СЛЁРМ

+

Southbridge

# Gitlab CI/CD



slurm.io

# Установка гитлаба

curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

sudo EXTERNAL_URL="https://gitlab.slurm.io" yum install -y gitlab-ce

# Обслуживание


```
0 1 * * * root /usr/bin/gitlab-rake gitlab:backup:create
    SKIP=registry,builds,artifacts >/dev/null 2>&1

0 1 * * * root sh -c '(umask 0077; tar -czf /var/opt/gitlab/backups/$(date
    "+etc-gitlab-\%s_\%Y_\%m_\%d.tgz") -C / etc/gitlab) >/dev/null 2>&1'

0 0 * * * root find /var/opt/gitlab/backups/ -type f -mmin +4320 -delete

0 4 * * * root /usr/bin/gitlab_registry_prune -c 5 -r
    >/var/log/gitlab/registry_gc.log 2>&1

10 4 * * * root /usr/bin/gitlab-ctl registry-garbage-collect >/dev/null 2>&1
```

# Установка gitlab-runner


curl -L
https://packages.gitlab.com/install/repositories/runner/gitlab-runner/script.rpm.sh | sudo bash

yum install gitlab-runner –y

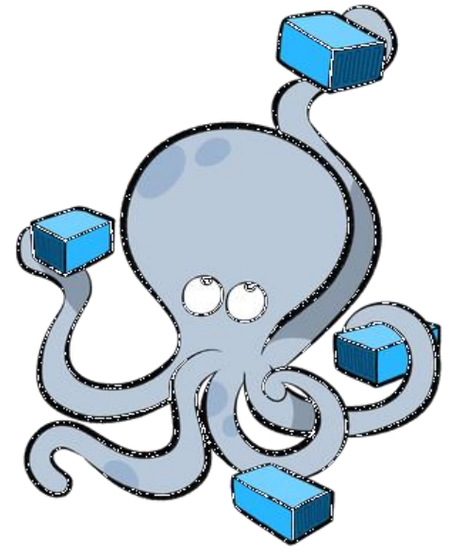
gitlab-ci-multi-runner register

# Этапы CI/CD

pdf с подробностями

- Build
- Test
- Cleanup test
- Push
- Deploy

# Test



**Запускаем через docker-compose**

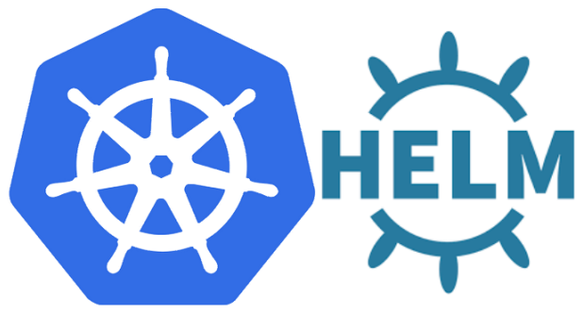
**Добавляем зависимости, чтобы успели запустится контейнеры с базой данных**

```
depends_on:
  db:
    condition: service_healthy
```

**Выключаем ненужные логи, чтобы не засоряли вывод**

```
logging:
  driver: none
```

# Deploy



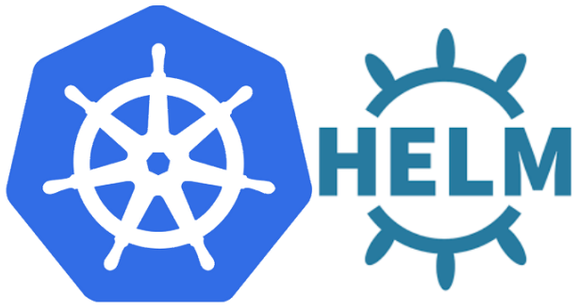

**Создаем окружение  и пользователя в kube для деплоя**
https://github.com/centosadmin/slurm/blob/master/practice/ci-cd/setup.sh

**Создаем Deploy Token в Gitlab**
https://gitlab.slurm.io/slurm.io/example/settings/repository

**Прописываем как секрет**
kubectl create secret docker-registry --docker-username 'YYYY' --docker-password 'xxxxxxxxxx' --docker-server <$CI_REGISTRY> --docker-email 'admin@slurm.io' <$CI_PROJECT_PATH_SLUG>-gitlab-registry --namespace <$CI_PROJECT_PATH_SLUG-$CI_ENVIRONMENT_NAME>

**Указываем в ImagePullSecrets: в deployment**
imagePullSecrets:
  - name: <$CI_PROJECT_PATH_SLUG>-gitlab-registry

# Секреты



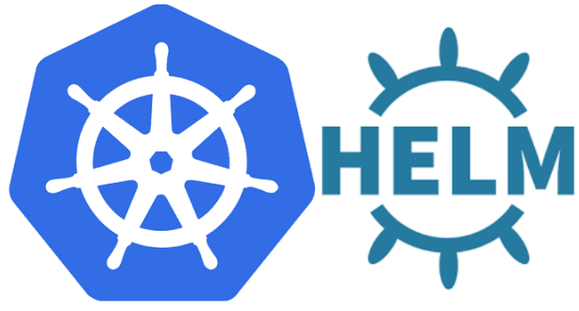

Хранятся в kube secrets

```
kubectl create secret generic slurmio --from-literal secret-key-
    base='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'
    --from-literal db-user='XXXXXXXXXXXXXXXX'
    --from-literal login-password='xxxxxxx' --namespace slurm-io-production
```

При деплое подставляются в env контейнера
helm генерирует из списка в темплейте

```
- name: LOGIN_PASSWORD
    valueFrom:
        secretKeyRef:
            key: login-password
            name: slurmio
```

# Deploy




**Запускаем helm из образа centosadmin/kubernetes-helm:v2.9**

**Настраиваем контекст доступа**

```
kubectl config set-cluster k8s --insecure-skip-tls-verify=true --server=$K8S_API_URL
kubectl config set-credentials ci --token=$K8S_CI_TOKEN
kubectl config set-context ci --cluster=k8s --user=ci
kubectl config use-context ci
```

**helm upgrade**

```
helm upgrade --install $CI_PROJECT_PATH_SLUG .helm \
 --set image=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME \
 --set imageTag=$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID \
 --wait --timeout 180 --debug \
 --namespace $CI_PROJECT_PATH_SLUG-$CI_ENVIRONMENT_NAME \
 --tiller-namespace=$CI_PROJECT_PATH_SLUG-$CI_ENVIRONMENT_NAME
```

# Deploy


```
|| (helm history --max 2 \
--tiller-namespace=$CI_PROJECT_PATH_SLUG-
$CI_ENVIRONMENT_NAME \
$CI_PROJECT_PATH_SLUG | \
head -n 2 | \
tail -n 1 | \
awk "{print \$1}" | \
xargs helm rollback \
--tiller-namespace=$CI_PROJECT_PATH_SLUG-
$CI_ENVIRONMENT_NAME \
$CI_PROJECT_PATH_SLUG && exit 1)
```

# Helm chart

**image:**
**imageTag:**
**imagePullSecret:**

**env:**
**NAME: value**
**любое количество переменных с их значениями по одному на строчку (будут добавлены в деплоймент)**

**envSecret:**
**NAME: secret-name см. описание выше**
**Секретные переменные, значения будут подставлены из secret-name**

**…**