**kubectl** - Инструмент командной строки, позволяет запускать команды для кластеров Kubernetes

**kubelet *—***сервис, который запускается и работает на каждом узле кластера. Следит за работоспособностью подов.

**kubeadm *—***утилита для управления кластером Kubernetes.

**LABELS - МЕТКИ**

kubectl get pods --show-labels #просмотр подов с метками -l вывод опр меток)

kubectl get pods -L prod #большая L метки через запятую- просмотр меток prod в собств столбце

kubectl get pods -l prod=ver1 #малая l - вывод списком подов с меткой prod=ver1

kubectl get pods -l ‘!env’ #вывод подов не имеющих метку env

kubectl label pods hello prod=ver1 #добавить к поду hello метку prod=ver1

kubectl label pods hello prod=ver2 --overwrite #заменить метку с ver1 на ver2

kubectl delete pods -l prod=ver1 #удалить все поды с меткой prod=ver1

-----

spec: #спецификация запуска пода на ноде с меткой gpu=true в блоке описания пода

nodeSelector:

gpu: "true"

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**NAMESPACE - ПРОСТРАНСТВА ИМЁН --- по умолчанию, если не вводить название пространства выводит из default, можно разгруппировать поды по разным NS (например по специалистам работающим в кластере), внутри разных NS могут быть поды с одинаковыми именами**

kubectl get namespace (или ns) #просмотр пространств имен

kubectl get po --namespace (или -n) kube-system #просмотр подов в пространстве имен kube-system

kubectl describe po -n kube-system имя-пода #просмотр пода в пространстве имен kube-system

kubectl get pods --all-namespaces -o wide #просмотр подов во всех простанствах имен

kubectl apply -f select.yml -n andrey-namespace #создание пода в пространсте имен andrey-namespace

kubectl -n kube-system logs <имя нода> #просмотр логов в пространстве имен kube-system

**--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#REPLICATION CONTROLLER (устаревший способ репликации)**

**Управляет подами по меткам выбранным в селекторе меток, если ИЗМЕНИТЬ метку в поде - он выведется из под контроля RC и создасть новый под с меткой из селектора**

**Добавив метку в RC поды не удалятся, если удалить самому - добавится новый под с уже новыми метками**

kubectl edit rc replicacontr #редактирования созданного RC replicacontr

kubectl delete rc replicacontr --cascade=orphan #удалить RC без удаления подов

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#REPLICA SET**

**В отличии от RC может работать с метками подов у которых разные ключи, выбирать поды в которых отсутсвует опр метка, либо содержащие опр ключ метки или группу ключей например env=\***

selector:

matchExpressions:

- key: app #селектор требует, чтобы под содержал метку с ключом app

operator: In #In - должно совпадать с value, NotIn - не должно совпадать, Exists - под должен иметь метку с указанным ключом (в этом операторе не нужно поле value)

values: #DoesNotExist - под не должен содержать метку с указанным ключом, свойство value не указывается

- prod #значение метки должно быть prod

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**DAEMONSET**

**#DAEMONSET запускает один под на каждом выбранном узле, при присоединении нового узла, добавит под.**

**Используется nodeSelector для выбора типа узлов по меткам. Например установить ssd мониторинг на узлы с ssd дисками, после создание DS можно добавить метку на узел, туда установится под**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**StatefulSet**

Запускает приложения с сохранением состояния (например базы данных) каждый под и его имя уникальные (в отличии от deployment где все поды одинаковые)

Имена выдаются по порядку от нуля: pod-0 pod-1 pod-2 без пропусков (если удалить pod-1 создастся такой же а не pod-3)

При удалении не удаляется PVC, при удалении/создании подов новые поды будут использовать старые pvc (старые данные)

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**Taints и tolerations**

**Taints: на НОДЕ** зараза - объявляем часть узлов заразными и k8s перестаёт запускать на этих нодах поды, кроме тех подов в которых есть сопротивление к заразе (**Tolerations)**

**Ключ:значение**

NoSchedule не устанавливать поды на ноде с этой заразой

NoExecute действует так же и на запущенные поды, если на ноде появляется зараза, то поды без сопротивления удаляются(если из RS то запускаются на др нодах)

**Tolerations: на ПОДЕ** сопротивление заразе

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**STATIC POD запускается НЕ из API а манифест файлом из /etc/kubernetes/manifest**

Удаляется, редактируется тоже только в файле на ноде в папке

Например запускает на мастер нодах кластерные поды etcd api controller scheduler

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#JOBS запуск модулей выполняющих одну заканчиваемую задачу (неперезапускаемого контейнера). При отключении узла, передается на другой пока не завершит работу.**

apiVersion: batch/v1

kind: Job

metadata:

name: multi-completion-batch-job

spec:

completions: 5 # Установка completions равной 5заставляет это задание выполнитьпоследовательно пять модулей

parallelism: 2 # задание создаёт два пода и запускает их параллельно

template:

metadata:

labels:

app: batch-job

spec:

restartPolicy: OnFailure

containers:

- name: main

image: luksa/batch-job

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#CRONJOBS запуск job по расписанию, которая (job) запускает поды**

apiVersion: batch/v1beta1

kind: CronJob

metadata:

name: hello

spec:

schedule: "\*/1 \* \* \* \*" #расписание, тут каждую минуту

concurrencyPolicy: Allow #если за предыдущий под еще не запустился, allow разрешает запустить следующий паралельно Forbid запрещает паралельный запуск

jobTemplate: #Replace если первый не успел запуститься, завершить его и запустить новый

spec:

backoffLimit: 2 #две попытки запуска

activeDeadlineSeconds: 100 #100 секунд ожидания запуска

template:

spec:

containers:

- name: hello

image: quay.io/prometheus/busybox

args:

- /bin/sh

- -c

- date; echo Hello from the Kubernetes cluster

restartPolicy: Never

**Successful Job History Limit: 3** #в дискрайб кронджоба - кол-во сохраняемых джобов

**Failed Job History Limit: 1** #сохраняет один крашнутый джоб

**startingDeadlineSeconds 20** #время на которое можно просрочить запуск джоба

после 100 неудачных попыток запуска джоба - кронджоб считается нерабочим и выключается

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#SERVICES - доступ к группе подов. Служба имеет не меняющиеся ip и порт с которых маршрутизируются запросы в один подов этой службы (с балансировкой между всеми подами)**

Kubectl get endpoints *имя\_службы* #просмотр конечных точек службы - адреса и порт подов подкл к службе

kubectl get svc (services) #просмотр сервисов

kubectl expose deploy andrey-deploy --type=ClusterIP --port 80 #открыть порт 80 внутри кластера

kubectl expose deploy andrey-deploy --type=NodePort --port 80 #открыть порт наружу на всех нодах

kubectl expose deploy andrey-deploy --type=LoadBalancer --port 80 #балансировщик нагрузки (в облаке)

**-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

**#DEPLOYMENTS - развертывания позволяют управлять экземплярами подов, контролируется их восстановление, автобалансировка нагрузки**

**Деплойментс создает репилка сет, реплика сет создёт поды. В отличии от RS можно менять image в deployment (обновить версию)**

kubectl create deploy andrey-deploy --image x3m5555/k8s #создать развертывание из имеджа x3m5555/k8s

kubectl scale deploy andrey-deploy --replicas 4 # масшатабирование развертывания andrey-deploy на 4шт

kubectl autoscale deploy andrey-deploy --min=4 --max=6 --cpu-percent=80 #автоматически масштабировать при загрузке проца больше 80

kubectl get hpa #просмотр параметров горизонтальной балансировки HorizontalPodAutoscaler

kubectl get rs #просмотр реплик ReplicaSet

kubectl set Image deploy/andrey-deploy k8s=nginx --record #изменить образ для развертывания на nginx с записью в историю изменений (--record)

kubectl rollout history deploy/andrey-deploy #история изменения развертывания andrey-deploy

kubectl rollout status deploy/andrey-deploy #показать статус

kubectl rollout undo deploy/andrey-deploy #вернуть на предыдущую версию

kubectl rollout undo deploy/andrey-deploy --to-revision=4 #вернуть на 4ую версию из истории изменений

kubectl rollout restart deploy/andrey-deploy #перегрузить версию развертывания

kubectl apply -f create-deploy-autoscaling.yml #запустить из манифест файла развертывание

kubectl delete -f create-deploy-autoscaling.yml # удалить все что было сделано манифест файлом

kubectl delete deploy --all #удалить все развертывания

**-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------**

kubectl get nodes #просмотр нодов

kubectl get pods #просмотр подов

kubectl get deploy #просмотр развёртываний

kubect run hello --image=x3m5555/k8s --port=80 #создать под с названием hello из docker образа x3m5555/k8s и открыть порт 80

kubectl delete pods hello #удалить под hello

kubectl describe pods hello #полная информация о поде hello

kubectl exec -it hello -- sh #запустить в поде команду (sh оболочку)

kubectl logs hello #просмотр логов пода hello

kubectl port-forward hello --address 192.168.0.39 7788:80 #форвард порта 80(докера) на внешний порт 7788

kubectl apply -f pod-myweb-ver3.yml #создать объекты из манифест файла pod-myweb-ver3.yml

kubectl delete -f pod-myweb-ver3.yml #удалить объекты из манифест файла pod-myweb-ver3.yml

kubectl delete all --all #удалить ресурсы всех типов all и их экземпляров --all (не все ресурсы, например secret надо явно)

**kubectl logs -n kube-system deploy/metrics-server #логи сервера метрик**

kubectl api-resources #API ресурсы

kubectl explain pods(hpa и тд) #просмотр полей объектов (kind, metadata, spec, status)

kubectl explain pod.spec #просмотр отдельно поля spec для подов

kubectl get pv #просмотр томов PersistentVolume

kubectl get pvc #просмотр заявок PersistentVolumeClaim

kubectl get apiservice #просмтор API сервисов

**П**росмотр сертификатов - kubeadm cert check-expiration

**minikube** - это инструмент, который позволяет запускать Kubernetes локально, запускает одноузловой кластер Kubernetes на вашем персональном компьютере

место хранения настройек, конфигураций:

C:\Users\Admin\.kube #конфигурация kubectl с k8s cluster

C:\Users\Admin\.minikube #файлы ISO и остальное для создания кластера

kubeadm init --pod-network-cidr=10.244.0.0/16 #Инициализация кластера:

kubeadm token create --print-join-command #Команда для получения команды по подключению рабочей ноды:

source <(kubectl completion bash | sed s/kubectl/k/g) #автодополнение команд по TAB

**#команды**

minikube version #показать версию

minikube start (stop delete) # запуск кластера (остановить удалить)

minikube start --cpus=2 --memory=1800mb --disk-size=5gb #запуск с настройками железа

kubectl get componentstatuses # состояние k8s (статус компонентов)

kubectl cluster-info # информация о кластере

kubectl get nodes #из каких серверов состоит кластер

minikube ssh #залогиниться в докер из консоли

- сделать логин на вирт машину с кластером

username: docker

password: tcuser

или root без пароля