Федеральное государственное автономное образовательное учреждение высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

# ОТЧЕТ ПО ПРАКТИЧЕСКОМУ ЗАДАНИЮ №7

**«Знакомство с Kubernetes»**

Практическая работа

по дисциплине «Современные технологии программирования» студента 1 курса группы ПИ-б-о-231

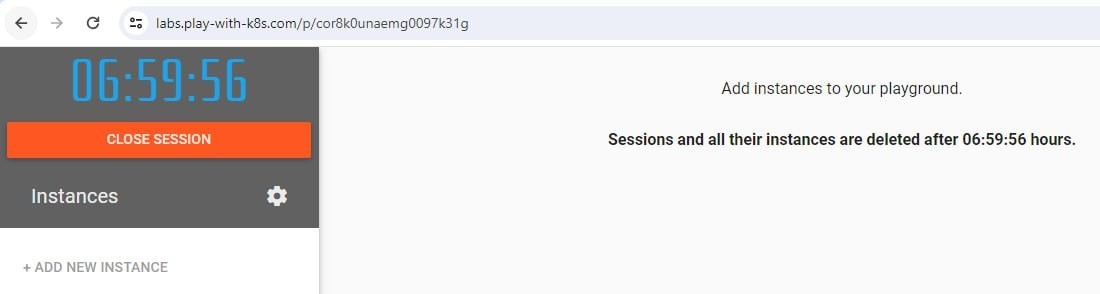
Покидько Максим Сергеевич

направления подготовки 09.03.04 «Программная инженерия»

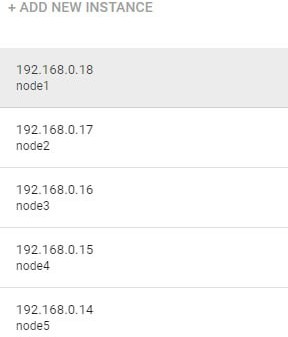
Симферополь, 2024

**Цель:** Ознакомиться на практике с инструментом оркестрации контейнерирированных приложений Kubernetes.

# Ход выполнения задания: Объединение серверов в кластер

1. Перейдём на сайт https://labs.play-with-k8s.com/ и залогинимся при помощи учётной записи GitHub
2. После того, как мы нажали на кнопку "Start" для нас создана 7-и часовая сессия, в пределах которой можно будет создать до 5-ти серверов (node) с уже установленными компонентами kubernetes.

Создадим максимум нод при помощи кнопки "Add new instance"



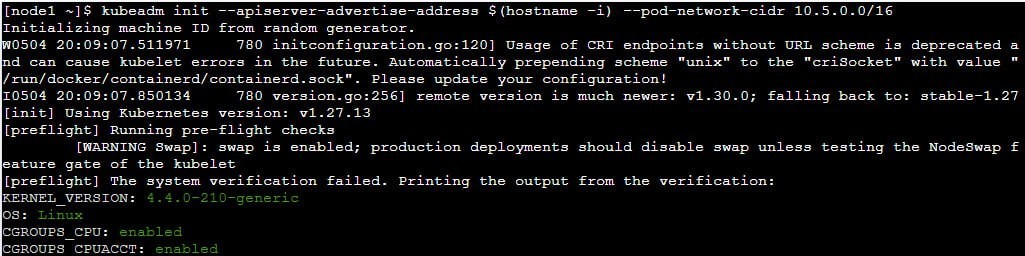
1. Каждую ноду можно воспринимать как отдельный сервер подключённый к подсети 192.168.0.0. Несмотря на то, что это локальная сеть (т.е. недоступная из интернета) каждой ноде присвоено доменное имя (url) на поддомене

direct.labs.play-with-k8s.com по которому уже можно будет достучаться до сервера

1. На каждой ноде мы увидим одинаковое приветствие с предложением выполнить несколько команд, которые (1) инициируют на машине мастер- ноду, (2) создадут сеть внутри кластера и (3) установят приложение "nginx"
2. В данном случае мы будем создавать кластер с одной мастер-нодай (хотя можно и больше) и 4-мя рабочими нодами. Поэтому:

На первой ноде запустим команду (Ctrl + Shift + С для вставки):

# kubeadm init --apiserver-advertise-address $(hostname -i) --pod-network-cidr 10.5.0.0/16

Это приведёт к инициализации на текущей машине управляющих компонент кластера ([Control Plane](Control Plane)).

После завершения процесса инициализации нас попросят указать путь к конфиг-файлу кластера. По умолчанию, kubernetes ищет конфиг в рабочем каталоге пользователя в папке ~/.kube/config или в переменной окружения KUBECONFIG.

Мы пропускаем этот этап, т.к. kubernetes сам создал в нашем рабочем каталоге файл с конфигом, но это произошло потому, что мы запустили процедуру установки из под этого пользователя, и если бы мы хотели передать возможность работать с нашим кластером другому пользователю, то конфиг нужно было бы скопировать ему в рабочий каталог. Т.е. под выбранным для управления кластером пользователем нужно было бы выполнить команды:

# mkdir -p $HOME/.kube

**sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config**

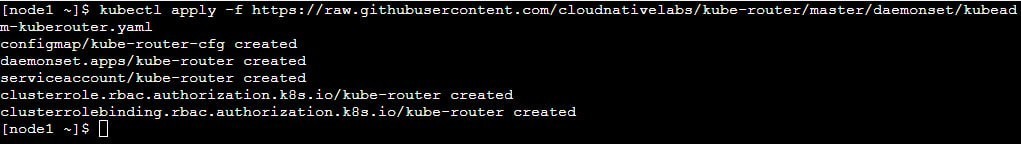
Здесь же будет показана строка которую нужно будет запускать на рабочих узлах для подключения к кластеру. Найдём её и скопируем в текстовый файл, она нам ещё пригодится. В моём случае она выглядит так:



На мастер-ноде запустим команду:

# kubectl apply -f https://raw.githubusercontent.com/cloudnativelabs/kube- router/master/daemonset/kubeadm-kuberouter.yaml

Она приведёт к инициализации сети внутри кластера

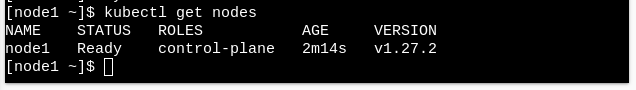


Теперь у нас есть полностью готовая к работе мастер-нода

1. Выполним команду:

# kubectl get nodes

В результате мы увидим список подключённых к кластеру нод. В нашем случае, в списке, пока что доступна только одна нода



1. Выполним команду:

# kubectl get pods -A

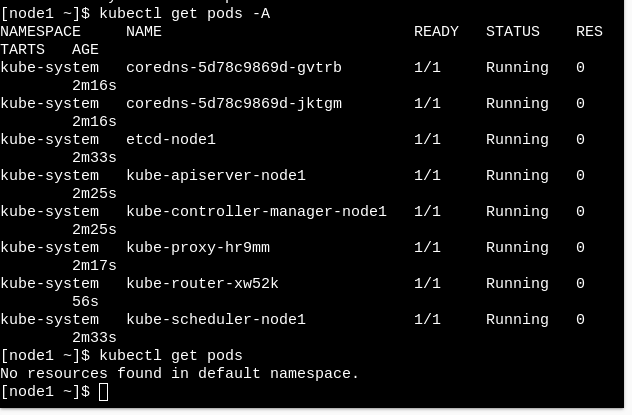
а затем

# kubectl get pods

В результате мы увидим полный (-A) список подов запущенных в кластере и только те поды, которые находятся в пространстве имён "default". В первом

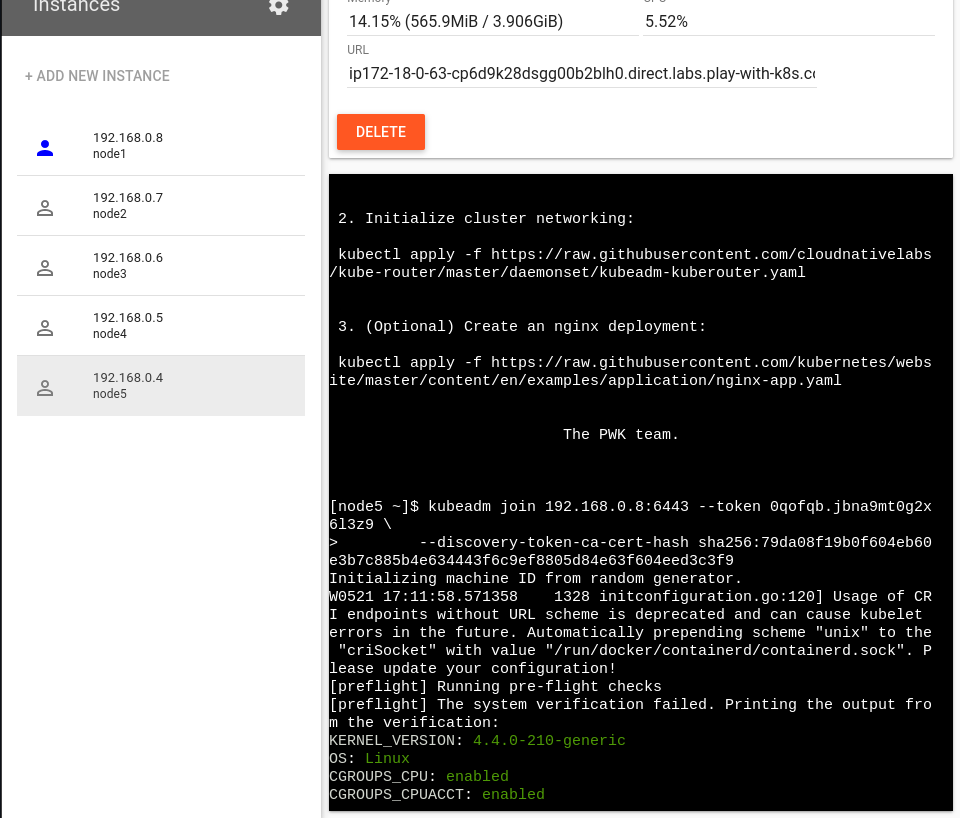
приближении поды можно воспринимать как аналог процессов на обычном компьютере.

Все поды распределяются по пространствам имён (namespaces) полный список которых можно посмотреть при помощи команды: kubectl get namespaces. Если команды выполняется без явного указания пространства имён (ключ --namespace), то подразумевается пространство имён "default"



1. Подключим остальные сервера к кластеру.

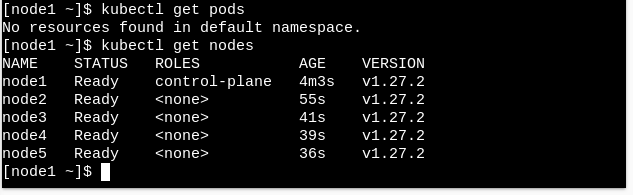
Для этого на оставшихся 4-х нодах запустим команду kubeadm join с параметрами, которые мы копировали ранее в текстовый файл.



1. Перейдём на мастер-ноду и снова выполним команду:

# kubectl get nodes

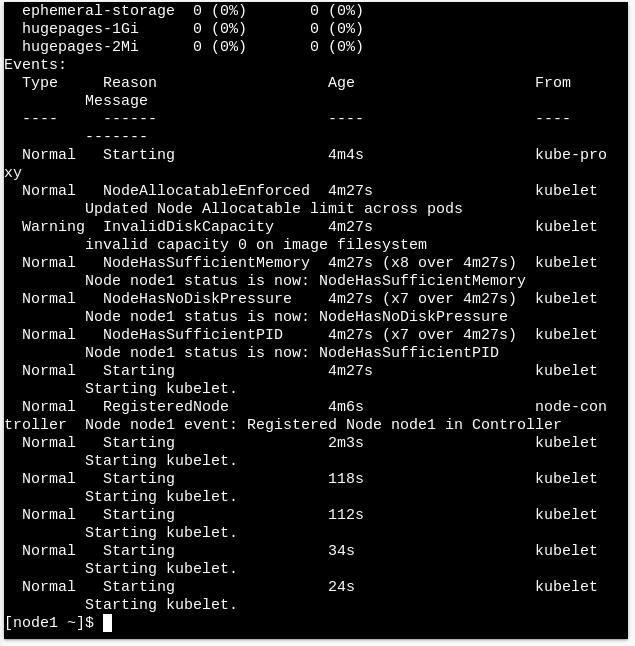
В результате видим, что в списке теперь 5 нод



1. Выполним команду:

# kubectl describe nodes node1

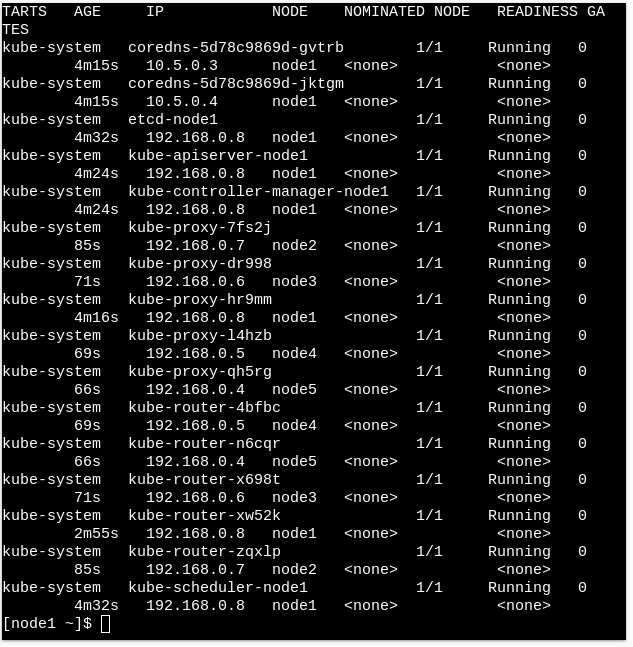
Данная команда позволит нам получить подробную информацию по ноде с именем "node1"



1. Выполним команду:

# kubectl get pods -o wide -A

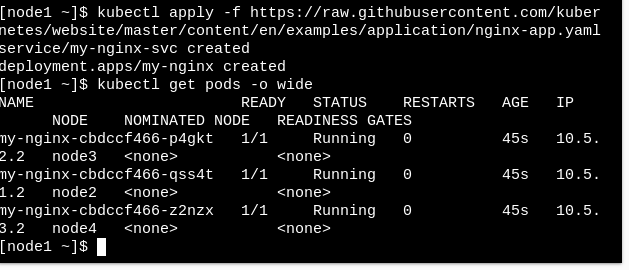
В результате получим список всех подов с расширенной информацией по каждому. Обратим внимание, что некоторые поды автоматически запустились на новых нодах



1. Выполните команду:

# kubectl apply -f https://raw.githubusercontent.com/kubernetes/website/master/content/en/ examples/application/nginx-app.yaml

В результате будет запущено 3 пода с сервером "nginx". Проверим это при помощи команды kubectl get pods -o wide. В столбце "Status" указано "Running"

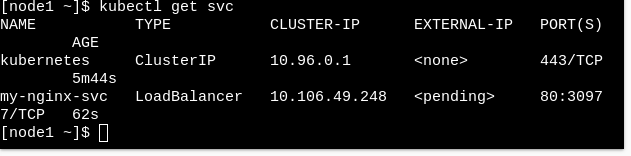


1. Обычно поды "заперты" во внутренней сети кластера и снаружи к ним доступа нет. Но командой выше мы не только запустили 3 пода с "nginx", но и создали элемент кластера "Service", который пробросил некоторый внешний порт на 80й порт внутри кластера. Т.к. "nginx" - это веб-сервер то он слушает как раз 80й порт. Определим, какой же порт проброшен.

Для этого введём команду:

# kubectl get svc

Эта команда покажет список элементов типа "Service". Найдём в нём сервис типа "LoadBalancer" с названием "my-nginx-svc" и в столбце "Port" определите внешний порт (у меня это 31231)

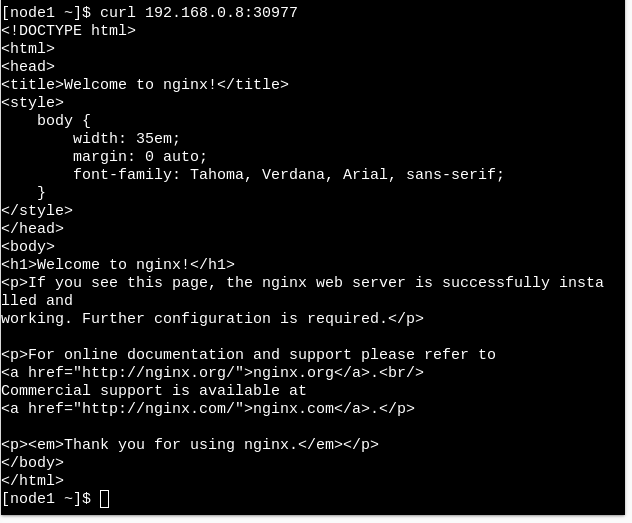


1. Проверим, что "nginx" работает при помощи утилиты curl:

# curl 192.168.0.18:31231

Здесь 192.168.0.18 - это ip-адрес моей мастер-ноды, а 31231 - проброшенный внутрь порт.

В результате видим код html-страницы с приветствием от "nginx"

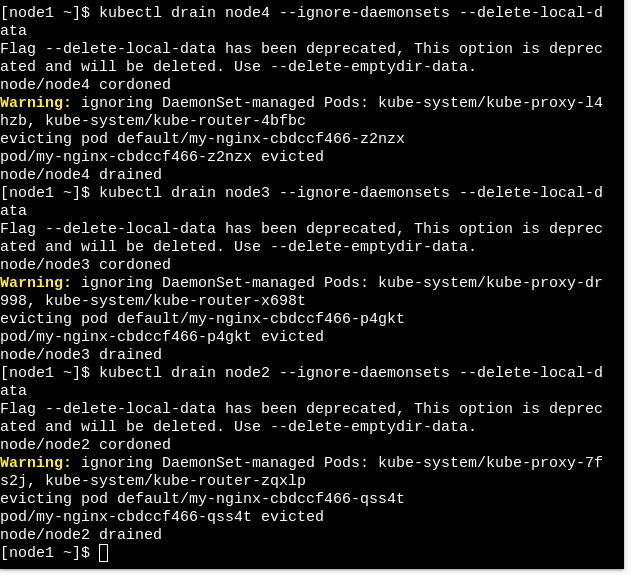


1. Отключим одну или несколько нод на которых работают поды с "nginx". Определим на каких нодах запущены поды с "nginx" (у меня это node4, node5, node2).

Введём команду (со своим именем ноды):

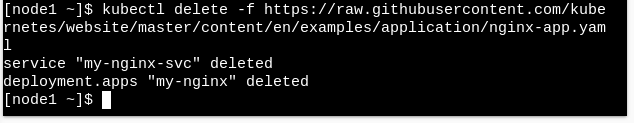
# kubectl drain node4 --ignore-daemonsets --delete-local-data

Данная команда проинформирует ноду о том, что нужно завершить все свои поды и в результате они будут автоматически запущены на оставшихся нодах.

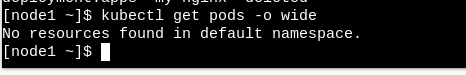


1. Остановим запущенное приложение "nginx". Для этого выполним команду запуска, но вместо applay укажем delete, т.е.:

# kubectl delete –f https://raw.githubusercontent.com/kubernetes/website/master/content/en/ examples/application/nginx-app.yaml



1. Проверим список запущенных подов - он пуст



Вопросы:

* 1. Чтобы узнать IP-адрес ноды по её имени в кластере, вы можете воспользоваться командой kubectl get pods -o wide, если используете Kubernetes. Эта команда покажет IP-адреса всех подов в вашем кластере, включая их имена.
  2. Предварительный этап для запуска приложения из исходных кодов в кластере может включать настройку среды разработки, установку необходимых зависимостей и инструментов сборки, а также создание манифестов Kubernetes для вашего приложения.
  3. Для подключения рабочей ноды к кластеру нужно знать IP-адреса мастер-нод, учетные данные для аутентификации (например, сертификаты или токены доступа) и соответствующие разрешения (RBAC), если они используются в вашем кластере.
  4. Да, в одном кластере может быть несколько мастер-нод. Это обеспечивает отказоустойчивость и распределение нагрузки.
  5. Да, кластер может состоять только из мастер-ноды, которая одновременно является и рабочей нодой. Это часто используется в небольших или тестовых средах, где нет необходимости в выделенных рабочих нодах. Однако в продукционных средах обычно используется не менее двух мастер-нод для обеспечения отказоустойчивости.