

# Kubernetes secrets. Hashicorp Vault.

# Не забудь включить запись!



# План

- Работа с секретами в kubernetes
- Вспомнить как работает SSL
- Хранение секретов в vault
- Использование vault в k8s

# Хранение секретов в k8s

- Создание секрета

```
kubectl create secret generic dev-db-secret --from-literal=username=devuser \  
--from-literal=password='S!B\*d$zDsb'
```

- Создание секрета из файла

```
kubectl create secret generic db-user-pass --from-file=./username.txt --from-  
file=./password.txt
```

- Создание секрета из yaml'a

- для записи в yaml используем base64

- Хранение сертификатов

```
kubectl create secret tls vault-certs --cert=vault.crt --key=vault_gke.key
```

# Использование секретов в k8s

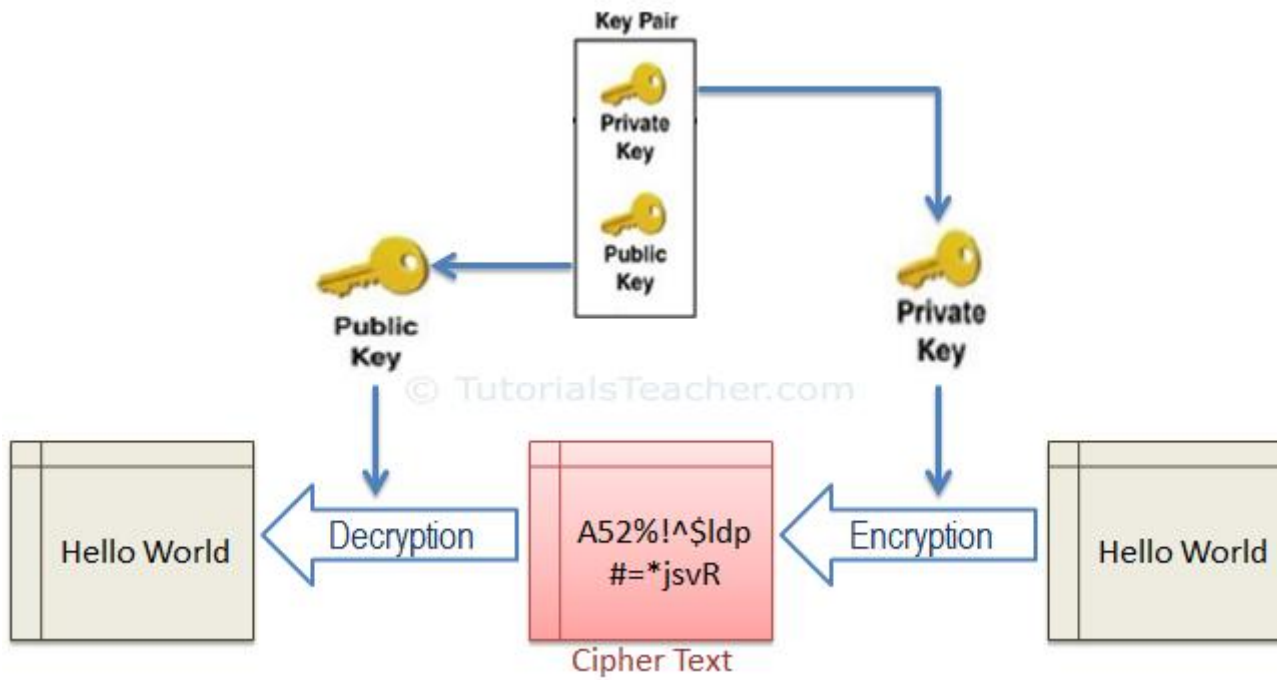
- Монтирование секрета через volumeMounts в файл
- Мэппинг в переменные окружения пода

# Шифрование хранилища секретов

- По умолчанию секреты сохраняются в etcd в нешифрованном виде
- Но есть возможность зашифровать с использованием одного из провайдеров
  - aescbc
  - secretbox
  - aesgcm
  - kms

# Демо. Создаем и используем секреты в k8s

# Что здесь не так?

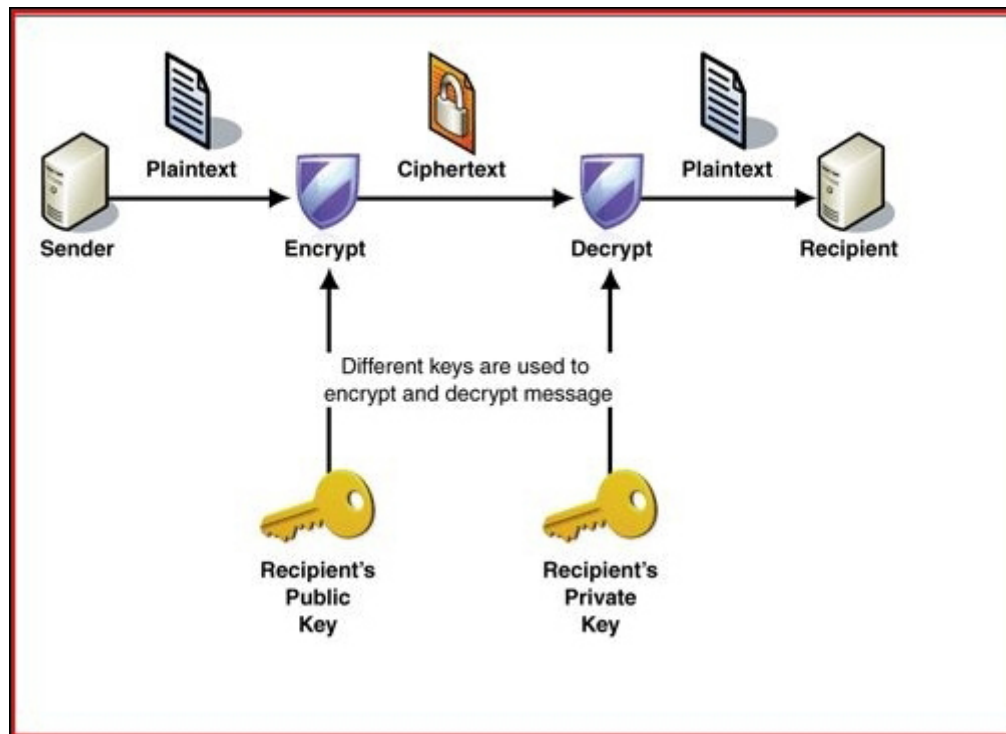




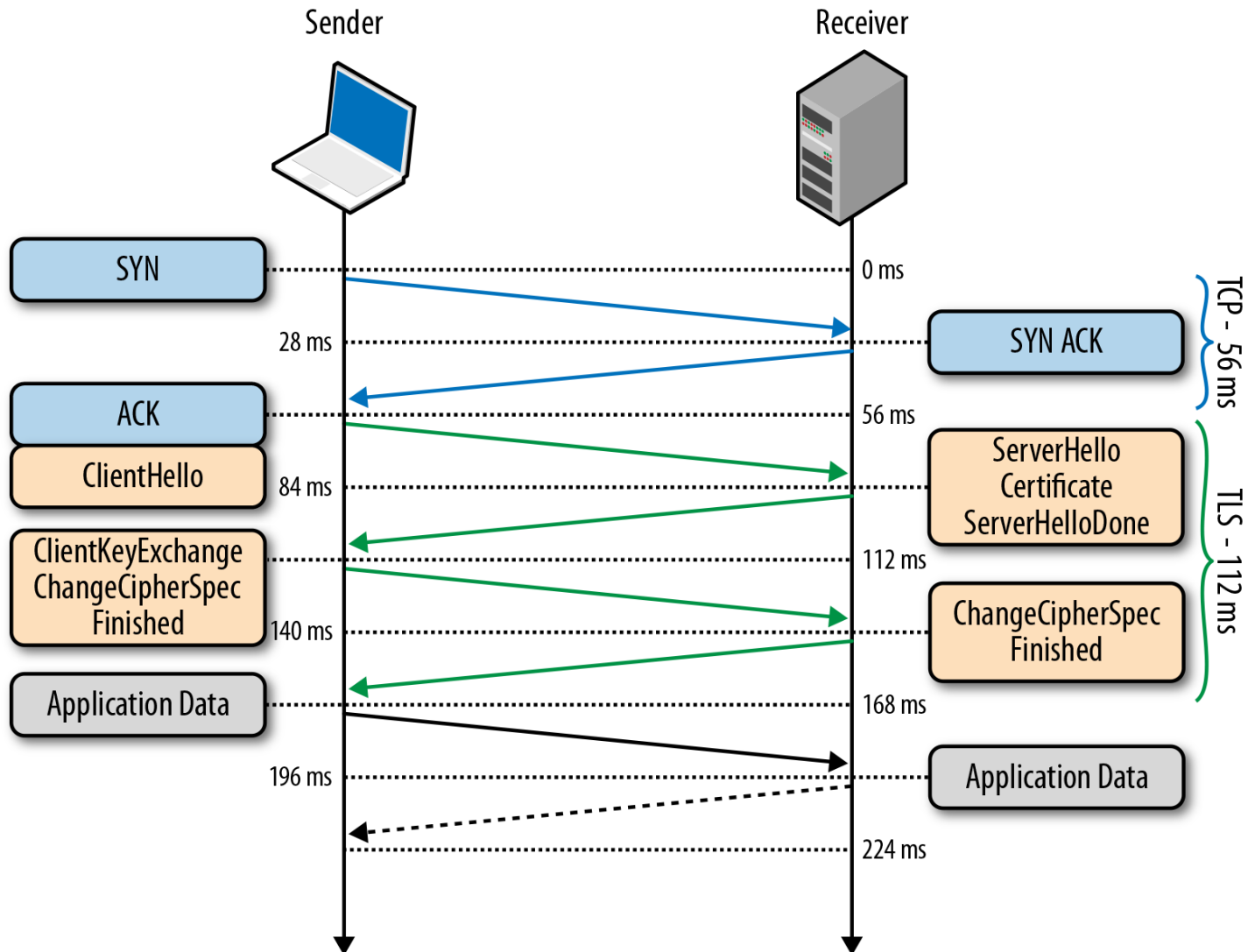
# Ассиметричное/симметричное шифрование

- Симметричный алгоритм шифрования — алгоритм, при котором для шифрования и дешифрования используется один и тот же ключ.
- В ассиметричном шифровании используется 2 ключа — открытый и закрытый(тайный). Открытый ключ для шифрования, закрытый — для дешифрования.

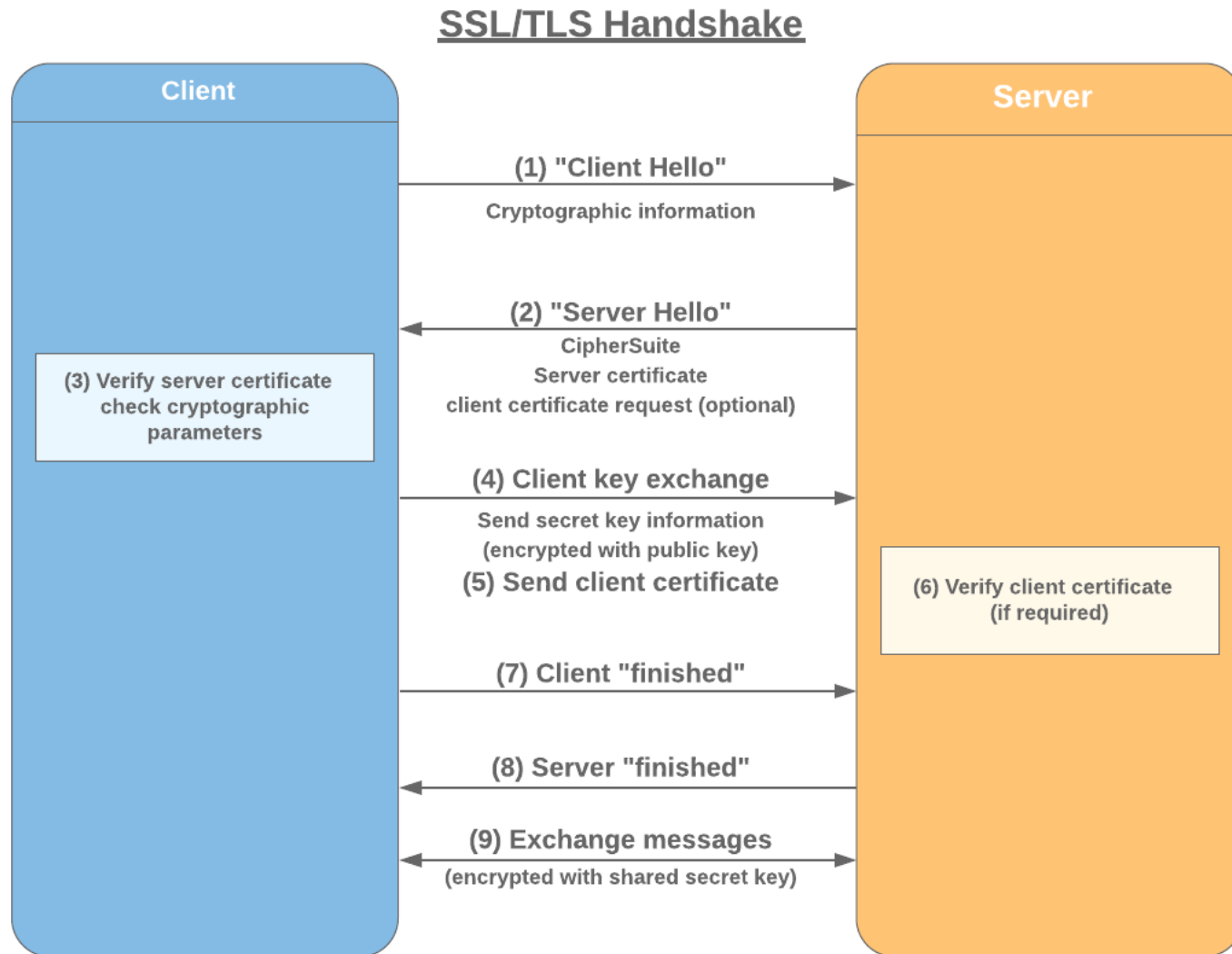
# Ассиметричное шифрование



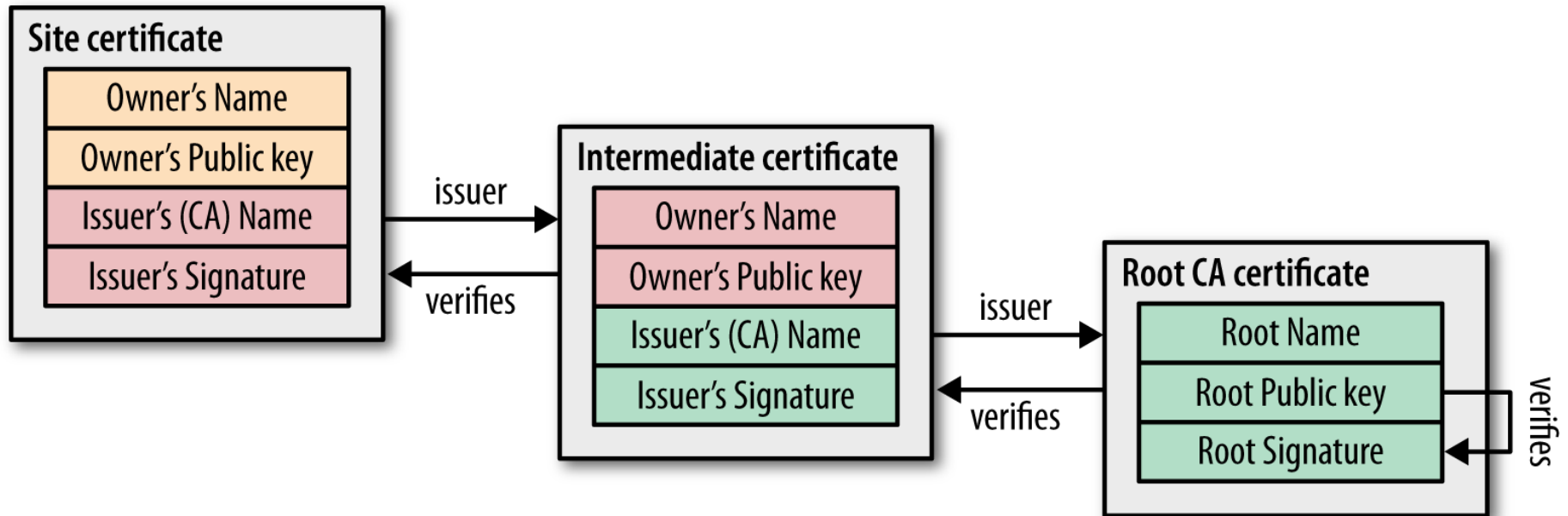
# TLS Handshake



# TLS Handshake



# Certificate's chain



# SSL/TLS: терминология

- SSL - Secure Socket Layer
- TLS - Transport Layer Security (актуальная версия 1.3)
- PKI - Public Key Infrastructure
- CA - certificate authorities
- Intermediate certificate
- CRL - Certificate Revocation List

# Hashicorp vault



HashiCorp  
**Vault**

# Особенности

- REST, JSON
- Безопасно хранит и управляет ключами.
- хранилища: file, consul, etcd, mysql, mongodb ...
- кластеризуется (в том числе в k8s)
- удобная система политик
- наличие API для взаимодействия из приложений напрямую
- различные варианты аутентификации: userpass, tls, токены, внешние API ...



# Проблемы и решения от Vault

## Проблемы

## Возможности vault

Хранение секретов везде.

Одно общее хранилище.

Хранение в открытом виде.

Встроенное шифрование, включая транзитное

Сложности с динамическими секретами

Встроенная возможность динамической генерации

Сложно выделять на время и отзываться

Доступы к секретам могут быть выданы и отозваны

Сложно отслеживать кто имел доступ

Встроенный аудит на генерацию и использование секретов

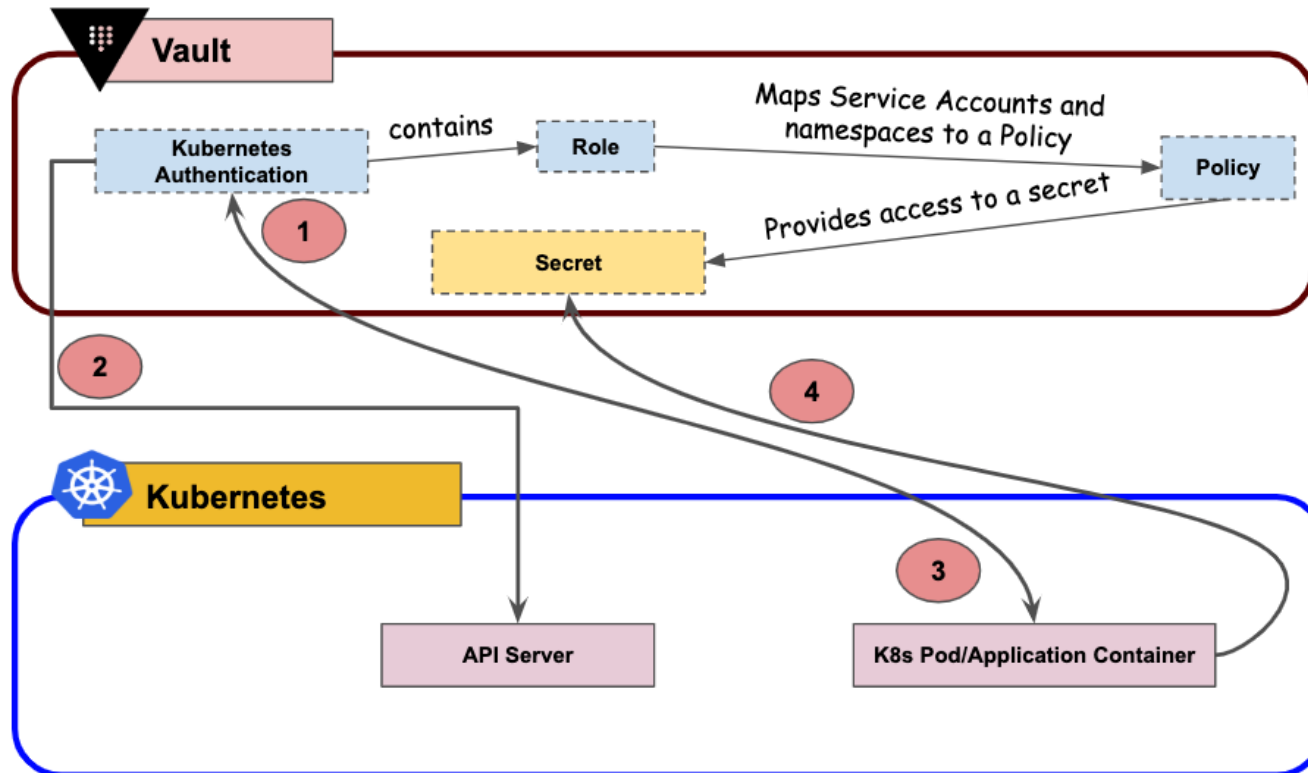
# Инициализация и открытие хранилища

- После установки требуется инициализация хранилища
  - `vault operator init --key-shares=1 --key-threshold=1`
- После инициализации отдает ключи
  - root'a
  - unseal ключ
- По дефолту работает через HTTP
  - необходимо включить https в конфиге

# Vault аутентификация

- Поддерживает несколько auth методов: userpass, token, tls, kubernetes
- По умолчанию каждый тип авторизации хранится по пути `auth/<type>`
- Для использования метода его необходимо включить
  - `vault auth enable userpass`
  - `vault write sys/auth/my-auth type=userpass`

# Kubernetes auth



1. Pod (application supplied JWT with Service Account and Namespace) authenticates using SA JWT and a role
2. Vault validates JWT, namespace and Service Account
3. Vault returns an access token
4. Pod access secrets using the returned access token from Vault

# Kubernetes auth

*# включили аутентификацию через k8s*

```
vault auth enable kubernetes
```

*# прописали куда обращаться за api*

```
vault write auth/kubernetes/config \  
token_reviewer_jwt="$SA_JWT_TOKEN" \  
kubernetes_host="$K8S_HOST" \  
kubernetes_ca_cert="$SA_CA_CRT"
```

*# создаем роль с привязкой к сервис аккаунту*

```
vault write auth/kubernetes/role/demo \  
    bound_service_account_names=vault-auth \  
    bound_service_account_namespaces=default \  
    policies=default \  
    ttl=1h
```

# Kubernetes Service Account

```
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: role-tokenreview-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
- kind: ServiceAccount
  name: vault-auth
  namespace: default
```

# Vault secrets engines

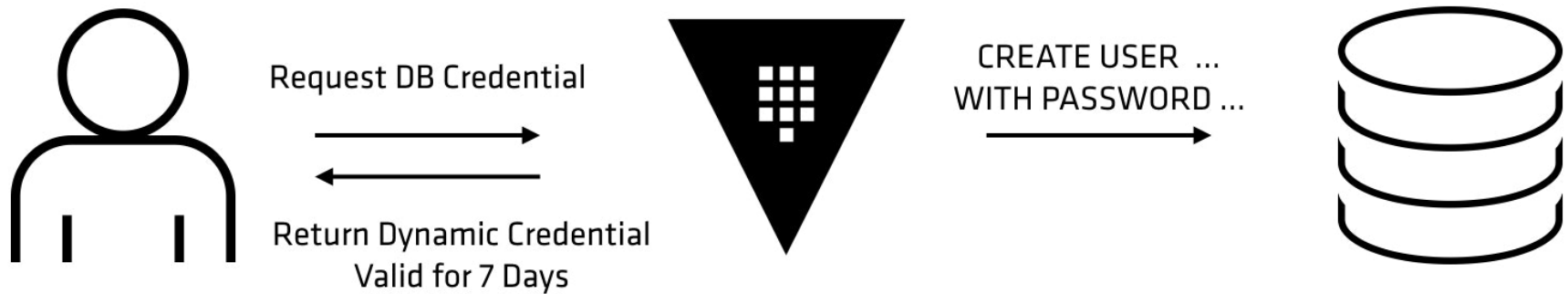
- Компоненты для хранения, генерации и шифрования секретов
  - проще воспринимать как функцию для реализации этих методов
- Может хранить локально, может обращаться на внешний API
- Обращение к компоненту происходит через `path`
- Жизненный цикл secret engines
  - `enable/disable`
  - `move`
  - `tune`
- Помощь по конкретному энжину `vault path-help`

# Динамические секреты

- Создаются по требованию
- ограниченный доступ согласно роли
- могут быть ограничены сроком жизни
- могут быть отозваны
- доступ к секрету логируется (включен аудит)



# Динамические секреты



# Vault leases

- Метаинформация о динамических секретах и токенах
  - содержит ttl, возможность обновления итд
- При чтении выдается Leaseld, через который можно
  - `vault renew my-lease-id 3600`
  - `vault revoke my-lease-id`

# Vault wrapping

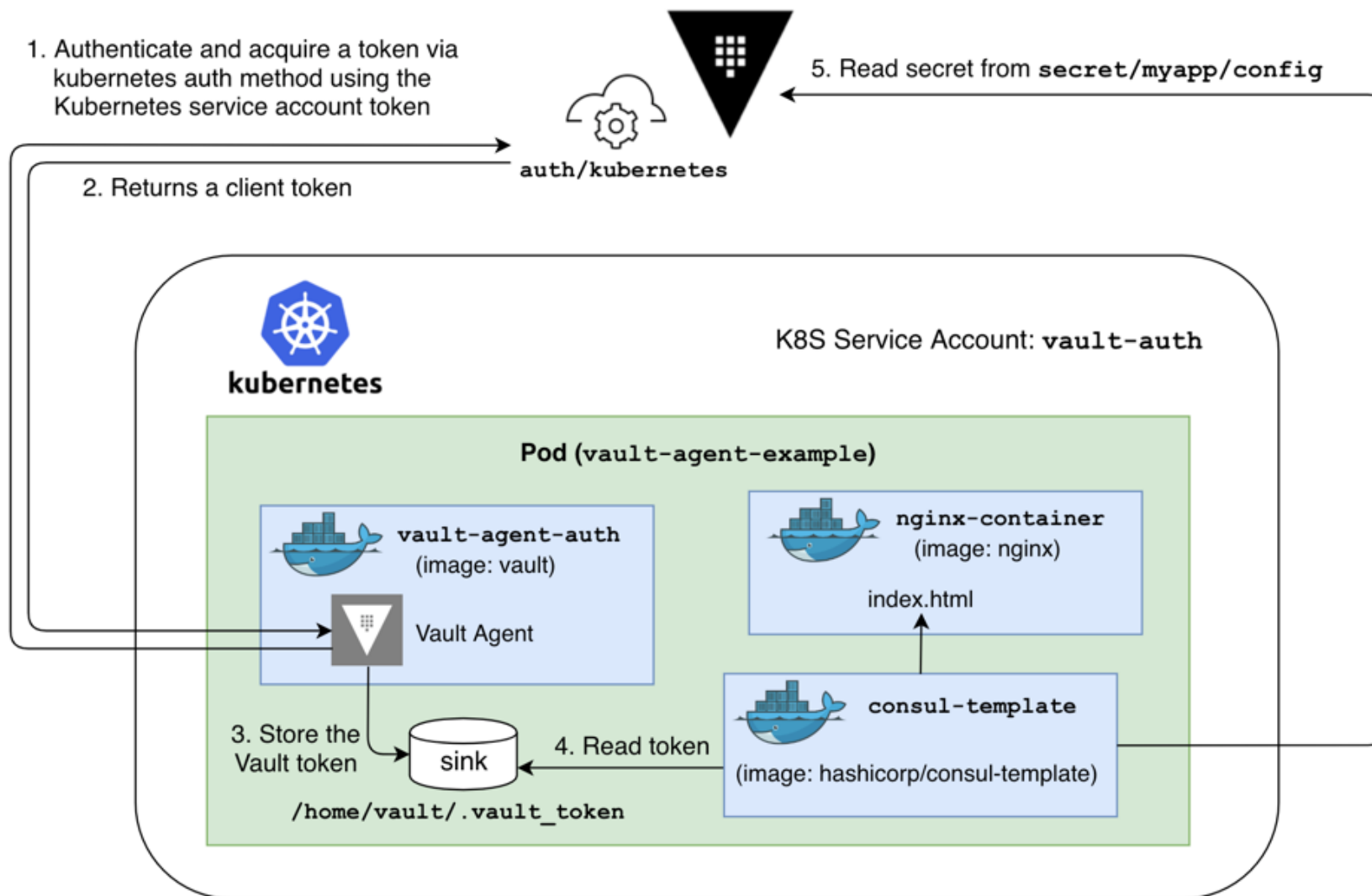
- Доступ к секрету через временный токен
  - при чтении секрета создается временный токен
  - у токена ограничен TTL
  - передается токен
  - по токену читается секрет

# Vault agent

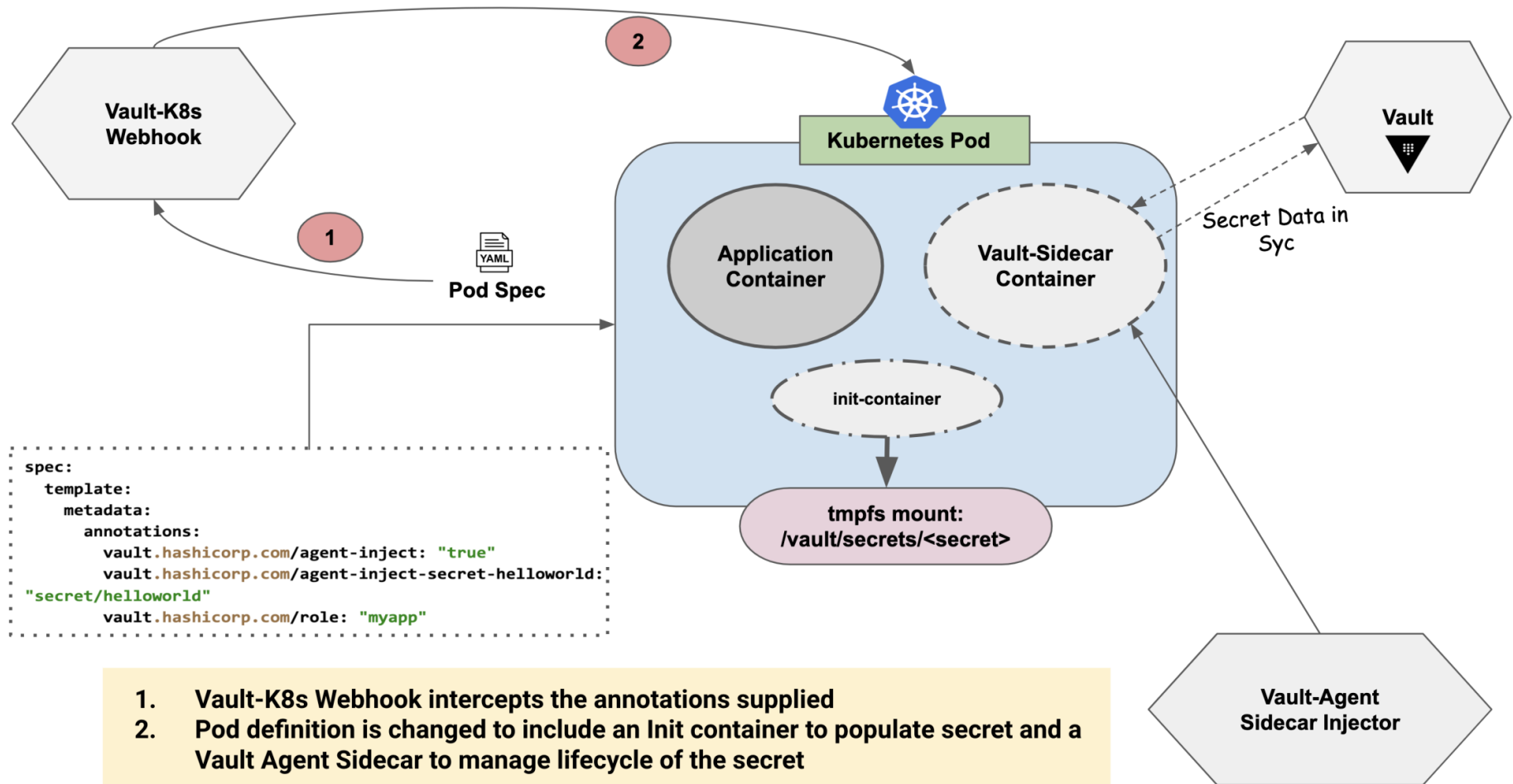
Клиентский демон со следующим функционалом

- Auto-auth
  - автоматическая аутентификация и автообновление токена
- Caching
  - кэширование токенов
- Templating
  - рендер шаблонов с автоподстановкой новых секретов и возможностями вызова команд

# Vault agent



# Vault inject



# Vault policies

- Задаются файлами правил
  - `vault policy write my_policy my_policy.hcl`
  - `my-policy.hcl`

```
path "secret/app/*" {  
  capabilities = ["create", "read", "update", "delete", "list"]  
}  
path "secret/db/*" {  
  policy = "read"  
}
```

# Транзитное шифрование

```
vault secrets enable transit  
vault write -f transit/keys/otus  
vault write transit/encrypt/otus plaintext=$(base64 <<< "Hell, kitty")  
vault write transit/decrypt/otus ciphertext=<cipher> | base64 -d
```



# Policy capabilities

- create - позволяет создавать секреты по заданному пути
- read - разрешает чтение
- update - разрешает обновление существующих секретов
- delete - разрешает удаление существующих секретов
- list - просмотр списка секретов
- sudo - позволяет доступ к root-protected секретам
- deny - запрещает доступ

# Vault audit

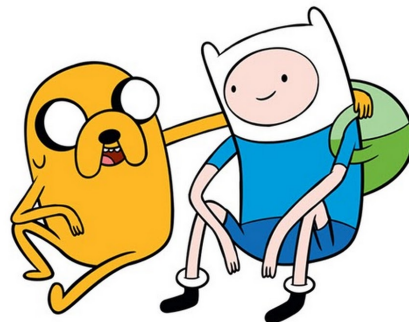
```
# file
vault audit enable file file_path=/var/log/vault_audit.log
# stdout
vault audit enable file file_path=stdout
# syslog
vault audit enable syslog tag="vault" facility="AUTH"
# tcp/udp/unix socket
vault audit enable syslog tag="vault" facility="AUTH" format="json"
```

# Kubevault

- <https://kubevault.com>
- автоматическая инициализация и unsealing
- управления политиками vault
- AWS/Azure/GCP/Database secrets engines
- мониторинг через Prometheus
- реализация через operator

# Демо.

- Vault в k8s через tls.
- Создаем пользователей, секреты и политики в vault
- используем Vault в приложении



# Спасибо за внимание!

## Время для ваших вопросов!