Kubernetes для непрерывной поставки (CI/CD). Интеграция с CI-сервисом.

Не забудь включить запись!

План

- Организация непрерывной поставки
- Gitlab и Kubernetes
- Вопрос к организации CI/CD в kubernetes

Непрерывная интеграция и поставка

Непрерывная интеграция и поставка

- Непрерывная интеграция
  - Интеграция нового функционала в основной код
- Непрерывная поставка
  - Автоматическое развертывание сервисов доступных для конечного пользователя
  - Изменения конфигов
  - Миграции в БД

Этапы непрерывной поставки

Типичные этапы (stages)

- Build
- Test
- Review
- Deploy
- Report

Gitlab + Kubentes

- Интеграция с кубернетес
  - tiller
  - gitlab-runner (в привилегированном режиме!!!)
  - kubeconfig
  - shared runners
- AutoDevOps

Настройка интеграции

- Прописать доступ к k8s кластеру
  - `kubectl cluster-info | grep 'Kubernetes master' | awk '/http/ {print $NF}'`
- Прописать ca сертификат из секрета default-token-xxxxx
  - `kubectl get secret <secret name> -o jsonpath="{['data']['ca\.crt']}" | base64 --decode`
- создать gitlab-admin service account
- прописать токен
  - `kubectl -n kube-system describe secret $(kubectl -n kube-system get secret | grep gitlab-admin | awk '{print $1}')`

Создание сервис аккаунта

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitlab-admin
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: gitlab-admin
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: gitlab-admin
  namespace: kube-system
```

Демо

- смотрим, как это все настраивается в gitlab

вопросы CI/CD в Kubernetes

- сборка образов
  - dind (привилегированный режим, проброс сокета с хоста)
  - kaniko
- разные проекты в разные неймспейсы
  - tiller
- разные этапы в разные кластера
  - kubeconfig
- разные проекты и этапы в один кластер в разные неймспейсы
  - лимиты, сертификаты, очистка

Демо.

- как затереть из другого репозитория чужую инсталляцию

Заведем свой тиллер в отдельном неймспейсе

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: tiller
  namespace: otus-gitlab
---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: tiller-manager
  namespace: otus-gitlab
rules:
- apiGroups: ["", "batch", "extensions", "apps"]
  resources: ["*"]
  verbs: ["*"]
```

Заведем свой тиллер в отдельном неймспейсе

```
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: tiller-binding
  namespace: otus-gitlab
subjects:
- kind: ServiceAccount
  name: tiller
  namespace: otus-gitlab
roleRef:
  kind: Role
  name: tiller-manager
  apiGroup: rbac.authorization.k8s.io
```

Заведем cluster-role helm в отдельном неймспейсе

```yaml
kind: Namespace
apiVersion: v1
metadata:
  name: sre
---
kind: ServiceAccount
apiVersion: v1
metadata:
  name: helm
  namespace: sre
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: helm-clusterrole
rules:
  - apiGroups: [""]
    resources: ["pods/portforward"]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["list", "get"]
```

Заведем cluster-role helm в отдельном неймспейсе

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: helm-clusterrolebinding
roleRef:
  kind: ClusterRole
  apiGroup: rbac.authorization.k8s.io
  name: helm-clusterrole
subjects:
  - kind: ServiceAccount
    name: helm
    namespace: sre
```

Соберем отдельный kubeconfig

- Почитать подробнее
- kubernetes_add_service_account_kubeconfig.sh
- `./kubernetes_add_service_account_kubeconfig.sh <service_account_name> <namespace>`
- Использование в пайплайне
  - KUBECONFIG=<ваш конфиг файл>
  - конфиг кодируем base64 и сохраняем в gitlab variables
  - и можем использовать его в пайпалайнах с использованием хелма
  - `- echo $KUBECONFIG_VAR | base64 -d > ~/.kube/config`

Установка gitlab-runner самостоятельно

- скачать gitlan-runner helm
- прописать в секрет если нужно <your-gitlab-domain>.crt
  - `kubectl create secret generic gitlab-domain-cert --from-file=gitlab.devlab.ru.crt -n`
- правим values
  - токен гитлаба
  - `certsSecretName: gitlab-domain-cert`
  - выключаем привилегированный режим
  - указываем используемый service-account
- `helm install --name gitlab-runner -f values.yaml gitlab/gitlab-runner`

Проблемы поставки в разные неймспейсы

- Поставка связанных секретов и конфигов
  - используйте helm-hooks
- Генерация сертификатов для каждого тестового неймспейса
  - letsencrypt (обращаем внимаение на лимиты)
  - свой CA
- Миграции БД

Пример деплоя

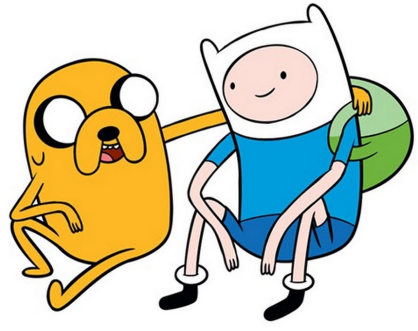
```
deploy:test:
  extends: .helm_common
  stage: deploy-test
  variables:
    IMAGE_NAME_BASE: $REGISTRY_HOST/$REGISTRY_ID/$REGISTRY_NAMESPACE
    IMAGE_TAG: $CI_COMMIT_REF_SLUG-$CI_COMMIT_SHORT_SHA
    KUBECONFIG_GITLAB_B64: $TEST_KUBECONFIG_GITLAB_B64
  script:
    - apk update && apk add gnupg git bash curl
    - /bin/bash -c "gpg --import <(echo $GPG_KEY_B64 | base64 -d)"
    - helm secrets upgrade
      --install app Deployments/helm/app
      --namespace app-test
      -f Deployments/environments/all/secrets.yaml
      -f Deployments/environments/test/values.yaml
      -f Deployments/environments/test/secrets.yaml
      --set app1.image.name=$IMAGE_NAME_BASE/app1
      --set app1.image.tag=$IMAGE_TAG
      --set app2.image.name=$IMAGE_NAME_BASE/app2
      --set app2.image.tag=$IMAGE_TAG
```

Версионирование

- Версия приложения
- Версии образов (в проекте их бывает много)
  - перетэгирование образа
- Изменение чарта при изменении образа
- Версия хелм чарта
  - история чартов

Демо.

- CI/CD для телеграмм-бота



Спасибо за внимание!

Время для ваших вопросов!