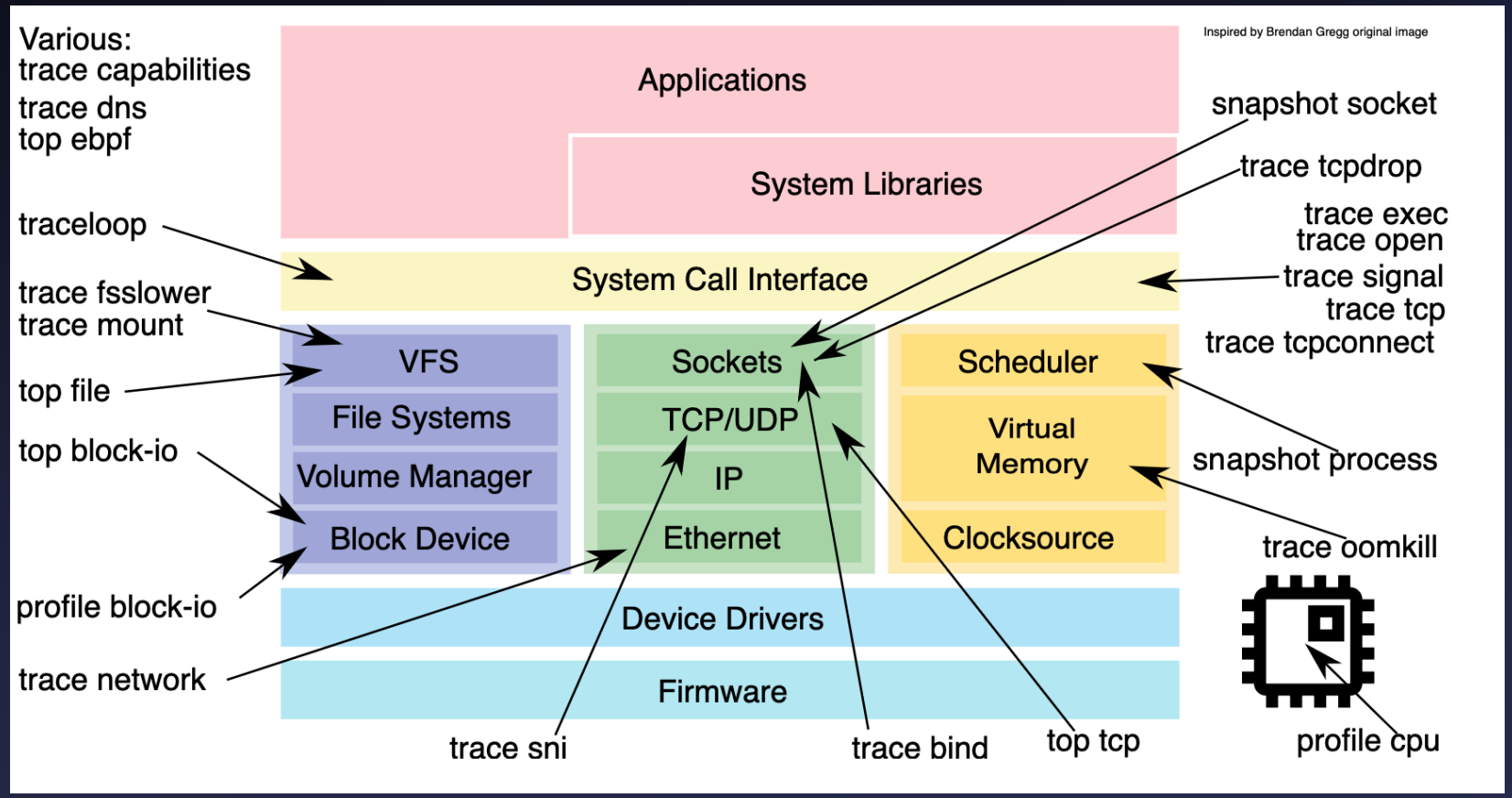
Cilium Editor

- Можно легко накликивать нужные политики
- Есть возможность автоматической генерации по логам
- `hubble observe --output jsonpb --last 1000 --follow --namespace your-namespace > your-namespace-flows.json`
- Сервис висит в Интернете
- Генерация политик по логам в конкретный момент времени

Cilium Editor

- Можно легко накликивать нужные политики
- Есть возможность автоматической генерации по логам
- `hubble observe --output jsonpb --last 1000 --follow --namespace your-namespace > your-namespace-flows.json`
- Сервис висит в Интернете
- Генерация политик по логам в конкретный момент времени
- Завязка на конкретный CNI и дополнительные ресурсы на сервис hubble

Inspektor gadget



Inspektor gadget – generate network policy

- `kubectl krew install gadget`

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`
- `kubectl gadget advise network-policy monitor -n demo --output ./networktrace.log`

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`
- `kubectl gadget advise network-policy monitor -n demo --output ./networktrace.log`
- `kubectl gadget advise network-policy report --input ./networktrace.log > network-policy.yaml`

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`
- `kubectl gadget advise network-policy monitor -n demo --output ./networktrace.log`
- `kubectl gadget advise network-policy report --input ./networktrace.log > network-policy.yaml`
- Нужны дополнительные ресурсы

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`
- `kubectl gadget advise network-policy monitor -n demo --output ./networktrace.log`
- `kubectl gadget advise network-policy report --input ./networktrace.log > network-policy.yaml`
- Нужны дополнительные ресурсы
- Вся генерация только в конкретный момент времени

Inspektor gadget – generate network policy

- `kubectl krew install gadget`
- `kubectl gadget deploy`
- `kubectl gadget advise network-policy monitor -n demo --output ./networktrace.log`
- `kubectl gadget advise network-policy report --input ./networktrace.log > network-policy.yaml`
- Нужны дополнительные ресурсы
- Вся генерация только в конкретный момент времени
- Есть инструменты, где получение политик автоматизировано, но как правило они платные

Контроль ресурсов

Используем CODEOWNERS

- Есть в платной версии Gitlab

Используем CODEOWNERS

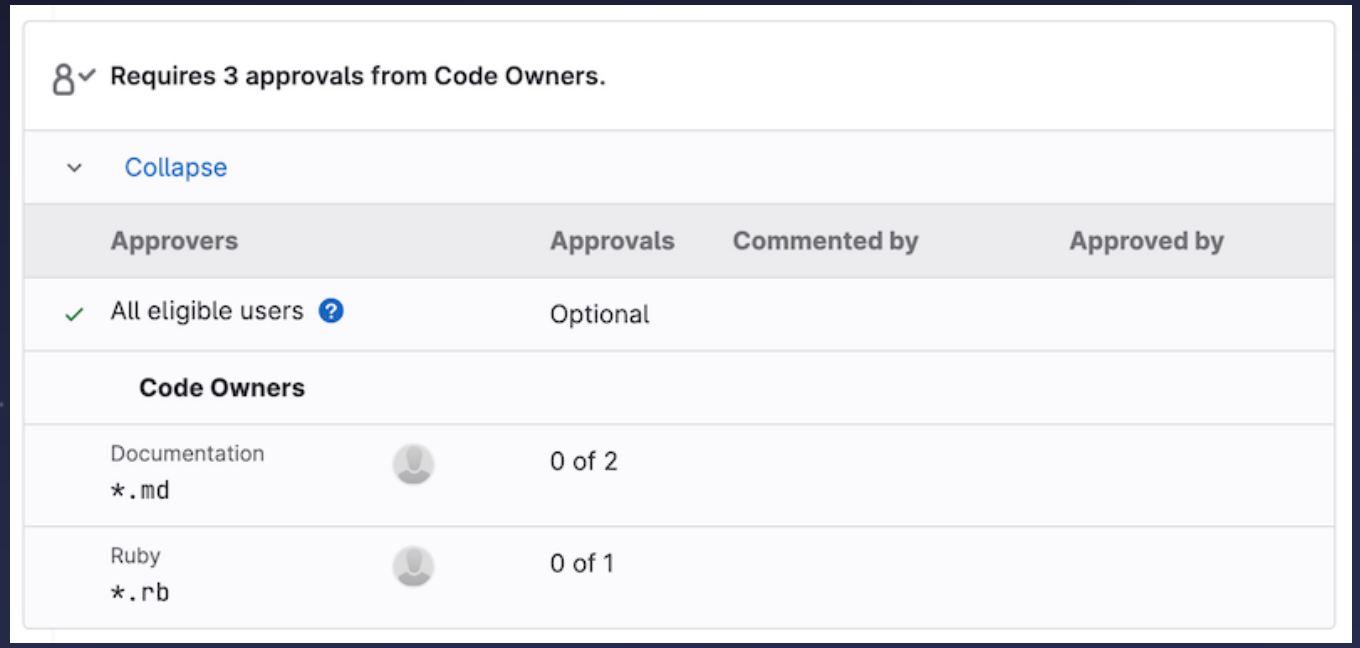
- Есть в платной версии Gitlab
- Ревьюверы назначаются автоматически

Используем CODEOWNERS

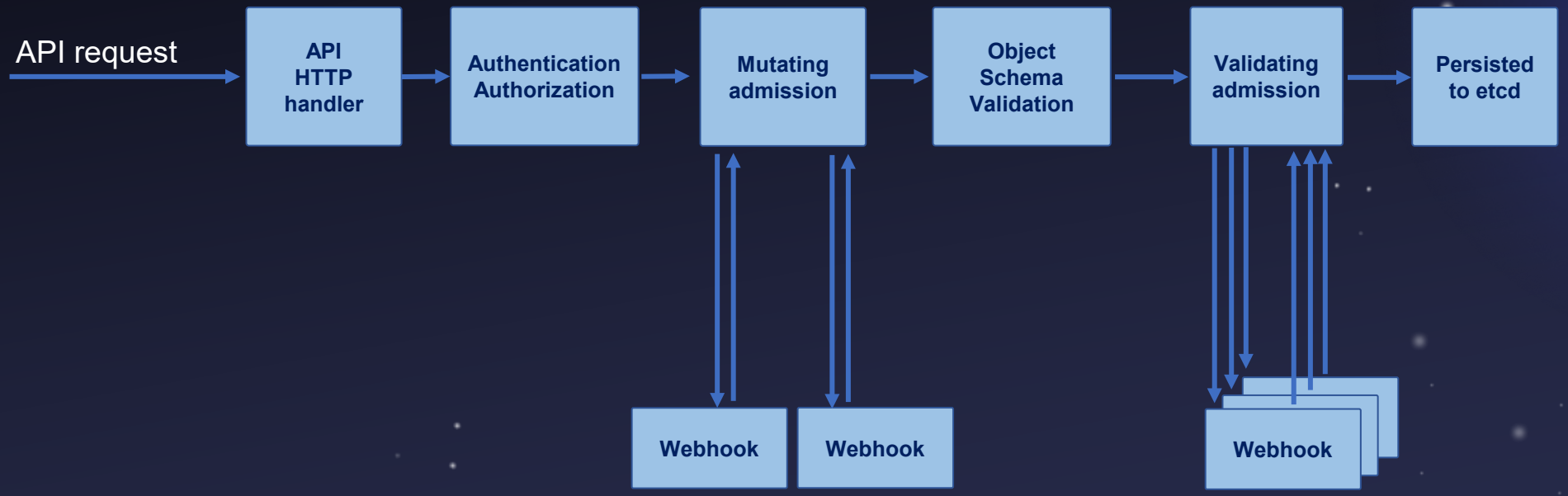
- Есть в платной версии Gitlab
- Ревьюверы назначаются автоматически
- Мердж не пройдет пока владелец файла не одобрит реквест

Используем CODEOWNERS

- Есть в платной версии Gitlab
- Ревьюверы назначаются автоматически
- Мердж не пройдет пока владелец файла не одобрит реквест



Путь до деплоя



Используем PolicyEngine

- В Kyverno есть возможность писать Validation Rule, учитывая наличие подписи у ресурса

Используем PolicyEngine

- В Kyverno есть возможность писать Validation Rule, учитывая наличие подписи у ресурса
- Генерируем пары ключей для cosign

```
$ cosign generate-key-pair
```

Используем PolicyEngine

- В Kyverno есть возможность писать Validation Rule, учитывая наличие подписи у ресурса
- Генерируем пары ключей для cosign

```
$ cosign generate-key-pair
```

- С помощью утилиты kubectl-sigstore подписываем наш манифест закрытым ключом; на выходе получаем новый подписанный манифест

```
$ kubectl-sigstore sign -f secret.yaml -k cosign.key --tarball no -o
networkpolicy-signed.yaml
```

Используем PolicyEngine

- В Kyverno есть возможность писать Validation Rule, учитывая наличие подписи у ресурса
- Генерируем пары ключей для cosign

```
$ cosign generate-key-pair
```

- С помощью утилиты kubectl-sigstore подписываем наш манифест закрытым ключом; на выходе получаем новый подписанный манифест

```
$ kubectl-sigstore sign -f secret.yaml -k cosign.key --tarball no -o
secret-signed.yaml
```

- В политике для kyverno указываем открытый ключ, полученный на первом этапе

Используем PolicyEngine

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: validate-secrets
spec:
  validationFailureAction: Enforce
  background: true
  rules:
    - name: validate-secrets
      match:
        any:
        - resources:
            kinds:
              - NetworkPolicy
      validate:
        manifests:
          attestors:
          - count: 1
            entries:
            - keys:
                publicKeys: |-
                  -----BEGIN PUBLIC KEY-----
                  MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEStoX3dPCFYFD2uPgTjZOf1I5UFTa
                  1tIu7uoGoyTxJqqEq7K2aqU+vy+aK76uQ5mcllc+TymVtcLk10kcKvb3FQ==
```

Добавляем GitOps оператор

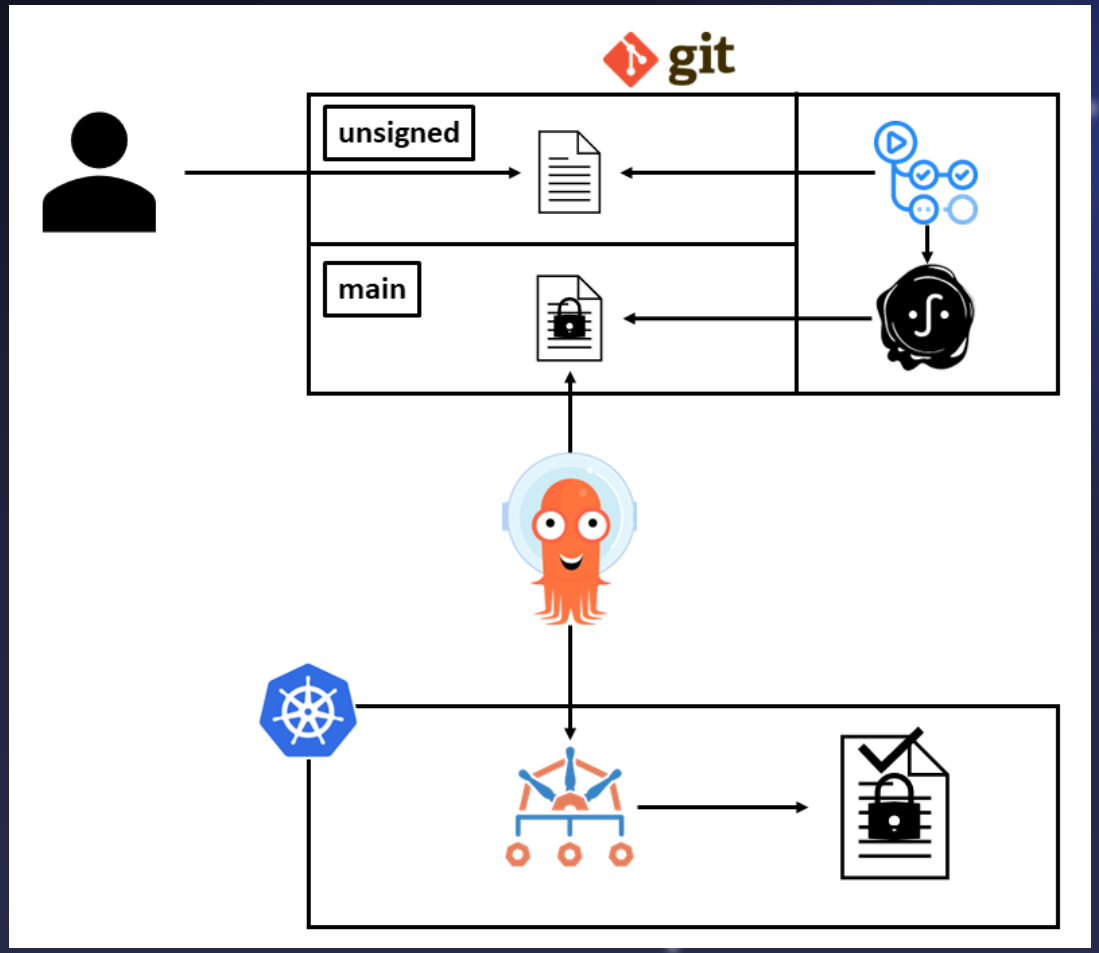
- Kyverno отлично встраивается в CI/CD

Добавляем GitOps оператор

- Kyverno отлично встраивается в CI/CD
- Можно подписывать ресурсы через keyless signing

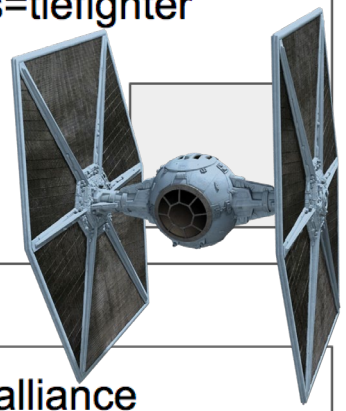
Добавляем GitOps оператор

- Kyverno отлично встраивается в CI/CD
- Можно подписывать ресурсы через keyless signing
- Настраиваем ArgoCD таким образом, чтобы деплоились ресурсы только из подписанной ветки

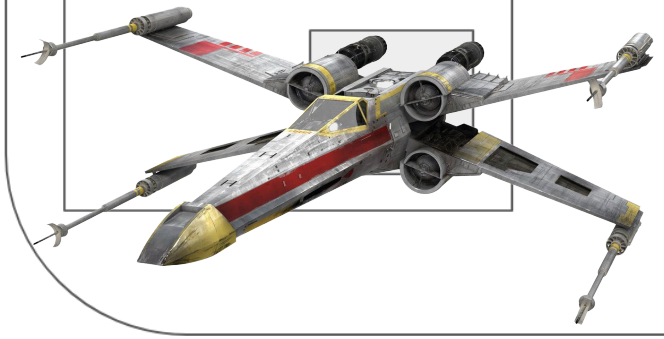


Demo time

org=empire
class=tiefighter

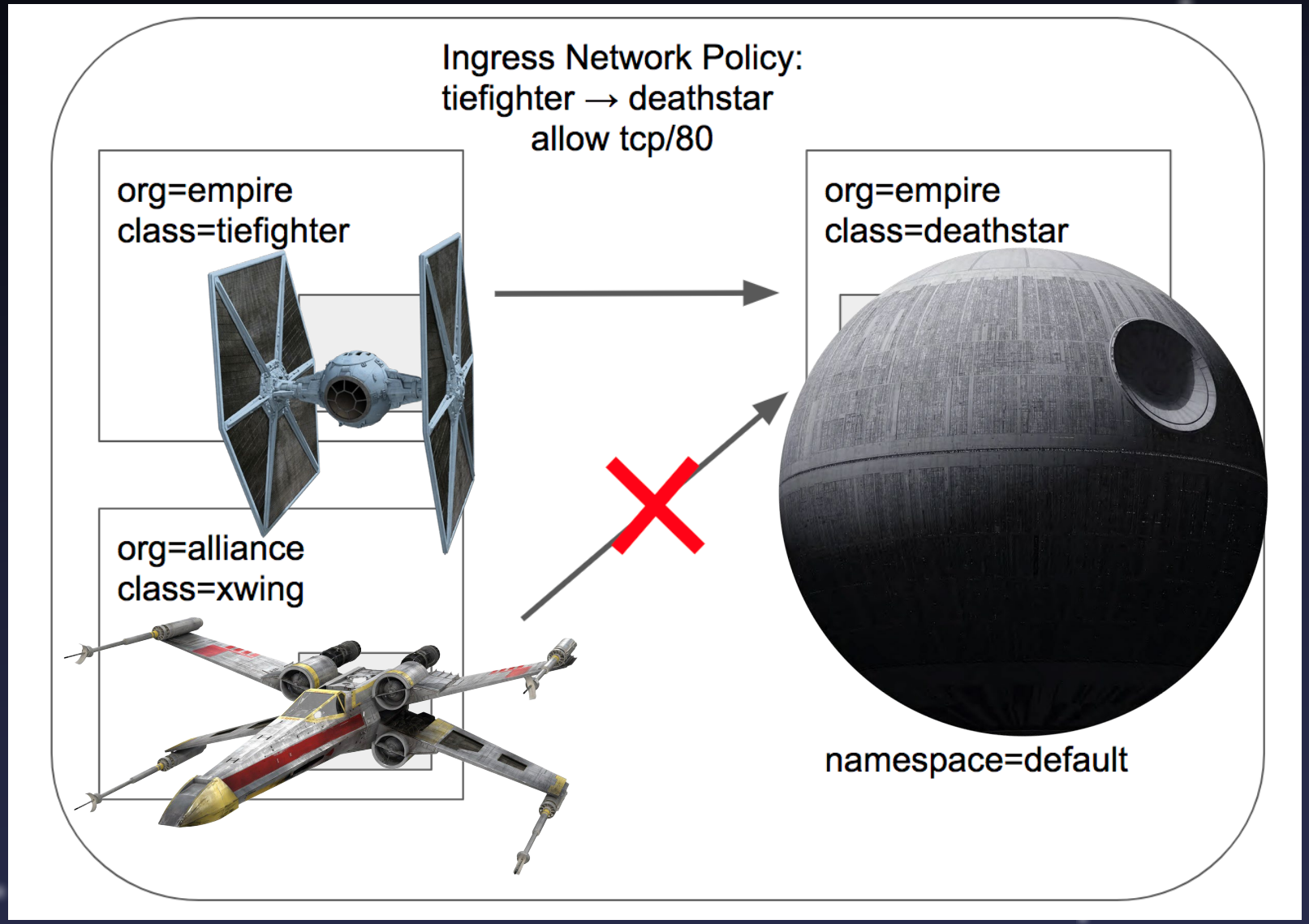


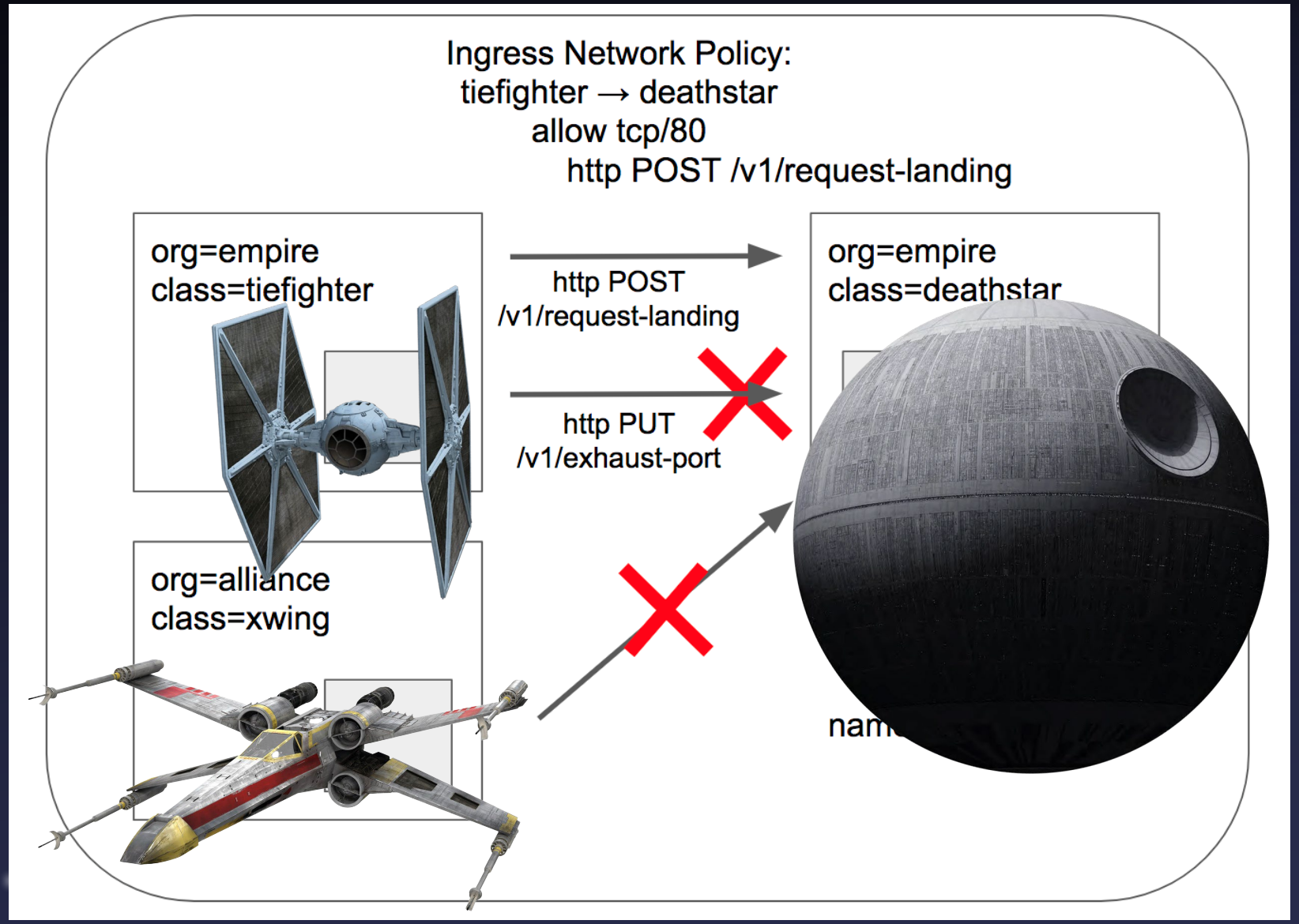
org=alliance
class=xwing



org=empire
class=deathstar

namespace=default


Ingress Network Policy:
tiefighter → deathstar
allow tcp/80
org=empire
class=tiefighter
org=alliance
class=xwing
org=empire
class=deathstar
namespace=default


Ingress Network Policy:
tiefighter → deathstar
allow tcp/80
http POST /v1/request-landing
org=empire
class=tiefighter
org=alliance
class=xwing
http POST
/v1/request-landing
http PUT
/v1/exhaust-port
org=empire
class=deathstar

Выводы

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса

Выводы

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые Network Policy – труд нескольких департаментов

Выводы

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые Network Policy – труд нескольких департаментов
- Разработчик лучше всех (должен) понимает бизнес-логику микросервиса

Выводы

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые Network Policy – труд нескольких департаментов
- Разработчик лучше всех (должен) понимает бизнес-логику микросервиса
- С помощью Network Policy можно очень гибко ограничивать сетевое взаимодействие ваших сервисов

Выводы

- В реальных окружениях сетевое взаимодействие намного сложнее, чем то что лежит в ответственности разработчика микросервиса
- Итоговые Network Policy – труд нескольких департаментов
- Разработчик лучше всех (должен) понимает бизнес-логику микросервиса
- С помощью Network Policy можно очень гибко ограничивать сетевое взаимодействие ваших сервисов
- Микросегментация – важный аспект безопасности и концепции ZeroTrust

Спасибо за внимание!

Telegram: @r0binak

E-mail: sk@luntry.ru

luntry.ru