2 ИЮНЯ 2026, МОСКВА, ЛОФТ ГОЭЛРО

БЕКОН LUNTRY '26

ЕДИНСТВЕННАЯ КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Validating Admission Policy замена Policy Engine?

Александр Алёхин | ecom.tech

БЕКОН'26 ecom.tech

Validating Admission Policy

Александр Алёхин

TechLead

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

whoami & kubernetes в ECOM

БЕКОН'26

Техлид продукта эксплуатации платформы k8s

01

Управляю k8s там, где он работает одинаково, а облака — нет

02

VAP — мой основной инструмент для контура базовой валидации

03

kubernetes в ECOM:

30+
32
кластера;

11
локаций;

5
активов организаций;

- OLTP
- DATA
- ML
- INFRA
- SANDBOX

- Bare metal
- Virtual
- Cloud

План

БЕКОН'26

Проблематика 01

Почему CEL и откуда он взялся 02

Структура и ключевые фичи VAP 03

Ограничения 04

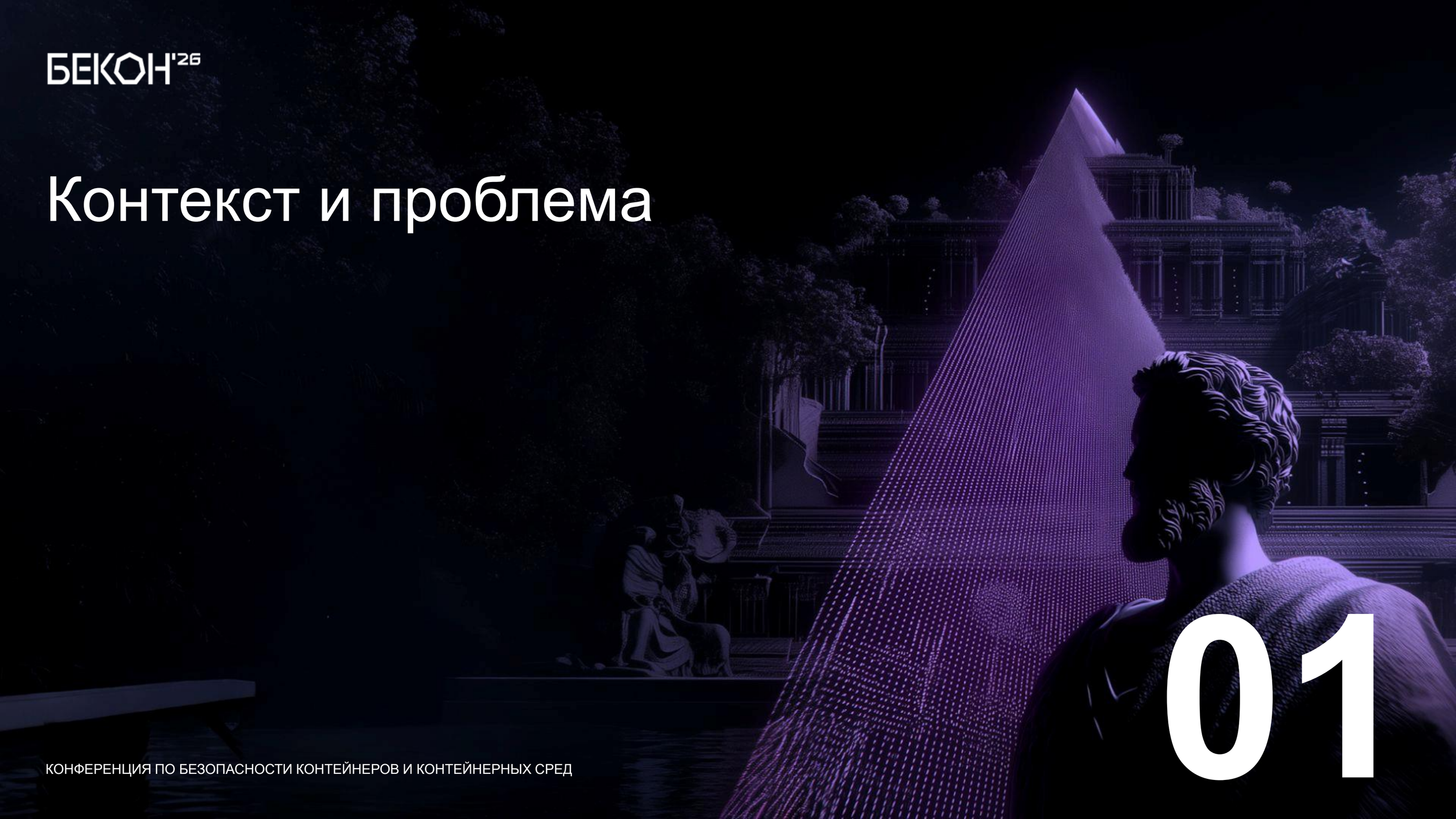
Миграция 05

Эксплуатация и наблюдаемость 06

Итоги 07

БЕКОН'26

Контекст и проблема



01

Потребности: кто и зачем пишет политики

БЕКОН'26

Простые

DevSecOps

require-run-as-nonroot
disallow-privileged-containers
require-ro-rootfs

01

DevOps & SRE

require-pod-requests-limits
disallow-latest-tag
require-deployments-have-multiple-replicas

02

Containers

restrict-node-port
pdb-maxunavailable
restrict-storageclass

03

Посложнее

DevSecOps

- Проверки подписей образов (airgap)

01

DevOps & SRE

- unique-ingress-host-and-path
  require-unique-external-dns

02

Containers

- restrict-pod-count

03

Почему kyverno перестал справляться

БЕКОН'26

Kyverno — отличный инструмент. Но его эксплуатация нетривиальна.

Кейс	Последствия
Включили background-скан для сбора исключений на data-кластере	Заафектили кластер k8s
Генерируется большое количество отчётов, отчёты не удаляются	Раздувает etcd
Переносили kyverno в другой namespace	Заблокировали всё =)
Создали мутацию с походом в внешнюю систему	Поды создаются очень долго
Захотели обновиться с v2 на v3	Обновление без простоя не предусмотрено =)

QR с ссылкой на прошлогодний доклад про рецепты правильного приготовления kyverno в конце презентации

БЕКОН'26

Озвученные проблемы характерны для webhook-подхода

80–90% политик - это простейшая валидация.

VAP* vs webhooks

БЕКОН'26

Критерий	VAP	Вебхуки (Kyverno / OPA)
Где исполняется	Внутри API-сервера	Отдельные поды
Дополнительная инфра	✗ Не нужна	✓ Нужна
Простая валидация	✓ Идеально	! Избыточно
Сложная логика	✗ Не подходит	! Технически подходит
Мутации	✓ Есть (1.36, MAP*)	✓ Есть
Надежность	✓ Высокая	! Ниже чем у VAP
Гранулярные исключения	✓ Есть	✓ Есть
Скан существующих ресурсов	✗ Нет	✓ Есть
Отчёты	✓ Есть	✓ Есть

*MAP - Mutating Admission Policy

Сравнение ValidatingAdmissionPolicy и ValidatingPolicy

БЕКОН'26

Feature	ValidatingAdmissionPolicy	ValidatingPolicy
Enforcement	Admission	Admission, Background, Pipelines, ...
Payloads	Kubernetes	Kubernetes, Any JSON or YAML
Distribution	Kubernetes	Helm, CLI, Web Service, API, SDK
CEL Library	Basic	Extended
Bindings	Manual	Automatic
Auto-generation	-	Pod Controllers, ValidatingAdmissionPolicy
External Data	_	Kubernetes resources or API calls
Caching	_	Global Context, image verification results
Background scans	-	Periodic, On policy creation or change
Exceptions	-	Fine-grained exceptions
Reporting	_	Policy WG Reports API
Testing	_	Kyverno CLI (unit), Chainsaw (e2e)

БЕКОН'26

VAP берем для простой валидации
kyverno оставляем для сложных кейсов
Функциональность VAP используем напрямую

БЕКОН'26

CEL

02

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Common Expression Language (CEL)

БЕКОН'26

Простой синтаксис

01

Быстрый
Нано/микросекунды
Компиляция один раз

02

Встраиваемый

03

Неполный по Тьюрингу = нет циклов

04

Пример про безопасность

Проверяем, что URL безопасный

```
| isURL(self) && url(self).getScheme() == 'https'
```

05

Как CEL появился в kubernetes

БЕКОН'26

KEP #2876

CRD Validation 01

Август 2022

1.25 (beta)

VAP 02

Август 2023

1.28 (beta)

CRD Validation 03

Декабрь 2023

1.29 (GA)

VAP 04

Апрель 2024

1.30 (GA)

Что дали CRD Validation rules?

БЕКОН'26

CEL-выражения
прямо в схеме CRD

01

Проверка каждого объекта CRD
при создании / обновлении

02

Без вебхуков, без
дополнительной инфраструктуры

03

```
openAPIV3Schema:
      type: object
      properties:
        spec:
          type: object
          x-kubernetes-validations:
            - rule: "self.minReplicas <= self.replicas && self.replicas
<= self.maxReplicas"
              message: "replicas should be in the range
minReplicas..maxReplicas."
          properties:
            replicas:
              type: integer
---
openAPIV3Schema:
  properties:
    value:
      maxLength: 512
      type: string
      x-kubernetes-validations:
      - message: Value is immutable
        rule: self == oldSelf
  required:
  - value
  type: object
```

Что это дало для VAP?

БЕКОН'26

“This KEP builds on the capabilities of the CRD Validation Rules”

01	02	03
Не изобретён велосипед	Переиспользован CRD Validation Rules	Механизм расширен на admission control (парамерты, авторизационные проверки)

БЕКОН'26

“Webhooks are operationally burdensome for cluster administrators to manage” (c) KEP-3488

KEP определяет четыре ключевые проблемы вебхуков

БЕКОН'26

Инфраструктура:

отдельный сервис,
сертификаты, ресурсы

01

Задержки:

каждое обращение = сетевой хоп

02

fail closed vs fail open:

или безопасность,
или отказоустойчивость

03

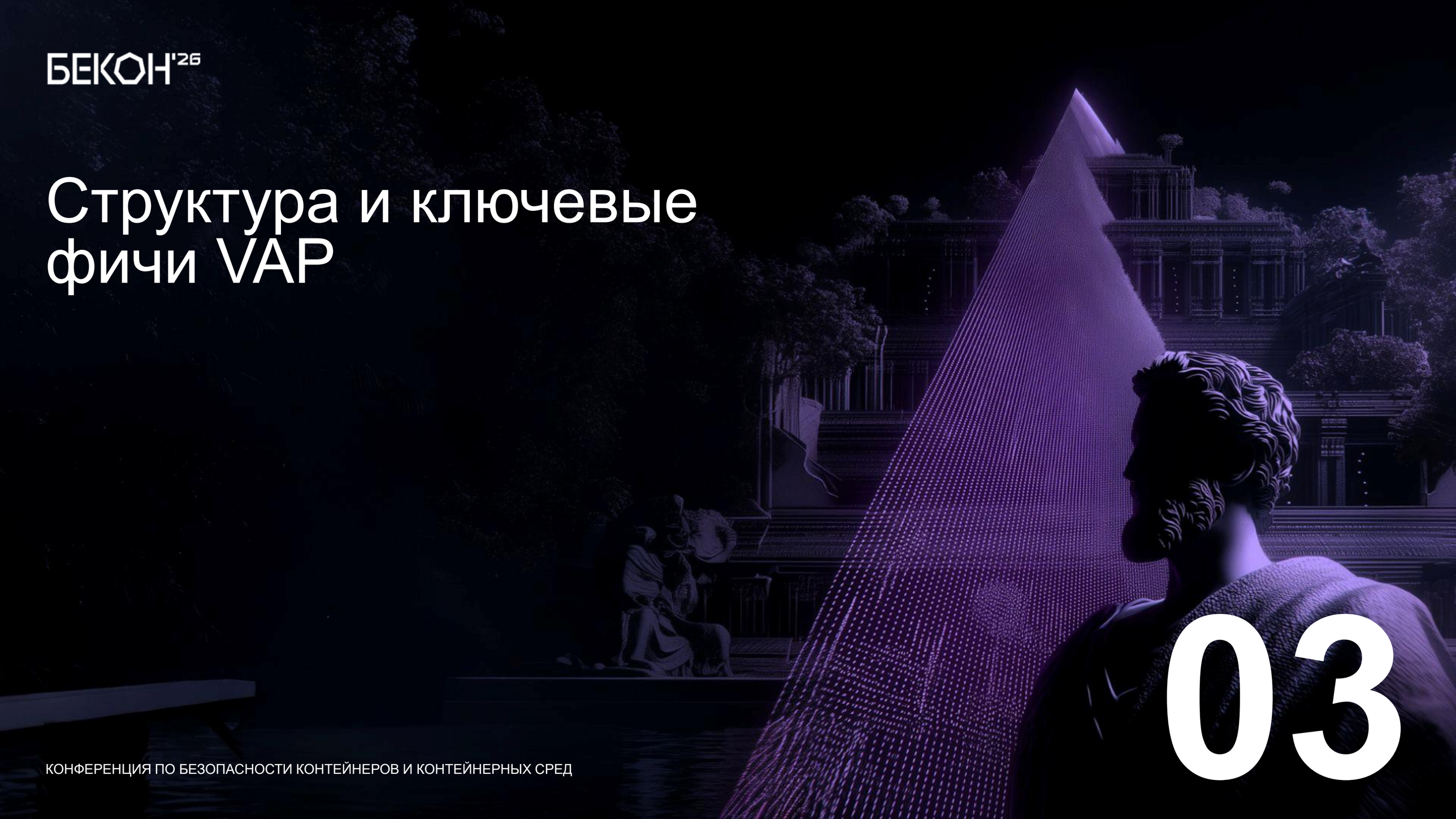
Операционная нагрузка:

обновления, наблюдаемость,
производительность

04

БЕКОН'26

Структура и ключевые фичи VAP



03

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Принцип разделения ответственности в VAP

БЕКОН'26

ValidatingAdmissionPolicy	ValidatingAdmissionPolicyBinding	Parameter Resource (CRD)
Кто пишет правила (заказчик)	Кто включает (администратор кластера)	Кто настраивает (владелец сервиса)

ValidatingAdmissionPolicy — «что проверяем?»

БЕКОН'26

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: capabilities
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: [""]
        apiVersions: ["v1"]
        operations: ["CREATE","UPDATE"]
        resources: ["pods","pods/ephemeralcontainers”]
        scope: '*'
  variables:
    - expression: "object.spec.containers + object.spec.?initContainers.orValue([]) + object.spec.?ephemeralContainers.orValue([])"
      name: allContainers
  validations:
    - expression:
        variables.allContainers.all(container, container.?securityContext.?capabilities.?drop.orValue([]).exists(capability, capability.upperAscii()
== 'ALL'))
      message: "Containers must drop `ALL` capabilities."
```

ValidatingAdmissionPolicyBinding — «где и как проверяем?»

БЕКОН'26

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: capabilities
spec:
  validationActions: [Audit, Deny]
  policyName: capabilities
  matchResources:
    namespaceSelector:
      matchExpressions:
        - key: "vap.example.com/capabilities"
          operator: NotIn
          values: ["excluded"]
```

policyName

(ссылка на политику)

01

paramRef

(где брать данные)

02

matchResources

(какие ресурсы попадают под политику)

03

validationActions

(Deny / Warn / Audit)

04

ValidatingAdmissionPolicyBinding - «где и как проверяем?»

БЕКОН'26

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: capabilities
spec:
  validationActions: [Audit, Deny]
  policyName: capabilities
  matchResources:
    namespaceSelector:
      matchExpressions:
        - key: kubernetes.io/metadata.name
          operator: NotIn
          values:
            - kube-system
```

Если нужно более жёстко контролировать список исключений

Параметризация

БЕКОН'26

Policy

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: capabilities
spec:
  paramKind:
    apiVersion: vap.example.com/v1
    kind: Parameter
  variables:
    - name: "globallyAllowedCapabilities"
      expression: "[
        'NET_BIND_SERVICE',
      ]"
  validations:
    - expression: >
        object.spec.containers.all(c,
c.?securityContext.capabilities.add.orValue([]).all(cap,
            cap in variables.globallyAllowedCapabilities ||
            cap in params.allowedCapabilities
          )
        )
```

Binding

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: capabilities
spec:
  validationActions: [Deny]
  policyName: capabilities
  paramRef:
    name: params
    parameterNotFoundAction: Allow
  matchResources:
    namespaceSelector:
      matchExpressions:
        - key: "vap.example.com/capabilities"
          operator: NotIn
          values: ["excluded"]
```

Parameter CRD

```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: parameters.vap.example.com
spec:
  group: vap.example.com
  names:
    kind: Parameter
    listKind: ParameterList
    plural: parameters
    singular: parameter
  scope: Namespaced
```

Parameter Resource

```
apiversion: vap.example.com/v1
kind: Parameter
allowedCapabilities:
  - "NET_ADMIN"
```

Variable Composition

БЕКОН'26

```
spec:
  variables:
    - name: hasVolumes
      expression: "!has(object.spec.volumes)"
    - name: volumes
      expression: "object.spec.volumes"
    - name: volumesWithHostPath
      expression: "variables.volumes.filter(volume,
has(volume.hostPath))"
  validations:
    - expression: >-
        variables.hasVolumes ||
        variables.volumesWithHostPath.all(volume,
!volume.hostPath.path.matches('/var/run/docker.sock'))
    - expression: >-
        variables.hasVolumes ||
        variables.volumesWithHostPath.all(volume,
!volume.hostPath.path.matches('/var/run/containerd/containerd.s
ock'))
    - expression: >-
        variables.hasVolumes ||
        variables.volumesWithHostPath.all(volume,
!volume.hostPath.path.matches('/var/run/crio/crio.sock'))
    - expression: >-
        variables.hasVolumes ||
        variables.volumesWithHostPath.all(volume,
!volume.hostPath.path.matches('/var/run/cri-dockerd.sock'))
```

Variable Composition — переиспользование CEL-выражений. Особенности:

ленивое вычисление; 01

запоминаются результаты; 02

можно ссылаться только на ранее объявленные переменные. 03

matchConditions: фильтрация до валидации

БЕКОН'26

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: example1
spec:
  matchConditions:
    - expression: (object.metadata.name).find
("service1|service2") == ""
      name: 'exempt-by-name'

---
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: example2
spec:
  matchConditions
    - name: 'exempt-by-namespace-and-name'
      expression: |
        (
          {
            "namespace1": ["name1", "name11"],
            "namespace2": ["name2", "name22"]
          }[object.metadata.namespace]
        ).exists(prefix, object.metadata.name.startsWith(prefix))
```

Как это работает:

все условия matchConditions должны вернуть true, чтобы перейти к валидации; 01

если false или ошибка — политика пропускается; 02

если true — переходим к валидации (основные проверки); 03

если ошибка CEL — срабатывает failurePolicy; 04

Авторизационные проверки

БЕКОН'26

```
validations:
  # отсекаем имперсонацию - все запросы должны содержать имя при использовании sa
  - expression: "has(request.userInfo.extra) && ('authentication.kubernetes.io/node-name' inrequest.userInfo.extra)"
    message: "this user must have a \"authentication.kubernetes.io/node-name\" claim"
  # все запросы должны отправляться с ноды, где работает под
  - expression: "object.metadata.name == request.userInfo.extra[\"authentication.kubernetes.io/node-name\"][0]"
    messageExpression: "'updates to Node ' + string(object.metadata.name) + ' may only be effected from the pod running on the same node'"
```

Другие примеры:

```
authorizer.group('').
resource('pods').
namespace('default').
check('create').allowed()
```

01

```
authorizer.path
('healthz').check('get').
allowed()
```

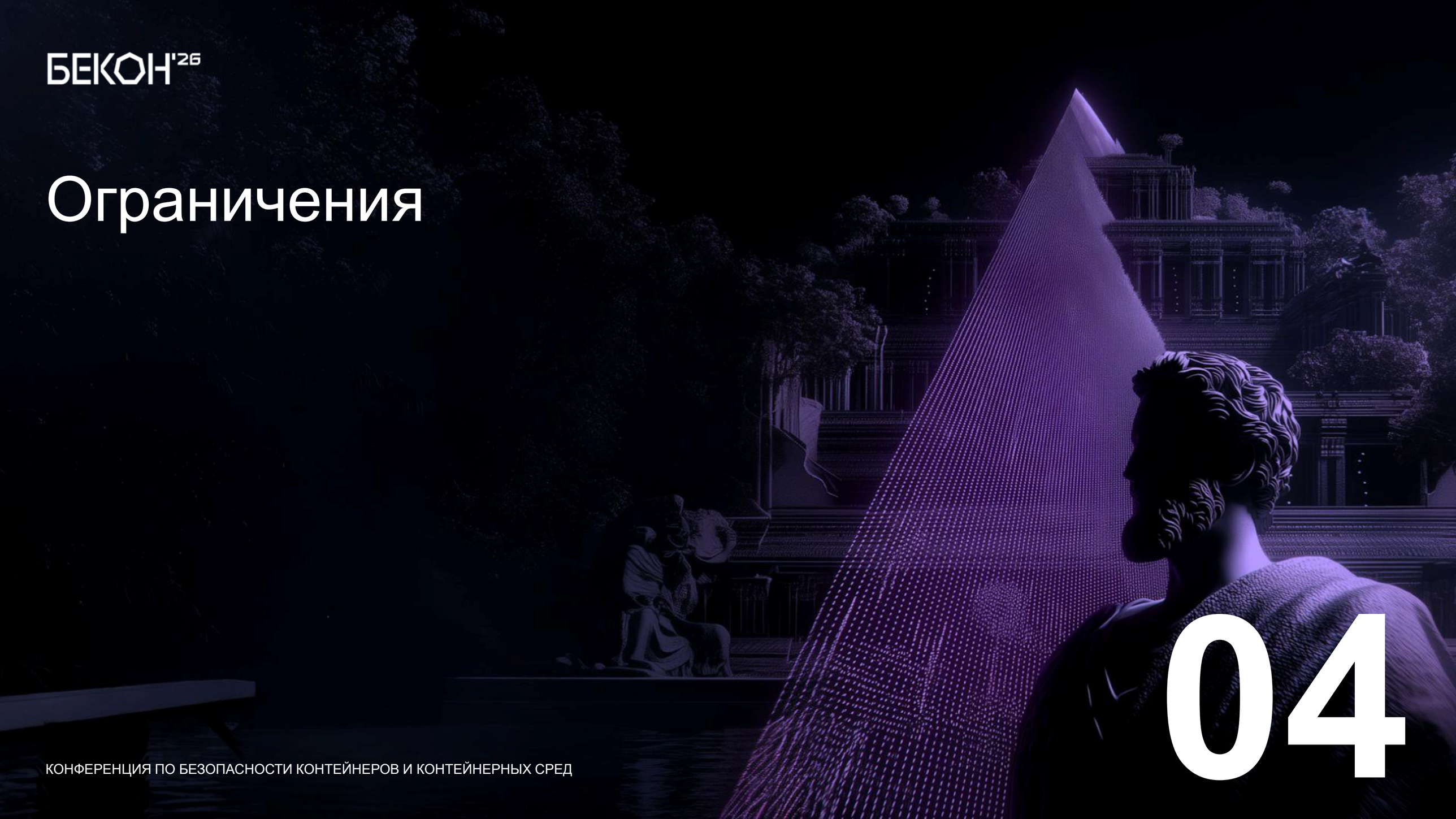
02

```
authorizer.service
Account('default',
'myserviceaccount').
resource('deployments').
check('delete').allowed()
```

03

БЕКОН'26

Ограничения



04

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Бюджеты CEL: защита от DoS

БЕКОН'26

Два уровня защиты kubernetes

При применении объекта (estimated cost):

Если в худшем случае расчетное время выполнения окажется непомерно высоким, API-сервер предотвратит запись соответствующего выражения

При выполнении (runtime cost):

на каждое правило (percall)	10 000 000 (~1 секунда)
на всю политику целиком	100 000 000 (~10 секунд)

При превышении failurePolicy определяет результат

Проверка image spec через регулярку

БЕКОН'26

```
self.matches('^(((?:[a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9-]*[a-zA-Z0-9])(?:\\.(?:[a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9-]*[a-zA-Z0-9]))+(?::[0-9]+)?)|(localhost(?::[0-9]+)?))(?:(?:/[a-z0-9]+(?:(?:(?:[._]|__|[-]*)[a-z0-9]+)+)?)+)?$|^(((?:[a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9-]*[a-zA-Z0-9])(?:\\.(?:[a-zA-Z0-9]|[a-zA-Z0-9][a-zA-Z0-9-]*[a-zA-Z0-9]))+(?::[0-9]+)?)|(localhost(?::[0-9]+)?))(?:(?:/[a-z0-9]+(?:(?:(?:[._]|__|[-]*)[a-z0-9]+)+)?)+)?$')
```

```
spec.validation.openAPIV3Schema.properties[spec].properties[scopes].items.x-kubernetes-validations[0].rule: Forbidden: estimated rule cost exceeds budget by factor of more than 100x (try simplifying the rule, or adding maxItems, maxProperties, and maxLength where arrays, maps, and strings are declared)
```

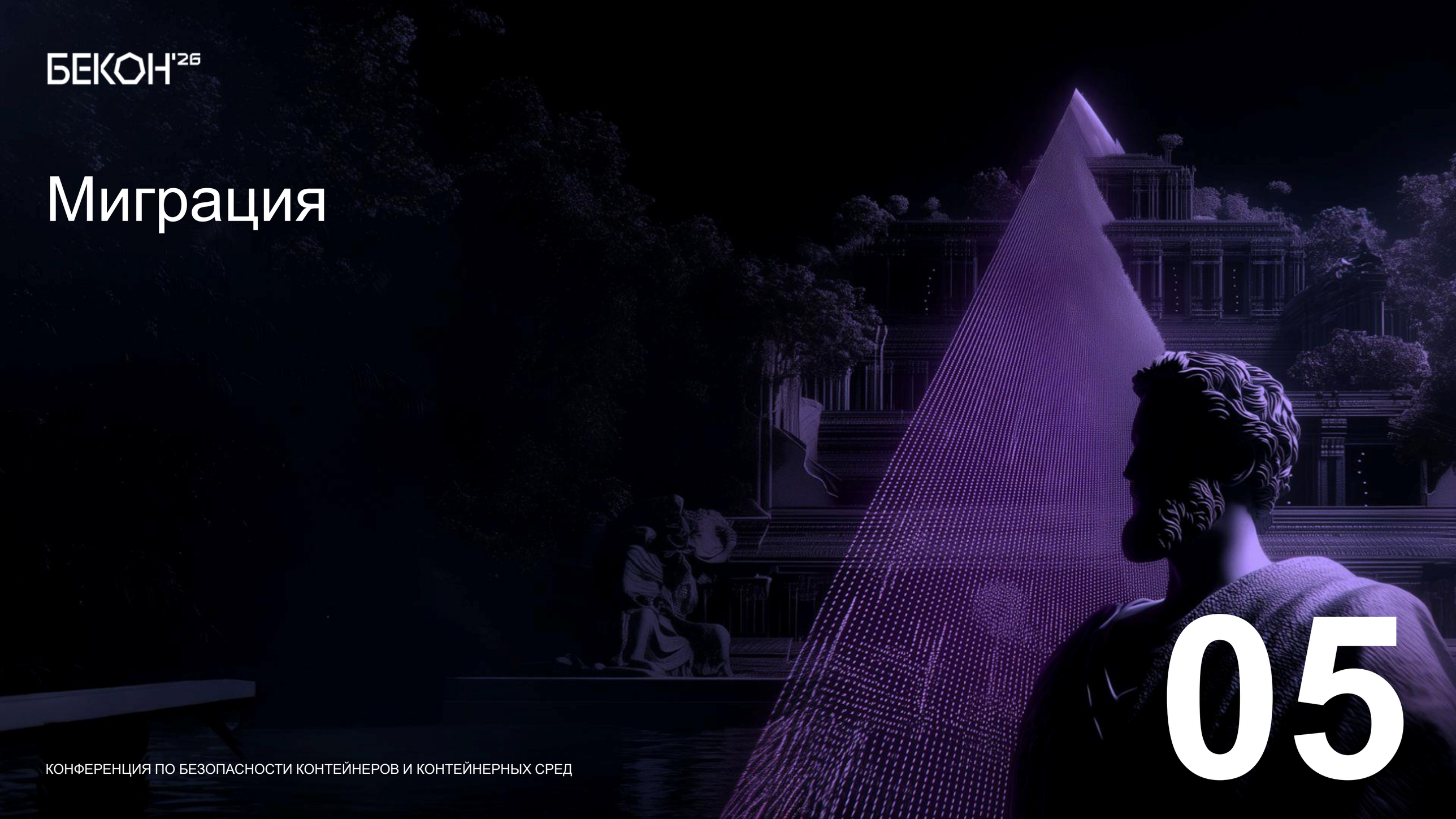
matchConditions: бюджет на выполнение

БЕКОН'26

```
17  package cel
18
19  const (
20    // PerCallLimit specify the actual cost limit per CEL validation call
21    // current PerCallLimit gives roughly 0.1 second for each expression validation call
22    PerCallLimit = 1000000
23
24    // RuntimeCELCostBudget is the overall cost budget for runtime CEL validation cost per ValidatingAdmissionPolicyBinding
    or Custom Resource
25    // current RuntimeCELCostBudget gives roughly 1 seconds for the validation
26    RuntimeCELCostBudget = 10000000
27
28    // RuntimeCELCostBudgetMatchConditionsistheoverallcostbudgetforruntime CEL
validationcostonmatchConditionsperobjectwithmatchConditions
29    // this is per webhook for validatingwebhookconfigurations and mutatingwebhookconfigurations or per
ValidatingAdmissionPolicyBinding
30    // current RuntimeCELCostBudgetMatchConditions gives roughly 1/4 seconds for the validation
31    RuntimeCELCostBudgetMatchConditions = 2500000
32
33 )
```

БЕКОН'26

Миграция



05

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Enforcement Actions

БЕКОН'26

Режимы validationsActions на биндинге:

- DENY (отклоняем запрос)
- WARN (предупреждение в kubectl)
- AUDIT (пишем в Audit Log)

01

✅ Можно комбинировать:

[Warn, Audit] или [Deny, Audit]

02

❌ Нельзя:

[Deny, Warn] дублирует информацию

03

- Шаг 1: WARN + AUDIT;
- Шаг 2: собираем исключения (анализируем логи);
- Шаг 3: DENY + AUDIT (включаем enforce);

04

Порядок действий в цепочке admission

- VAP проверки выполняются первыми;
- Webhook (kyverno) выполняются следом

05

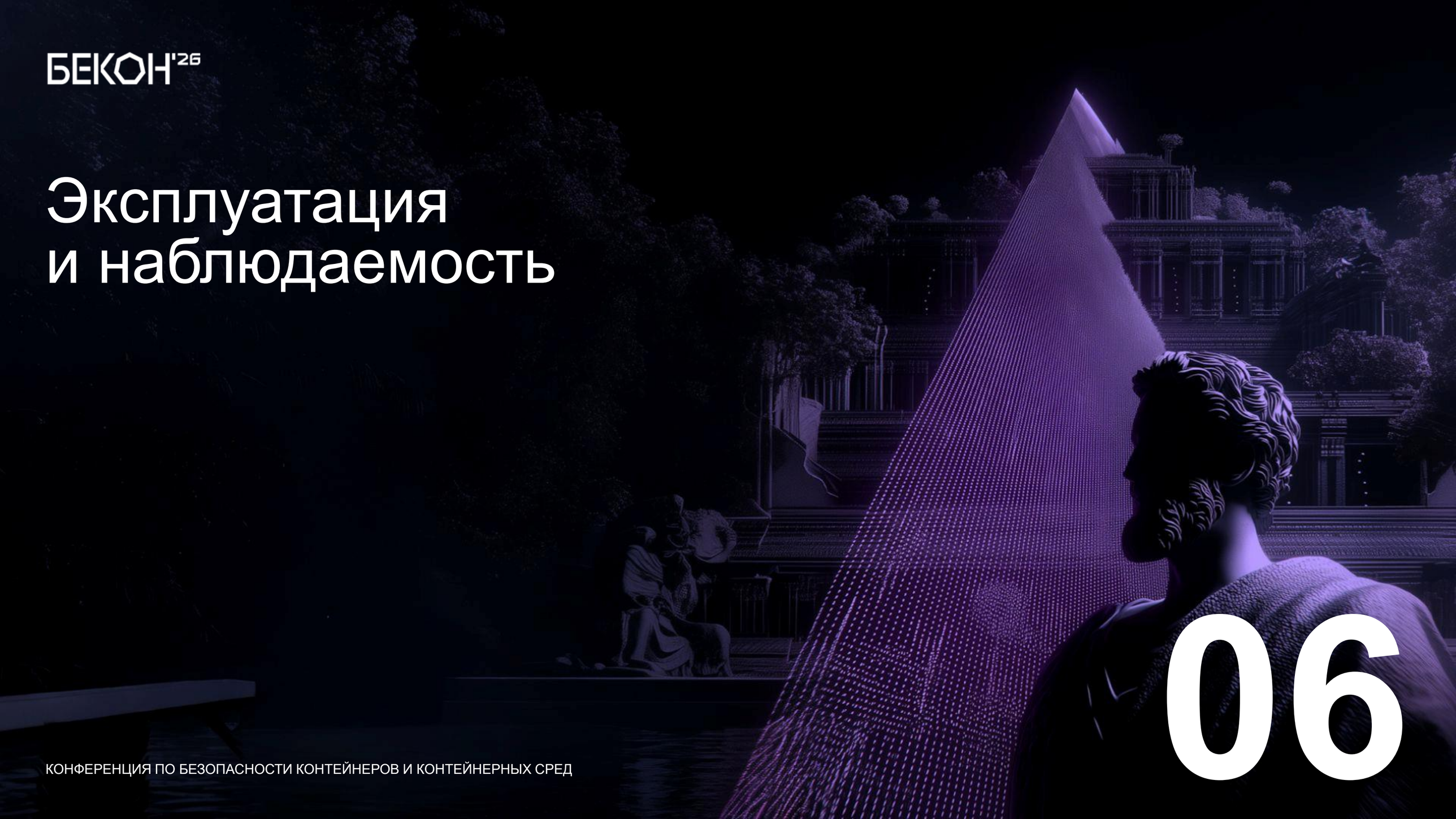
При миграции:

- VAP в [Warn, Audit], kyverno в Deny;
- Каждый отказ kyverno должен иметь предупреждение VAP; если нет = политики не эквивалентны.

06

БЕКОН'26

Эксплуатация и наблюдаемость



06

КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД

Аудит вместо хранения отчётов в etcd

БЕКОН'26

https://kubernetes.io/docs/reference/access-authn-authz/validating-admission-policy/#audit-annotations

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingAdmissionPolicy
metadata:
  name: "demo-policy.example.com"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
    - apiGroups:   ["apps"]
      apiVersions: ["v1"]
      operations:  ["CREATE", "UPDATE"]
      resources:   ["deployments"]
  validations:
    - expression: "object.spec.replicas > 50"
      messageExpression: "'Deployment spec.replicas set to ' +
string(object.spec.replicas)"
  auditAnnotations:
    - key: "high-replica-count"
      valueExpression: "'Deployment spec.replicas set to ' +
string(object.spec.replicas)"
```

```
# the audit event recorded
{
    "kind": "Event",
    "apiVersion": "audit.k8s.io/v1",
    "annotations": {
        "demo-policy.example.com/high-replica-count":
"Deployment spec.replicas set
to 128"
        # other annotations
        ...
    }
    # other fields
    ...
}
```

Мониторинг

БЕКОН'26

https://kubernetes.io/blog/2024/04/24/validating-admission-policy-ga/#set-up-monitoring

общее количество проверок политики

apiserver_validating_admission_policy_check_total
Validation admission policy check total, labeled by policy and further identified by binding and enforcement action taken.
StabilityLevel:BETA
Type: Counter
Labels:enforcement_actionerror_typepolicypolicy_binding

длительность проверок

apiserver_validating_admission_policy_check_duration_seconds
Validation admission latency for individual validation expressions in seconds, labeled by policy and further including binding and enforcementaction taken.
StabilityLevel:BETA
Type: Histogram
Labels:enforcement_actionerror_typepolicypolicy_binding

95-й перцентиль длительности для политики

histogram_quantile(0.95,sum(rate(apiserver_validating_admission_policy_check_duration_seconds_bucket{policy="name"}[5m])) by (le))

частота выполнения

rate(apiserver_validating_admission_policy_check_total{policy="name"}[5m])

Мониторинг

БЕКОН'26

apiserver_cel_compilation_duration_seconds
CEL compilation time in seconds.
StabilityLevel:BETA
Type: Histogram
Components:
kube-apiserver (/metrics)

apiserver_cel_evaluation_duration_seconds
CEL evaluation time in seconds.
Stability Level:BETA
Type: Histogram

Как тестировать политики без работающего кластера (envtest-подход)

БЕКОН'26

Как тестировать политики без работающего кластера (envtest-подход)

- Схема с etcd + kube-apiserver + политики VAP + kubectl apply --dry-run=server

01

- В качестве основы можно взять kubebuilder / envtest или собрать кастомный образ с supervisord

02

- kubebuilderenvtest: https://book.kubebuilder.io/reference/envtest

03

Что можно сделать?

Плейграунд VAP

- UI-форма (Go)
- Загружаем политики при старте
- Разработчик вбивает манифест и получает вердикт
- Без доступа к основному кластеру

01

Day2

- Gates для CICD
- Мониторинг «мертвых» исключений

02

VAP Playground

$ Вставьте YAML манифест для проверки

Pod | Ingress | Service

```
---
apiVersion: v1
kind: Pod
metadata:
  name: bad-pod
  namespace: example
spec:
  containers:
    - image: nginx
      name: nginx
      command: [ "/bin/sh", "-c", "sleep 9999" ]
      volumeMounts:
        - mountPath: /host
          name: my-volume
  volumes:
    - name: my-volume
      hostPath:
        path: /
```

$ Проверить

VAP Playground

$ Вставьте YAML манифест для проверки

Pod | Ingress | Service

```
---
apiVersion: v1
kind: Pod
metadata:
  name: bad-pod
  namespace: example
spec:
  containers:
    - image: nginx
      name: nginx
      command: [ "/bin/sh", "-c", "sleep 9999" ]
      volumeMounts:
        - mountPath: /host
          name: my-volume
  volumes:
    - name: my-volume
      hostPath:
        path: /
```

$ Проверить

$ Результат

```
❌ Ошибка проверки

📦 Ресурс: pods/bad-pod
📋 Политика: security-forbidden-host-path
🔗 Привязка: security-forbidden-host-path
📌 Описание: HostPath volumes are forbidden. The field spec.volumes[*].hostPath must be unset.
```

Manifest-based admission control

БЕКОН'26

“these policies are always on”

- ✗ Не заменяют API-based VAP
- ✗ Синхронизация конфигураций не предусмотрена
- ✗ Нет параметров
- ✗ Возникают вопросы по доставке
- ✗ Нужны процессы
- ✓ Есть метрики для обнаружения расхождений
- ✓ Защищают API-based admission-ресурсы

Типы API, которые освобождены от проверок доступа через обычный VAP

ValidatingAdmissionPolicies
ValidatingAdmissionPolicyBindings
MutatingAdmissionPolicies
MutatingAdmissionPolicyBindings
TokenReviews
LocalSubjectAccessReviews
SelfSubjectAccessReviews
SelfSubjectReviews

Выводы

БЕКОН'26

- VAP — для базовой валидации. Просто, быстро, надежно
- kyverno / OPA — для остального: background-сканы, сложная логика
- Не используйте вебхуки там, где они не нужны

Полезные ссылки

БЕКОН'26

Доклад “Kyverno: рецепты правильного приготовления”

CEL-плейграунд

Документация


БЕКОН'26
КОНФЕРЕНЦИЯ ПО БЕЗОПАСНОСТИ КОНТЕЙНЕРОВ И КОНТЕЙНЕРНЫХ СРЕД
Александр Алёхин
+7 (993) 980 – 80 – 51