

DevOps
Conf **2022**

МультиклUSTERНЫЙ масштабируемый Vault на тысячи сервисов – это ОК

Иван Буймов
Одноклассники



- ~10 лет в ИТ
- 4 года в ОК
- Занимаюсь автоматизацией
- Много пишу на Python
- Немного пишу на Go

На фото я чиню сервера во время летнего корпоратива на прошлом месте работы

Иван Буймов
Одноклассники

О чём сегодня поговорим

1

Инфраструктура
ОК

2

Интеграция с
нашими системами
и аутентификация

3

Отказ ДЦ и
масштабирование

4

Решение проблем,
что получилось

5

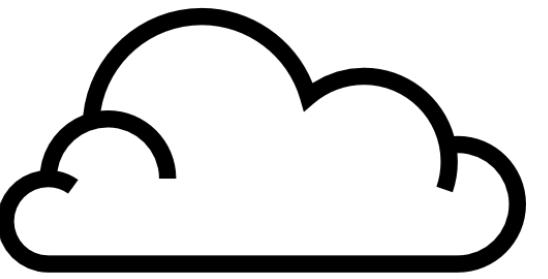
Вопросы

Одноклассники сегодня

о
х

Одноклассники сегодня

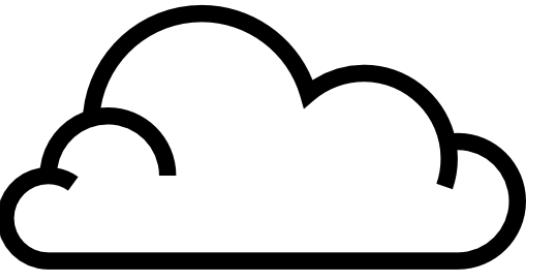
о
х



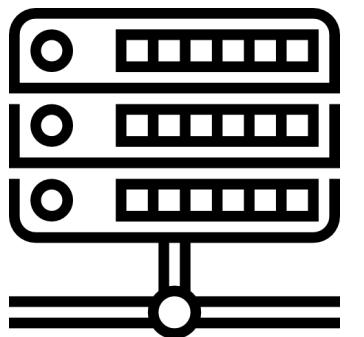
Десятки тысяч
контейнеров
в облаке one-cloud

Одноклассники сегодня

ОК

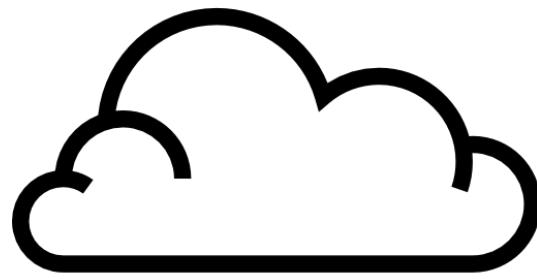


Десятки тысяч
контейнеров
в облаке one-cloud

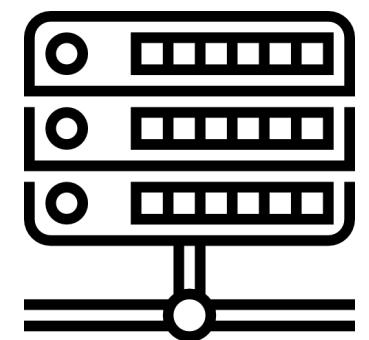


10k серверов

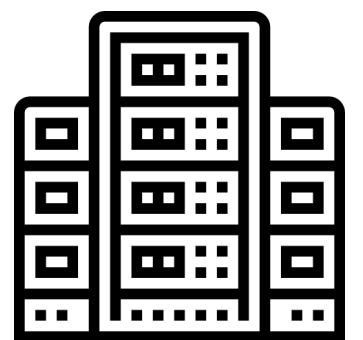
Одноклассники сегодня



Десятки тысяч
контейнеров
в облаке one-cloud



10k серверов

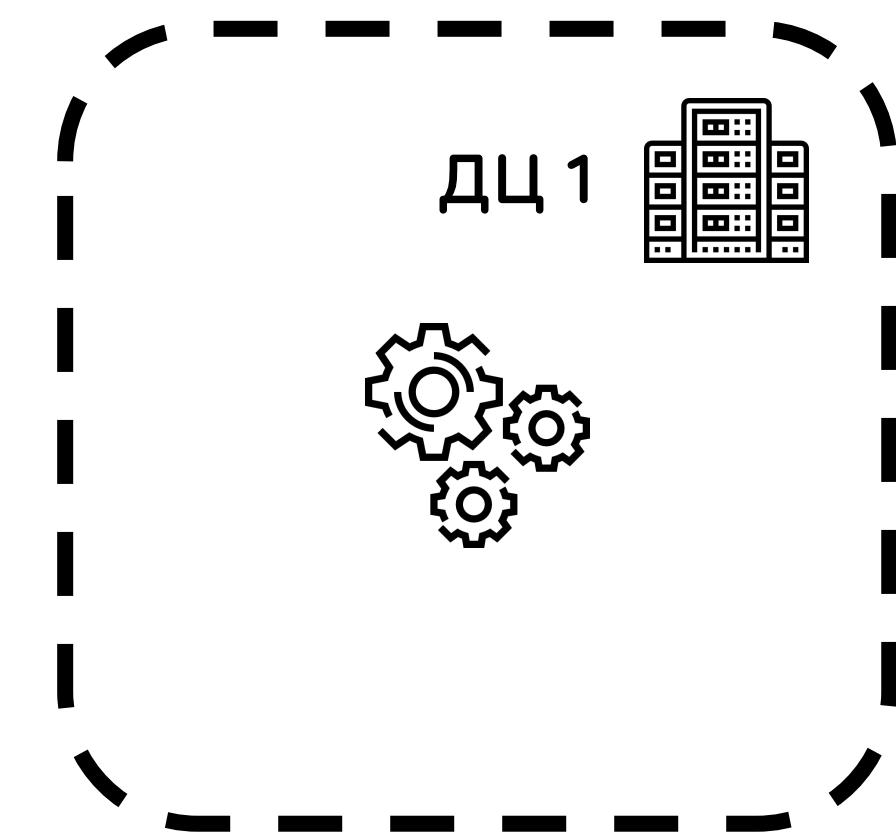


7 дата-центров

Наши стандарты

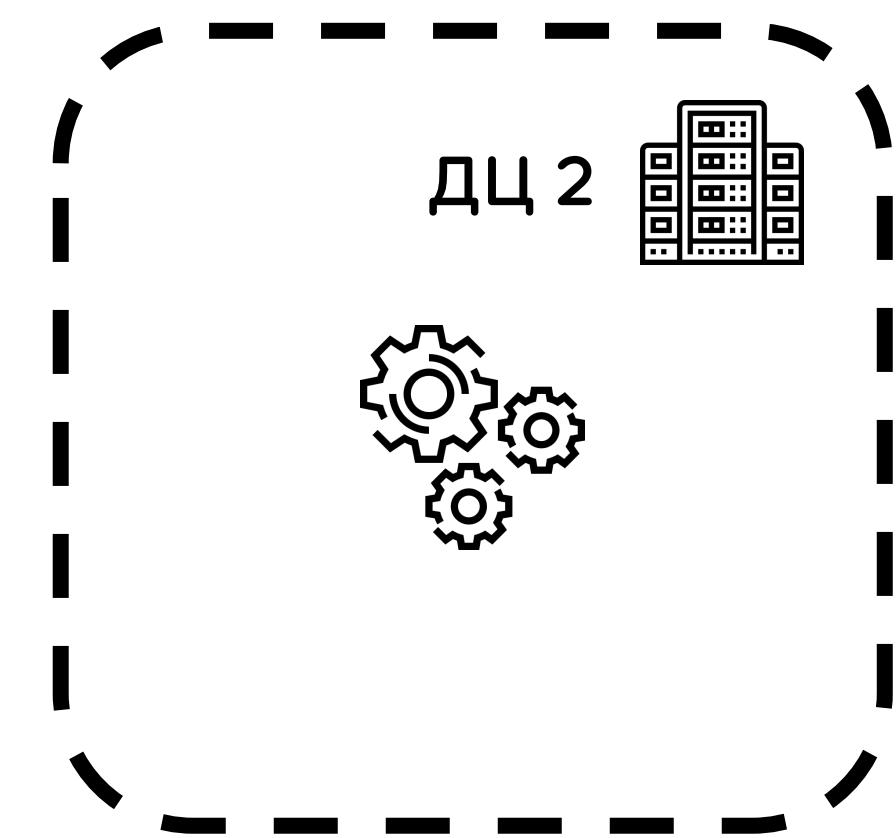
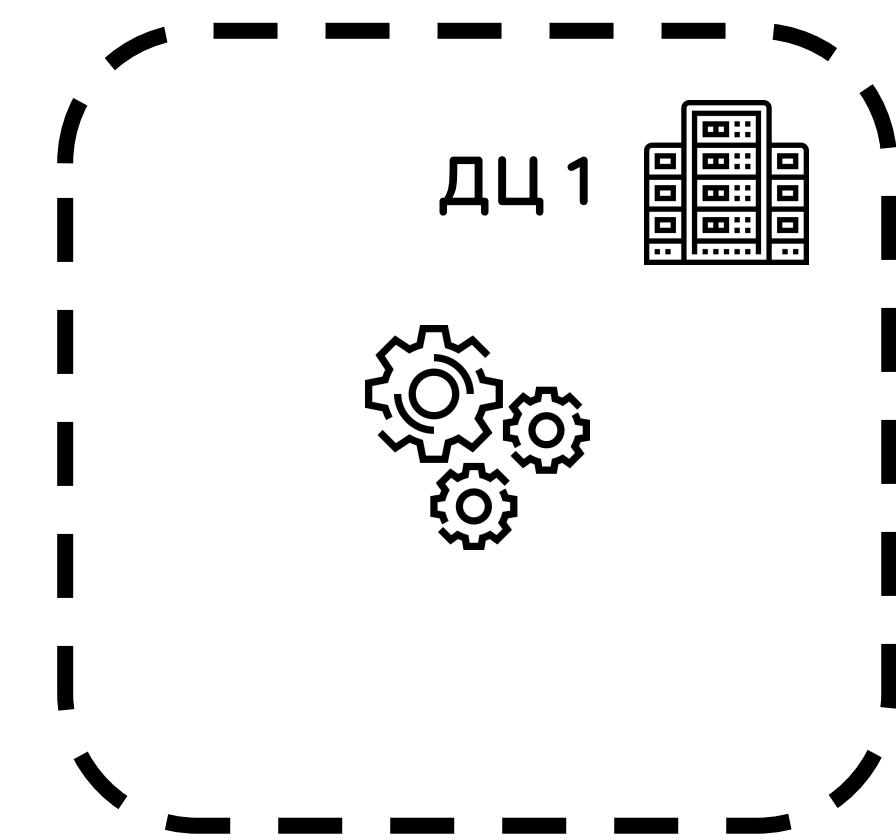
Наши стандарты

RF3 – обязательно



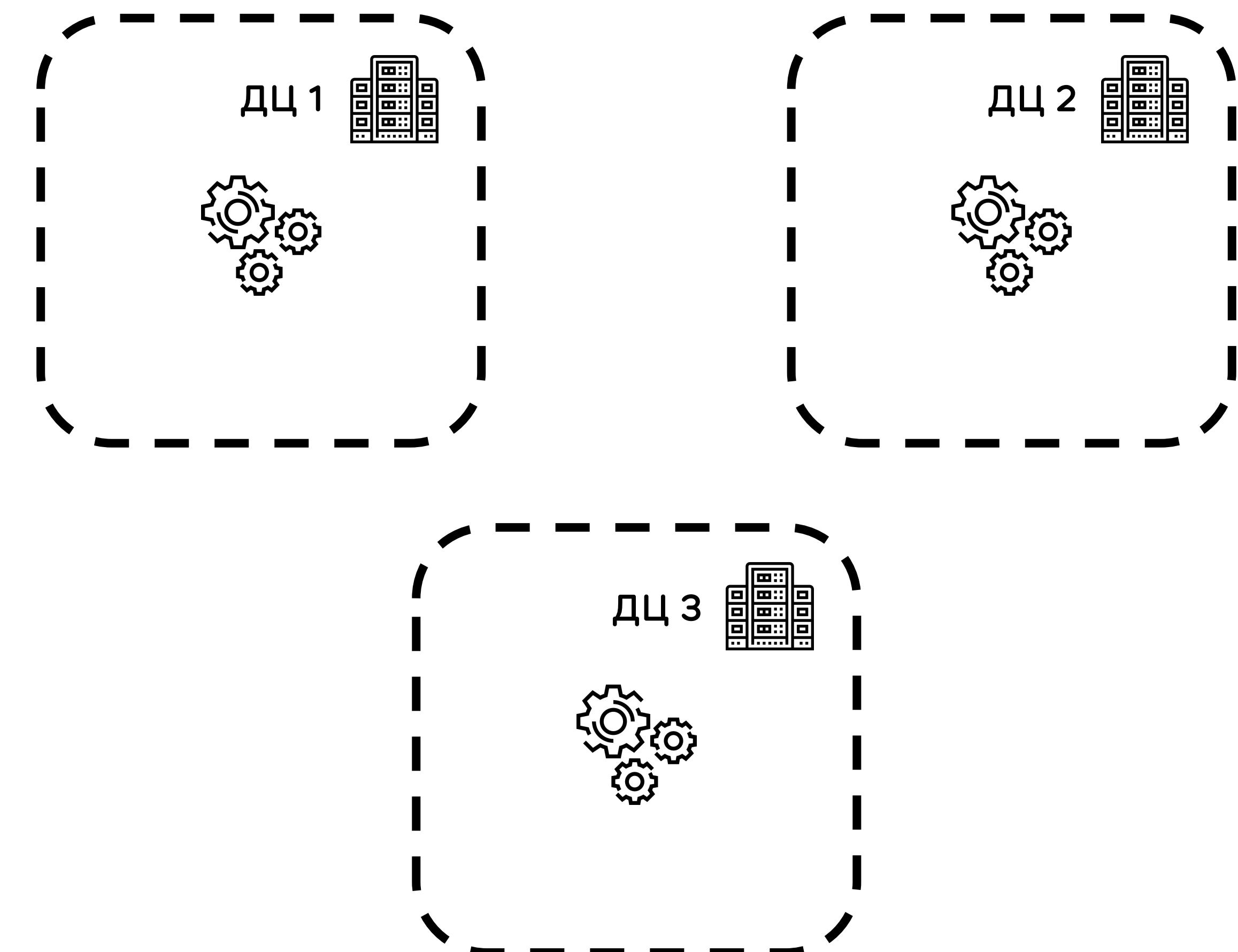
Наши стандарты

RF3 – обязательно



Наши стандарты

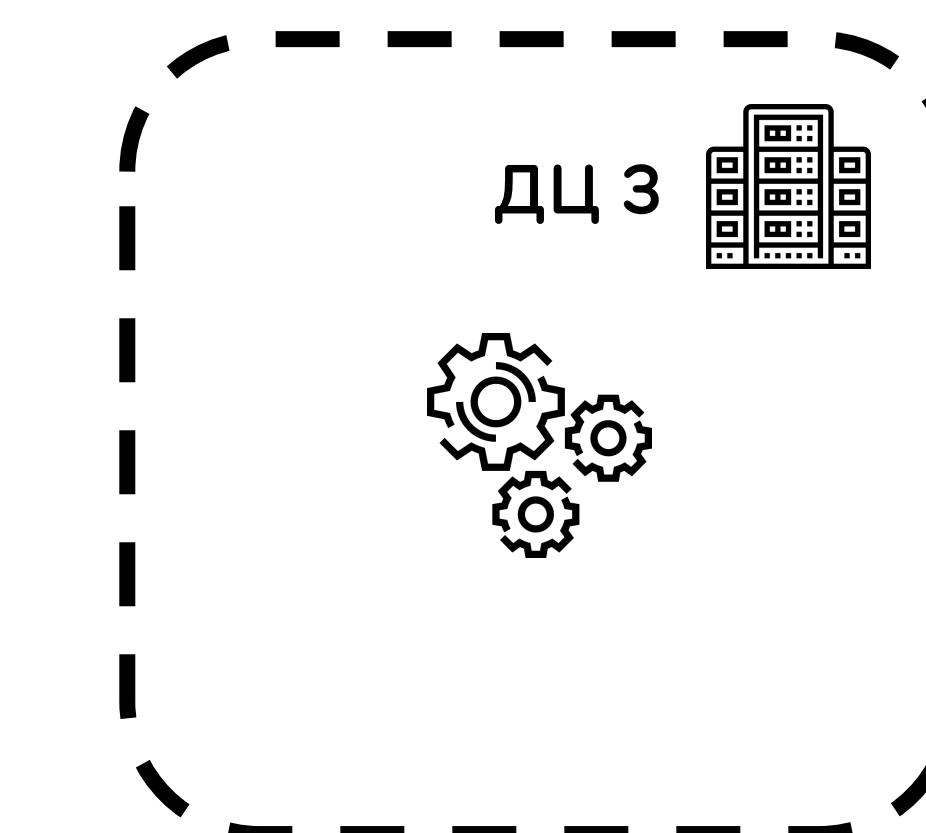
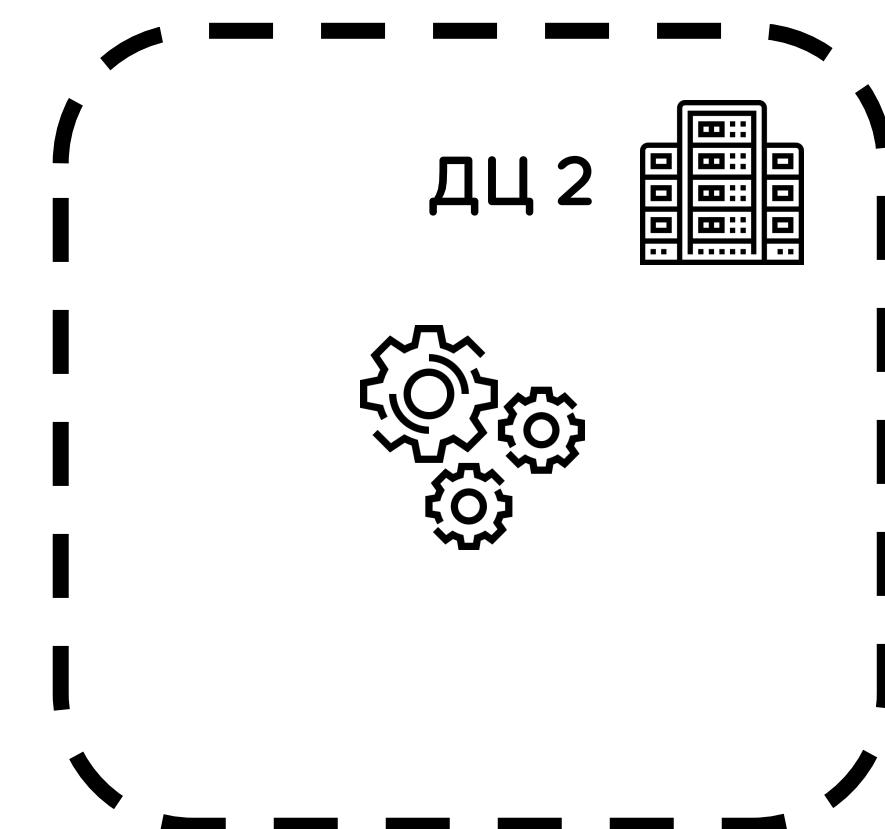
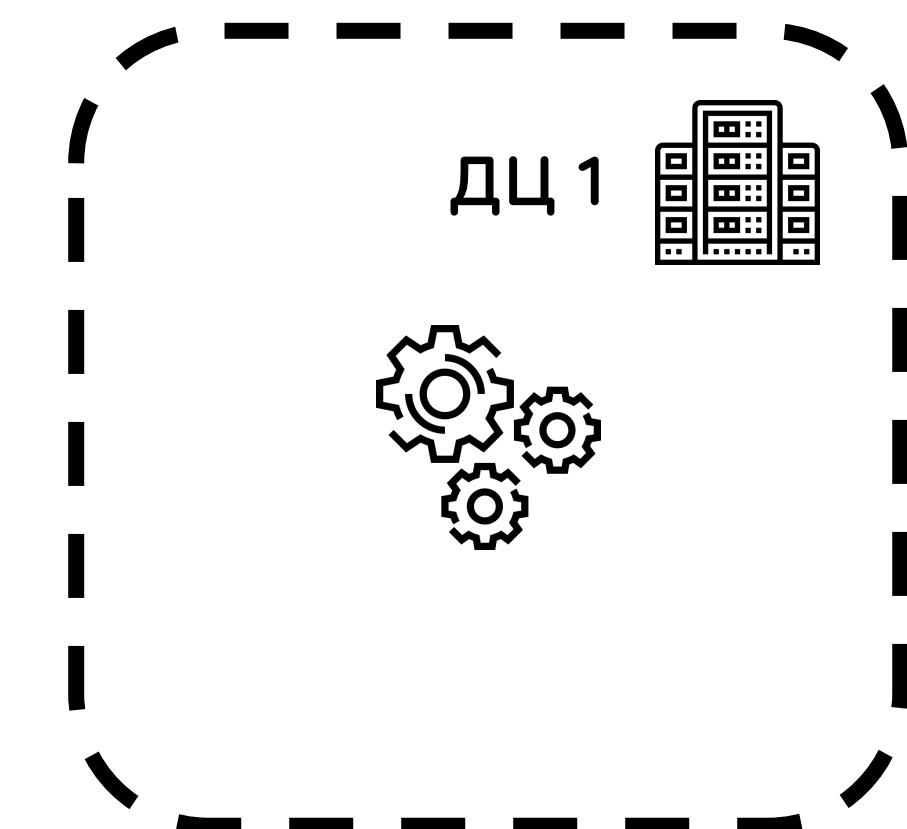
RF3 – обязательно



Наши стандарты

RF3 – обязательно

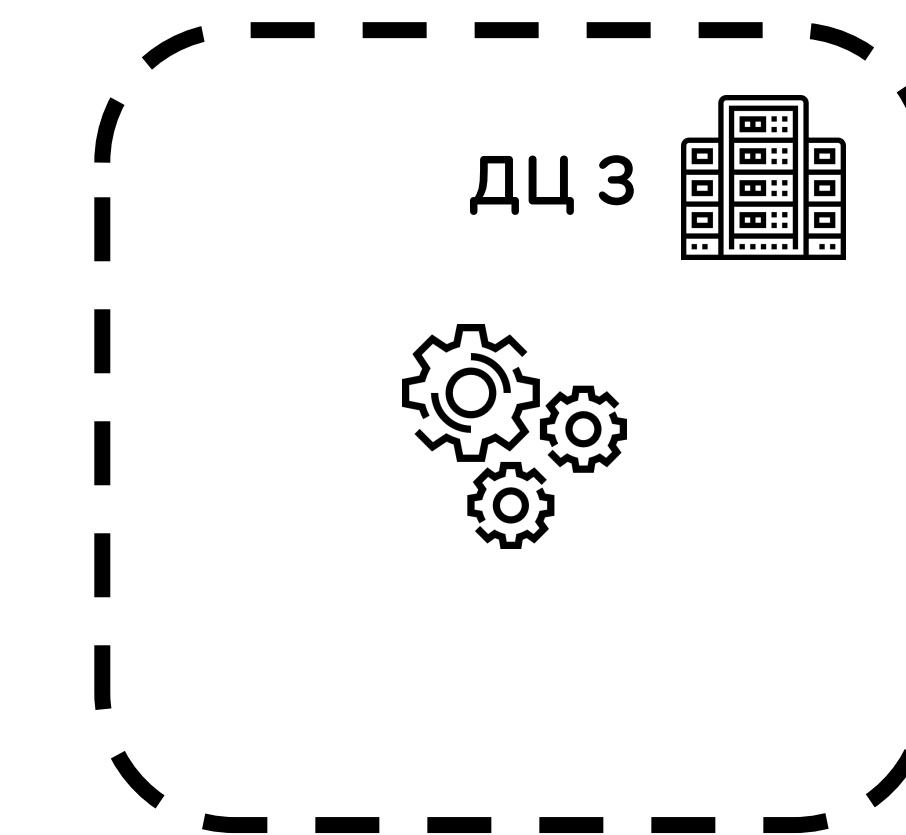
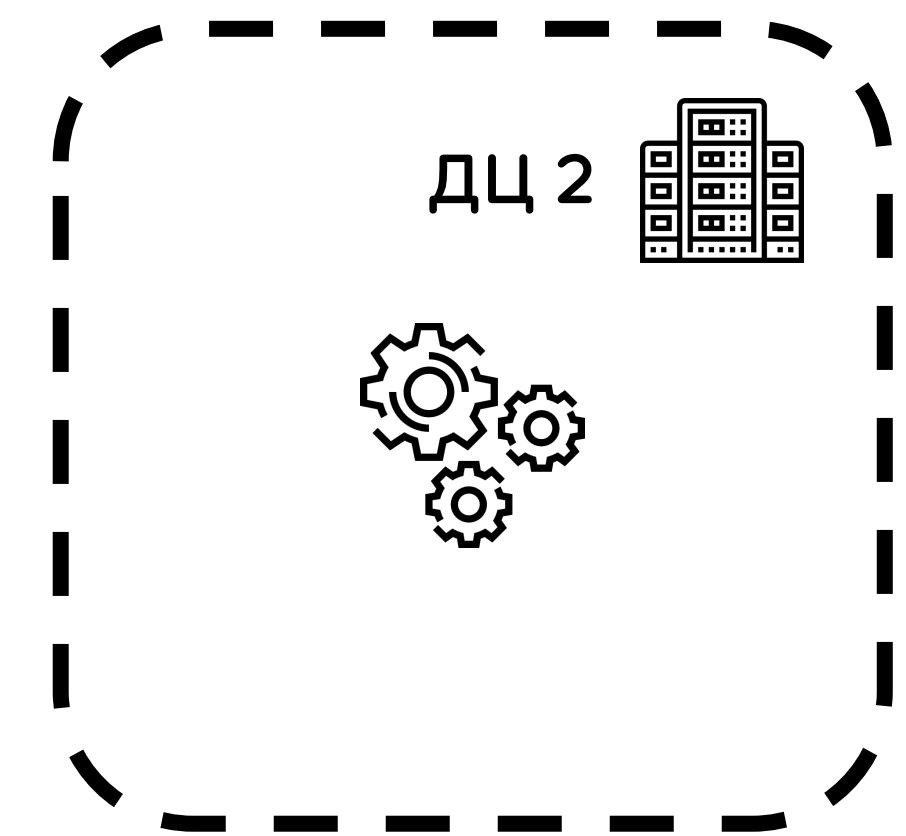
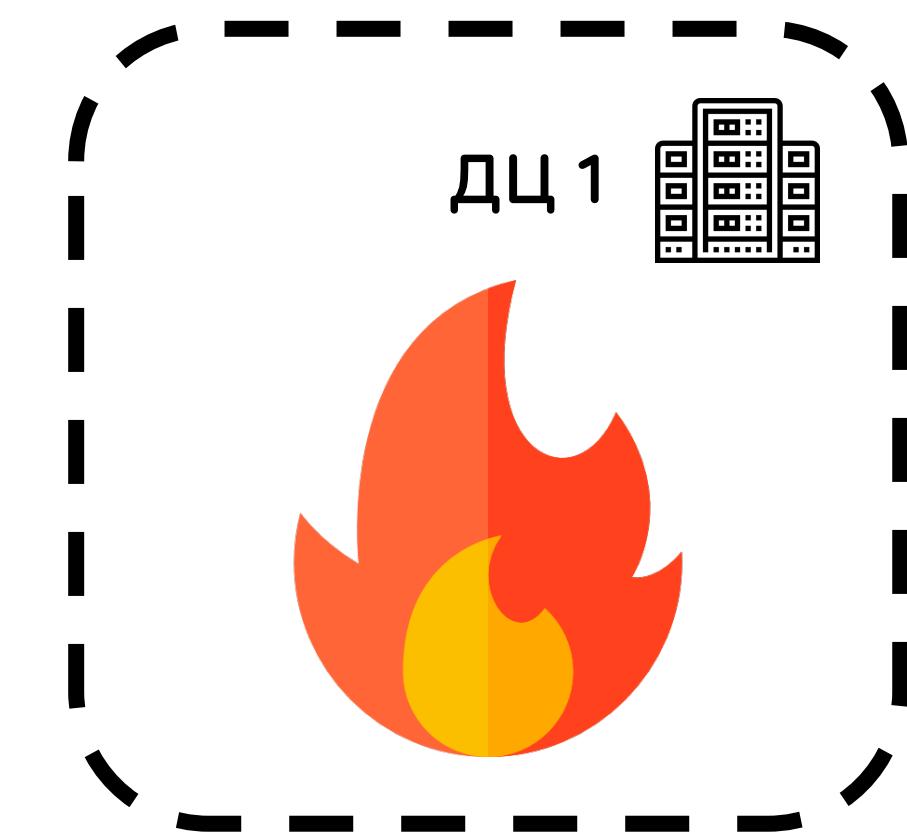
Отказ дата-центра



Наши стандарты

RF3 – обязательно

Отказ дата-центра

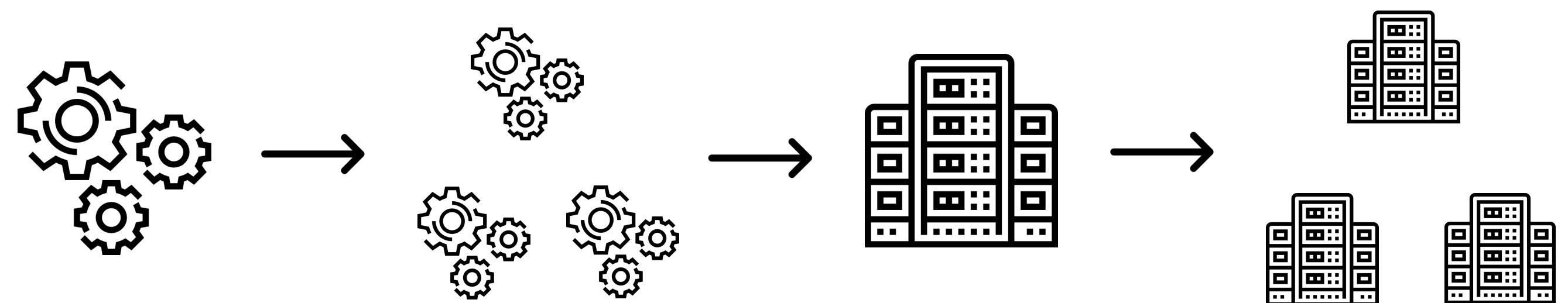


Наши стандарты

RF3 – обязательно

Отказ дата-центра

Плавное применение
изменений

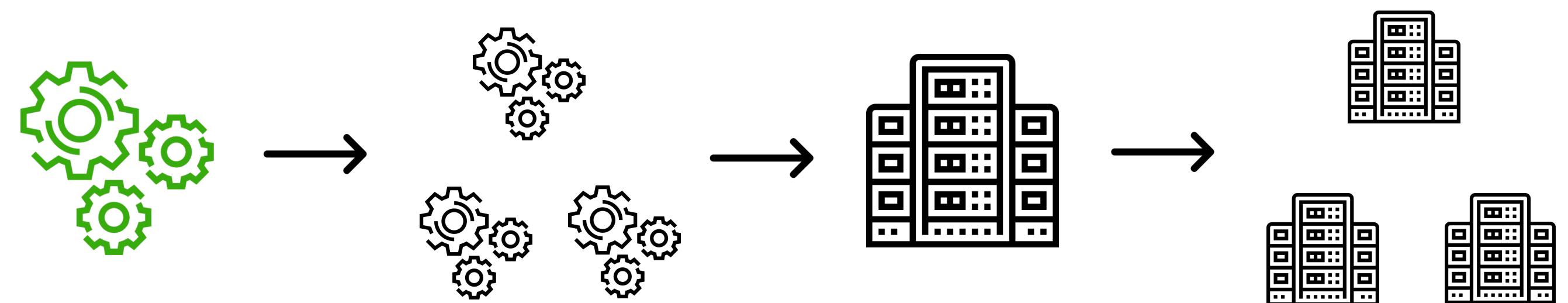


Наши стандарты

RF3 – обязательно

Отказ дата-центра

Плавное применение
изменений

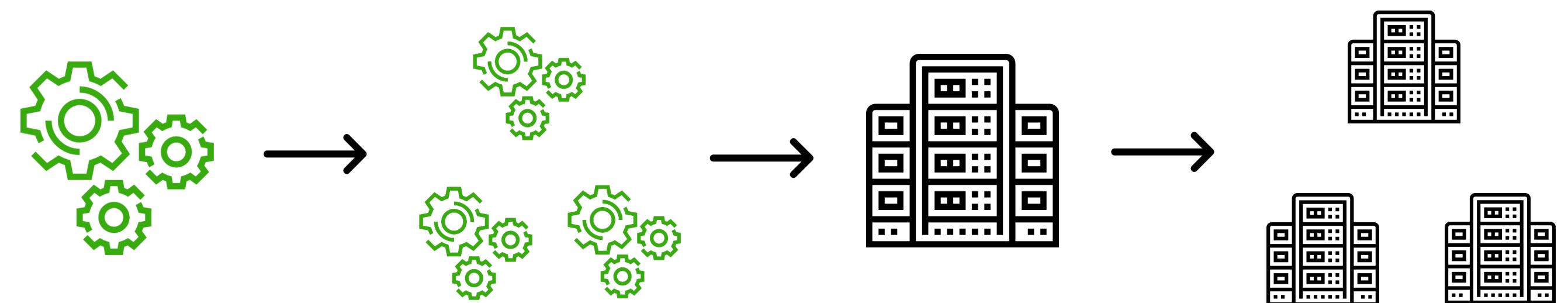


Наши стандарты

RF3 – обязательно

Отказ дата-центра

Плавное применение
изменений

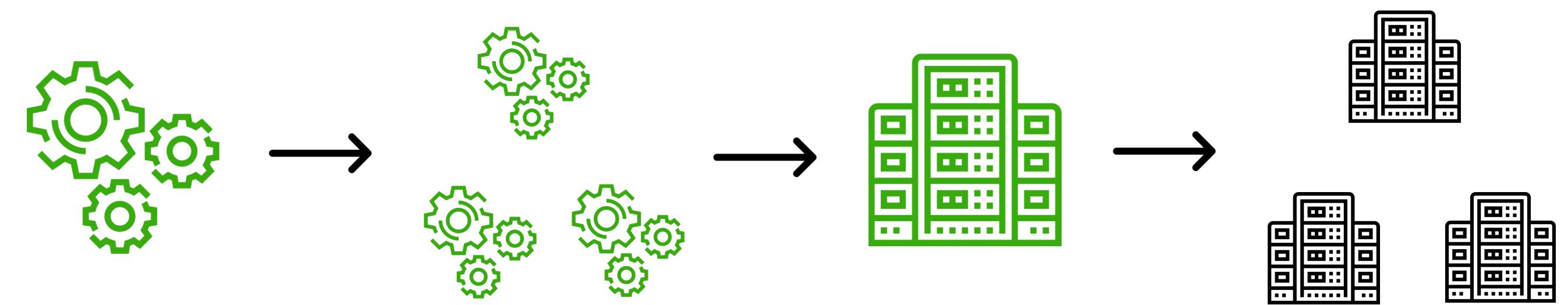


Наши стандарты

RF3 – обязательно

Отказ дата-центра

Плавное применение
изменений

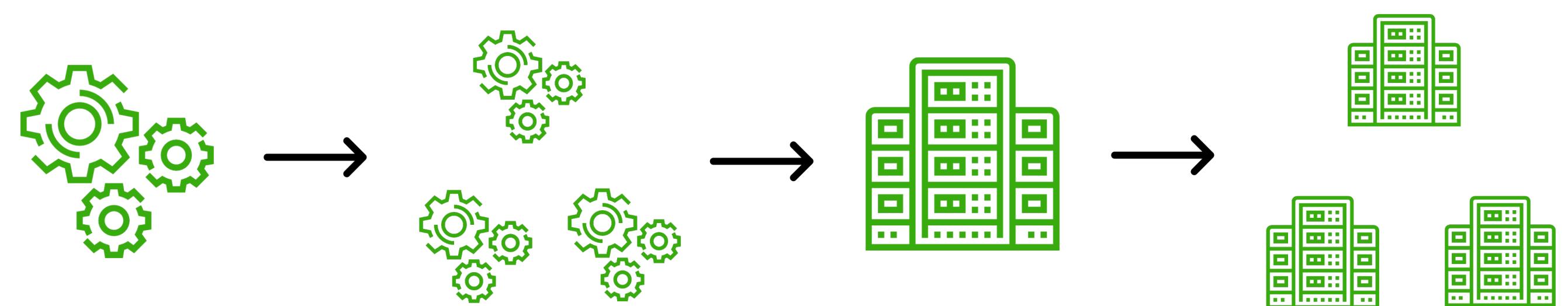


Наши стандарты

RF3 – обязательно

Отказ дата-центра

Плавное применение
изменений



Требования

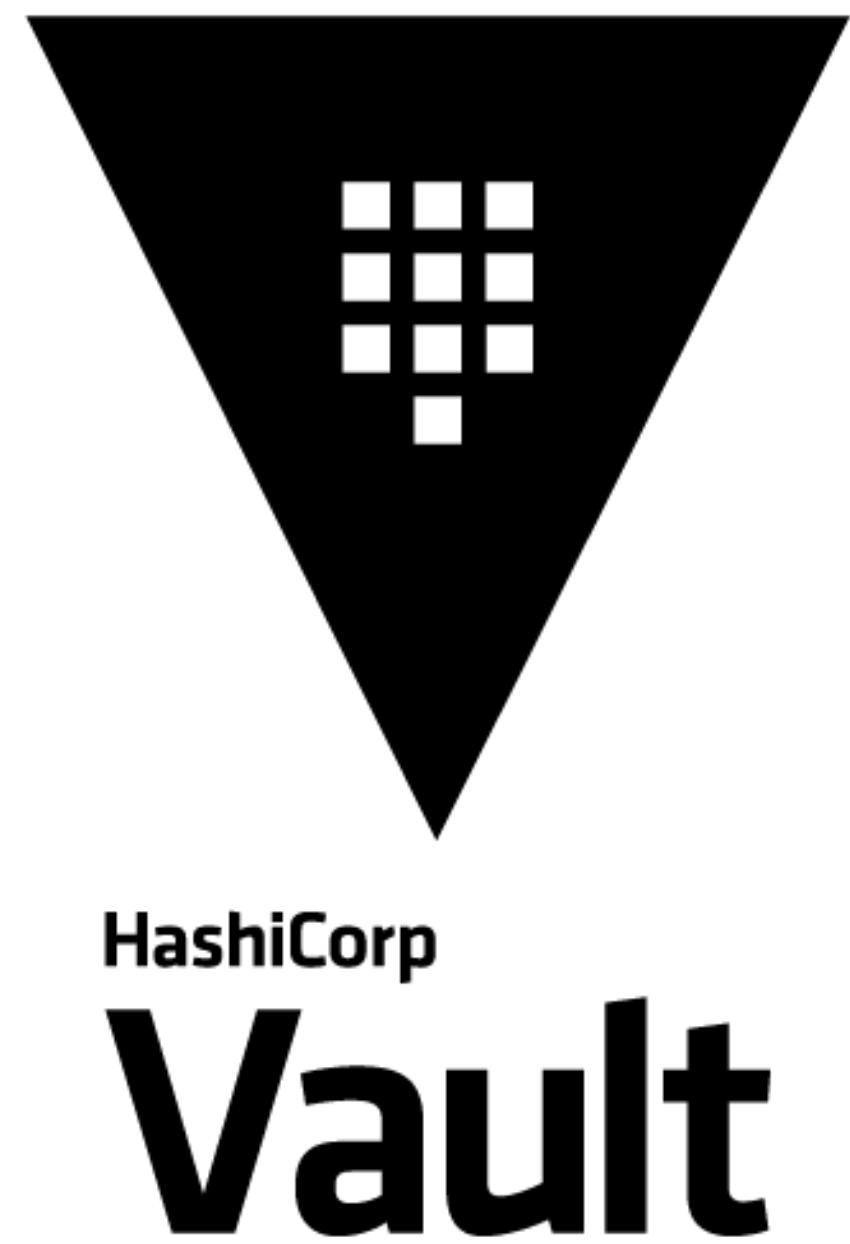


- Шифрование
- Аудит
- Удобный доступ для людей и машин (UI, API, cli)
- Версионирование секретов
- Интеграция с нашими системами: CMDB, CM
(Ansible CFEngine), one-cloud
- Наши стандарты

HashiCorp Vault

ХО

- Де-факто является стандартом в индустрии
- Поддержка в стороннем ПО (TeamCity, Ansible)
- OpenSource
- Web UI, REST API, cli, libs





Отлично! Но чего-то
не хватает...



- Интеграция с нашими системами
- Выписка сертификатов
- Отказ дата-центра
- Масштабирование

ХО

Что нужно для чтения секрета?

ХО

Vault-токен

Что нужно для чтения секрета?



Vault-токен

Как получить токен?

Что нужно для чтения секрета?



Vault-токен

Как получить токен?

Пройти аутентификацию в Vault

Что нужно для чтения секрета?



Vault-токен

Как получить токен?

Пройти аутентификацию в Vault

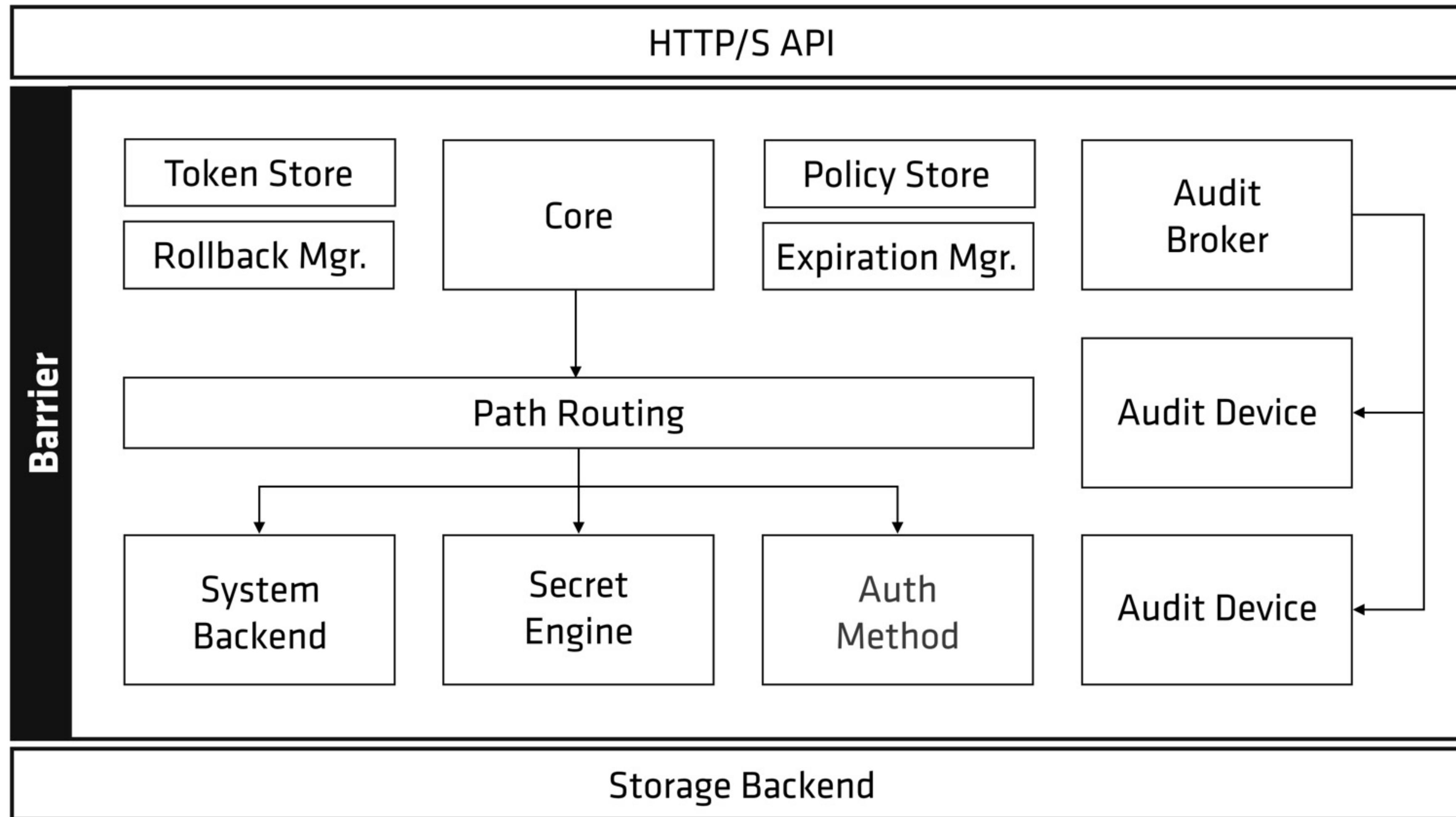
Как пройти аутентификацию?

Секрет для доступа к секретам

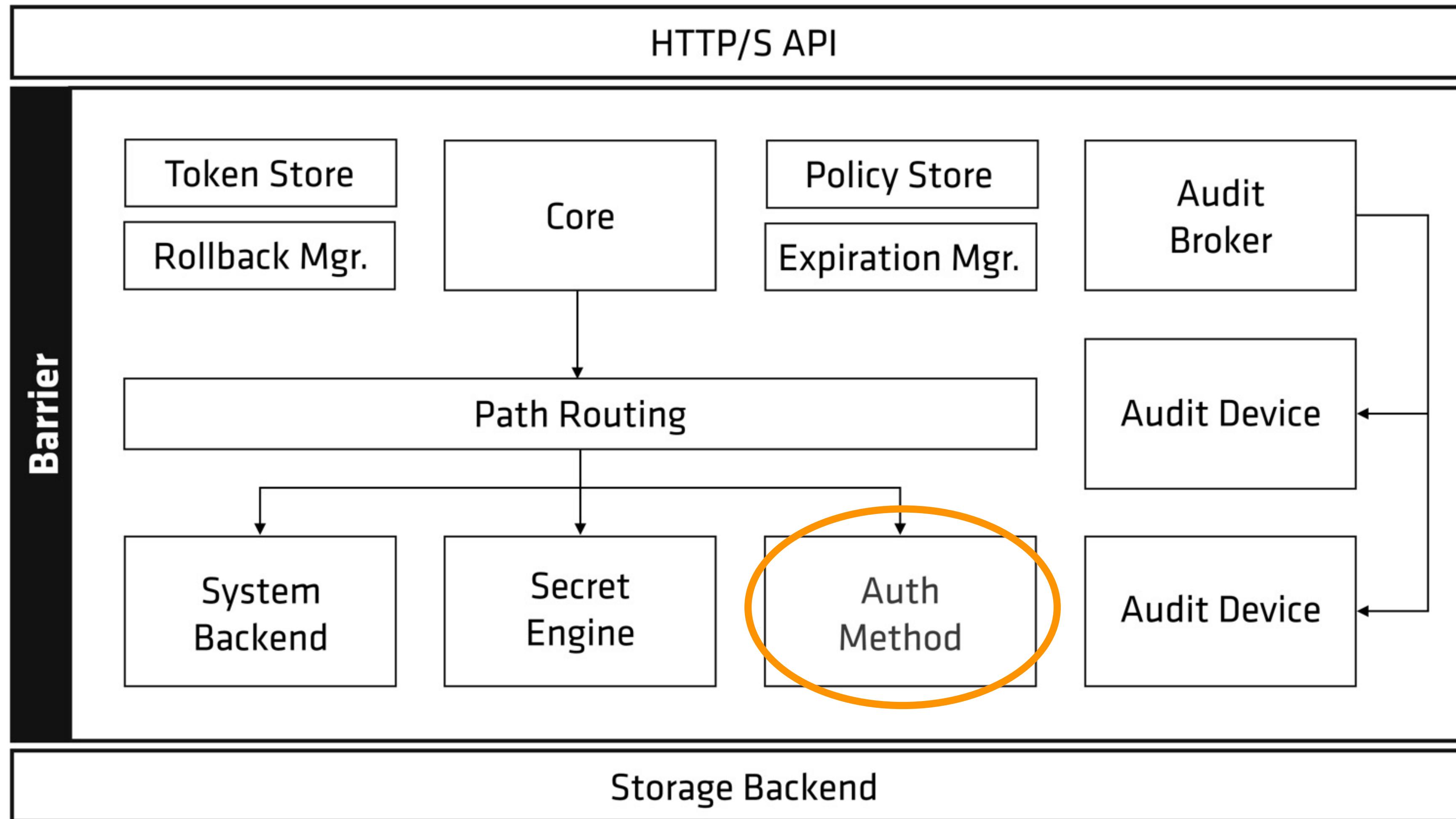
- Нужен доверенный источник, обычно это системы **CI/CD** или **Configuration Management**



Архитектура Vault



Архитектура Vault



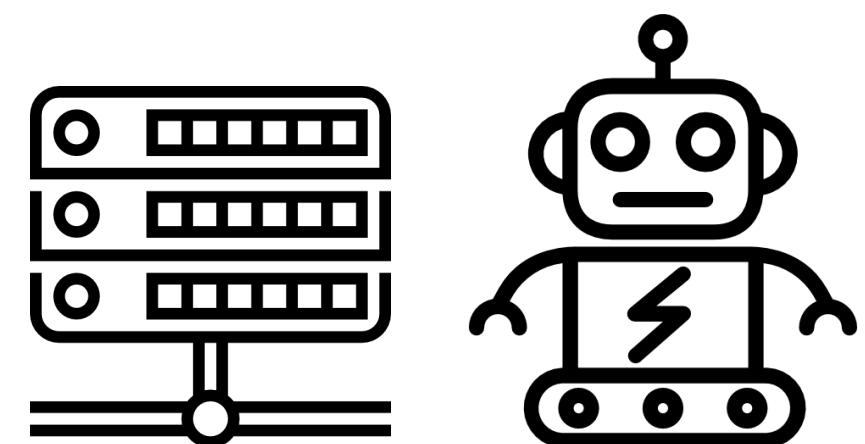
Как пройти аутентификацию?

ХО



Как пройти аутентификацию?

ХО

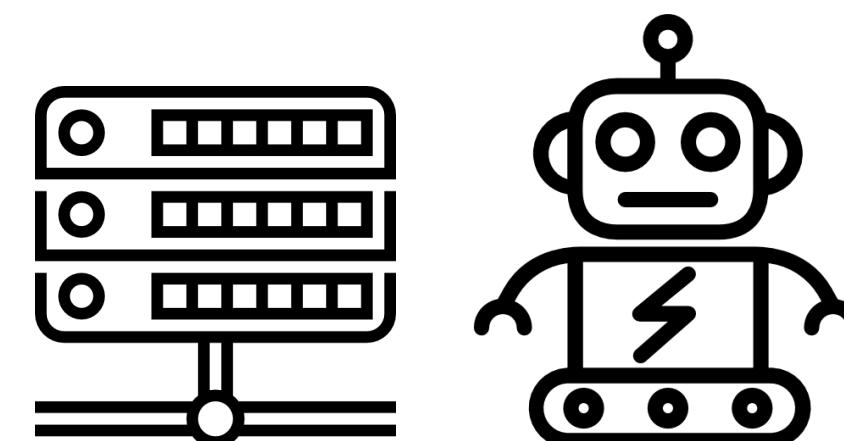


Как пройти аутентификацию?

ХО



LDAP

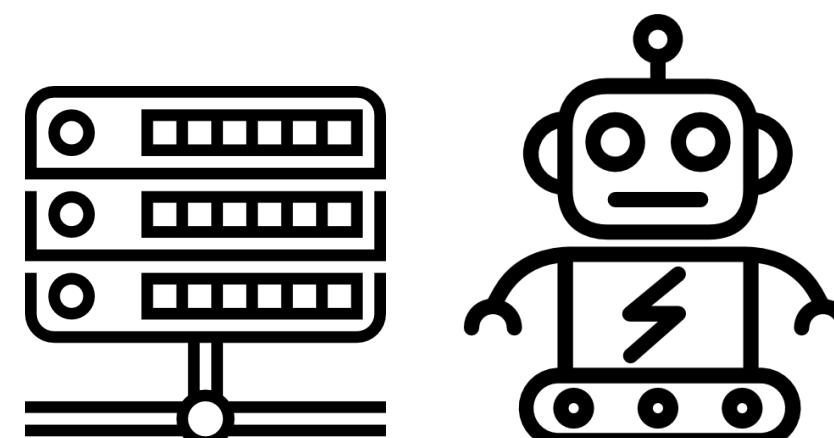


Как пройти аутентификацию?

Х



LDAP



?



Auth Methods

Overview

AppRole

AliCloud

AWS

Azure

Cloud Foundry

GitHub

Google Cloud

JWT/OIDC

Kerberos

Kubernetes

LDAP

Login MFA

Oracle Cloud Infrastructure

Okta

RADIUS

TLS Certificates

Tokens

Username & Password

Kubernetes

AliCloud

LDAP

TLS Certificates

AWS

Cloud Foundry

GitHub

Google Cloud

JWT/OIDC

Tokens

Username & Password

Oracle Cloud Infrastructure

AppRole

RADIUS



Auth Methods

Overview

AppRole

AliCloud

AWS

Azure

Cloud Foundry

GitHub

Google Cloud

JWT/OIDC

Kerberos

Kubernetes

LDAP

Login MFA

Oracle Cloud Infrastructure

Okta

RADIUS

TLS Certificates

Tokens

Username & Password

Kubernetes

AliCloud

LDAP

Tokens

TLS Certificates

AWS

Cloud Foundry

GitHub

Google Cloud

Username & Password

Oracle Cloud Infrastructure

AppRole

JWT/OIDC

RADIUS

JWT: json web token

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
JoZWxsbyI6IndvcmxkIiwiZGF0YSI6InNvbWUgZ  
GF0YSIsImZvbyI6ImJhciJ9.0mtc-  
LssSGdoVj9Tzx1F2J3f7-guDvgWaGaKtGB3D_U
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "hello": "world",  
  "data": "some data",  
  "foo": "bar"  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  very-secure-key  
)  secret base64 encoded
```

JWT: json web token

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
JoZWxsbyI6IndvcmxkIiwiZGF0YSI6InNvbWUgZ  
GF0YSIsImZvbyI6ImJhciJ9.0mtc-  
LssSGdoVj9Tzx1F2J3f7-guDvgWaGaKtGB3D_U
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

PAYOUT: DATA

```
{  
  "hello": "world",  
  "data": "some data",  
  "foo": "bar"  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  very-secure-key  
)  secret base64 encoded
```

Заголовок

JWT: json web token

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
JoZWxsbyI6IndvcmxkIiwiZGF0YSI6InNvbWUgZ  
GF0YSIsImZvbyI6ImJhciJ9.0mtc-  
LssSGdoVj9Tzx1F2J3f7-guDvgWaGaKtGB3D_U
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Заголовок

PAYOUT: DATA

```
{  
  "hello": "world",  
  "data": "some data",  
  "foo": "bar"  
}
```

Данные (payload)

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  very-secure-key  
)  secret base64 encoded
```

JWT: json web token

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ  
JoZWxsbyI6IndvcmxkIiwiZGF0YSI6InNvbWUgZ  
GF0YSIsImZvbyI6ImJhciJ9.0mtc-  
LssSGdoVj9Tzx1F2J3f7-guDvgWaGaKtGB3D_U
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```

Заголовок

PAYOUT: DATA

```
{  
  "hello": "world",  
  "data": "some data",  
  "foo": "bar"  
}
```

Данные (payload)

VERIFY SIGNATURE

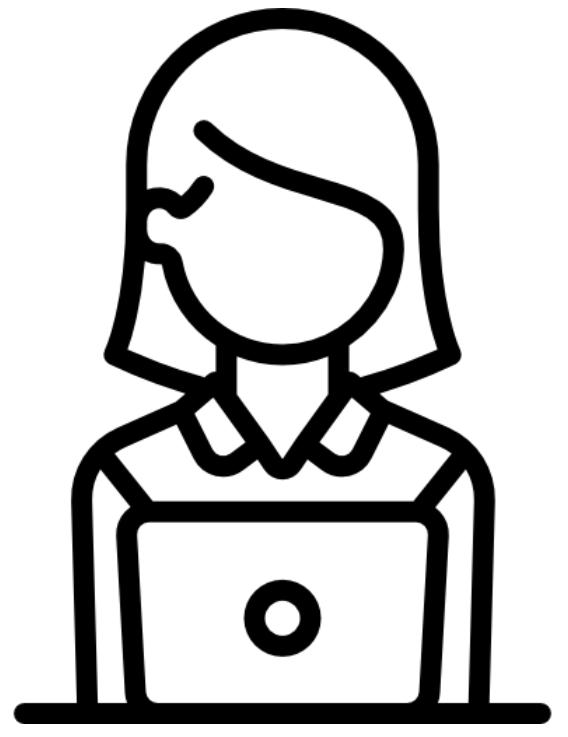
```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  very-secure-key  
)  secret base64 encoded
```

Подпись

JWT: json web token

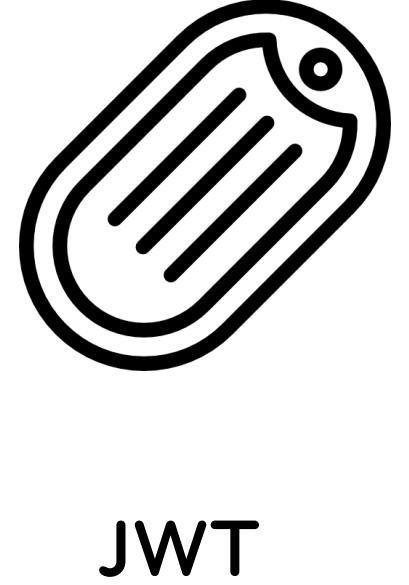


Алиса



write data

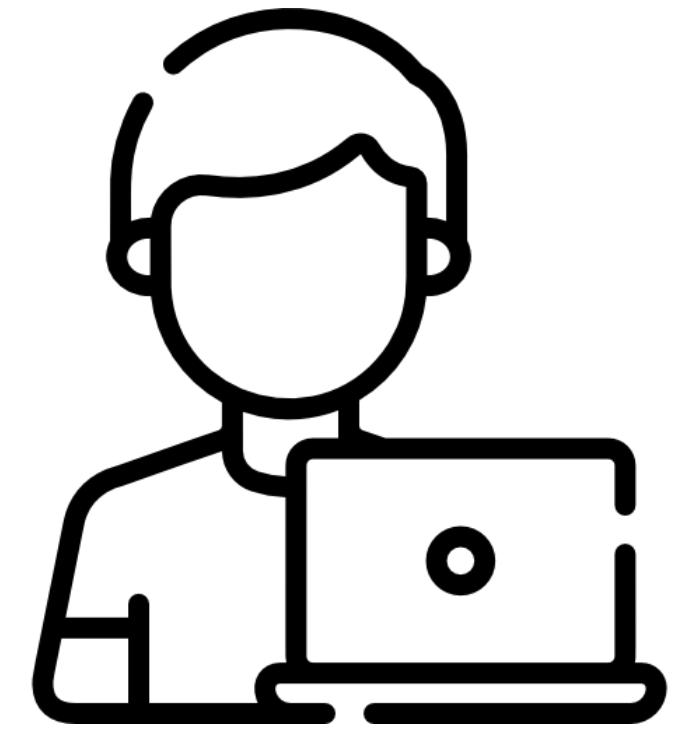
sign with **private key**



read data

check sign with **public key**

Боб

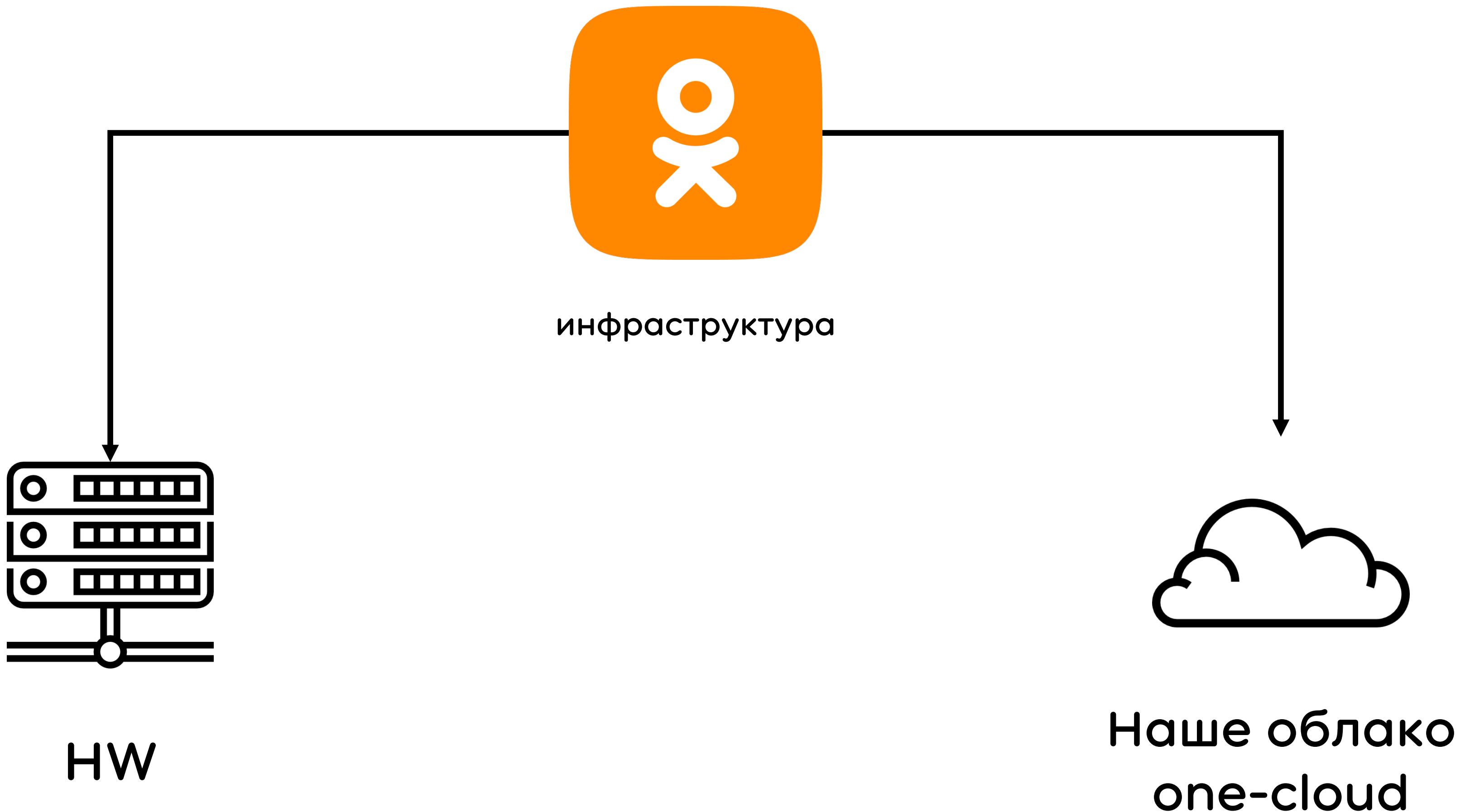


```
{  
  "data": {  
    "name": "Ivan",  
    "Age": 31  
  }  
}
```

```
{  
  "data": {  
    "name": "Ivan",  
    "Age": 31  
  }  
}
```



Посмотрим по сторонам



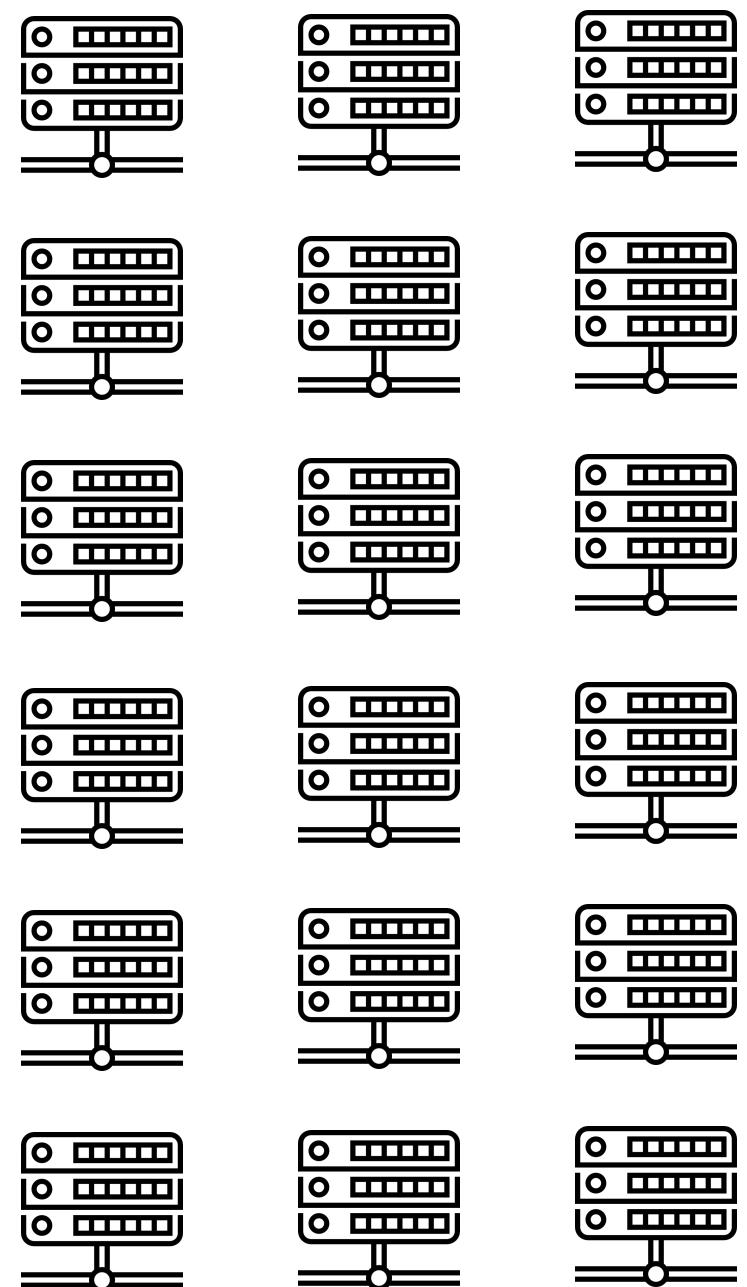
Посмотрим по сторонам: HW

ХО

HW

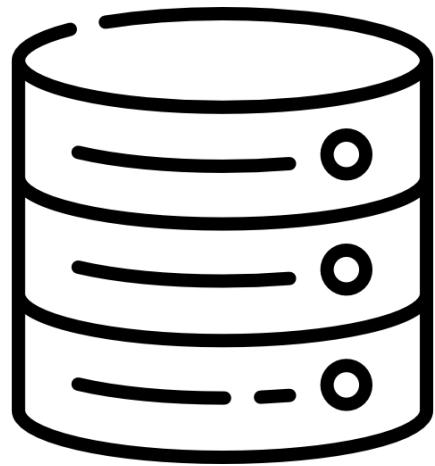
Посмотрим по сторонам: HW

HW

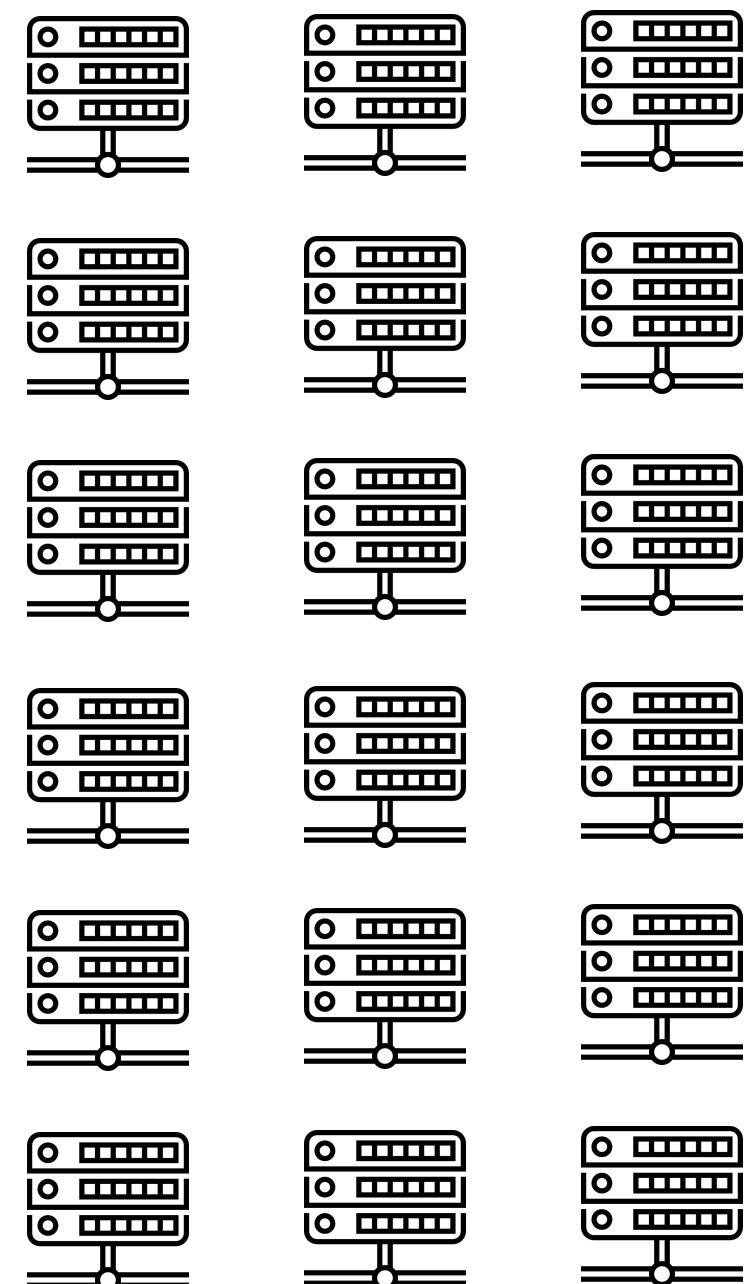


Посмотрим по сторонам: HW

HW



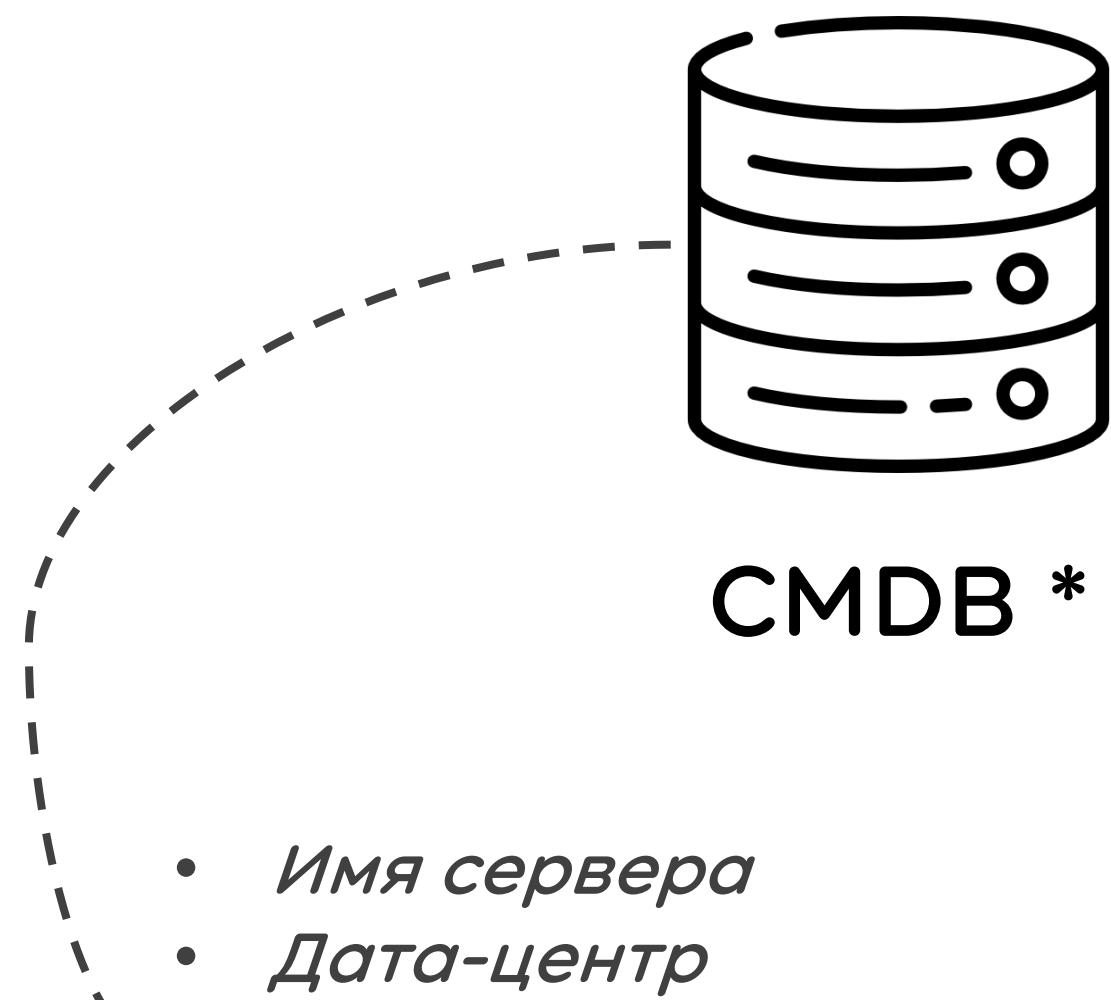
CMDB *



* Configuration management database

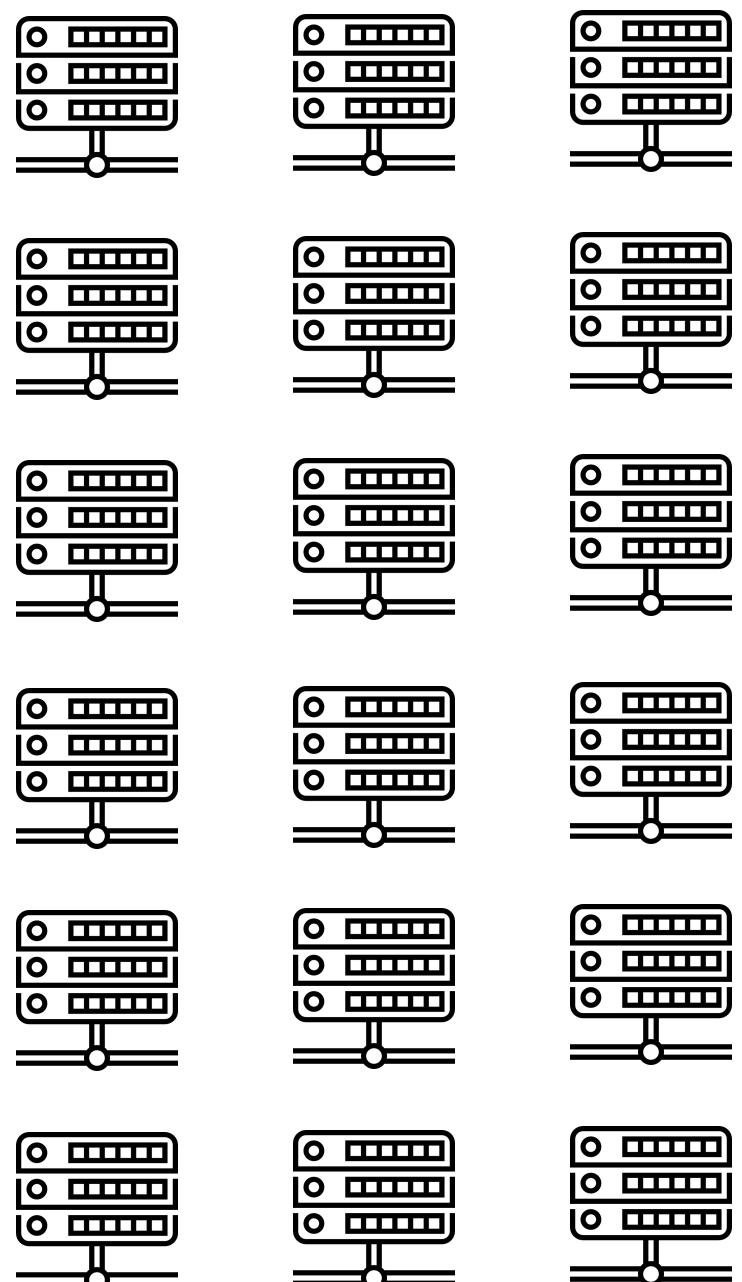
Посмотрим по сторонам: HW

HW



CMDB *

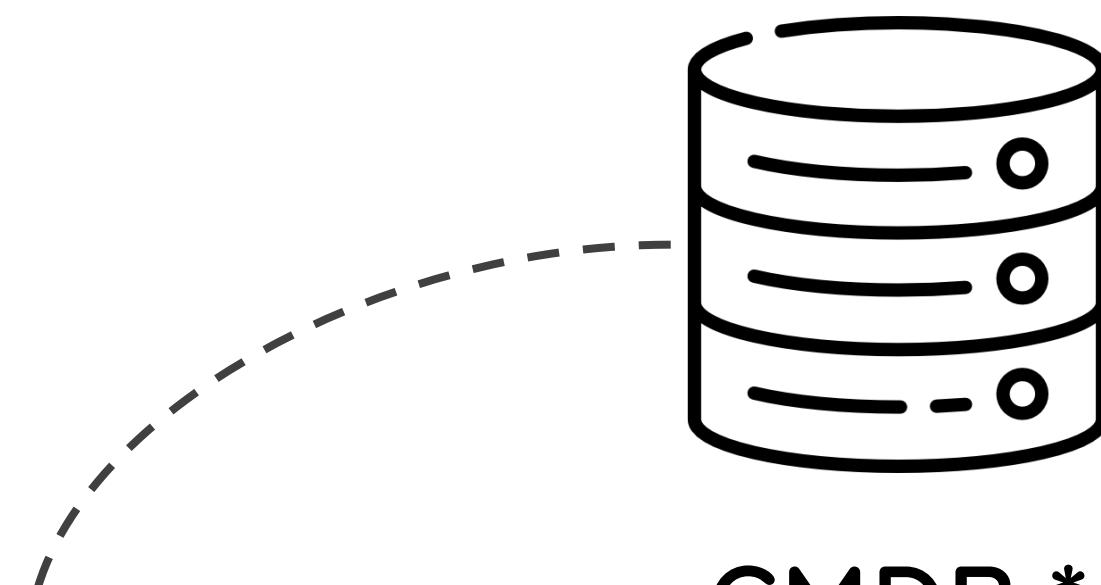
- Имя сервера
- Дата-центр
- Настройки сети
- Группа (*cloud-minion, cdb, video-download*)



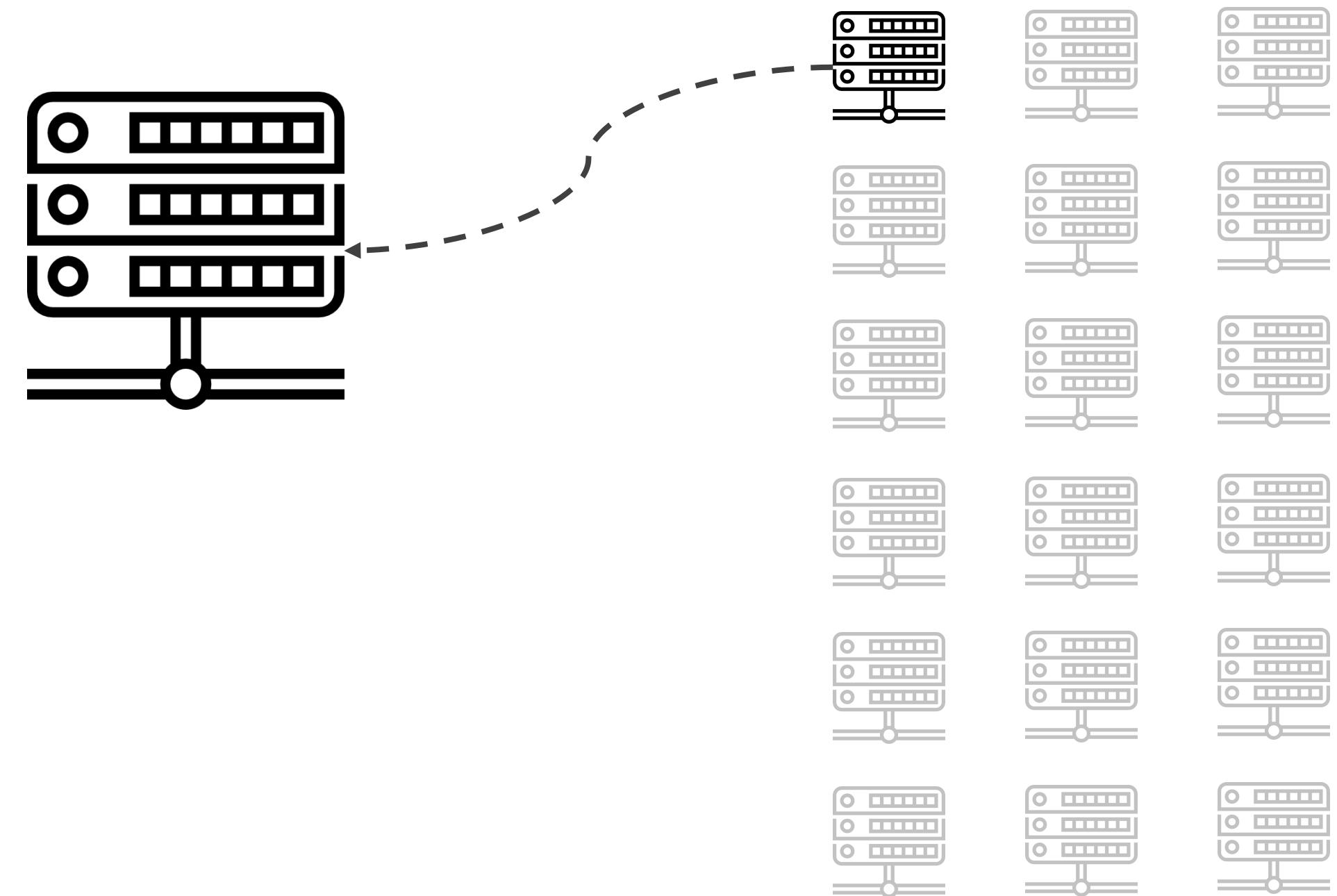
* Configuration management database

Посмотрим по сторонам: HW

HW

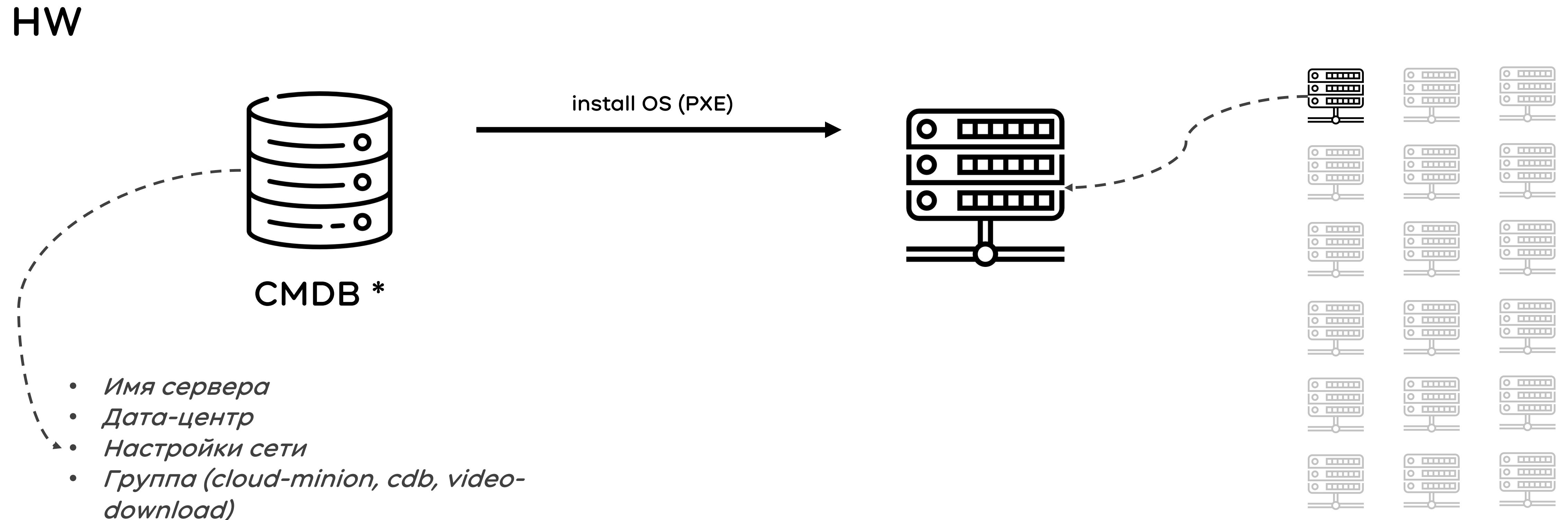


- Имя сервера
- Дата-центр
- Настройки сети
- Группа (*cloud-minion, cdb, video-download*)



* Configuration management database

Посмотрим по сторонам: HW

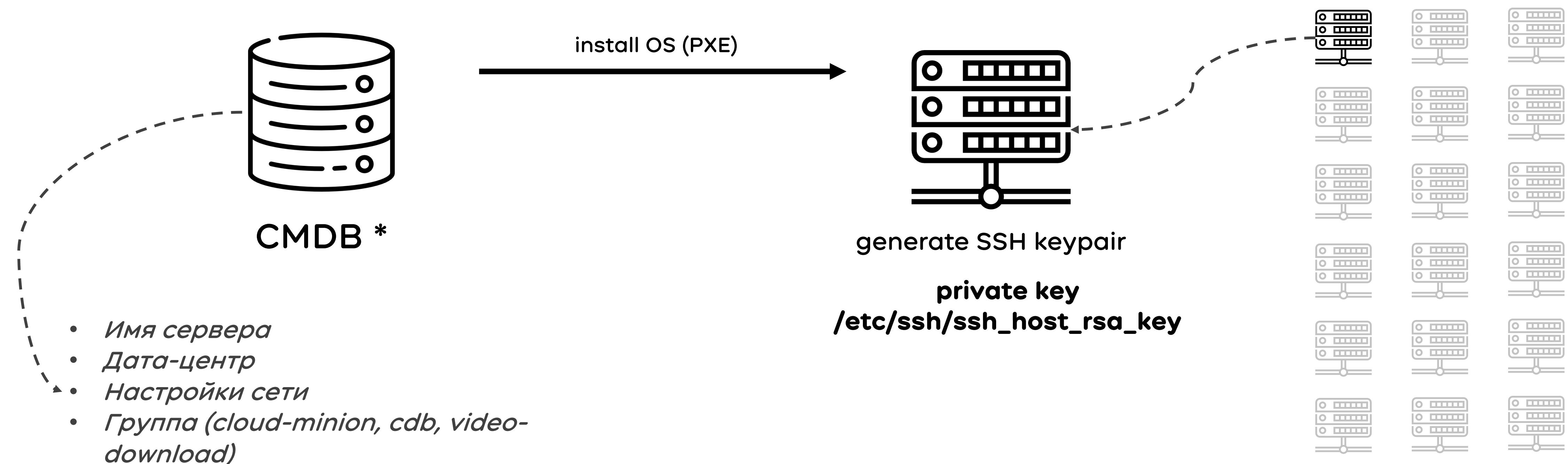


* Configuration management database

Посмотрим по сторонам: HW



HW

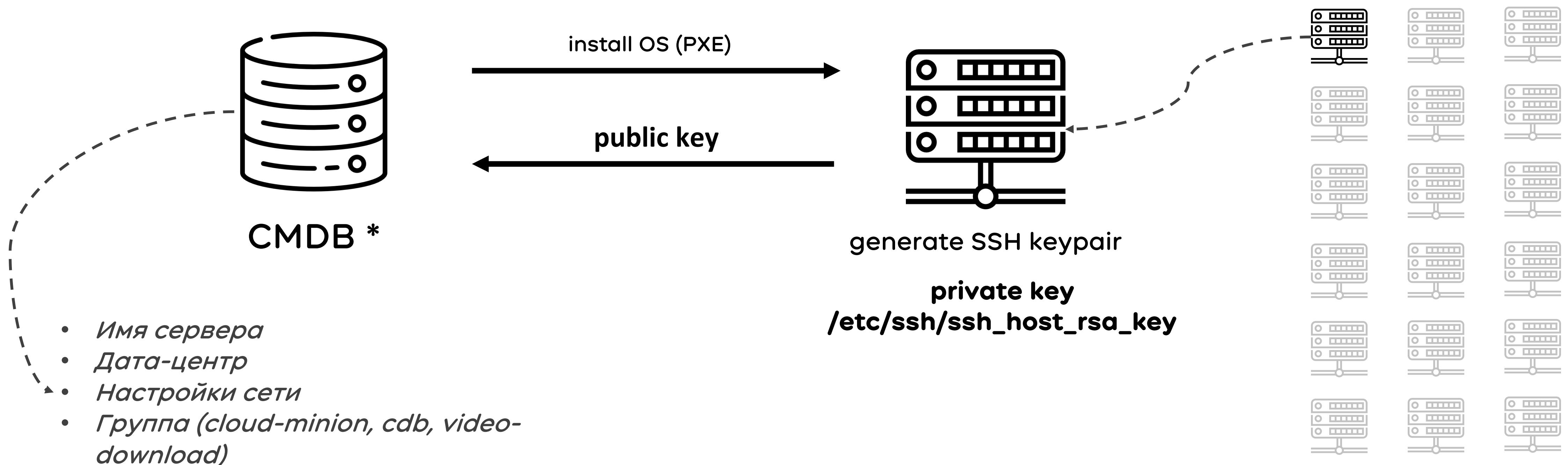


* Configuration management database

Посмотрим по сторонам: HW



HW



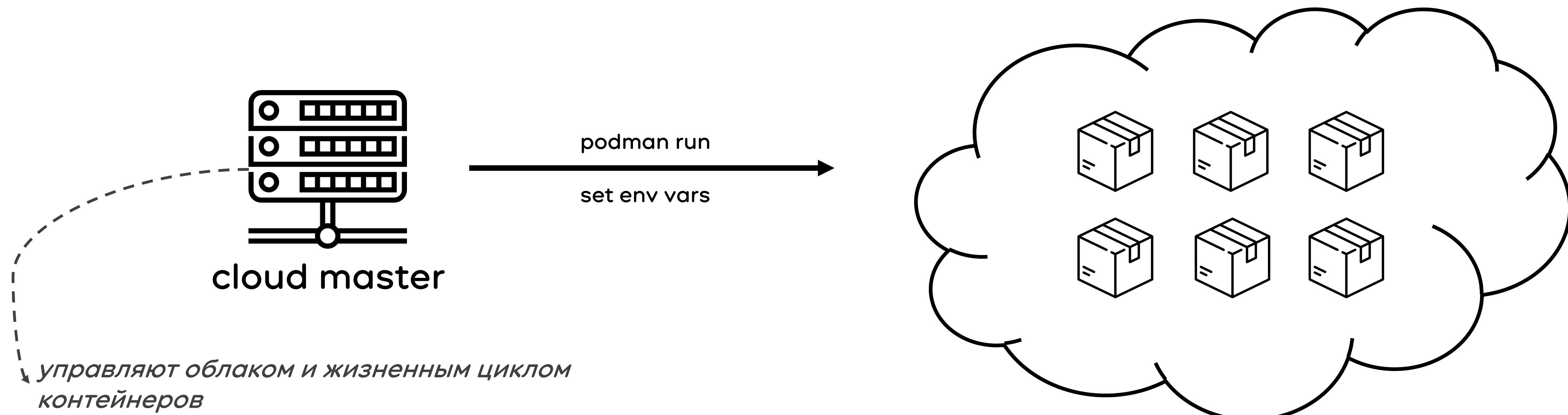
* Configuration management database

Посмотрим по сторонам: Облако



One-cloud

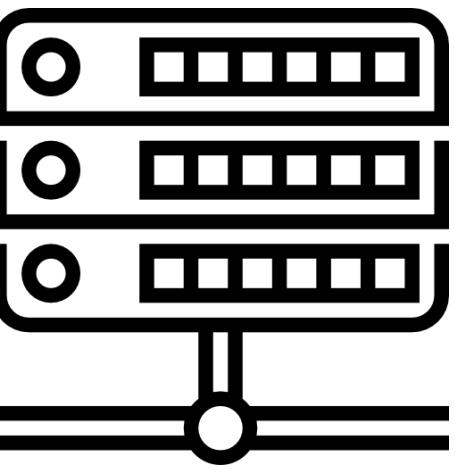
- У контейнеров нет процедуры деплоя ОС
- В контейнерах нет приватного ключа (готового секрета)



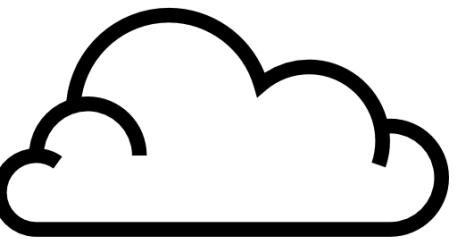
Есть все для JWT!



- На всех серверах есть **приватные ключи**
- Все **публичные ключи** есть в CMDB
- **Полезная нагрузка** – информация из CMDB



HW-сервер может сам
создать свой JWT



Мастер облака – тоже
HW-сервер

Создает JWT для
контейнеров и кладет в
ENV при запуске
контейнера

Есть все для JWT!

ХО

- Секрет для доступа к секретам – **JWT**
- Доверенные источники – **CMDB** и мастера **One-cloud**
- Нет новых сущностей
- Не нужно поддерживать новые решения





Auth Methods

Overview

AppRole

AliCloud

AWS

Azure

Cloud Foundry

GitHub

Google Cloud

JWT/OIDC

Kerberos

Kubernetes

LDAP

Login MFA

Oracle Cloud Infrastructure

Okta

RADIUS

TLS Certificates

Tokens

Username & Password

Kubernetes

AliCloud

LDAP

Tokens

TLS Certificates

AWS

Cloud Foundry

GitHub

Google Cloud

Username & Password

AppRole

RADIUS

JWT/OIDC

JWT в Vault «из коробки»

```
> vault auth enable -path=jwt jwt  
Success! Enabled jwt auth method at: jwt/
```

Три метода проверки подписи:

- **Static Keys** ----->

```
> vault write auth/jwt/config jwt_validation_pubkeys=@key.pub.pem  
Success! Data written to: auth/jwt/config  
> vault read auth/jwt/config  
Key Value  
---  
...  
jwt_validation_pubkeys [-----BEGIN PUBLIC KEY-----  
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE4+SFvPw0y0miy/FiT  
T05HnwjpEbSq+7+1q9BFxAkzjgKnIkXk5qxhzXQvRmS4w9ZsskoTZtu  
UI+XX7conJhzCQ==  
-----END PUBLIC KEY-----]  
...
```

- Нужно поддерживать список в vault

JWT в Vault «из коробки»

```
› vault auth enable -path=jwt jwt  
Success! Enabled jwt auth method at: jwt/
```

Три метода проверки подписи:

- Static Keys
- **JWKS (JSON Web Key Set)**

```
› vault write auth/jwt/config jwks_url="http://127.0.0.1:8000"  
Success! Data written to: auth/jwt/config
```

Key	Value
jwks_url	http://127.0.0.1:8000

- Нужно поддерживать список
- Нужно гонять данные по сети (все ключи)

JWT в Vault «из коробки»



Сомнительный механизм проверки подписи:

JWT в Vault «из коробки»



Сомнительный механизм проверки подписи:

```
var valid bool
for _, key := range config.ParsedJWTPubKeys {
    if err := parsedJWT.Claims(key, &claims, &allClaims); err == nil {
        valid = true
        break
    }
}
```

тысячи проходов



JWT в Vault «из коробки»

```
› vault auth enable -path=jwt jwt  
Success! Enabled jwt auth method at: jwt/
```

Три метода проверки подписи:

- Static Keys
- JWKS (JSON Web Key Set)
- **OIDC Discovery**

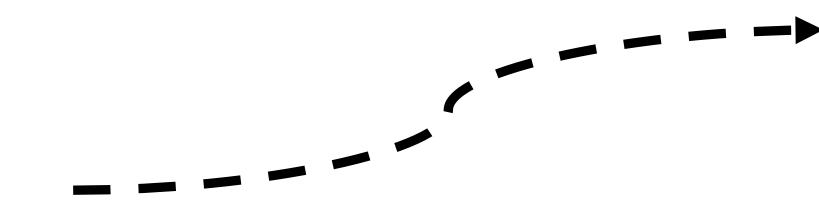
- Рассматривали как основное решение
- Не нужно перебирать ключи
- **Нужно написать свой OIDC-сервер**
- **Нужен умный клиент**

JWT в Vault «из коробки»

```
› vault auth enable -path=jwt jwt  
Success! Enabled jwt auth method at: jwt/
```

Три метода проверки подписи:

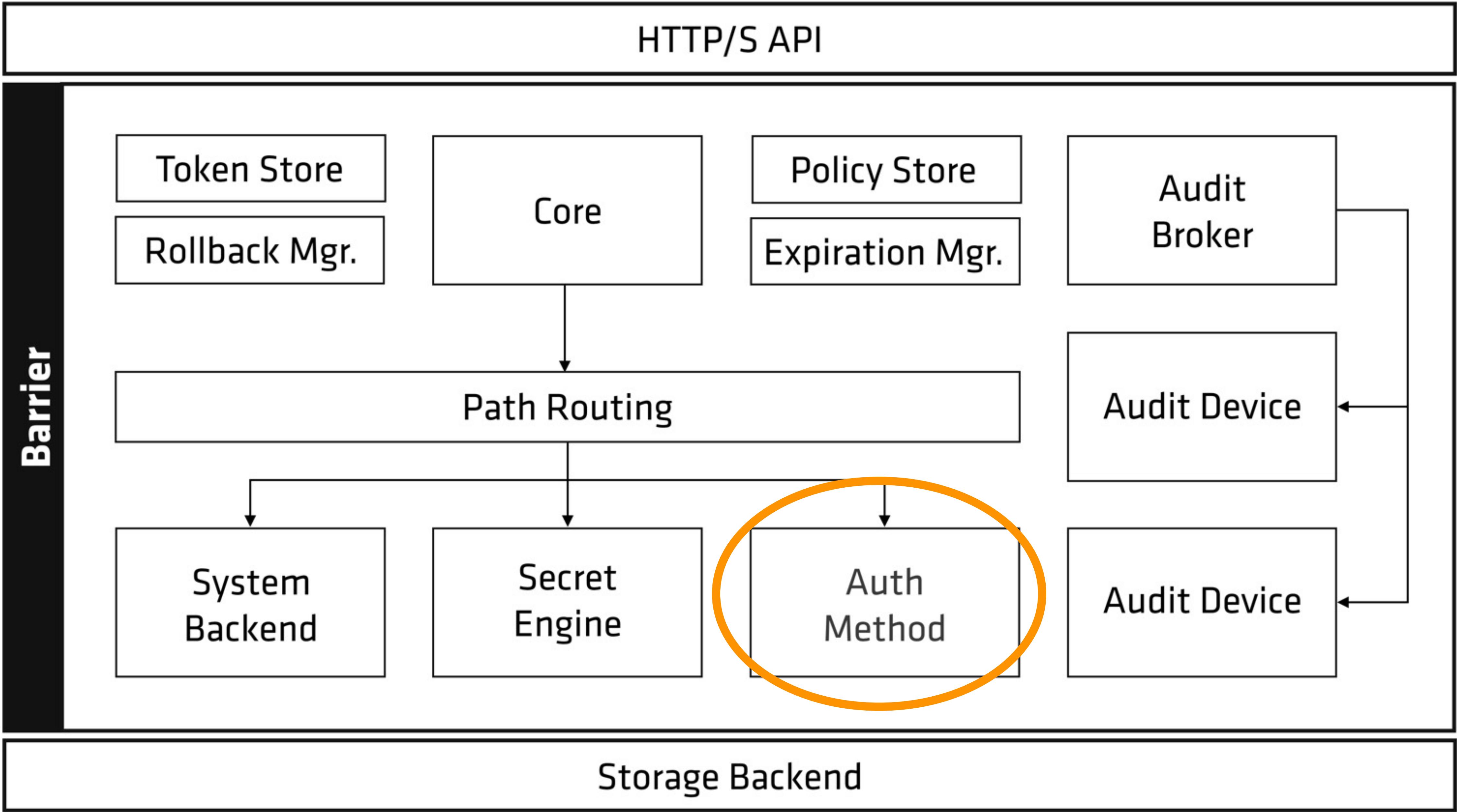
- Static Keys
- JWKS (JSON Web Key Set)
- OIDC Discovery



- Нужно заранее создавать роли
- Набор политик задается в ролях при их создании
- Роли нужно поддерживать



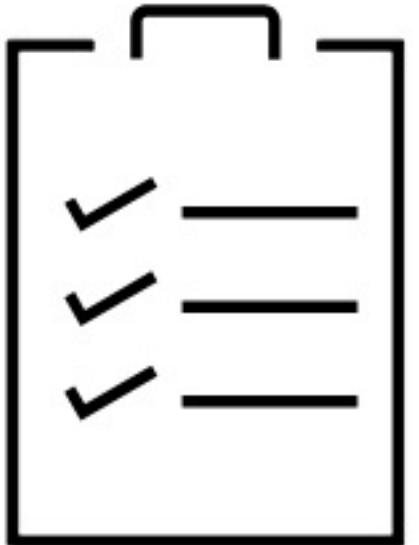
Сделаем аутентификацию сами!



Политики

После аутентификации в Vault клиент получается **Vault-токен**

У Vault-токана есть **метаданные и политики**

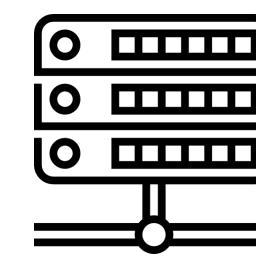


```
› vault token lookup s.wFCv4MhhScDKaSuwhNZRF4yv
```

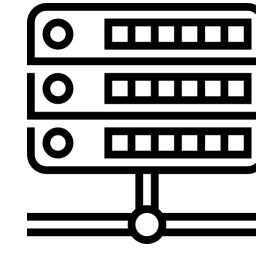
Key	Value
---	-----
...	
accessor	cWwjU76HZsIXhPIdBfUI547X
creation_ttl	24h
meta	map[cmdb_group:web service:web-1 hostname:srvd1234]
display_name	jwt-srvd1234
path	auth/jwt/login
policies	[hw_host, hw_host_srd1234, cmdb_group_web, development]
...	

Политики

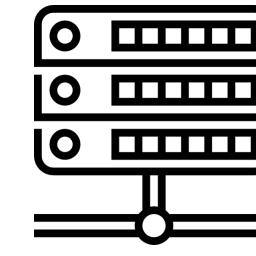
В политиках описаны пути и права доступа



srvd1234

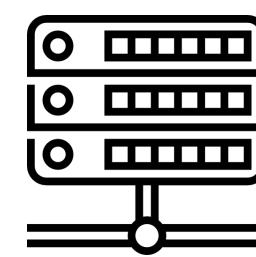


srvd4567



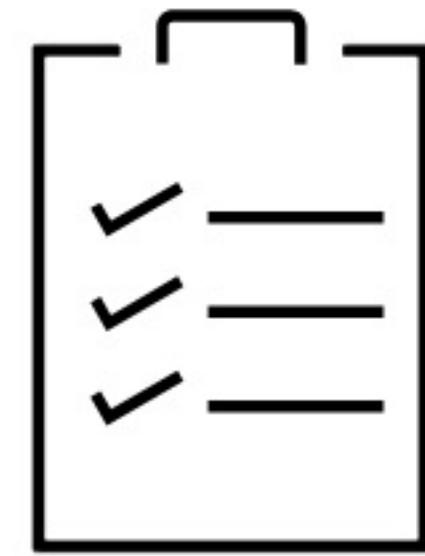
srvd7890

...



srvdXXX

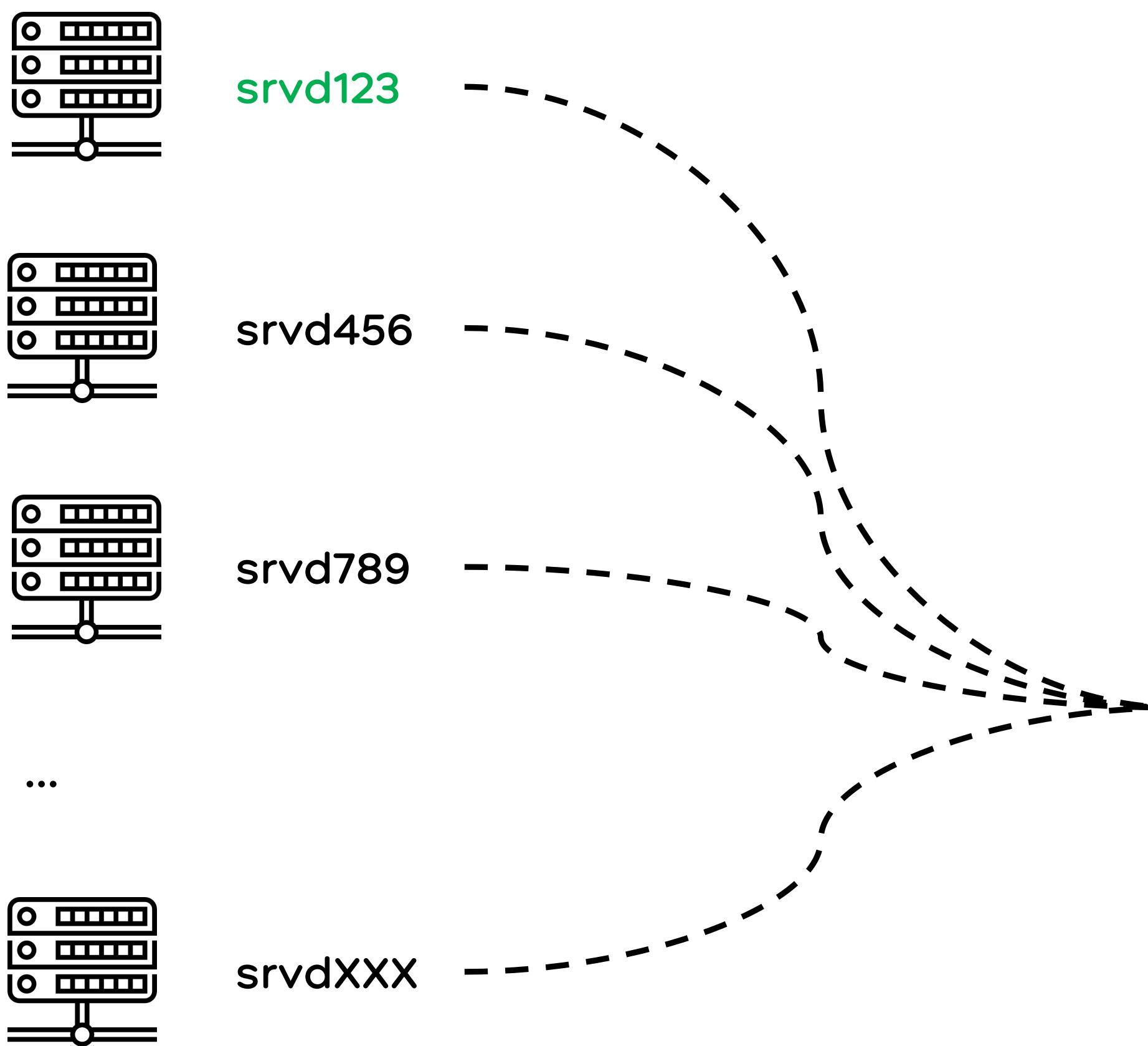
```
> cat policies/srvd1234.yml
---
paths:
  - path: kv/secrets/srvd1234/*
capabilities:
  - read
  - update
```



Писать 100500 политик для каждого хоста?..

Политики

В политиках можно использовать шаблоны



```
> cat policies/srvd1234.yml
```

```
---
```

```
paths:
```

- path: kv/secrets/**srvd1234/***
- ```
capabilities:
```
- read
  - update

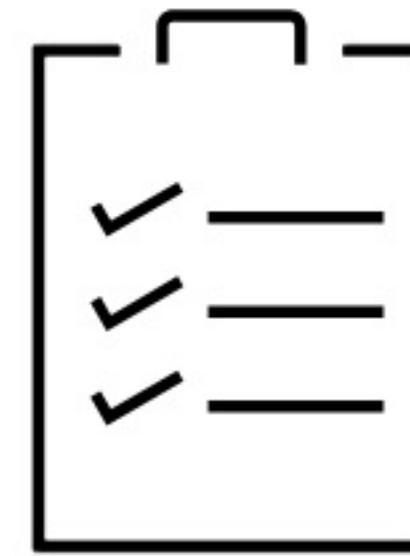
```
> cat policies/hw_host.yml
```

```

```

```
paths:
```

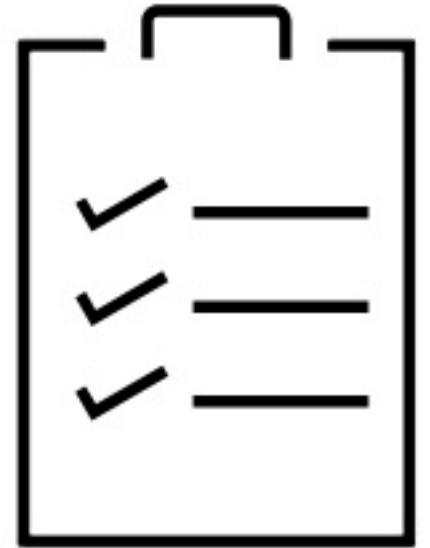
- path: kv/secrets/{{ hostname }}/\*"
- ```
capabilities:
```
- read
 - update



Политики

В политиках можно использовать шаблоны

```
> cat policies/hw_host.yml
---
paths:
  - path: kv/secrets/{{ hostname }}/+
capabilities:
  - read
  - update
```



```
> vault token lookup s.wFCv4MhhScDKaSuwhNZRF4yv
```

Key	Value
---	-----
...	
accessor	cWwjU76HZsIXhPIdBfUI547X
creation_ttl	24h
meta	map[cmdb_group:web service:web-1 hostname:srvd1234]
display_name	jwt-srvd1234
path	auth/jwt/login
policies	[hw_host , hw_host_srd1234, cmdb_group_web, development]
...	

Политики

HCL

```
> cat development.hcl  
  
path "kv/dev/users/+" {  
    capabilities = [  
        "read",  
        "update",  
    ]  
}  
  
path "kv/users/{{ service }}" {  
    capabilities = [  
        "read",  
    ]  
}
```



YAML

```
> cat development.yaml  
  
- path: kv/dev/users/+  
  capabilities:  
    - read  
    - update  
  
- path: kv/users/{{ service }}
```

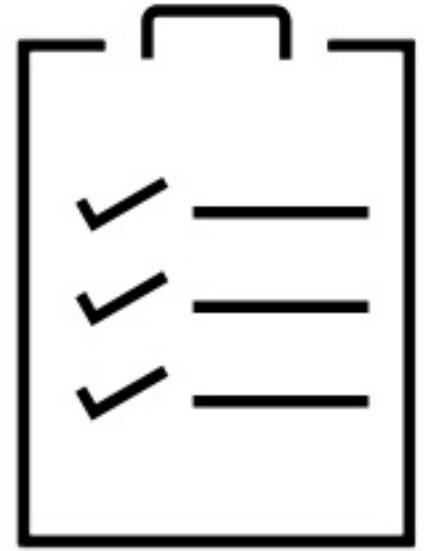
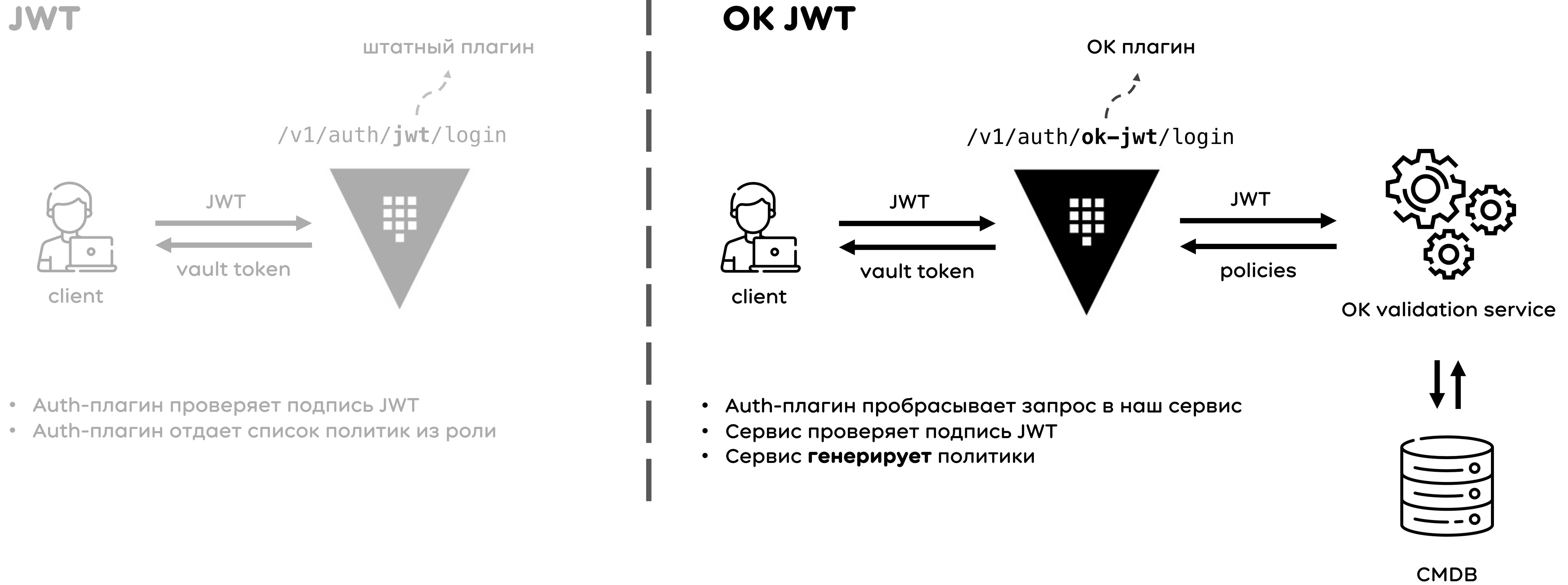


Схема аутентификации



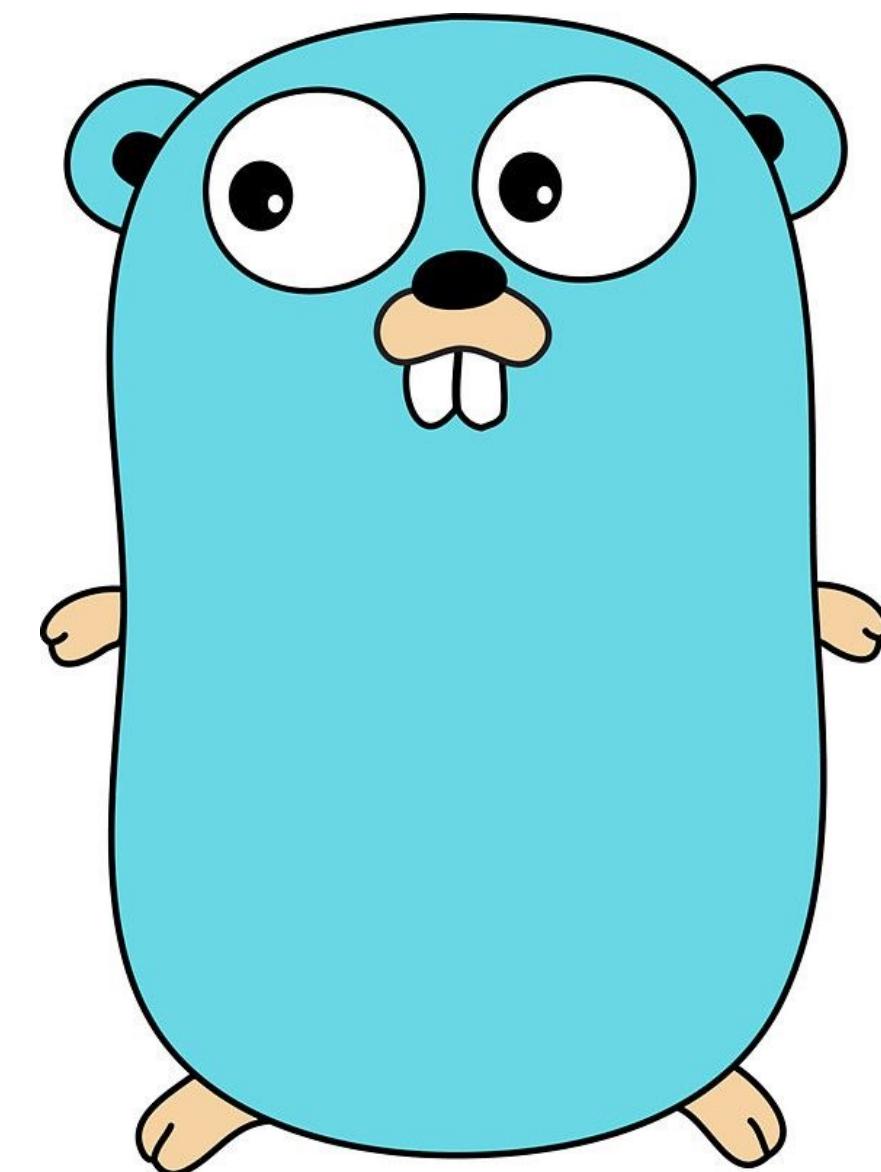
Аутентификация: OK JWT

OK

Auth-плагин для Vault

```
> vault write sys/plugins/catalog/auth/ok-jwt command="ok-jwt" sha_256=$SHA  
> vault auth enable -path=ok-jwt ok-jwt
```

- Бинарный файл (Go)
- Vault проверяет SHA плагинов
- Основан на [hashicorp/vault-plugin-auth-jwt](#)
- Проксирует запросы в наш сервис валидации
- Добавляет в токен **политики и метаданные**

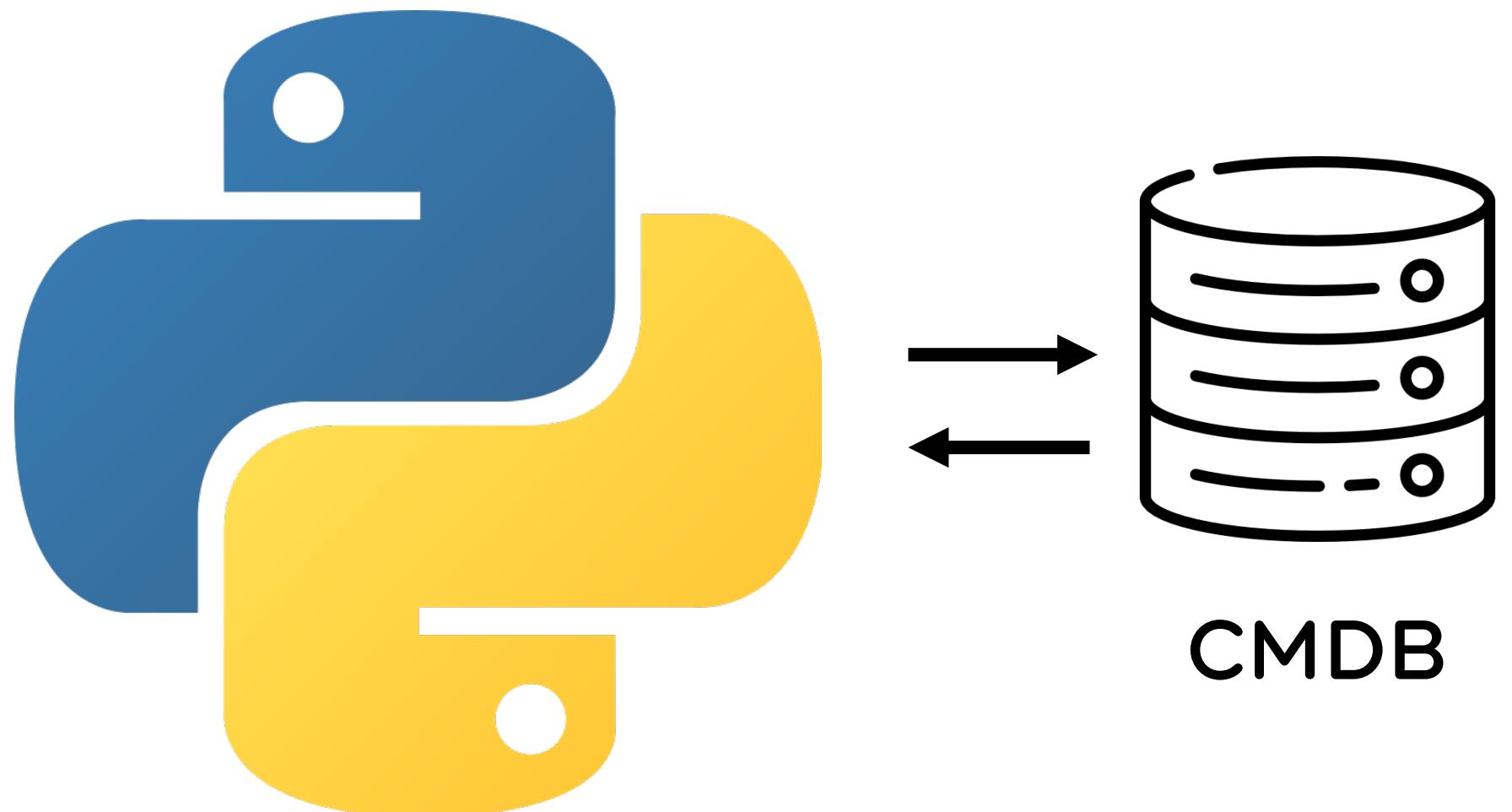


Аутентификация: OK JWT

Х

Сервис для валидации

- Python
- Проверяет подпись JWT
- Сверяет данные с информацией в CMDB
- Генерирует политики по нашим правилам
- Отдает список политик в Vault (наш плагин)



Аутентификация: OK JWT



```
func validateByOKService(token, OKAuthServiceURL string) ([]string, error) {  
  
    type response struct {  
        Policies []string `json:"policies"  
        Message  string   `json:"message"  
    }  
    var respData response  
    log.Printf("Send JWT for validation to: %s", OKAuthServiceURL)  
  
    var jsonStr = []byte(fmt.Sprintf(`{"JWT": "%s"}`, token))  
  
    req, err := http.NewRequest("POST", OKAuthServiceURL, bytes.NewBuffer(jsonStr))  
    if err != nil {  
        log.Printf("Can't create http request: %s", err)  
        return nil, err  
    }  
    ...  
    return respData.Policies  
}
```