Нестандартное применение Kyverno

Сергей Канибор

R&D/Container Security, Luntry

whoami

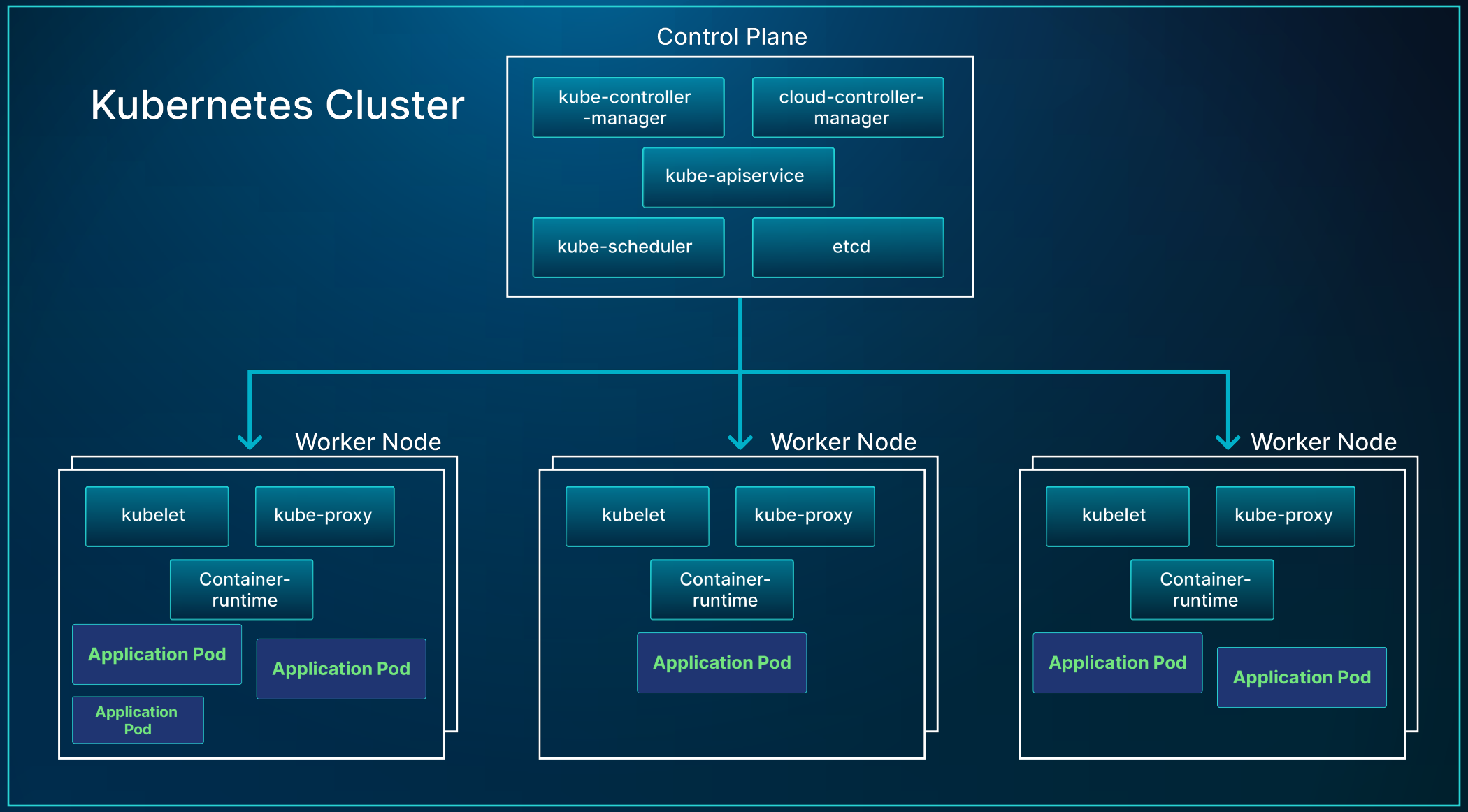
- R&D/Container Security в Luntry
- Специализируюсь на безопасности контейнеров и Kubernetes
- Багхантер
- Редактор Telegram-канала “k8s (in)security”
- Спикер: PHDays, OFFZONE, VK Kubernetes Conf, Devoops, HackConf, CyberCamp, БеКон и др.

Agenda

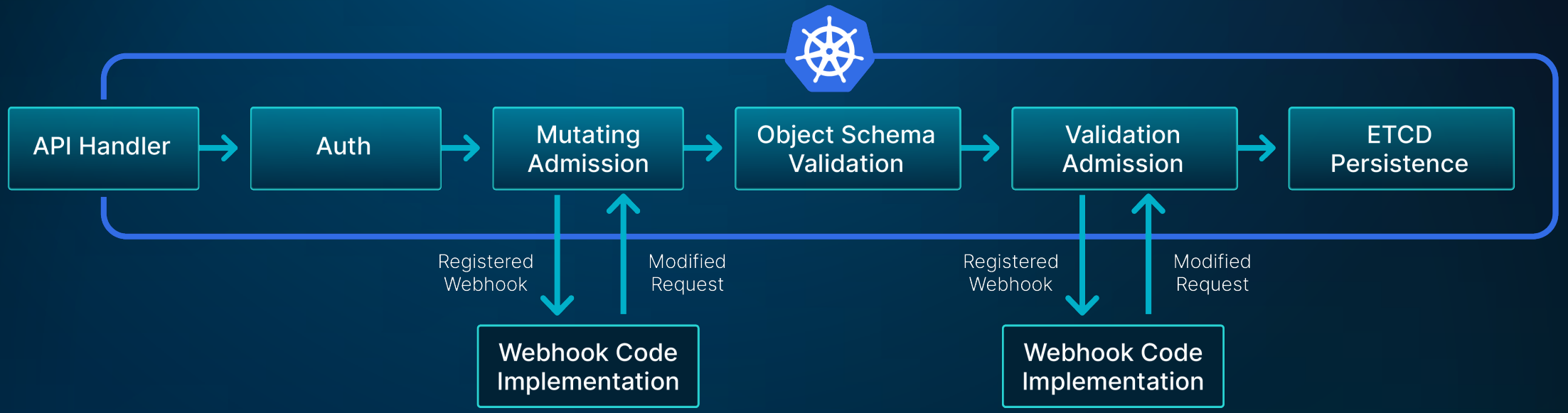
- Kubernetes 101
  - Admission Controller
- Стандартное применение
  - Валидация
  - Мутация
  - Генерация
  - Проверка подписи образов
- Нестандартное использование
  - ???

Kubernetes 101

What is Kubernetes?



Admission Controller Phases



Kubernetes Policy Engines

- Kyverno
- OPA Gatekeeper
- jsPolicy
- Kubewarden
- ...

Open Policy Agent

jsPolicy

KUBEWARDEN

Kyverno

- Декларативный подход в написании и управлении политиками
- Результаты политик хранятся в отдельных CRD ресурсах – PolicyReport и ClusterPolicyReport
- Валидация, мутация и генерация ресурсов
- Поддерживает все Kubernetes resources, включая CRD

Стандартное применение

Валидация

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: priv-exec-pod
  labels:
    app: pentest
spec:
  containers:
  - name: priv-pod
    image: ubuntu
    securityContext:
      privileged: true
    command: [ "/bin/sh", "-c", "--" ]
    args: [ "while true; do sleep 30; done;" ]
```

Валидация – Privileged containers

```yaml
spec:
  validationFailureAction: audit
  background: true
  rules:
    - name: privileged-containers
      match:
        any:
        - resources:
            kinds:
              - Pod
      validate:
        pattern:
          spec:
            =(ephemeralContainers):
              - =(securityContext):
                  =(privileged): "false"
            =(initContainers):
              - =(securityContext):
                  =(privileged): "false"
            containers:
              - =(securityContext):
                  =(privileged): "false"
```

Мутация – Add AppArmor Annotations

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-apparmor
spec:
  containers:
  - name: hello
    image: busybox:1.28
```

Мутация – Add AppArmor Annotations

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: hello-apparmor
  annotations:
    container.apparmor.security.beta.kubernetes.io/hello: runtime/default
spec:
  containers:
  - name: hello
    image: busybox:1.28
```

Мутация – Add AppArmor Annotations

```yaml
spec:
  rules:
  - name: apparmor-runtime-default
    match:
      any:
      - resources:
          kinds:
          - Pod
    preconditions:
      all:
      - key: "{{request.operation || 'BACKGROUND'}}"
        operator: AnyIn
        value:
          - CREATE
          - UPDATE
    mutate:
      foreach:
      - list: request.object.spec.containers[]
        patchStrategicMerge:
          metadata:
            annotations:
              container.apparmor.security.beta.kubernetes.io/{{element.name}}: runtime/default
```

Генерация – Add Network Policy for DNS

```yaml
spec:
  rules:
  - name: add-netpol-dns
    match:
      any:
      - resources:
          kinds:
          - Namespace
    generate:
      apiVersion: networking.k8s.io/v1
      kind: NetworkPolicy
      name: allow-dns
      namespace: "{{request.object.metadata.name}}"
      synchronize: false
      data:
        spec:
          podSelector:
            matchLabels: {}
          policyTypes:
          - Egress
          egress:
          - to:
            - namespaceSelector:
                matchLabels:
                  name: kube-system
            ports:
            - protocol: UDP
              port: 53
```

Проверка подписи образа

```yaml
spec:
  validationFailureAction: enforce
  background: false
  rules:
    - name: verify-image
      match:
        any:
        - resources:
            kinds:
              - Pod
      verifyImages:
      - imageReferences:
        - "ghcr.io/kyverno/test-verify-image*"
        mutateDigest: true
        attestors:
        - entries:
          - keys:
              publicKeys: |
                -----BEGIN PUBLIC KEY-----
                MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE8nXRh950IZbRj8Ra/N9sbqOPZrfM
                5/KAQN0/KjHcorm/J5yctVd7iEcnessRQjU917hmKO6JWVGHpDguIyakZA==
                -----END PUBLIC KEY-----
```

Нестандартное применение

Нестандартное применение

- Github + Trivy + Cosign + Kyverno
- Временные исключения из политик
- Петров не может деплоиться после 18:00
- OTP code
- OTP code с квотами
- External Data Sources
- Kyverno CLI
- Kyverno-json

Github + Trivy + Cosign + Kyverno

Github + Trivy + Cosign + Kyverno

Scan job

- Используем Trivy для сканирования образа

```yaml
scan:
  runs-on: ubuntu-20.04
  permissions:
    contents: read
  outputs:
    scan-digest: ${{ steps.calculate-scan-hash.outputs.scan_digest }}
  steps:
  - name: Scan for vulnerabilities
    uses: aquasecurity/trivy-action@1db49f532692e649dc5dc43c7c0444dac4790137
    with:
      image-ref: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest
      format: cosign-vuln
      ignore-unfixed: true
      output: scan.json

  - name: Calculate scan file hash
    id: calculate-scan-hash
    run: |
      SCAN_DIGEST=$(sha256sum scan.json | awk '{print $1}')
      echo "::set-output name=scan_digest::$SCAN_DIGEST"
      echo "Hash of scan.json is: $SCAN_DIGEST"

  - name: Upload vulnerability scan report
    uses: actions/upload-artifact@3cea5372237819ed00197afe530f5a7ea3e805c8
    with:
      name: scan.json
      path: scan.json
      if-no-files-found: error
```

Github + Trivy + Cosign + Kyverno

Scan job

- Используем Trivy для сканирования образа
- Считаем хэш

```yaml
scan:
  runs-on: ubuntu-20.04
  permissions:
    contents: read
  outputs:
    scan-digest: ${{ steps.calculate-scan-hash.outputs.scan_digest }}
  steps:
  - name: Scan for vulnerabilities
    uses: aquasecurity/trivy-action@1db49f532692e649dc5dc43c7c0444dac4790137
    with:
      image-ref: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest
      format: cosign-vuln
      ignore-unfixed: true
      output: scan.json

  - name: Calculate scan file hash
    id: calculate-scan-hash
    run: |
      SCAN_DIGEST=$(sha256sum scan.json | awk '{print $1}')
      echo "::set-output name=scan_digest::$SCAN_DIGEST"
      echo "Hash of scan.json is: $SCAN_DIGEST"

  - name: Upload vulnerability scan report
    uses: actions/upload-artifact@3cea5372237819ed00197afe530f5a7ea3e805c8
    with:
      name: scan.json
      path: scan.json
      if-no-files-found: error
```

Github + Trivy + Cosign + Kyverno

Scan job

- Используем Trivy для сканирования образа
- Считаем хэш
- Загружаем артефакт после сканирования в наш workflow

```yaml
scan:
  runs-on: ubuntu-20.04
  permissions:
    contents: read
  outputs:
    scan-digest: ${{ steps.calculate-scan-hash.outputs.scan_digest }}
  steps:
  - name: Scan for vulnerabilities
    uses: aquasecurity/trivy-action@1db49f532692e649dc5dc43c7c0444dac4790137
    with:
      image-ref: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest
      format: cosign-vuln
      ignore-unfixed: true
      output: scan.json

  - name: Calculate scan file hash
    id: calculate-scan-hash
    run: |
      SCAN_DIGEST=$(sha256sum scan.json | awk '{print $1}')
      echo "::set-output name=scan_digest::$SCAN_DIGEST"
      echo "Hash of scan.json is: $SCAN_DIGEST"

  - name: Upload vulnerability scan report
    uses: actions/upload-artifact@3cea5372237819ed00197afe530f5a7ea3e805c8
    with:
      name: scan.json
      path: scan.json
      if-no-files-found: error
```

Github + Trivy + Cosign + Kyverno

Attest job

- Скачиваем артефакт из предыдущей джобы о нашем сканировании

```
    - name: Download scan
      uses: actions/download-artifact@fb598a63ae348fa914e94cd0ff38f362e927b741 # v3.0.0
      with:
        name: scan.json
```

Github + Trivy + Cosign + Kyverno

Attest job

- Скачиваем артефакт из предыдущей джобы о нашем сканировании
- Убеждаемся в том, что хэш не изменился

```
- name: Verify scan
  run: |
    set -euo pipefail
    echo "Hash of scan.json should be: $SCAN_DIGEST"
    COMPUTED_HASH=$(sha256sum scan.json | awk '{print $1}')
    echo "The current computed hash for scan.json is: $COMPUTED_HASH"
    echo "If the two above hashes don't match, scan.json has been tampered with."
    echo "$SCAN_DIGEST scan.json" | sha256sum --strict --check --status || exit -2
```

Github + Trivy + Cosign + Kyverno

Attest job

- Скачиваем артефакт из предыдущей джобы о нашем сканировании
- Убеждаемся в том, что хэш не изменился
- Логинимся в GitHub Container Regisrty – именно там мы будем хранить signed attestation

```yaml
    - name: Log in to GHCR
      uses: docker/login-action@49ed152c8eca782a232dede0303416e8f356c37b # v2.0.0
      with:
        registry: ${{ env.REGISTRY }}
        username: ${{ github.actor }}
        password: ${{ secrets.GITHUB_TOKEN }}
```

Github + Trivy + Cosign + Kyverno

Attest job

- Скачиваем артефакт из предыдущей джобы о нашем сканировании
- Убеждаемся в том, что хэш не изменился
- Логинимся в GitHub Container Regisrty – именно там мы будем хранить signed attestation
- Используя keyless signing от Cosign подписываем образ и заменяем подпись, если она уже существует

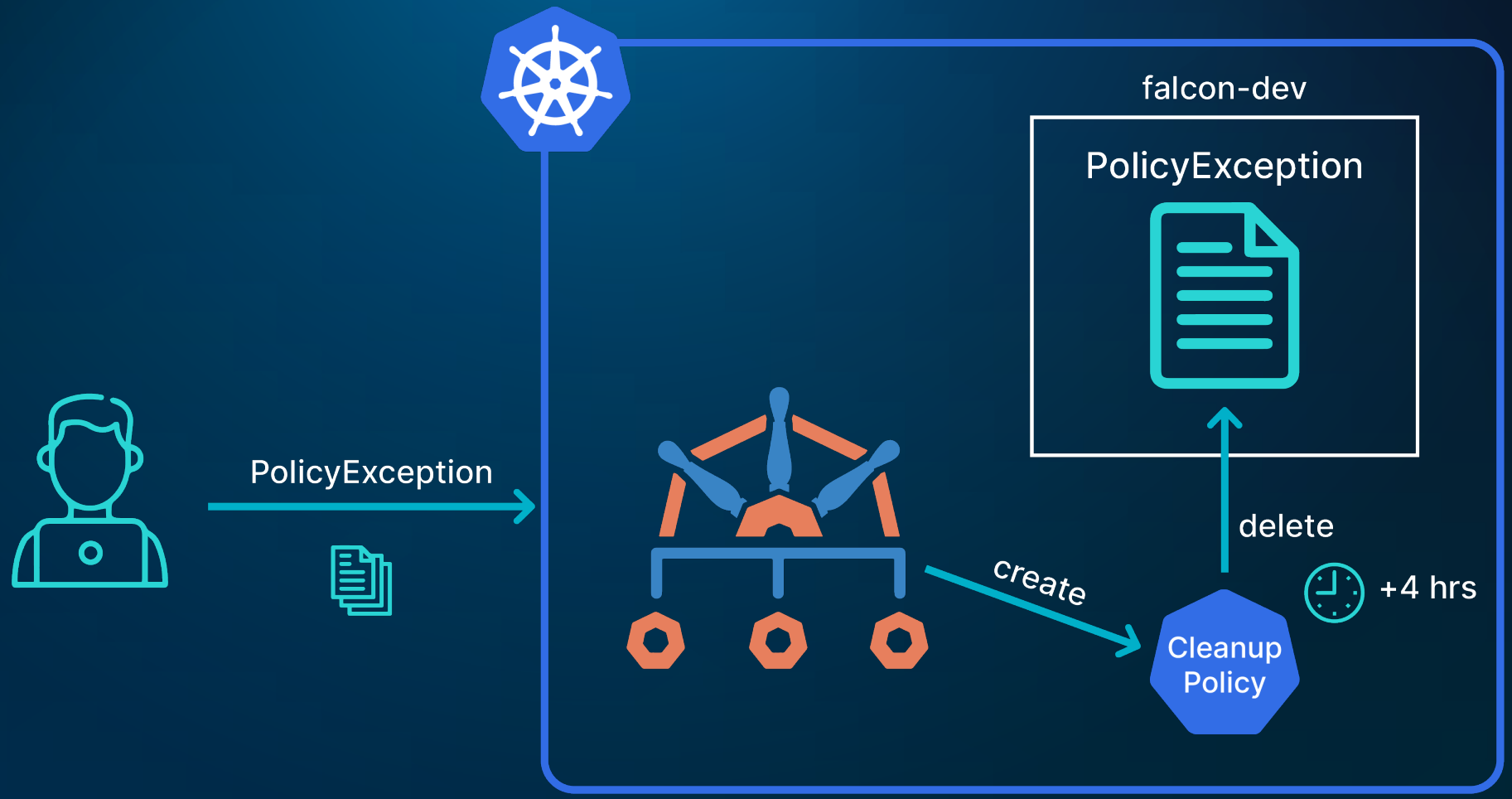
```
    - name: Attest Scan
      run: cosign attest --replace --predicate scan.json --type vuln ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}:latest
      env:
        COSIGN_EXPERIMENTAL: "true"
```

Github + Trivy + Cosign + Kyverno

```
verifyImages:
- imageReferences:
  - "ghcr.io/chipzoller/zulu:*"
  attestors:
  - entries:
    - keyless:
        subject: "https://github.com/chipzoller/zulu/.github/workflows/*"
        issuer: "https://token.actions.githubusercontent.com
```

Временные исключения из политик

Временные исключения из политик



Что для этого нужно?

- Выдать дополнительные привилегии Kyverno

```yaml
# allow a Kyverno generate rule to create ClusterCleanupPolicies
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/instance: kyverno
    app.kubernetes.io/name: kyverno
    app: kyverno
  name: kyverno:create-cleanups
rules:
- apiGroups:
  - kyverno.io
  resources:
  - clustercleanuppolicies
  verbs:
  - create
  - get
  - list
  - update
  - delete
---
# allow the Kyverno cleanup controller to remove PolicyExceptions
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/component: cleanup-controller
    app.kubernetes.io/instance: kyverno
    app.kubernetes.io/name: kyverno-cleanup-controller
  name: kyverno:cleanup-controller-polex
rules:
- apiGroups:
  - kyverno.io
  resources:
  - policyexceptions
  verbs:
  - list
  - delete
```

Что для этого нужно?

- Создать генеративную политику для создания ClusterCleanupPolicy

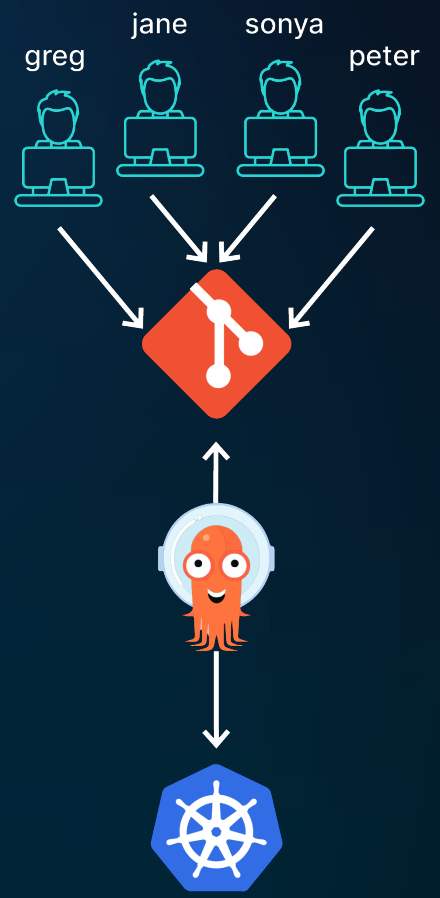
```
generate:
  apiVersion: kyverno.io/v2alpha1
  kind: ClusterCleanupPolicy
  name: polex-{{ request.namespace }}-{{ request.object.metadata.name }}-{{ random('[0-9a-z]{8}') }}
  synchronize: false
  data:
    metadata:
      labels:
        kyverno.io/automated: "true"
    spec:
      schedule: "{{ time_add('{{ time_now_utc() }}','4h') | time_to_cron(@) }}"
      match:
        any:
        - resources:
            kinds:
              - PolicyException
            namespaces:
            - "{{ request.namespace }}"
            names:
            - "{{ request.object.metadata.name }}
```

Петров не может деплоиться после 18:00

Петров не может деплоиться после 18:00

- Создаем мутирующую политику, которая будет добавляет лейбл по автору мерджа

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: add-labels
spec:
  background: false
  rules:
  - name: add-author
    match:
      any:
      - resources:
          kinds:
          - "*"
    mutate:
      patchStrategicMerge:
        metadata:
          labels:
            corp.org/author: "{{request.githubprauthor}}"
```



Петров не может деплоиться после 18:00

- Добавляем джобу в CI

```
    - name: Write author
      run: |
        curl -sLO https://github.com/kyverno/kyverno/releases/download/${{ env.VERSION }}/kyverno-cli_${{ env.VERSION
}}_linux_x86_64.tar.gz
        tar -xf kyverno-cli_${{ env.VERSION }}_linux_x86_64.tar.gz
        ./kyverno version
        for f in $(ls ./incoming)
        do
        if [[ "$f" = *\.yaml ]]
        then
            echo "Adding authorship to incoming/$f"
            ./kyverno apply author.yaml -r incoming/$f --set request.githubprauthor=${{github.event.pull_request.user.login}} -o
outgoing/temp.yaml
            sed '/^[[:space:]]*$/d' outgoing/temp.yaml > outgoing/$f
            rm incoming/$f
            rm outgoing/temp.yaml
        fi
        done
```

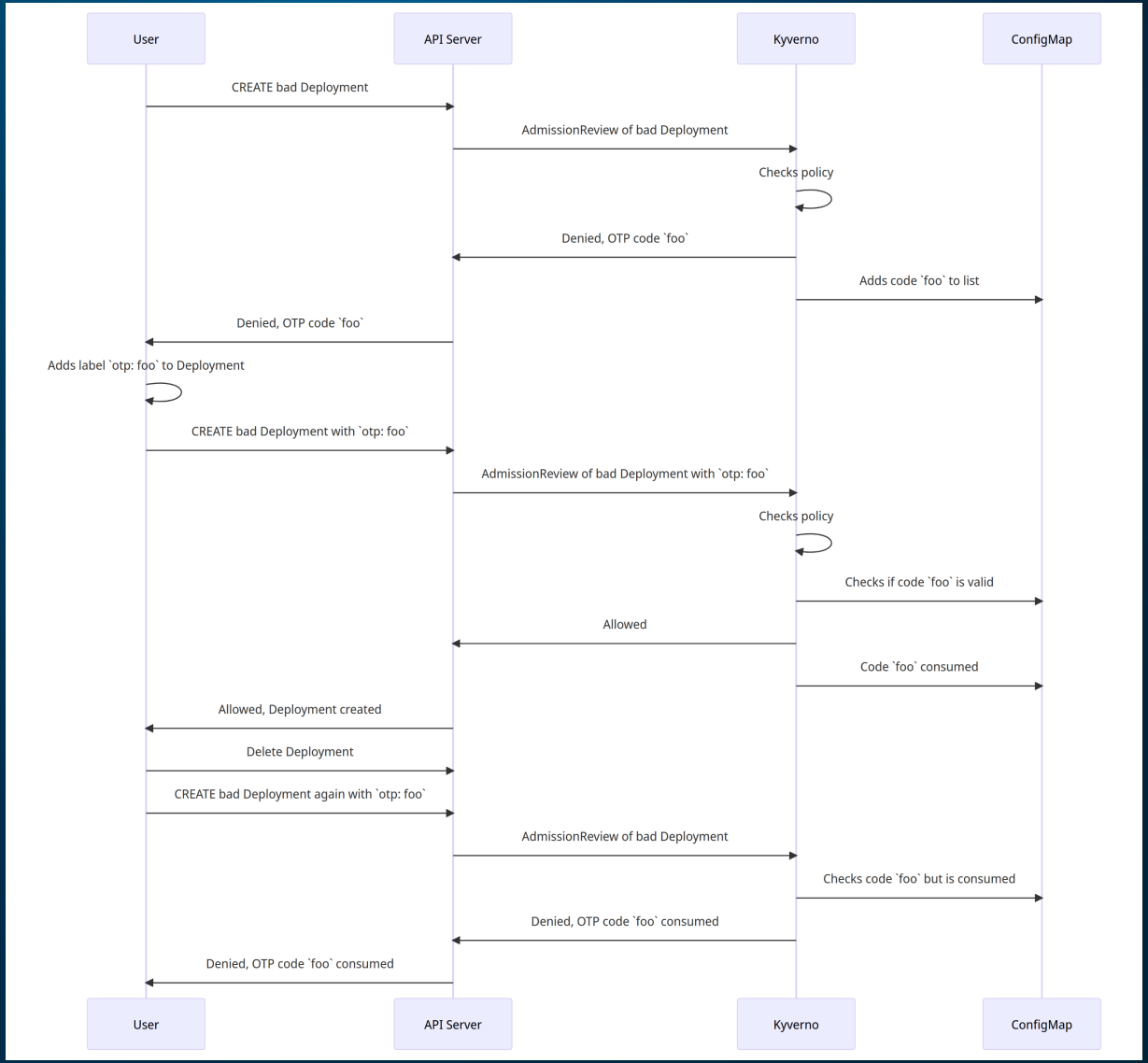
Петров не может деплоиться после 18:00

- Создаем валидирующую политику, которая будет привязываться по лейблам и ограничивать деплой по времени

```
match:
  any:
  - resources:
      selector:
        matchLabels:
          corp.org/author: petrov
 validate:
    message: "Petrov pls stop"
    deny:
      conditions:
        all:
        - key: "{{ time_after('{{time_now_utc() }}','2023-01-12T00:00:00Z') }}"
          operator: Equals
          value: true
```

OTP code

OTP code


User
API Server
Kyverno
ConfigMap
CREATE bad Deployment
AdmissionReview of bad Deployment
Checks policy
Denied, OTP code `foo`
Adds code `foo` to list
Denied, OTP code `foo`
Adds label `otp: foo` to Deployment
CREATE bad Deployment with `otp: foo`
AdmissionReview of bad Deployment with `otp: foo`
Checks policy
Checks if code `foo` is valid
Allowed
Code `foo` consumed
Allowed, Deployment created
Delete Deployment
CREATE bad Deployment again with `otp: foo`
AdmissionReview of bad Deployment
Checks code `foo` but is consumed
Denied, OTP code `foo` consumed
Denied, OTP code `foo` consumed
User
API Server
Kyverno
ConfigMap

OTP code

- Создаем ConfigMap, он будет выступать журналом с OTP кодами

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: otp
  namespace: platform
data:
  codes: |-
    - ua8v92pg
    - 9akvm2o7
```

OTP code

- Создаем политику, которая будет проверять, использован ли OTP код, а также интересующую нас валидацию, например – проверка использования Host Namespaces

```
    - name: otp
      configMap:
        name: otp
        namespace: platform
    preconditions:
      all:
      - key: "{{ request.object.metadata.labels.otp }}"
        operator: AnyNotIn
        value: "{{ parse_yaml(otp.data.codes) }}"
    validate:
      message: The code {{ request.object.metadata.labels.otp }} is invalid or has already been used
      deny: {}
```

OTP code

- Создаем ещё одну политику, которая будет проверять, валидное ли значение OTP используется в Deployment, если да, то изменяем ConfigMap, проставляем timestamp и username пользователя, который его использовал

```yaml
preconditions:
  all:
  - key: "{{ request.object.reason }}"
    operator: Equals
    value: PolicyViolation
  - key: "{{ contains(request.object.message, 'one-time pass code') }}"
    operator: Equals
    value: true
context:
- name: otp
  variable:
    jmesPath: split(request.object.message,'"') | [1]
mutate:
  targets:
    - apiVersion: v1
      kind: ConfigMap
      name: otp
      namespace: platform
  patchStrategicMerge:
    data:
      codes: |-
        {{ @ }}
        - {{ otp }}
```

OTP code

```
  - name: manage-otp
    match:
      any:
      - resources:
          kinds:
            - Deployment
          operations:
            - CREATE
          selector:
            matchLabels:
              otp: "?*"
    context:
    - name: otp
      configMap:
        name: otp
        namespace: platform
    preconditions:
      all:
      - key: "{{ request.object.metadata.labels.otp }}"
        operator: AnyIn
        value: "{{ parse_yaml(otp.data.codes) }}"
    mutate:
      targets:
        - apiVersion: v1
          kind: ConfigMap
          name: otp
          namespace: platform
          context:
          - name: used
            variable:
              jmesPath:
replace_all(target.data.codes,'{{request.object.metadata.labels.otp}}','{{request.object.metadata.labels.otp}}-{{time_now_utc()}}-
{{request.userInfo.username}}')
        patchStrategicMerge:
          data:
            codes: |-
                          {{ used }}
```

OTP code с квотами

OTP code с квотами

- В предыдущей версии пользователь мог неограниченно запрашивать коды
- В этой версии добавлены квоты на выдачу кодов, которые можно выдавать на определенный период, например на месяц

```yaml
mutate:
  targets:
    - apiVersion: v1
      kind: ConfigMap
      name: otp
      namespace: platform
  patchStrategicMerge:
    data:
      chip: "5"
      mark: "2"
```

External Data Sources

External Data Source

- Помимо самого ресурса, над которым сработает политика, можно использовать внешние источники данных
- ConfigMaps
- Kubernetes API Server Calls
- Service Calls
- Image Registries

External Data Source: ConfigMaps

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: some-config-map
  namespace: some-namespace
data:
  env: production
```

External Data Source: ConfigMaps

```yaml
context:
- name: dictionary
  configMap:
    name: some-config-map
    namespace: some-namespace
mutate:
  patchStrategicMerge:
    metadata:
      labels:
        my-environment-name: "{{dictionary.data.env}}
```

External Data Source: Kubernetes API Server Call

```yaml
context:
- name: serviceCount
  apiCall:
    urlPath: "/api/v1/namespaces/{{ request.namespace }}/services"
    jmesPath: "items[?spec.type == 'LoadBalancer'] | length(@)"
validate:
  message: "Only one LoadBalancer service is allowed per namespace"
  deny:
    conditions:
      any:
      - key: "{{ serviceCount }}"
        operator: GreaterThan
        value: 1
```

External Data Source: Service Calls

```yaml
context:
- name: result
  apiCall:
    method: POST
    data:
    - key: namespace
      value: "{{request.namespace}}"
    service:
      url: http://sample.kyverno-extension/check-namespace
      caBundle: |-
        -----BEGIN CERTIFICATE-----
        <snip>
        -----END CERTIFICATE-----
validate:
  message: "namespace {{request.namespace}} is not allowed"
  deny:
    conditions:
      all:
      - key: "{{ result.allowed }}"
        operator: Equals
        value: false
```

External Data Source: Image Registries

```
validate:
  message: "Images run as root are not allowed."
  foreach:
  - list: "request.object.spec.containers"
    context:
    - name: imageData
      imageRegistry:
        reference: "{{ element.image }}"
    deny:
      conditions:
        any:
          - key: "{{ imageData.configData.config.User || ''}}"
            operator: Equals
            value: ""
```

Kyverno CLI

Kyverno CLI

- Можно использовать в CI/CD пайплайнах в процессе создания ресурсов, чтобы убедиться в их соответствии политикам до развертывания

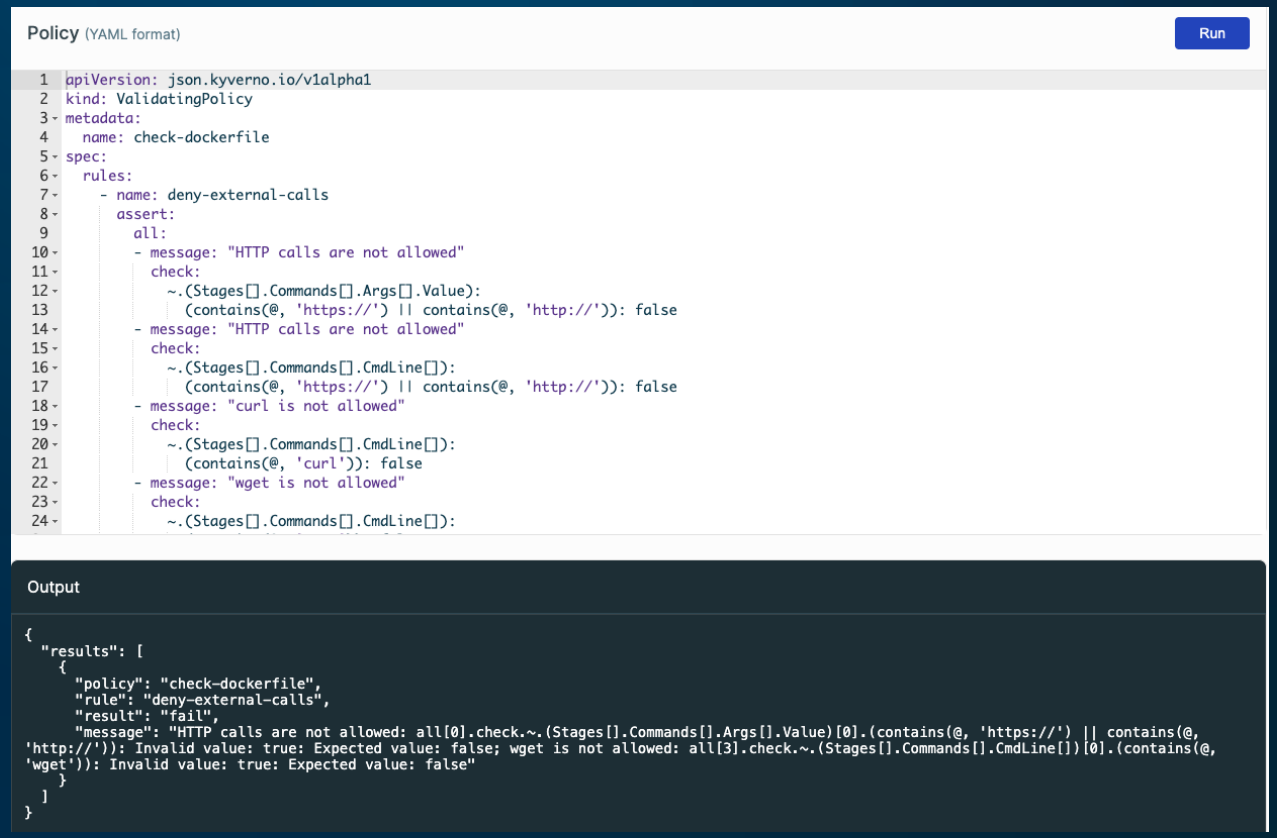
```yaml
apiVersion:
cli.kyverno.io/v1alpha1
kind: Test
metadata:
  name: disallow_latest_tag
policies:
  - disallow_latest_tag.yaml
resources:
  - resource.yaml
results:
  - policy: disallow-latest-tag
    rule: require-image-tag
    resource: myapp-pod
    kind: Pod
    result: pass
  - policy: disallow-latest-tag
    rule: validate-image-tag
    resource: myapp-pod
    kind: Pod
    result: pass
```

```
$ kyverno test .

Executing disallow_latest_tag...
applying 1 policy to 1 resource...

|---|-----------------------|-----------------------|---------------------------|--------|
| # | POLICY                | RULE                  | RESOURCE                  | RESULT |
|---|-----------------------|-----------------------|---------------------------|--------|
| 1 | disallow-latest-tag   | require-image-tag     | default/Pod/myapp-pod     | Pass   |
| 2 | disallow-latest-tag   | validate-image-tag    | default/Pod/myapp-pod     | Pass   |
|---|-----------------------|-----------------------|---------------------------|--------|

Test Summary: 2 tests passed and 0 tests failed
```

Kyverno-json

kyverno-json

- ~~Kyverno только для Kubernetes~~
- Можно использовать как CLI, web-приложение, а также интегрировать к себе, используя Go библиотеку
- Умеет валидировать Terraform files, Dockerfiles, Cloud configurations, Authorization requests



Полезные ссылки

- https://neonmirrors.net/
- https://kyverno.io/docs/
- https://github.com/kyverno/kyverno-json
- https://playground.kyverno.io/
- https://kyverno.github.io/kyverno-json/latest/_playground/

Выводы

1. Kyverno – must-have для вашего Kubernetes кластера
2. Это мощный инструмент, который способен покрыть огромное количество ситуаций
3. Не стоит изобретать велосипеды

Спасибо!

Сергей Канибор
R&D / Container Security

Email: sk@luntry.ru
Channel: @k8security
Site: www.luntry.ru

k8security luntrysolution