Kubernetes. Модель безопасности. Контроллеры задач.

Не забудь включить запись!

План

- Модель безопасности Kubernetes
- Контроллеры задач в Kubernetes

Модель безопасности Kubernetes

Namespaces

- Можно создать несколько виртуальных кластеров в рамках одного физического кластера
- Namespace - один виртуальный кластер

Namespaces

Часто применяются для:

- Обеспечения multitenancy (множественная аренда)
- Разграничения прав между командами
- Делегирования части административных функций доверенным пользователям
- Лимитирования ресурсов на проект с помощью квот (cpu, memory)

Namespaces

Достигается это путем:

- Создания области видимости имен (names)
- Подключения политик безопасности и авторизации к выделенной части кластера

Пример подключения к API:

```
$ curl -H "Authorization: Bearer $TOKEN"
https://$API_ADDRESS/api/v1/namespaces/<namespace>/pods/<name>
{
  "kind": "Pod",
  "apiVersion": "v1",
  "metadata": ...
}
```

Namespaces

В namespace нельзя увидеть:

- Сами namespace
- Ноды
- PersistentVolumes
- ...

Namespaces

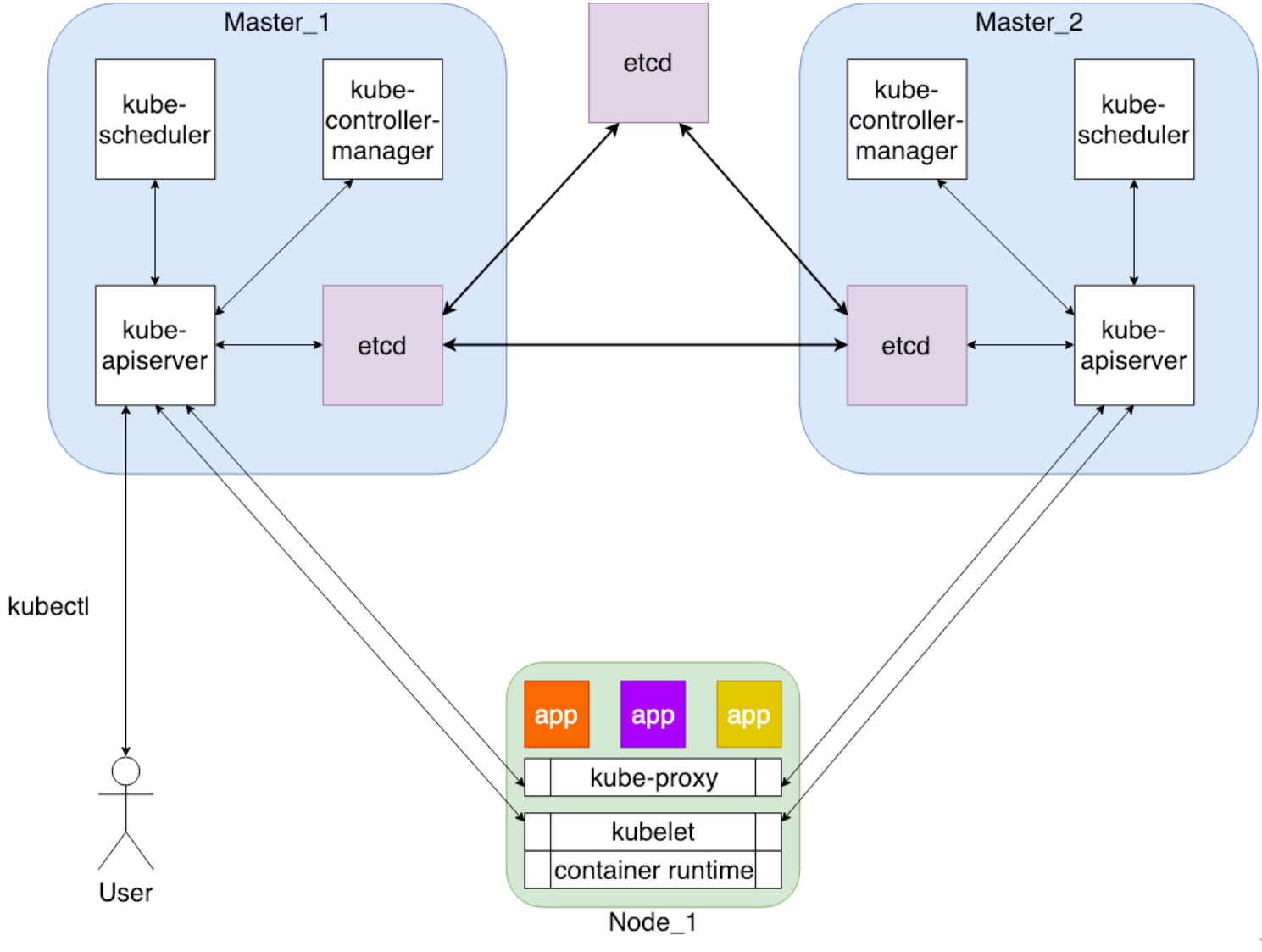
```
$ kubectl create ns dev-1
namespace "dev-1" created

$ kubectl create ns dev-2
namespace "dev-2" created

$ kubectl create ns services
namespace "services" created
```

Namespaces

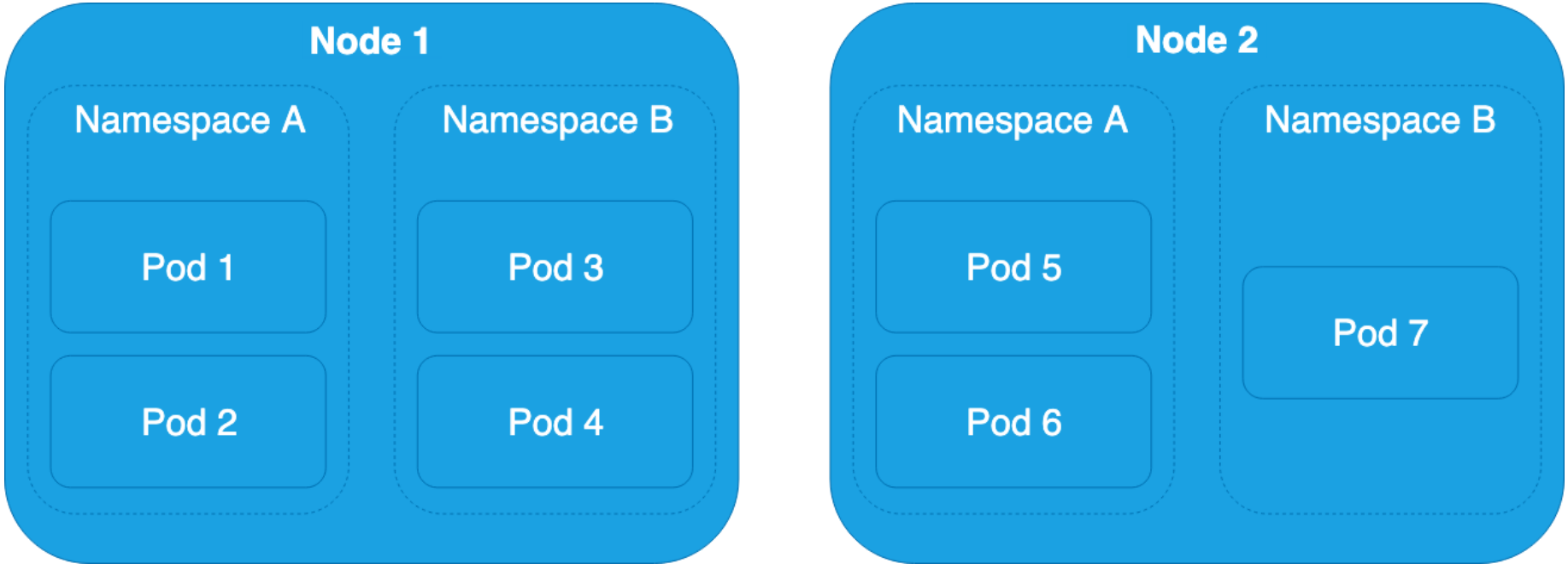
dev-1-ns.yml

```
apiVersion: v1
 kind: Namespace
 metadata:
   name: "dev-1"
```

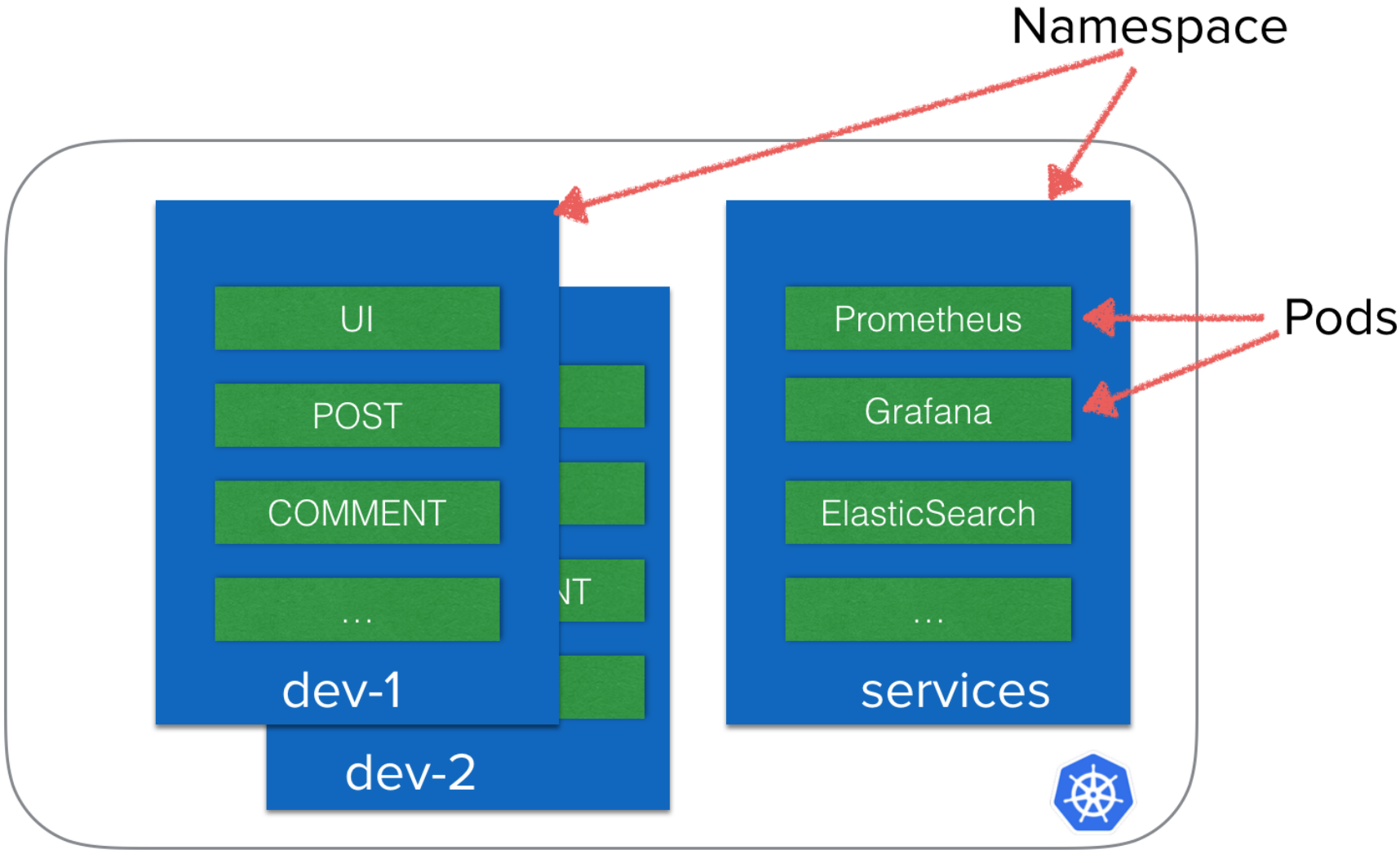
Namespaces


Master_1
kube-scheduler
kube-controller-manager
kube-apiserver
etcd
etcd
Master_2
kube-controller-manager
kube-scheduler
etcd
kube-apiserver
kubectl
User
app
app
app
kube-proxy
kubelet
container runtime
Node_1

Namespaces


Node 1
Namespace A
Pod 1
Pod 2
Namespace B
Pod 3
Pod 4
Node 2
Namespace A
Pod 5
Pod 6
Namespace B
Pod 7

Namespaces


Namespace
Pods
UI
POST
COMMENT
...
dev-1
dev-2
Prometheus
Grafana
ElasticSearch
...
services

Namespaces

Показать поды только в namespace dev-1

```
$ kubectl get pods -n dev-1
$ kubectl get pods --namespace dev-1
```

Namespaces

По умолчанию в кластере создается три namespace:

- **default** - для объектов у которых явно не определена принадлежность к другому namespace
- **kube-system** - для объектов созданных Kubernetes
- **kube-public** - для объектов к которым нужен доступ из любой точки кластера

Namespaces

У каждой команды есть возможность независимо от других использовать кластер. При этом у команды есть свои:

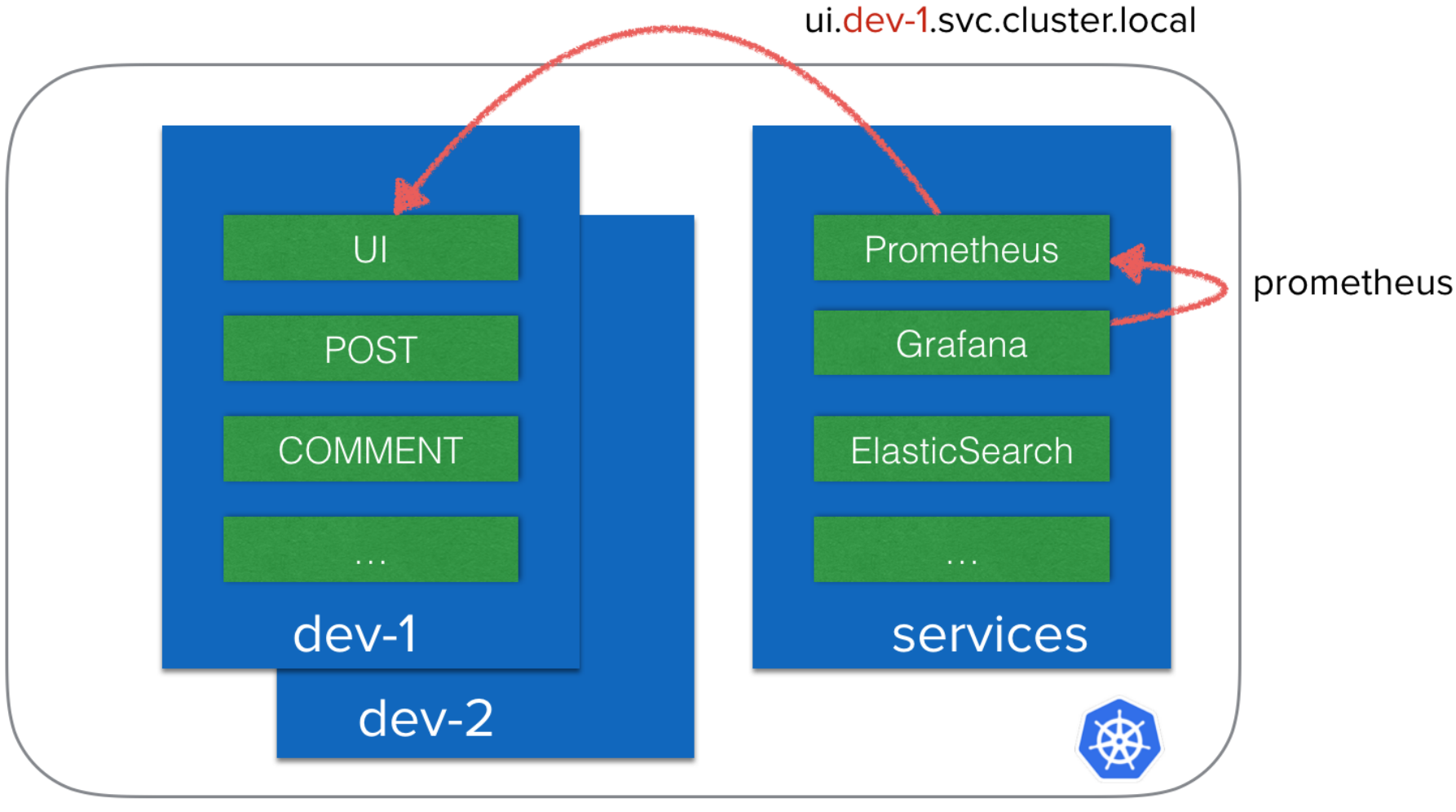
- Ресурсы (pod, service и т.д.)
- Политики (кому и какие действия можно совершать в кластере)
- Ограничения и квоты на ресурсы

Namespaces

Получить доступ к service запущенному внутри namespace можно по адресам:

- **service-name** - внутри одного namespace
- **service-name.namespace.svc.cluster.local** - из любой точки кластера

Namespaces


ui.dev-1.svc.cluster.local
UI
POST
COMMENT
...
dev-1
dev-2
Prometheus
Grafana
ElasticSearch
...
services
prometheus

ResouceQuotas

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-1-quota
spec:
  hard:
    pods: "4"
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

ResouceQuotas

```
Warning  FailedCreate  1m                 replicaset-controller  Error creating: pods
"ui-64f497f8fd-pj8vg" is forbidden: failed quota: dev-1-quota:must specify
limits.cpu,limits.memory,requests.cpu,requests.memory
```

LimitRanges

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: dev-1-limit-range
spec:
  limits:
  - default:
      memory: 128Mi
      cpu: 100m
    defaultRequest:
      memory: 64Mi
      cpu: 50m
    type: Container
```

Namespaces

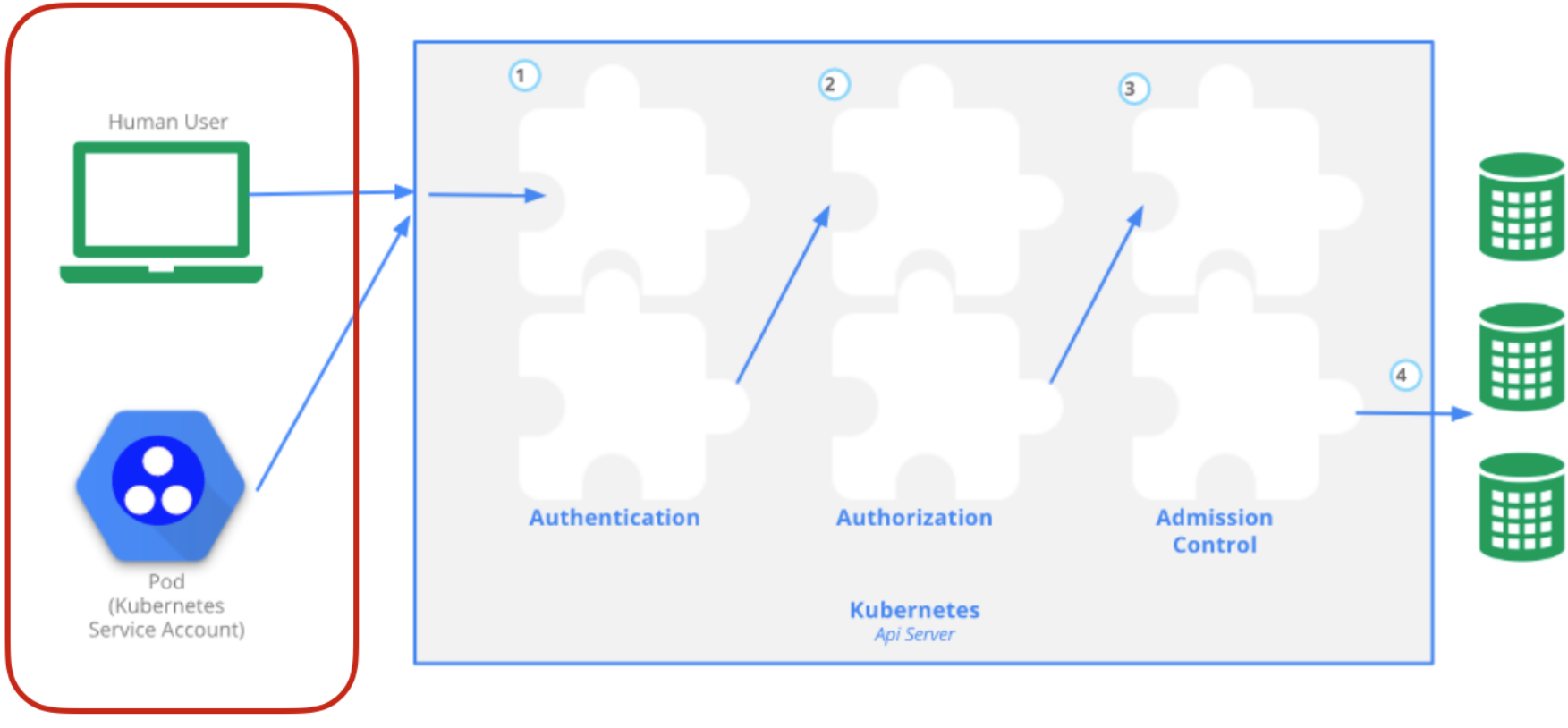
Namespace предоставляет каждой команде:

- Область видимости именованных ресурсов, чтобы избежать совпадения имен в пределах кластера
- Делегирование функций управления доверенным пользователям
- Возможность лимитировать доступные ресурсы

Demo

Identity Management

Accounts


Human User
Pod
(Kubernetes
Service Account)
1
2
3
4
Authentication
Authorization
Admission
Control
Kubernetes
Api Server

Accounts

- **User account** - учетная запись с использованием которой работает пользователь
- **Service account** - учетная запись с использованием которой которой процесс внутри pod работает с Kubernetes (например, с API)

Accounts

Service accounts	User accounts
Для задач в pod	Для людей
Разделены по namespace	Работают во всем кластере
Управляются Kubernetes	Управляются внешними сервисами
Конфигурация приложения может включать Service Account	

User Accounts

Не могут быть созданы через API Kubernetes (нет объекта в Kubernetes API). В качестве источников информации могут использоваться:

- Файлы (токены, сертификаты)
- LDAP
- SAML
- Kerberos

Service Accounts

- Service Account Admission Controller
- Token Controller
- Service Account Controller

Service Account Admission Controller

- Если у pod не установлен Service Account, то устанавливает Service Account "default"
- Проверяет, что Service Account pod существует
- Если у pod нет своих ImagePullSecrets, то устанавливает ImagePullSecrets Service Account
- Подключает к pod volume с ключом для API: /var/run/secrets/kubernetes.io/serviceaccount

Service Account

prometheus-sa.yml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
  namespace: services
...
```

prometheus.yml

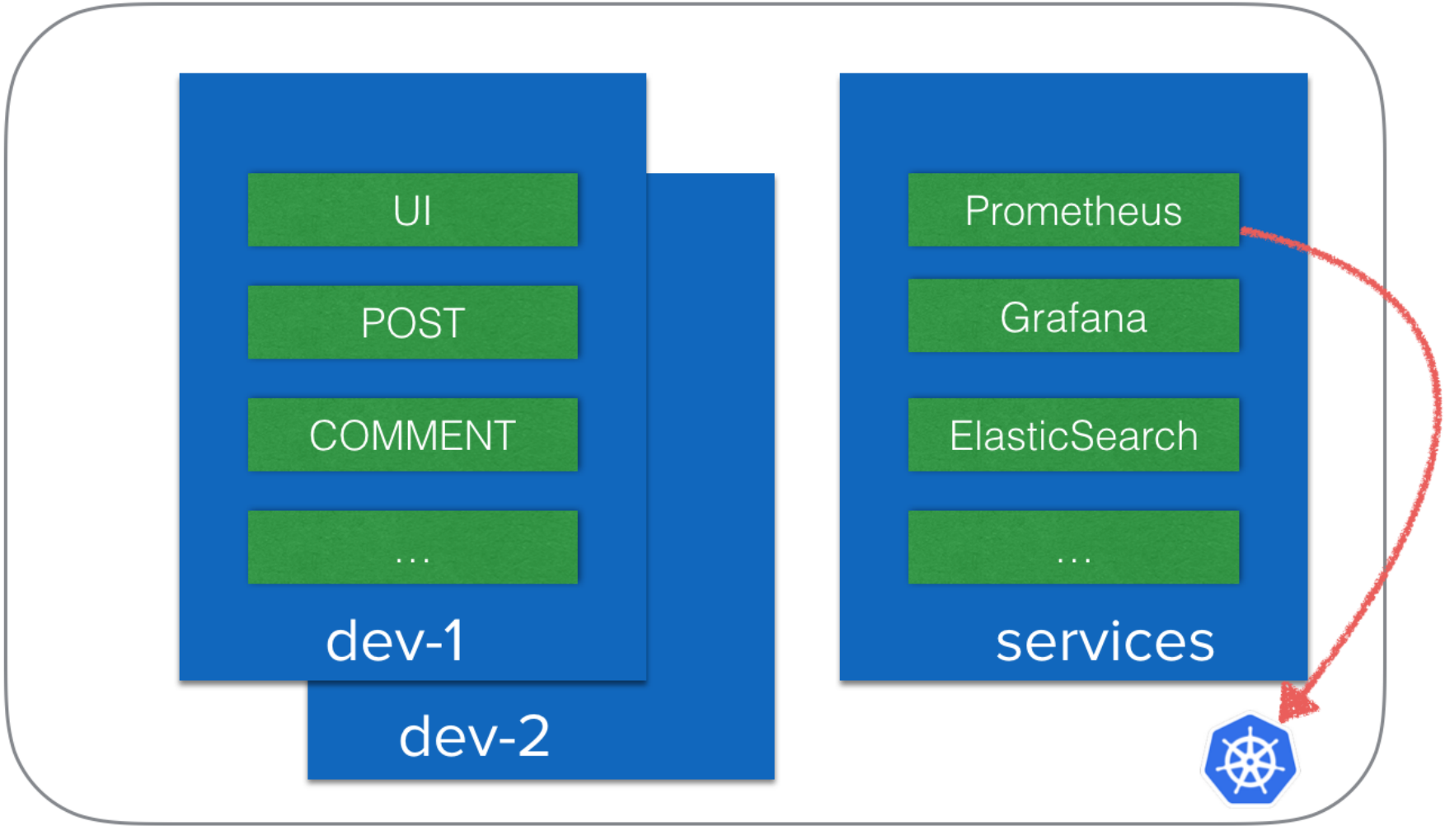
```
…
     - job_name: 'kubernetes-nodes'
       tls_config:
         ca_file: /var/run/secrets/kubernetes.io/serviceaccount/ca.crt
         insecure_skip_verify: true
       bearer_token_file: /var/run/secrets/kubernetes.io/serviceaccount/token
```

Service Account

prometheus-deployment.yml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: prometheus-core
  namespace: services
spec:
 replicas: 1
  template:
    metadata:
      name: prometheus-main
      labels:
        k8s-app: prometheus
        component: core
    spec:
     serviceAccountName: prometheus
...
```

Accounts


UI
POST
COMMENT
…
dev-1
dev-2
Prometheus
Grafana
ElasticSearch
…
services

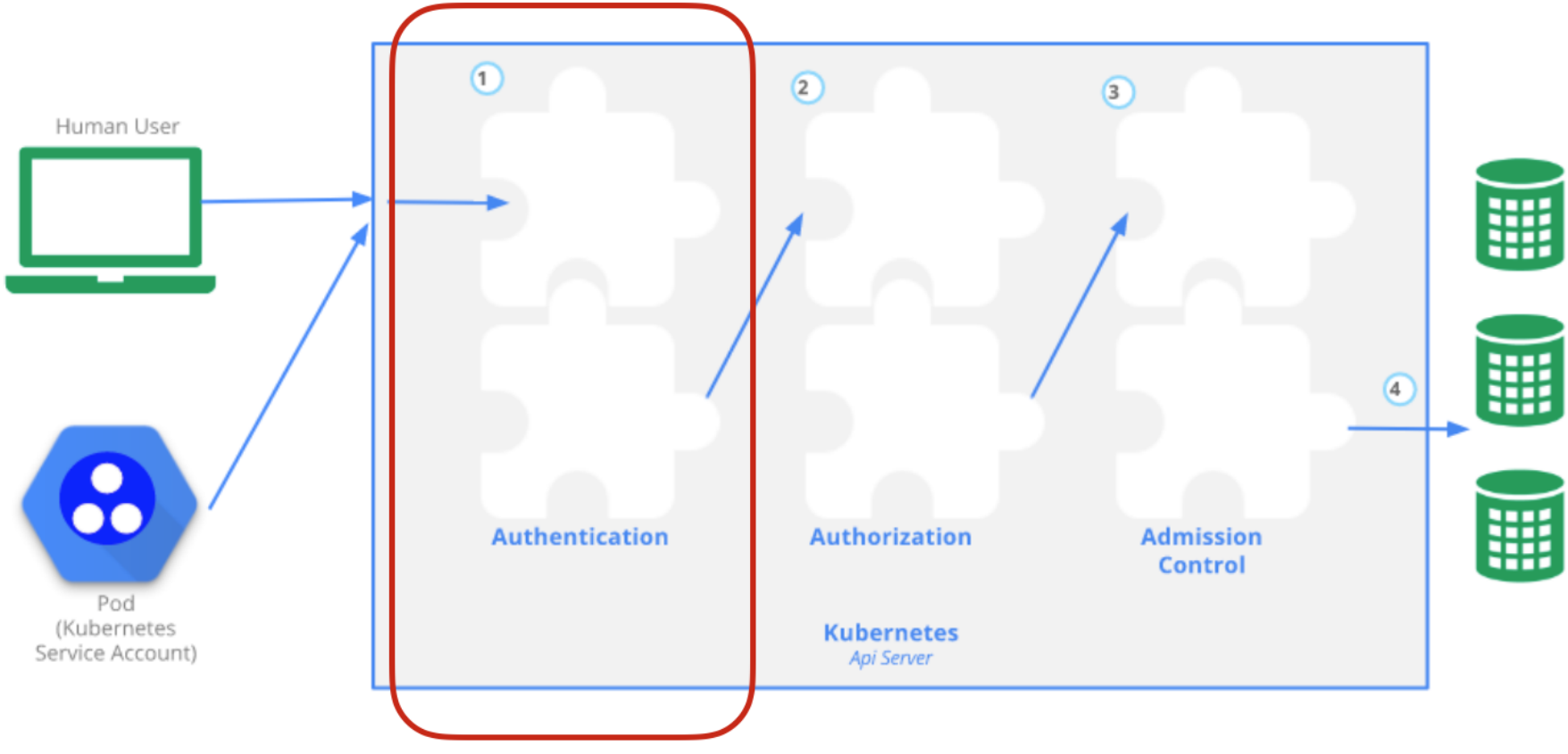
Service Account Controller

- Управляет Service Account внутри namespace
- Следит, чтобы Service Account "default" существовал во всех активных namespace

Token Controller

- Следит за созданием и удалением токенов для Service Account
- Следит, чтобы ключи всегда соответствовали существующему Service Account

Аутентификация



Аутентификация

- X509 сертификаты
- Статичные токены
- Bootstrap токены
- Статичные файлы с паролями
- Authenticate Proxy

Могут использоваться сразу несколько методов

Аутентификация

Основные поля:

- **Username** - идентификация конечного пользователя
- **UID** - уникальный идентификатор конечного пользователя
- **Groups** - объединение нескольких пользователей в группу

Static Token File

kube-apiserver запускается с флагами:

```
--token-auth-file=token-auth
```

token-auth

```
token,user,uid,"group1,group2"
```

Static Password File

kube-apiserver запускается с флагами:

```
--basic-auth-file=basic-auth
```

basic-auth

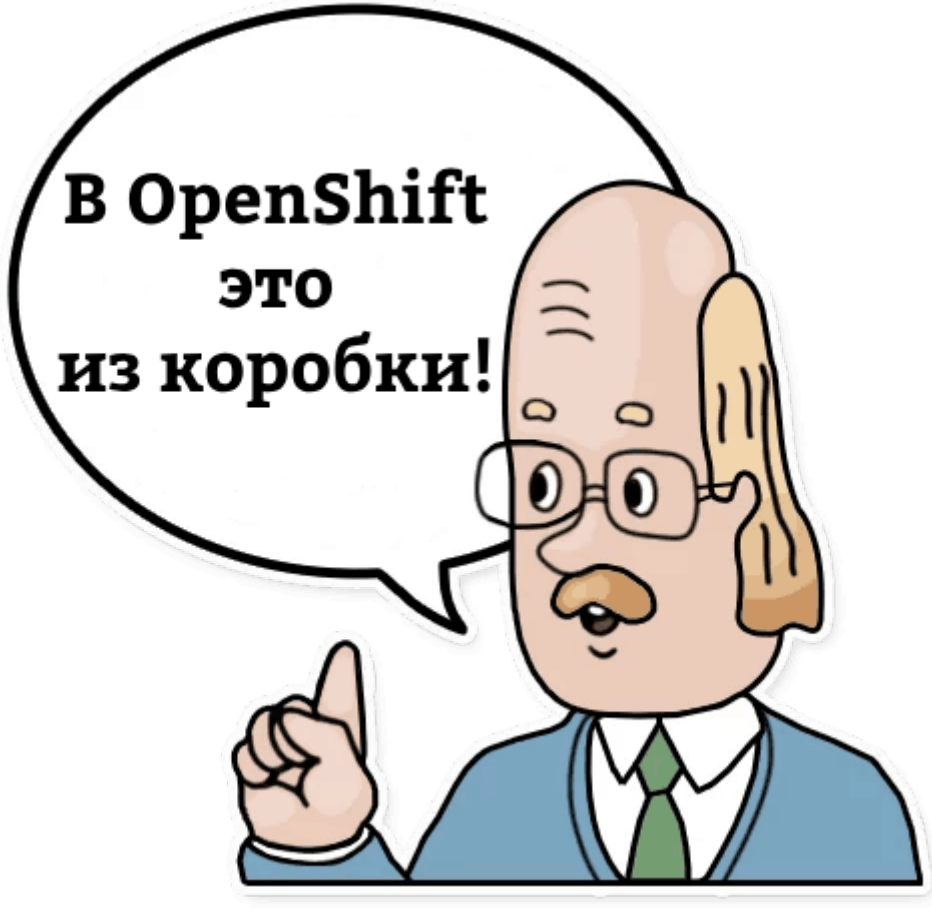
```
password,user,uid,"group1,group2"
```

X509

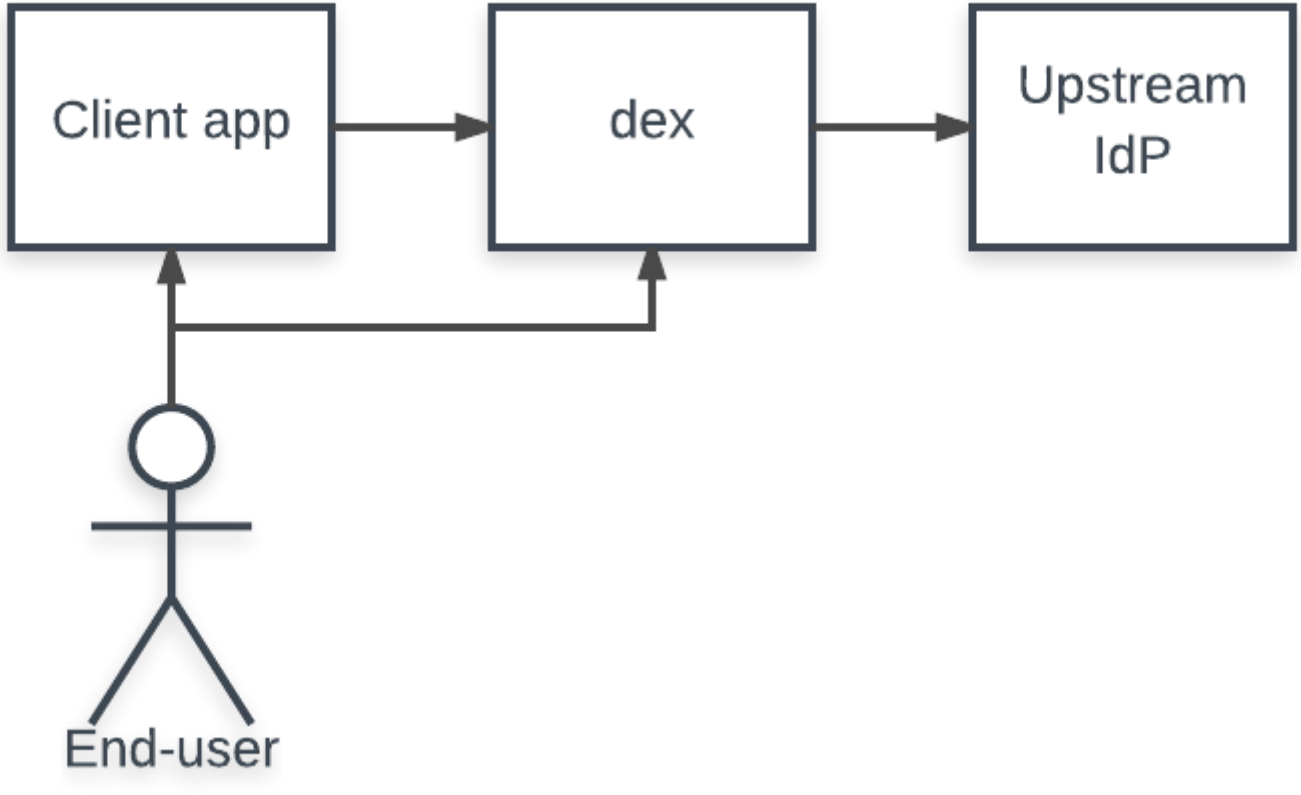
- Стандартные сертификаты
- Должны быть подписаны корневым сертификатом кластера

Demo

Active Directory



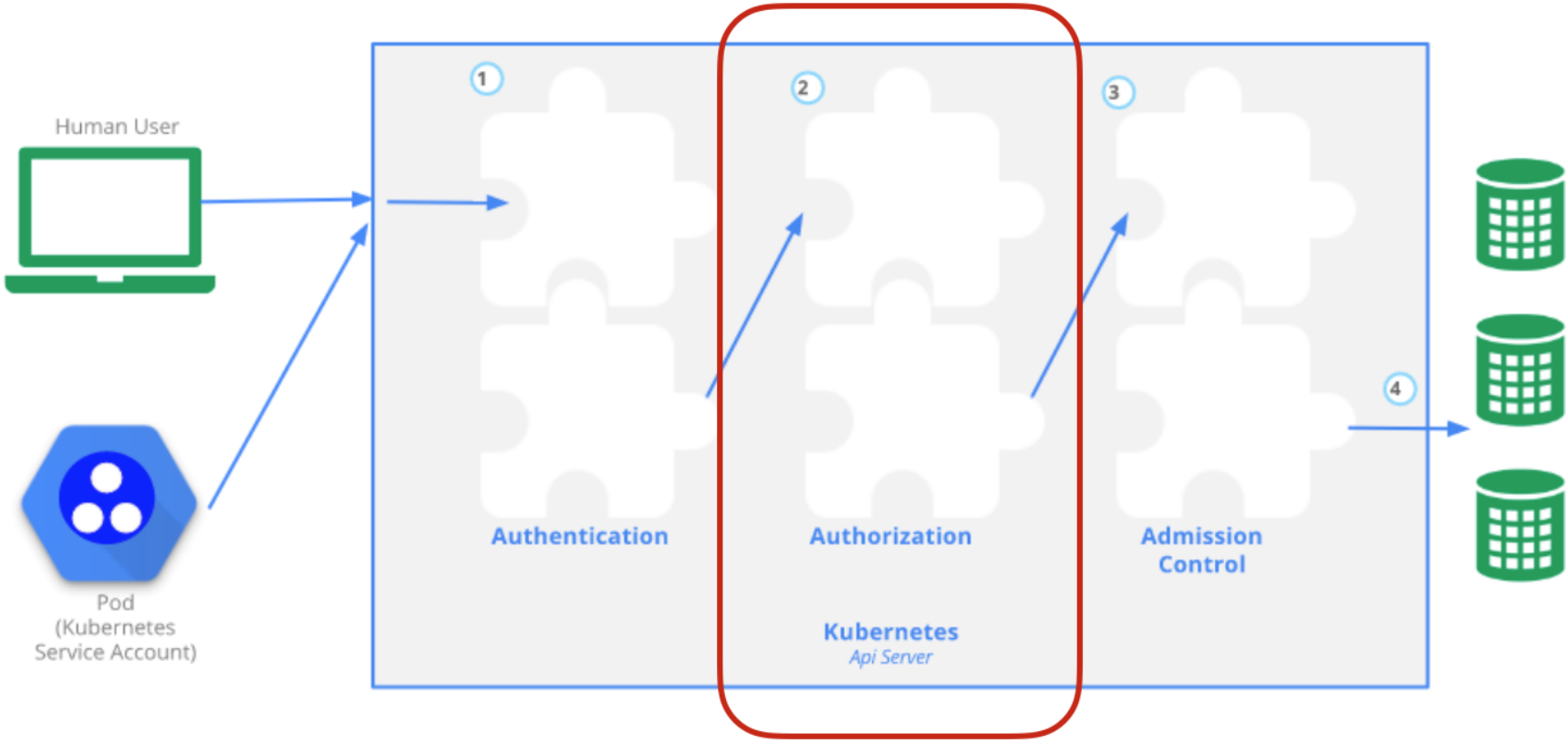
Active Directory



Active Directory

```
kubectl config set-credentials USER_NAME \
   --auth-provider=oidc \
   --auth-provider-arg=idp-issuer-url=( issuer url ) \
   --auth-provider-arg=client-id=( your client id ) \
   --auth-provider-arg=client-secret=( your client secret ) \
   --auth-provider-arg=refresh-token=( your refresh token ) \
   --auth-provider-arg=idp-certificate-authority=( path to your ca certificate ) \
   --auth-provider-arg=id-token=( your id_token )
```

Авторизация



Авторизация

- Attribute-based access control (ABAC)
- **Role-based access control (RBAC)**
- Node
- Webhook

ABAC (legacy)

- Авторизирует действия пользователя на основе набора политик
- Политики состоят наборов атрибутов по которым происходит авторизация
- Виды атрибутов: пользовательские атрибуты, атрибуты ресурсов, объектов, окружений и т.д.

ABAC (legacy)

kube-apiserver запускается с флагами:

```
--authorization-policy-file=Policy.json
--authorization-mode=ABAC
```

Policy.json

```
{
  "apiVersion":
    "abac.authorization.kubernetes.io/v1beta1",
    "kind": "Policy",
    "spec": {
      "user": "alice",
      "namespace": "*",
      "resource": "*",
      "apiGroup": "*"
    }
}
```

Example file

RBAC

- Авторизация происходит на основании роли пользователя
- Роль - набор правил, задающих разрешения
  - **ClusterRole** - роль в рамках всего кластера
  - **Role** - роль в рамках одного namespace
- Роли пользователям назначаются с помощью привязок (RoleBindings, ClusterRoleBindings)

RBAC

Правило для чтения информации о pod в namespace default:

```
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # Группы ресурсов
  resources: ["pods"] # Ресурсы
  verbs: ["get", "watch", "list"] # Действия
```

RBAC

ClusterRole = **Role** + :

- Кластерные ресурсы (Nodes, PersistentVolumes)
- Нересурсные эндпоинты (“/healthz”)
- Ресурсы из нескольких или всех namespaces;

ClusterRole

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: prometheus
rules: # Блок правил
- apiGroups: [""]
  resources: # Блок ресурсов
  - nodes
  - nodes/proxy
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"] # Блок разрешенных действий
- apiGroups: ["extensions"]
  resources:
  - deployments
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
```

ClusterRoleBinding

Связываем роль и serviceAccount:

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef: # Что привязываем
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects: # Кому привязываем
- kind: ServiceAccount
  name: prometheus
  namespace: services
```

ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: services
- kind: User
  name: alice
  apiGroup: rbac.authorization.k8s.io
- kind: Group
  name: admin
  apiGroup: rbac.authorization.k8s.io
```

Demo

Node

- Специальный механизм авторизации для агентов kubelet
- Авторизирует на основе хостов на которых kubelet запущен

Webhook

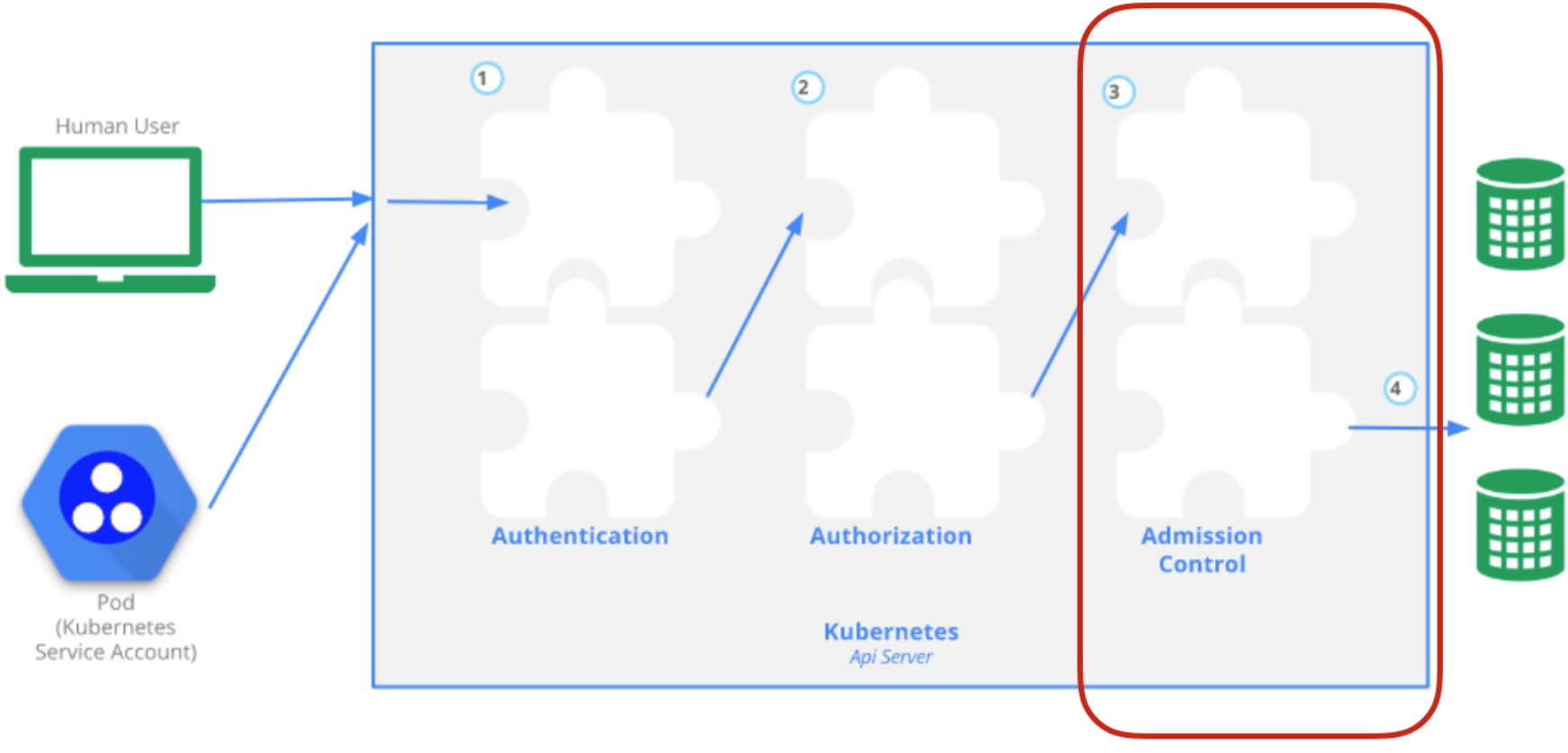
- Когда происходит событие, отправляется HTTP POST запрос
- Авторизация происходит на основании ответа на этот запрос

Webhook

Request

```
{
  "apiVersion": "authorization.k8s.io/v1beta1",
  "kind": "SubjectAccessReview",
  "spec": {
    "resourceAttributes": {
      "namespace": "dev-1",
      "verb": "get",
      "resource": "pods"
    },
    "user": "alice",
    "group": [
      "app1",
      "app2"
    ]
  }
}
```

Admission Control



Admission Controllers

Набор модулей

- AlwaysPullImages
- ResourceQuota
- PodSecurityPolicy
- ...

Demo

Security Context

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: prometheus
spec:
  replicas: 1
  selector:
    matchLabels:
      app: prometheus
  template:
    metadata:
      name: prometheus
      labels:
        app: prometheus
    spec:
      containers:
      - name: prometheus
        image: prom/prometheus
        securityContext:
          runAsUser: 1001
```

Controllers

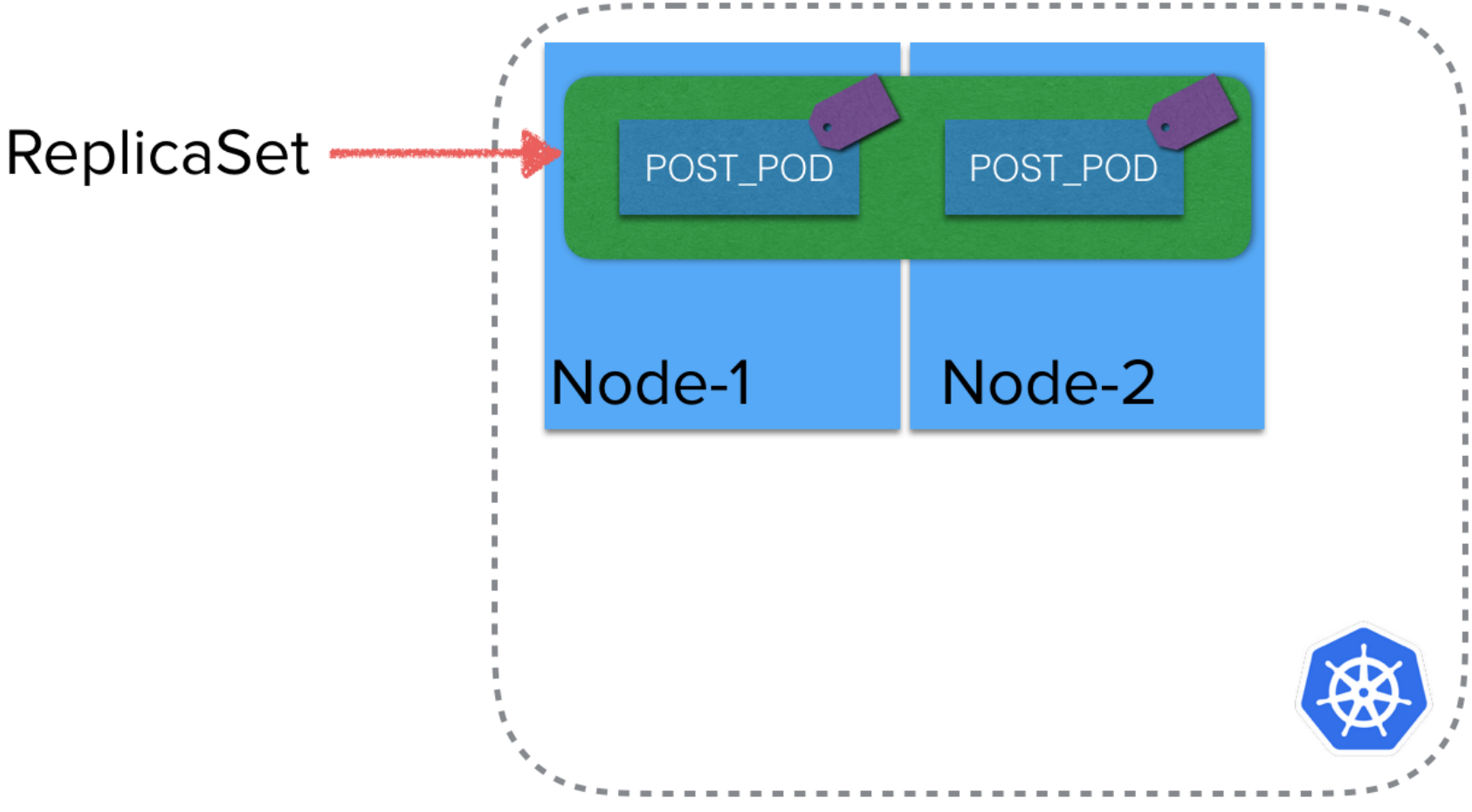
ReplicaSet

- ReplicaSet контролирует, что в любой момент времени запущено необходимое количество Pod'ов
- ReplicationController + set-based label selector
- Не умеет “гладко” выкатывать новые поды

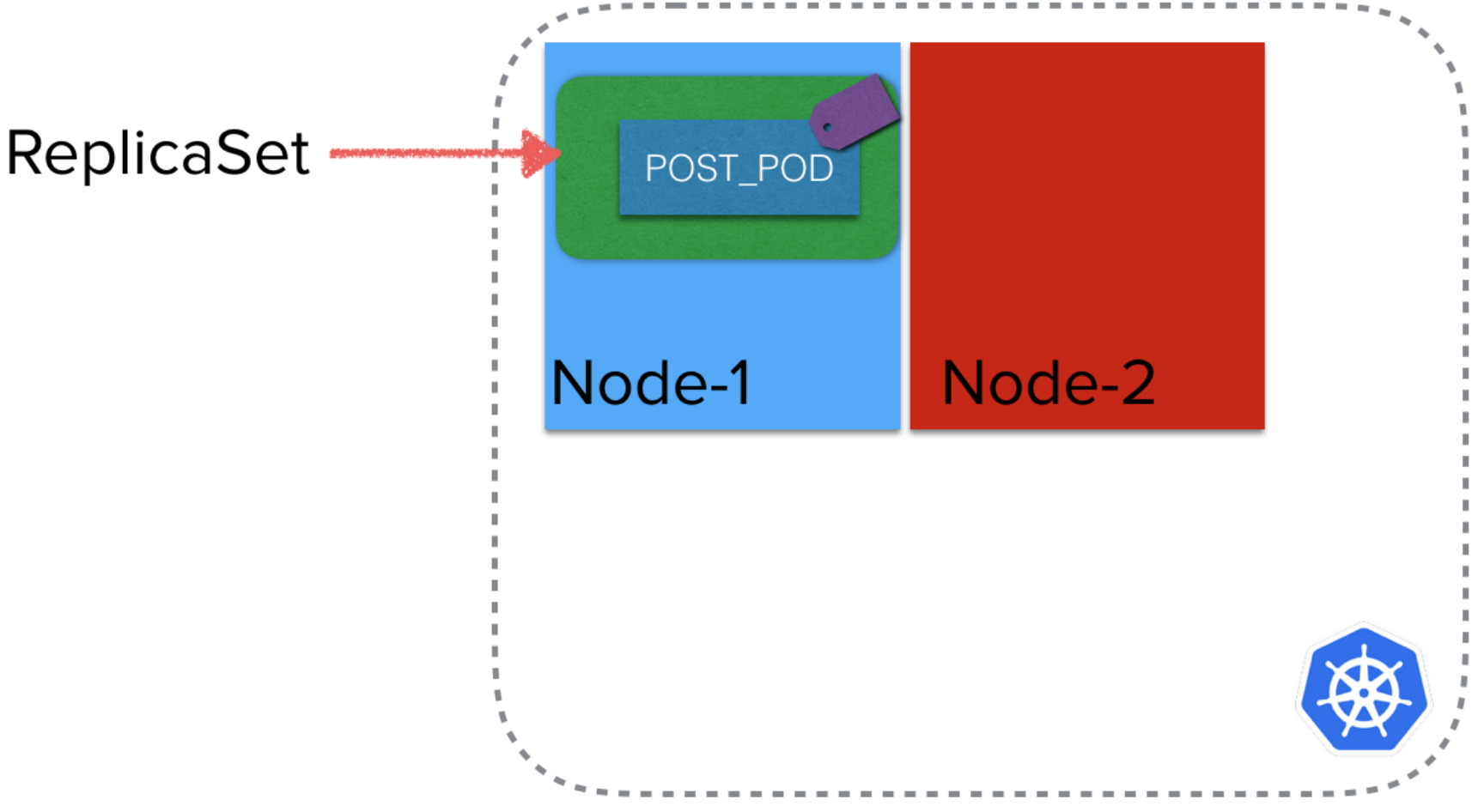
ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: post-rs
spec:
  replicas: 2
  selector:
    matchLabels:
      app: post
  template:
    metadata:
      name: post
      labels:
        app: post
    spec:
      containers:
      - image: avtandilko/post
        name: post
```

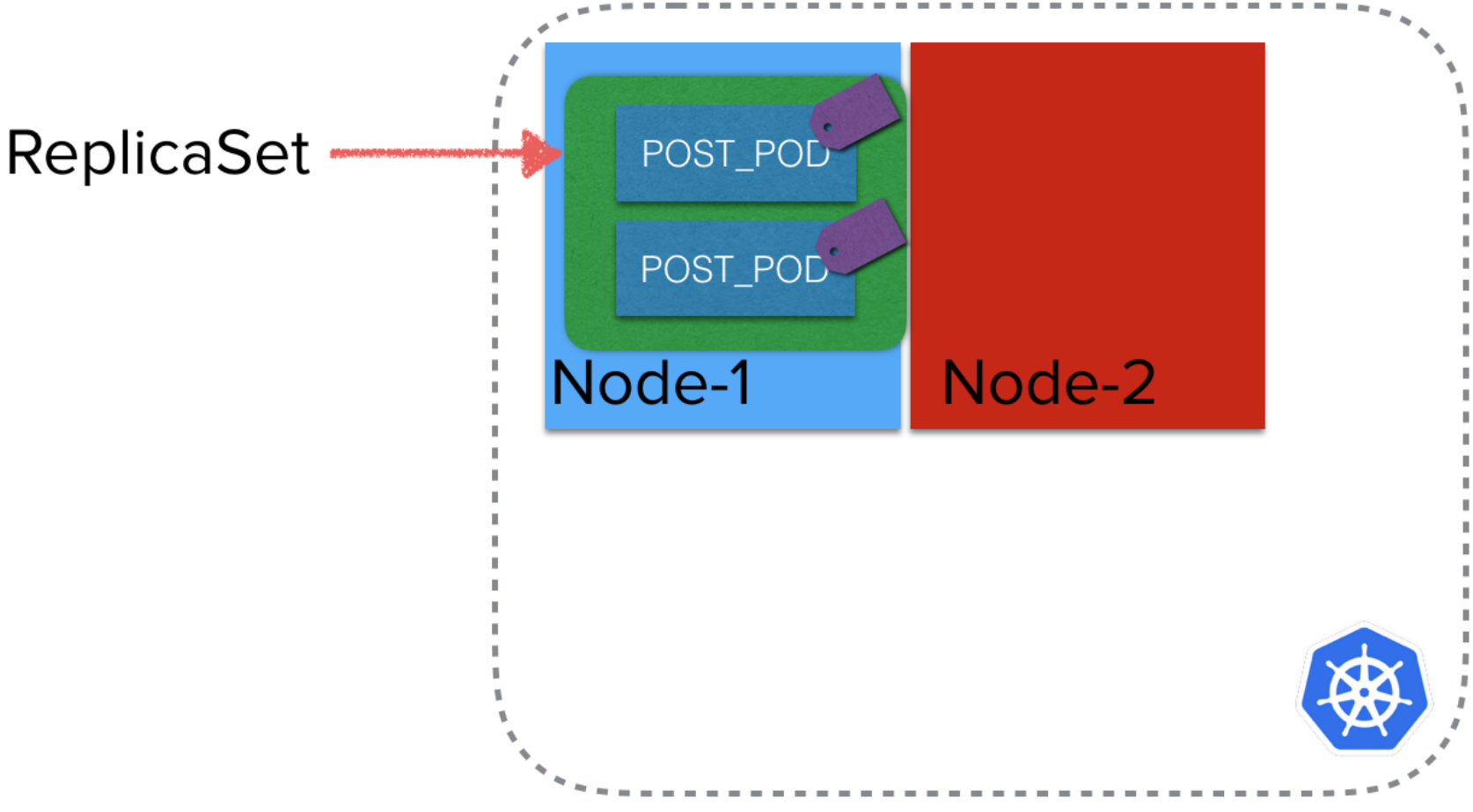
ReplicaSet



ReplicaSet



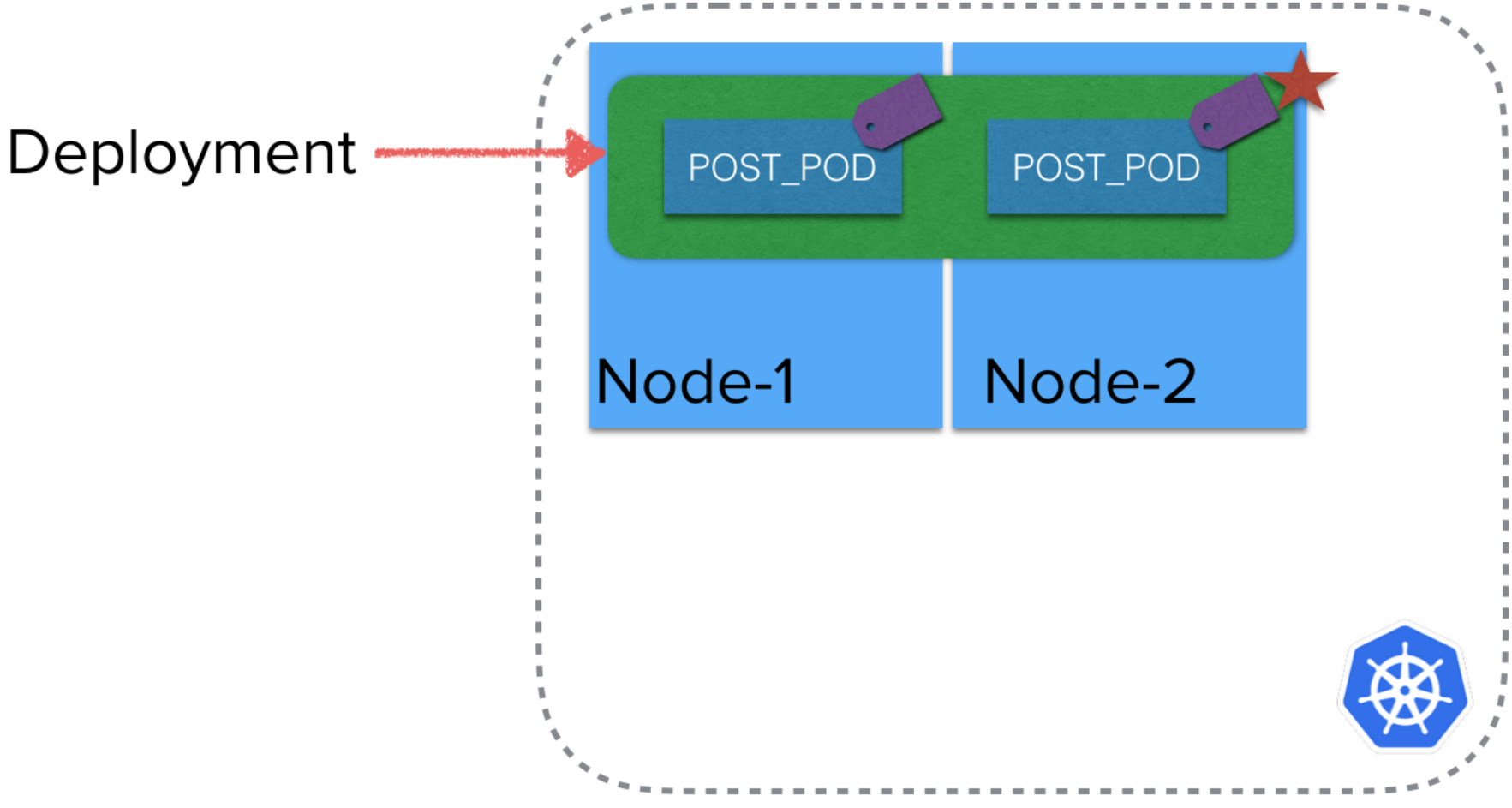
ReplicaSet



Deployment

- Deployment = ReplicaSet + ⭐:
- Текущее и желаемое состояние подов
- Контроль процесса обновления состояния

ReplicaSet



StatefulSet

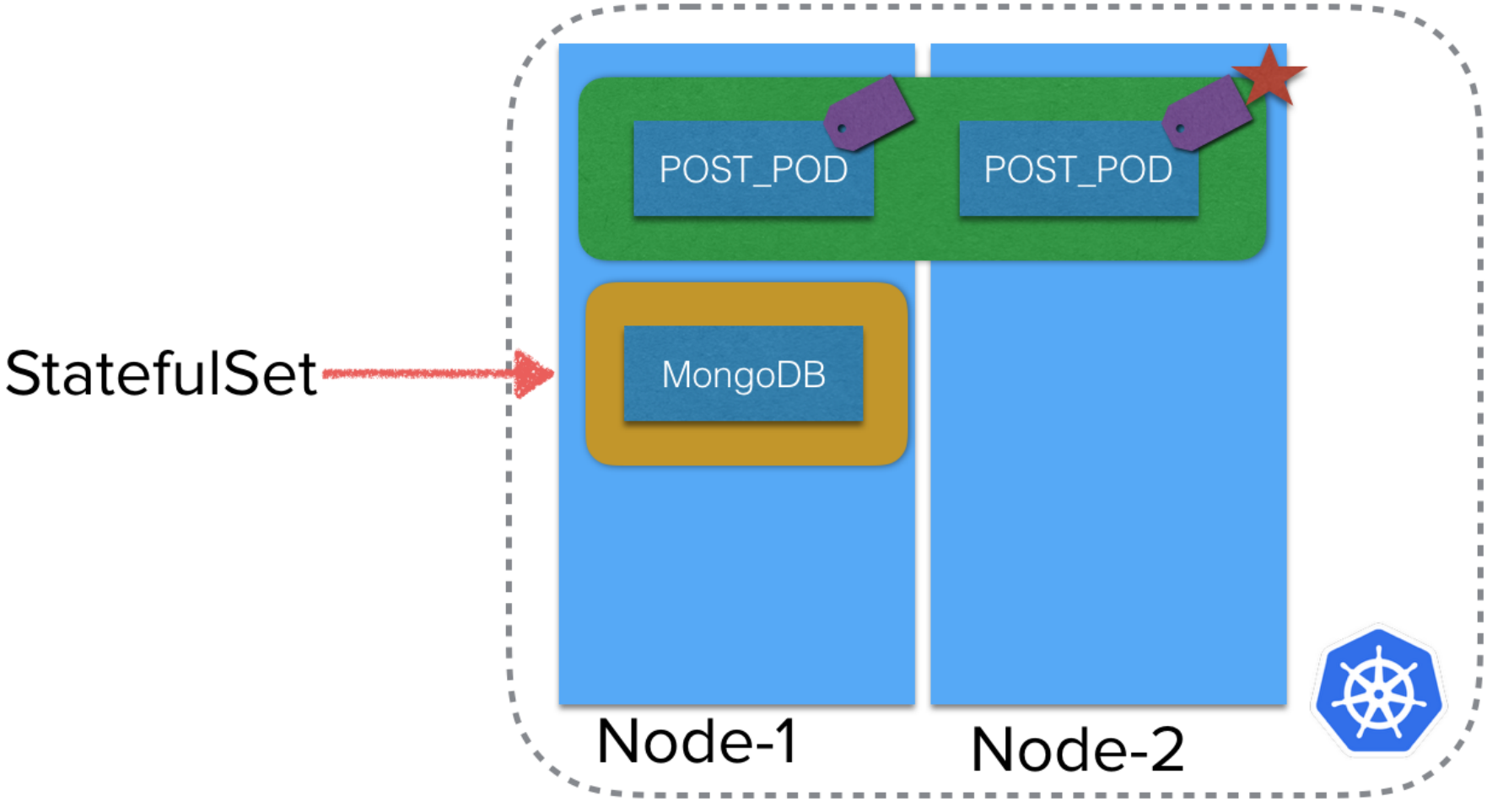
- Управляет выкаткой и масштабированием pod также как и Deployment
- Гарантирует очередность запуска и уникальность запущенных pod
- Гарантирует, что pod не сменит свои характеристики (name, pvc) при пересоздании
- Необходимо вручную обрабатывать отказы хостов с pod, управляемыми StatefulSet

StatefulSet

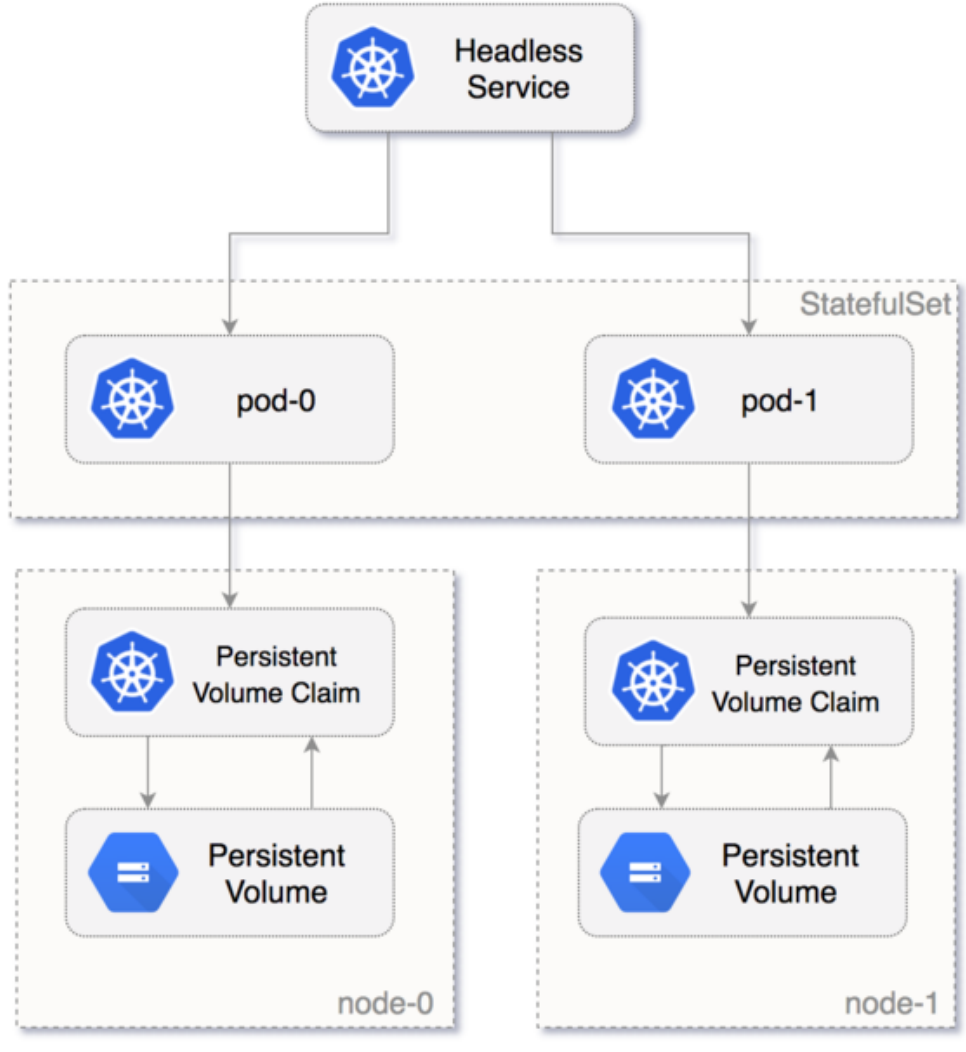
Применяется, когда приложению нужно:

- Постоянные уникальные сетевые идентификаторы
- Отличное от Deployment поведение при работе с volumes

StatefulSet



StatefulSet



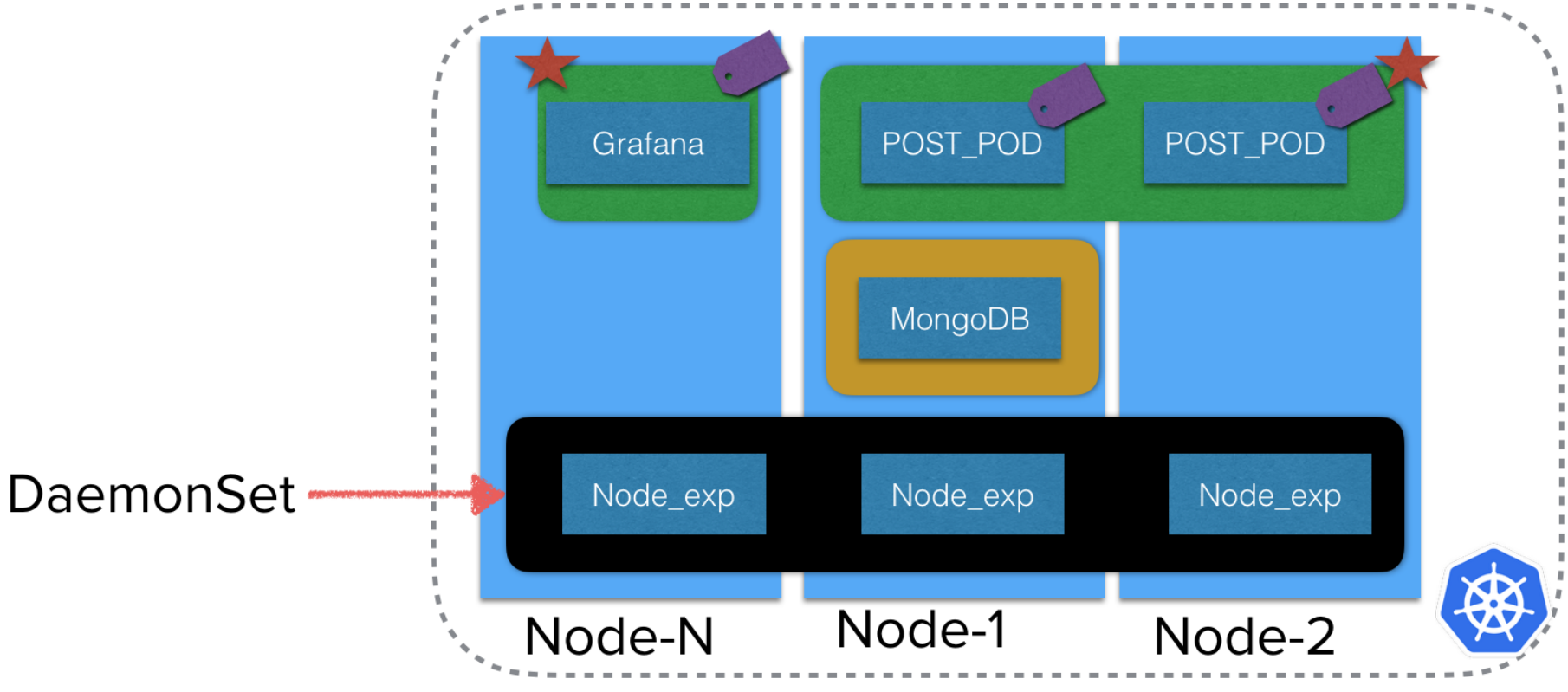
DaemonSet

- Запуск задачи на каждой ноде кластера
- DaemonSet могут быть запущены до старта Scheduler
- Будут запущены еще до запуска остальных задач
- Поддерживается Rolling Update

DaemonSet

```yaml
kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: prometheus-node-exporter
  namespace: Service
  labels:
    k8s-app: prometheus
    component: node-exporter
spec:
  template:
    metadata:
      name: prometheus-node-exporter
      labels:
        k8s-app: prometheus
        component: node-exporter
    spec:
      containers:
      - image: prom/node-exporter:0.17.0
        name: prometheus-node-exporter
        ports:
        - name: prom-node-exp
          containerPort: 9100
          hostPort: 9100
```

DaemonSet


DaemonSet
Grafana
POST_POD
POST_POD
MongoDB
Node_exp
Node_exp
Node_exp
Node-N
Node-1
Node-2

Job

Запуск одного или нескольких pod для выполнения разовых задач

```
apiVersion: batch/v1
kind: Job
metadata:
  name: grafana-import
spec:
  template:
    spec:
      containers:
      - name: grafana-import
        image: busybox
        args:
        - curl http://admin:admin@grafana:3000/api/dashboards/import
      restartPolicy: OnFailure
  backoffLimit: 4
```

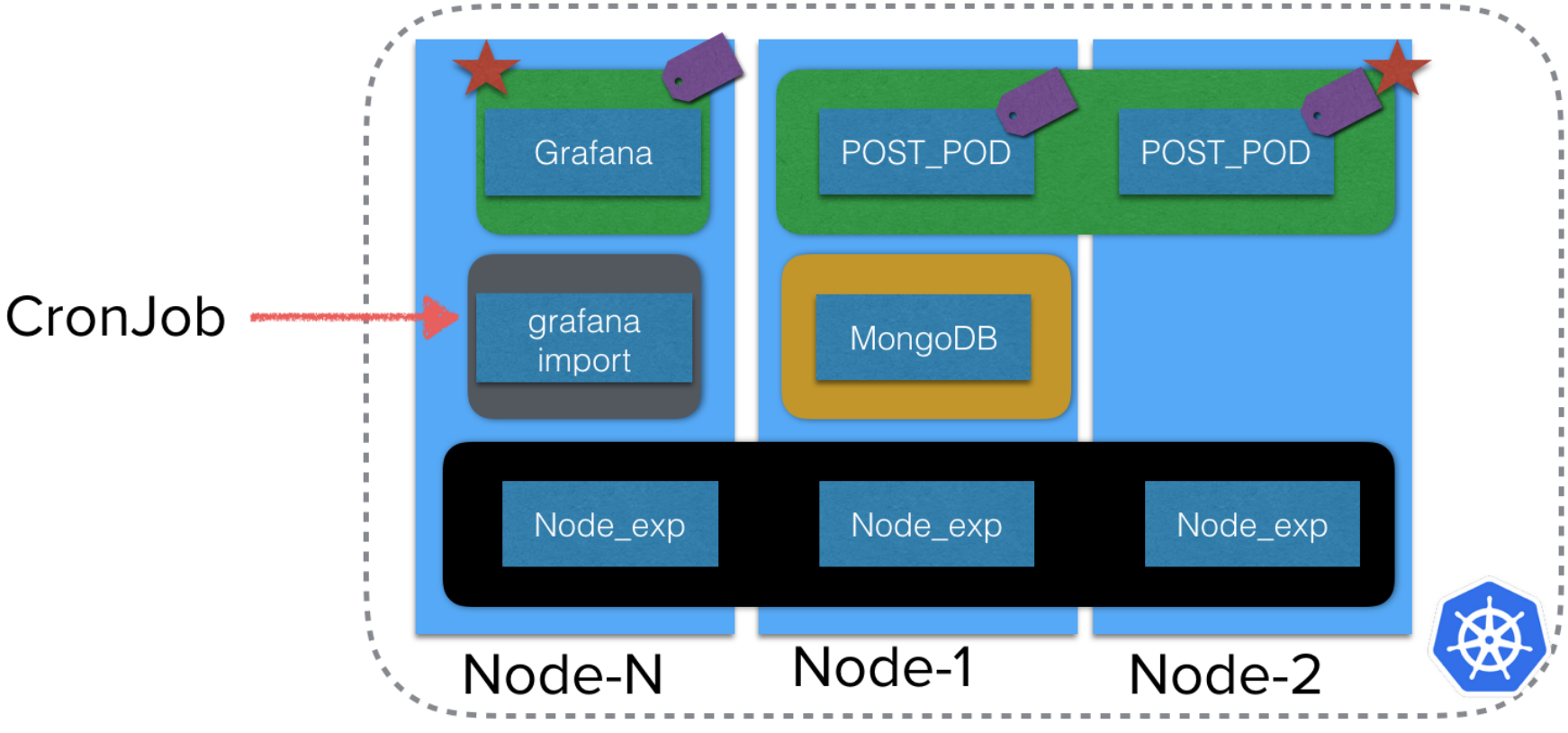
CronJob

- Запуск Job по расписанию в Cron:
  - Единожды в определенный момент времени
  - Периодически

CronJob

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: grafana-import-job
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: grafana-import
            image: busybox
            args:
            - curl http://admin:admin@grafana:3000/api/dashboards/import
          restartPolicy: OnFailure
```

CronJob


CronJob
Grafana
POST_POD
POST_POD
grafana import
MongoDB
Node_exp
Node_exp
Node_exp
Node-N
Node-1
Node-2

Полезные ссылки

- Безопасность в Kubernetes
- Страх и ненависть DevSecOps