1. **Конфигурирование gitlab, кластера kubernetes и private** **docker registry**

Кофигурация gitlab и кластера kubernetes для работы ci/cd сценариев со specific gitlab-runner описана в документе задачи [623](https://jira.gkh.mos.ru/browse/ASUEIRC-623).

В данном документе описываются работа ci/cd сценариев с shared gitlab-runner и конфигурация кластера kubernetes для работы с private docker registry

Общее описание шагов конфигурации

1. Получить данные private docker registry или создать его
2. Обеспечить связь мастера и всех нод кластера kubernetes с docker registry
3. В соответствии с документацией по задаче создать shared gitlab-runner и подключить его к проекту с ci/cd
4. В namespace kubernetes, в котором выполняется контейнер gitlab-runner, создать secret, обеспечивающий связь kubernetes с docker registry
5. Предоставить service account (учетной записи) kubernetes, под которой выполняется процесс ci/cd, права на чтение конфигураций и выполнение деплойментов
6. Настроить переменные проекта.
7. Настроить файлы ci/cd в тестовом проекте

Детали шагов конфигурации

1. Должен существовать private docker registry с basic authentication. Его можно создать командами

$ mkdir auth

$ docker run \

--entrypoint htpasswd \

registry:2 -Bbn testuser testpassword > auth/htpasswd

$ sudo docker run -d \

-p 5000:5000 \

--restart=always \

--name registry \

-v "$(pwd)"/auth:/auth \

-e "REGISTRY\_AUTH=htpasswd" \

-e "REGISTRY\_AUTH\_HTPASSWD\_REALM=Registry Realm" \

-e REGISTRY\_AUTH\_HTPASSWD\_PATH=/auth/htpasswd \

-v "$(pwd)"/certs:/certs \

-e REGISTRY\_HTTP\_TLS\_CERTIFICATE=/certs/domain.crt \

-e REGISTRY\_HTTP\_TLS\_KEY=/certs/domain.key \

registry:2

Подробности см здесь

<https://docs.docker.com/registry/deploying/>

Как генерировать и размещать сертификаты описано здесь:

<https://docs.docker.com/engine/security/certificates/>

Сертификаты можно выпустить командой

1. На всех нодах и мастере кластера kubernetes должен быть сконфигурирован доступ к private docker registry. Сертификаты из п 1 должны быть развернуты в соотв с <https://docs.docker.com/engine/security/certificates/>. В результате с каждой ноды и мастера kubernetes должна пройти команда  
   $ sudo docker login -u <usr> -p <pwd> <registry host>:<registry port>
2. В соотв с документом [по задаче 623](https://jira.gkh.mos.ru/browse/ASUEIRC-623) создать shared gitlab-runner и подключить его к проекту с ci/cd.  
   см http://<gitlabhost>:<gitlabport>/help/user/project/clusters/add\_remove\_clusters.md#add-existing-cluster  
   После чего нужно зайти на страницу Your Project/CI-CD/Runners. Нажать кнопку Enable Shared Runners
3. В namespace kubernetes, в котором выполняется контейнер gitlab-runner, создать secret, обеспечивающий связь kubernetes с docker registry. Команда выполняется из консоли мастера кластера kubernetes   
     
   $ kubectl create secret docker-registry regcred --docker-server=<private docker registry host> --docker-username=<registry username> --docker-password==<registry password> – n gitlab-managed-apps  
     
   gitlab-managed-apps – это namespace созданный в п 3. Название созданного secret   
   regcred
4. Процесс ci/cd выполняется в namespace gitlab-managed-apps под service account default. Этому service account нужно предоставить права на чтение конфигураций и выполнение деплойментов. Для этого нужно сформировать файл deployment.yaml

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRoleBinding

metadata:

name: gitlab-bootstrap-admin-managed-apps

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: default

namespace: gitlab-managed-apps  
  
и выполнить команду  
  
$ kubectl apply -f deployment.yaml   
См <https://sanderknape.com/2019/02/automated-deployments-kubernetes-gitlab/>

1. Для выполнения команд развертывания из файла .gitlab.ci, определяющего процесс ci нужно настроить контекст процесса. Для этого в контексте нужно определить данные кластера kubernetes (url мастера и сертификат кластера ) и данные service account’а (имя в нашем случае default и токен). Лучше всего конфигурацию выполнить используя переменные процесса в gitlab. Для задания их нужно пройти по адресу   
   Your Project/CI-CD/Variables  
   Нужны следующие переменные  
   SERVER url мастера kubernetes. Как получить см п 3  
     
   CERTIFICATE\_AUTHORITY\_DATA (сертификат кластера)  
   Нужно выполнить команду на мастере

$ kubectl config view --minify --raw --output 'jsonpath={..cluster.certificate-authority-data}'

Скопировать вывод и вставить в поле CERTIFICATE\_AUTHORITY\_DATA

USER\_TOKEN (токен service account default)   
Для получения выполняем команду   
$ kubectl get secrets -n gitlab-managed-apps

Вывод команды будет таким:

NAMESPACE NAME TYPE

gitlab-managed-apps default-token-d6qds kubernetes.io/service-account-token

gitlab-managed-apps regcred kubernetes.io/dockerconfigjson

gitlab-managed-apps runner-gitlab-runner-token-69fxw kubernetes.io/service-account-token   
  
Нужно выбрать secret default-token-d6qds и выполнить команду

$ kubectl describe secret default-token-d6qds -n gitlab-managed-apps

Скопировать значение поля токен и вставить в поле USER\_TOKEN

1. Нужно настроить формирование контекста в файле .gitlab.ci

В секции deploy нужно выполнить следующие стейтменты формирования контекста

- kubectl config set-cluster k8s --server="${SERVER}"

- kubectl config set clusters.k8s.certificate-authority-data "${CERTIFICATE\_AUTHORITY\_DATA}"

- kubectl config set-credentials default --token="${USER\_TOKEN}"

- kubectl config set-context default --cluster=kubernetes --user=default

- kubectl config use-context default

Значения SERVER, CERTIFICATE\_AUTHORITY\_DATA, USER\_TOKEN опрежделены в п 6. default – название service account, kubernetes – название кластера.

В результате выполнения .gitlab.ci из проекта происходит следующее:

1. Проект собирается maven’ом, на выходе получается jar
2. Образ собирается docker’ом и публикуется в private docker registry
3. Образ забирается из registry с помощью kubectl и публикуется на нодах кластера kubernetes. В примере публикуется 5 образов на разных нодах кластера
4. **Особенности конфигурирования gitlab, кластера kubernetes и private** **docker registry в новых версиях**
5. Ocoбенности установки docker registry. Версия docker 20.10.XX

a. Необходимо выпустить сертификат (self-signed certificate) с обязательным флагом

addext "subjectAltName = DNS:dev-. Версия docker 20.10.XX

docker-registry"

Здесь dev-docker-registry FQN машины с установленным docker registry; -days 365 означает что сертификат выпускается на один год.

Пример стейтмента:

openssl req \

-newkey rsa:4096 -nodes -sha256 -keyout certs/domain.key \

-addext "subjectAltName = DNS:dev-docker-registry" \

-x509 -days 365 -out certs/domain.crt

b. Скопировать файл domain.crt в директорию /etc/docker/certs.d/dev-docker-registry\:5000/ и переименовать в ca.crt

c. Генерация файла паролей и запуск docker registry осуществляется командой из п. I.1  
Остальные шаги конфигурации не отличаются от п. I.1.

В результате со всех нод kubernetes и с kubernetes master должна успешно выполняться команда

docker login -u test -p testpassword dev-docker-registry:5000

Ссылка на документацию:

https://docs.docker.com/registry/insecure/

##Use self-signed certificates

1. Ocoбенности установки gitlab. Версия gitlab 14.XX
2. Инсталляция выполняется командой:

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

Первоначальный пароль доступен в файле

/etc/gitlab/initial\_root\_password

1. Создание kubernetes cluster.

GitlabUI/Admin Area/Kubernetes/Connect Cluster with Certificate/Connect exisiting cluster.

Далее по инструкции.

Ссылки на документацию:

https://about.gitlab.com/install/#ubuntu - установка gitlab

1. Ocoбенности установки gitlab-runner. Версия gitlab-runner 14.XX
2. Создание gitlab-runner   
   GitlabUI/Admin Area/Overview/Runners/  
     
   При создании gitlab-runner будет доступно поле registrationtoken. Нужно запомнить значение. Оно будет нужно в переменной runnerRegistrationToken ниже
3. На машине с установленным kubernetes master нужно установить клиент helm.  
   Использовался вариант установки From Script. Документацию см ниже.  
   $ curl -fsSL -o get\_helm.sh <https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3>  
   $ chmod 700 get\_helm.sh  
   $ ./get\_helm.sh
4. Добавить репозиторий gitlab-runner в helm.

helm repo add gitlab https://charts.gitlab.io  
Скачать файл values.yaml по адресу <https://gitlab.com/gitlab-org/charts/gitlab-runner/blob/main/values.yaml> и отредактировать его. Необходимо определить следующие параметры:

gitlabUrl: <http://192.168.103.244:8887/> ##url api gitlab server

runnerRegistrationToken: rpLsW\_eexK4Mkt6JSF3r

runnerRegistrationToken – можно получить при .регистрации кластера kubernetes в UI Gitlab Admin. См п II.2

1. Добавить инсталляцию gitlab-runner в существующее пространство имен kubernetes gitlab-runner и в docker containter одной из нод kubernetes

helm install --namespace gitlab-runner gitlab-runner -f values.yaml gitlab/gitlab-runner

Создание пространства имен kubernetes gitlab-runner описано в документе ниже

Ссылки на документацию:

<https://helm.sh/docs/intro/install/> - установка клиента helm

https://docs.gitlab.com/runner/install/kubernetes.html#configuring-gitlab-runner-using-the-helm-chart - установка gitlab runner

https://git.mos.ru/kgh/eirc/arch/w-gitlab-ci-cd-test/-/blob/master/doc/%D0%A0%D0%B0%D0%B7%D0%B2%D0%B5%D1%80%D1%82%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D1%81%D0%B5%D1%80%D0%B2%D0%B8%D1%81%D0%B0%20%D0%B2%20%D1%81%D1%80%D0%B5%D0%B4%D0%B5%20GitLab.docx - создание пространства имен kubernetes gitlab-runner

1. Ocoбенности установки kubernetes. . Версия kubernetes 1.4.XX

После выполнения шага 3 и кластере kubernetes будет сконфигурировано пространство имен gitlab\_runner и pod gitlab\_runner. На одной из нод кластера kubernetes будет запущен docker контейнер gitlab\_runner. Нужно сделать следующие модификации:

1. Использовав команду

docker exec -it 9692dda7fc08 sh

нужно войти в shell container’а и отредактировать файл config.toml

vi /home/gitlab-runner/.gitlab-runner/config.toml

9692dda7fc08 – это id container’а. Нужно использовать правильный.

В файл config.toml нужно вставить следующие строчки в секцию [runners.kubernetes]

[[runners.kubernetes.volumes.host\_path]]

name = "docker"

mount\_path = "/var/run/docker.sock"

host\_path = "/var/run/docker.sock"

1. Нужно создать secret командой отличной от п I.4

kubectl create secret docker-registry regcred --docker-server=dev-docker-registry --docker-username=test --docker-password==testpassword

Пункт II.4.a является своеобразным обходным маневром. Регулярным путем конфигурации является модификация файла values.yaml из п II.3.c. путем включения в него строк

config: |

[[runners]]

[runners.kubernetes]

namespace = "{{.Release.Namespace}}"

image = "ubuntu:16.04"

[[runners.kubernetes.volumes.host\_path]]

name = "docker"

mount\_path = "/var/run/docker.sock"

host\_path = "/var/run/docker.sock"

Ссылки на документацию:

<https://helm.sh/docs/intro/install/> - установка клиента helm