```
 OS Image:                   CentOS Linux 7 (Core)
 Operating System:           linux
 Architecture:               amd64
 Container Runtime Version:  docker://18.9.7
 Kubelet Version:            v1.15.3
 Kube-Proxy Version:         v1.15.3
PodCIDR:                     10.233.64.0/24
Non-terminated Pods:         (6 in total)
  Namespace                  Name                                CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                  ----                                ------------  ----------  ---------------  -------------  ---
  kube-system                kube-apiserver-kub-master-1         250m (13%)    0 (0%)      0 (0%)           0 (0%)         9m35s
  kube-system                kube-controller-manager-kub-master-1  200m (11%)  0 (0%)      0 (0%)           0 (0%)         9m39s
  kube-system                kube-flannel-6tqmg                  150m (8%)     300m (16%)  64M (1%)         500M (14%)     6m32s
  kube-system                kube-proxy-ld95b                    0 (0%)        0 (0%)      0 (0%)           0 (0%)         7m17s
  kube-system                kube-scheduler-kub-master-1         100m (5%)     0 (0%)      0 (0%)           0 (0%)         9m27s
  kube-system                nodelocaldns-s9mvz                  100m (5%)     0 (0%)      70Mi (2%)        170Mi (5%)     5m18s
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests         Limits
  --------           --------         ------
  cpu                800m (44%)       300m (16%)
  memory             137400320 (4%)   678257920 (20%)
  ephemeral-storage  0 (0%)           0 (0%)
Events:
  Type    Reason                   Age                  From                     Message
  ----    ------                   ----                 ----                     -------
  Normal  NodeHasSufficientMemory  10m (x8 over 10m)    kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientMemory
  Normal  NodeHasNoDiskPressure    10m (x8 over 10m)    kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasNoDiskPressure
  Normal  NodeHasSufficientPID     10m (x7 over 10m)    kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientPID
  Normal  Starting                 10m                  kube-proxy, kub-master-1 Starting kube-proxy.
  Normal  Starting                 8m12s                kubelet, kub-master-1    Starting kubelet.
  Normal  NodeHasSufficientMemory  8m12s                kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientMemory
  Normal  NodeHasNoDiskPressure    8m12s                kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasNoDiskPressure
  Normal  NodeHasSufficientPID     8m12s                kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientPID
  Normal  NodeAllocatableEnforced  8m11s                kubelet, kub-master-1    Updated Node Allocatable limit across pods
  Normal  Starting                 7m28s                kube-proxy, kub-master-1 Starting kube-proxy.
  Normal  NodeReady                6m31s                kubelet, kub-master-1    Node kub-master-1 status is now: NodeReady
[root@kub-master-1 kubespray]#
```

Сегодня открываю новый цикл статей, которые будут иметь непосредственное отношение к Devops и всему, что с этим связано. Начну с того, что расскажу, как установить и запустить кластер Kubernetes на собственном железе. Буду использовать установку через Kubespray с использованием ingress контроллера в кластере.

Содержание:

Цели статьи

1. Коротко рассказать о том, что такое Kubernetes и для чего он нужен.
2. Перечислить основные системные требования для разворачивания своего кластера Kubernetes.
3. Выполнить непосредственно установку кластера на свое железо.

Введение

Хочу сразу обратить внимание на важный момент. У меня нет опыта промышленной эксплуатации кластера Kubernetes. Я нахожусь в состоянии обучения и исследования этого инструмента. Я прошел обучение Слёрм, это дало базовые знания и понимание принципов работы. Дальше стал разворачивать свои кластера и исследовать их. Изначально я не хотел писать статьи по этой теме до тех пор, пока не накопится достаточного опыта, но сейчас поменял свое мнение.

В рунете очень мало материалов по kubernetes с конкретикой и практикой, по которым можно было бы учиться. Думаю, что даже те знания, что есть сейчас у меня, будут многим полезны и интересны. Плюс, когда пишешь статьи, систематизируешь свои знания, запоминаешь и получаешь обратную связь. Ускоряется процесс обучения. Так что статьям по kubernetes и devops в целом быть. Думаю, что в ближайшее время я сфокусируюсь именно на этом.

Могу однозначно сказать, что если у вас есть необходимость в продакшене использовать Kubernetes, не тяните время и не откладывайте. Идите учиться на курсы. Самостоятельно вы не освоите в достаточном объеме материал, чтобы можно было переносить рабочую нагрузку в свой кластер. Очень много нюансов и подводных камней. Вы потратите больше времени, нервов и денег на самостоятельное освоение, если будете сами все с нуля изучать.

Лично у меня сейчас нет цели становиться администратором Kubernetes. Мой формат занятости не подразумевает обслуживание таких крупных систем и в планах этого тоже пока нет. Мне просто любопытно его исследовать, узнавать что-то новое, поэтому я этим занимаюсь. Знания карман не тянут, особенно современные и востребованные.

Kubernetes простыми словами

Попробую рассказать своими словами, что такое кластер Kubernetes для чайников, без отсылок к описаниям и документации. По своей сути это кластер для обслуживания docker контейнеров. Я слышал, что он может управлять не только докером, но практически ничего про это не знаю. Все в основном используют Kubernetes в связке с Docker.

В Kubernetes все крутится вокруг докер контейнеров. Это инструмент для их запуска, поднятия в случае падения, распределения ресурсов и т.д. Под капотом никакой магии. Там обычный docker, iptables, etcd, nat, dns, ceph, nfs и т.д. Просто все собрано в одном месте для решения конкретных задач. Таким образом, для эффективного управления кластером кубера нужен хороший бэкграунд классического linux админа. Без этих знаний будет трудно.

Есть много способов разворачивания кластера, так как он модульный. Не всем и не всегда нужны все его компоненты. К примеру, есть ingress контроллер для распределения входящих запросов по сервисам. Под капотом там обычный nginx в режиме proxy_pass, интегрированный в инфраструктуру кластера. Реализация сети в кластере тоже может быть разной — на уровне l2 или l3 с помощью тех или иных технологий. То же самое с файловыми хранилищами —

локальные хранилища серверов, nfs хранилища, ceph и т.д.

В зависимости от того, какой функционал вам нужен, выбирается способ установки кластера kubernetes. Мы можете его установить полностью вручную, добавляя один компонент за другим. А можно использовать готовое средство, к примеру Kubespray, где весь необходимый для установки функционал реализуется с помощью ролей ansible. На Слёрме нас учили ставить кластер, используя свой форк компании southbridge. Они там немного изменили функционал под свои потребности. Я ставил и по их форку, и по оригинальному Kubespray. Основное отличие от классического Kubespray в том, что не используется kubeadm и сертификаты для общения компонентов кластера сразу выпускаются то ли на 10, то ли на 100 лет, не помню точно. В Kubespray сертификаты выписываются только на год и надо отдельно следить за их актуальности и своевременно обновлять.

Подведу итог о том, что же такое Kubernetes. Кубернетис — средство оркестрации (управления) контейнерами Docker. Это удобный инструмент для их автоматического запуска, выделения ресурсов, контроля состояния, обновления.

Кому нужен Kubernetes

Теперь порассуждаем о том, кому может пригодиться Kubernetes. В первую очередь это крупные компании со своими разработками в ИТ и командами программистов, для которых нужна большая производственная среда. Кластер Кубера добавляет серьезные накладные расходы на свое содержание, поэтому в небольших проектах выгоды от него не будет. Нет смысла объединить 5 маленьких виртуалок в кластер и эксплуатировать его. Если только для тестов. Или если вы точно уверены, что у проекта будет серьезный рост в ближайшее время. Под небольшую структуру и нагрузки лучше подыскать решения попроще, чем кубернетис.

Kubernetes накладывает серьезные требования к приложениям, которые в нем работают. Они должны быть изначально спроектированы и написаны по принципу микросервисов. У вас не получится взять и перенести в кластер сайт на wordpress или bitrix, даже если они очень большие и нагруженные. Вернее, перенести то вы их сможете, но вряд ли вам от этого будет проще и удобнее. Основное преимущество кластера — гибкость в разработке, деплое приложения, а так же в распределении ресурсов.

Примерная схема работы с кластером кубернетис на пальцах будет такая:

1. Разработчики какого-то одного сервиса проекта выпускают обновление, запушив его в репозиторий.
2. Система сборки формирует докер контейнер с этими изменениями и кладет в registry.
3. Контейнер уезжает на тесты и если все в порядке, выкатывается в продакшн кластер kubernetes.

Это я очень просто и условно описал. Все процессы после пуша кода в репозиторий могут быть автоматизированы. Команд разработчиков, как и микросервисов, может быть десятки. Они могут условно независимо друг от друга выкатывать обновления. У каждого сервиса могут быть свои языки

программирования, свои системные требования, свои файловые хранилища и базы данных. Все эти сущности отдельно описываются в конфигурациях кластера. Каждый микросервис получает то, что ему нужно для работы.

Теперь смотрим на Bitrix или WordPress. Они являются монолитными приложениями, написанными на php с использованием базы Mysql. В них нет микросервисов. Вам нужно либо как-то разбивать их на части, либо постоянно выкатывать все целиком. Но в этом случае смысл кластера кубернетис теряется, его гибкость настроек и выделения ресурсов под потребности не используются. Вам проще поставить обычный балансер на вход, сделать несколько нод для обработки php и за ними кластер БД.

Резюмируя все сказанное. Kubernetes — нишевое решение под конкретные проекты. Оно подходит далеко не всем и не надо его пихать туда, где от него не будет толку. Думаю, находятся люди, которые так делают. Сужу по тому, что мне в комментариях к некоторым статьям, например, про установку zabbix, пишут, а почему вы не в докере его ставите. Люди не понимают, что такое докер, для чего он нужен и какие проблемы решает. Смысла в использовании zabbix в docker нет никакого вообще. Docker создан для удобной разработки и деплоя приложений в продакшн. Этакий расширенный пакетный менеджер. В первую очередь он инструмент разработчиков.

Системные требования

Как таковых жестких системных требований у Kubernetes нет. Он с очень маленьких установок расширяется до огромных кластеров. Для того, чтобы его просто попробовать и посмотреть, достаточно следующих виртуальных машин:

- 2-3 мастер ноды с 2 cpu и 4 gb ram
- ingress нода с 1 cpu и 2 gb ram
- рабочие ноды для контейнеров от 2 cpu и 4 gb ram

Для того, чтобы просто запустить кластер, достаточно буквально двух виртуальных машин, которые одновременно будут и мастер и рабочими нодами. Но я рекомендую сразу планировать более ли менее полную структуру, которую можно брать за основу для последующего превращения в рабочий кластер. Я буду разворачивать кластер на следующих виртуальных машинах.

Название	IP	CPU	RAM	HDD
kub-master-1	10.1.4.36	2	4G	50G
kub-master-2	10.1.4.37	2	4G	50G
kub-master-3	10.1.4.38	2	4G	50G
kub-ingress-1	10.1.4.39	2	4G	50G

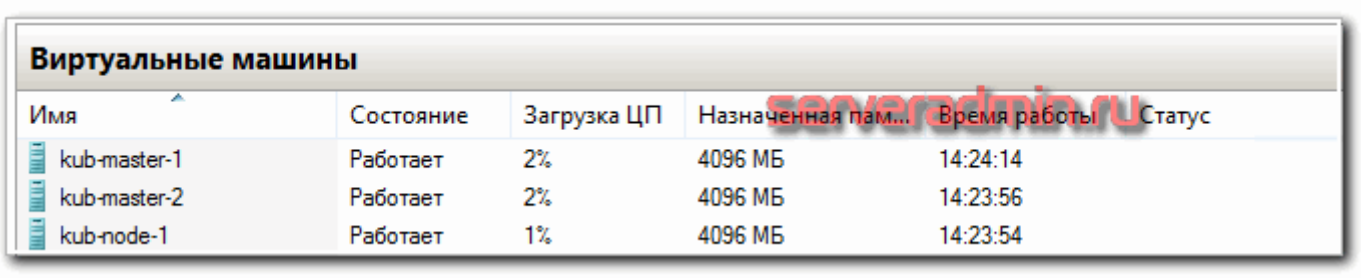
kub-node-1	10.1.4.32	2	4G	50G
kub-node-2	10.1.4.33	2	4G	50G

В моем случае это виртуальные машины на двух гипервизорах Hyper-V. Как я уже сказал в системных требованиях, для теста ресурсов можно и чуть меньше дать, но у меня есть запас, поэтому я такие ресурсы выделил для кластера Kubernetes. Перед установкой кластера рекомендую сделать снепшоты чистых систем, чтобы можно было оперативно вернуться к исходному состоянию, если что-то пойдет не так. Вручную готовить и переустанавливать виртуалки хлопотно.

По гипервизорам виртуальные машины распределил следующим образом.

Виртуальные машины

Имя	Состояние	Загрузка ЦП	Назначенная пам...	Время работы	Статус
kub-ingress-1	Работает	0%	4096 МБ	14:24:32	
kub-master-3	Работает	20%	4096 МБ	00:00:37	
kub-node-2	Работает	27%	4096 МБ	00:00:31	



Упомяну про еще одну рекомендацию. Мастер ноды с etcd дают приличную нагрузку на диск. Их рекомендуется **размещать на быстрых ssd дисках**. Чем больше кластер — тем больше нагрузка. В наших тестах сойдет и hdd диск под мастер. Но если будете использовать в продакшене с учетом расширения и роста, лучше сразу планируйте быстрые диски под мастера.

Подготовка к установке

Кластер Kubernetes я буду разворачивать на виртуальных машинах Centos 7. На них она установлена в минимальной конфигурации. Напоминаю, что установка будет проходить с помощью **Kubespray**. Я рекомендую склонировать к себе репозиторий, чтобы у вас сохранилась версия kubespray, с которой вы устанавливали кластер. Это позволит без проблем создавать копию кластера для тестов, дебага, обновления и т.д. Я для этого использую свой сервер Gitlab. Рекомендую озаботиться его наличием. Он нам очень пригодится и дальше в процессе знакомства и изучения кластера.

На виртуальных машинах нужно отключить следующие сущности:

1. SELinux (привет любителям безопасности, считающим, что selinux отключают только дилетанты).
2. Swap.
3. FirewallD, либо любой другой firewall.

На все сервера должен быть разрешен доступ пользователя root по ssh с одним и тем же паролем.

Установка кластера Kubernetes

Я буду устанавливать кластер Kubernetes с сервера **kub-master-1**. Установим на него некоторые пакеты, которые нам понадобятся в дальнейшем.

```
# yum install mc epel-release
# yum install wget curl git screen python-pip sshpass
```

Теперь клонируем себе локально репозиторий kubespray.

```
# cd ~
# git clone https://github.com/kubernetes-sigs/kubespray
```

Устанавливаем зависимости kubespray через pip, которые перечислены в файле **requirements.txt**.

```
# cd kubespray
# pip install -r requirements.txt
```

Теперь нам нужно заполнить инвентарь ansible, исходя из нашего набора серверов. Для этого скопируем стандартный инвентарь sample и будем редактировать его.

```
# cp -R ~/kubespray/inventory/sample ~/kubespray/inventory/dev
```

Приводим файл **inventory.ini** к следующему виду.

```
kub-master-1 ansible_host=10.1.4.36 ip=10.1.4.36
kub-master-2 ansible_host=10.1.4.37 ip=10.1.4.37
kub-master-3 ansible_host=10.1.4.38 ip=10.1.4.38
kub-ingress-1 ansible_host=10.1.4.39 ip=10.1.4.39
kub-node-1 ansible_host=10.1.4.32 ip=10.1.4.32
```

```
kub-node-2 ansible_host=10.1.4.33 ip=10.1.4.33

[kube-master]
kub-master-1
kub-master-2
kub-master-3

[etcd]
kub-master-1
kub-master-2
kub-master-3

[kube-node]
kub-node-1
kub-node-2
kub-ingress-1

[kube-ingress]
kub-ingress-1

[k8s-cluster:children]
kube-master
kube-node
```

В принципе, тут все понятно, если вы знакомы с работой ansible. Мы распределили хосты по ролям. Обращаю внимание, что сервер ingress по сути является обычной нодой, только с дополнительным функционалом, поэтому он присутствует в том числе в группе kube-node. Далее вы можете его использовать и как ingress, и как обычную ноду одновременно.

Теперь редактируем некоторые параметры. Для удобства восприятия, я их прокомментировал прямо тут, рядом со значениями. Переносить комментарии в реальные конфиги не надо. Они только для инфомрации на сайте. Начнем с файла *~/kubespray/inventory/dev/group_vars/all/all.yml*. Добавляем туда параметры:

```
kubelet_load_modules: true # автоматом загружает модули в ядро системы, не спрашивая админа сервера
kube_read_only_port: 10255 # порт для мониторинга кублетов, нужен, к примеру, для prometeus
```

В файл *~/kubespray/inventory/dev/group_vars/all/docker.yml* добавляем:

```
docker_storage_options: -s overlay2 # использует сторейдж overlay2 для докера
```

В файл *~/kubespray/inventory/dev/group_vars/etcd.yml* добавляем:

```
etcd_memory_limit: 0 # дефолтного ограничения в 512 мб может не хватать в больших кластерах, надо либо увеличить
значение, либо отключить ограничение
```

В файл *~/kubespray/inventory/dev/group_vars/k8s-cluster/k8s-cluster.yml* добавляем:

```
kube_network_plugin: flannel
kube_proxy_mode: iptables
kubeconfig_localhost: true # устанавливаем локально инструменты для управления кластером
```

Я буду использовать сетевой плагин flannel и iptables. Это хорошо проверенное и полностью готовое к production решение. Никаких особых настроек не требует, кроме пары параметров. Добавляем их в файл *~/kubespray/inventory/dev/group_vars/k8s-cluster/k8s-net-flannel.yml*.

```
flannel_interface_regexp: '10\\.1\\.4\\.\\d{1,3}'
flannel_backend_type: "host-gw"
```

В данном случае 10\\.1\\.4\\.\\d{1,3} это регулярное выражение, которое описывает подсеть 10.1.4.0/24, в которой у меня размещены виртуальные машины под кластер. Если у вас подсеть машин для кластера, к примеру, 192.168.55.0, то регулярка будет 192\\.168\\.55\\.\\d{1,3}

Теперь настроим **ingress**. Добавляем параметры в *~/kubespray/inventory/dev/group_vars/k8s-cluster/addons.yml*.

```
ingress_nginx_enabled: true

ingress_nginx_nodeselector:
  node-role.kubernetes.io/ingress: "true" # указываем, какая метка должна быть у ноды, чтобы туда установился ingress

ingress_nginx_tolerations:
  - key: "node-role.kubernetes.io/ingress"
    operator: "Exists"

ingress_nginx_configmap:
  server-tokens: "False"
  proxy-body-size: "2048M"
  proxy-buffer-size: "16k"
  worker-shutdown-timeout: "180"
```

Создаем файл *~/kubespray/inventory/dev/group_vars/kube-ingress.yml* и добавляем параметры:

```
node_labels:
  node.kubernetes.io/ingress: "true" # всем серверам в группе kube-ingress ставить соответствующую метку
node_taints:
  - "node.kubernetes.io/ingress=:NoSchedule"
```

Трудно кратко и понятно описать настройки ingress, так как тут используются не тривиальные возможности kubernetes в виде **taints** и **tolerations**. Общий смысл в том, что мы задаем метку для ingress и поведение на основе этой метки. На ноды в группе kube-ingress ставится ограничение NoSchedule (не распределять поды на ноду) с помощью taints. Это ограничение могут преодолевать только только те, у кого в tolerations прописана метка ingress. Таким

образом, на нодах ingress, кроме самого ингресса ничего запускаться не будет.

Вот и все. Мы готовы к тому, чтобы начать установку кластера Kubernetes. Запускаем ее с помощью **ansible-playbook**. Рекмоендую делать это в screen, чтобы не прерывался процесс из-за обрыва связи.

```
# ansible-playbook -u root -k -i inventory/dev/inventory.ini -b — diff cluster.yml
```

Процесс обычно длится 15-20 минут. У меня сервера старые, на hdd, длилось 30 минут. В конце вы должны увидеть примерно такую картину.

```
*****************************************************************************
Wednesday 18 September 2019  22:39:09 +0300 (0:00:00.221)       0:31:08.433 ***

TASK [kubernetes/preinstall : check fs type] ***********************************************************************
*****************************************************************************
Wednesday 18 September 2019  22:39:09 +0300 (0:00:00.215)       0:31:08.649 ***

TASK [kubernetes/preinstall : run growpart] ************************************************************************
*****************************************************************************
Wednesday 18 September 2019  22:39:10 +0300 (0:00:00.226)       0:31:08.876 ***

TASK [kubernetes/preinstall : run xfs_growfs] **********************************************************************
*****************************************************************************
Wednesday 18 September 2019  22:39:10 +0300 (0:00:00.213)       0:31:09.089 ***

PLAY RECAP *********************************************************************************************************
*****************************************************************************
kub-ingress-1              : ok=407  changed=74   unreachable=0    failed=0
kub-master-1               : ok=622  changed=126  unreachable=0    failed=0
kub-master-2               : ok=536  changed=105  unreachable=0    failed=0
kub-master-3               : ok=538  changed=106  unreachable=0    failed=0
kub-node-1                 : ok=409  changed=74   unreachable=0    failed=0
kub-node-2                 : ok=407  changed=74   unreachable=0    failed=0
localhost                  : ok=1    changed=0    unreachable=0    failed=0

Wednesday 18 September 2019  22:39:10 +0300 (0:00:00.223)       0:31:09.312 ***
===============================================================================
download : download_file | Download item ---------------------------------------------------------------------
------------------------------------------------------------- 278.59s
container-engine/docker : ensure docker packages are installed -----------------------------------------------
------------------------------------------------------------- 147.84s
kubernetes/master : kubeadm | Init other uninitialized masters -----------------------------------------------
------------------------------------------------------------- 83.20s
download : download_file | Download item ---------------------------------------------------------------------
------------------------------------------------------------- 60.11s
download : download_container | Download image if required ---------------------------------------------------
------------------------------------------------------------- 52.66s
kubernetes/master : kubeadm | Initialize first master -------------------------------------------------------
```

Ошибок быть не должно. Если есть ошибки, внимательно ищите их в выводе ansible и исправляйте. Основные ошибки возникают из-за неправильно заполненного инвентаря, из-за неправильной маски ip в свойствах flannel, из-за ошибок загрузки докер образов в процессе установки. После исправления ошибки можно запускать этот же плейбкук снова. Чаще всего все будет нормально донастроено.

Проверить состояние кластера можно командой:

```
# kubectl get nodes
```

```
[root@kub-master-1 kubespray]# kubectl get nodes
NAME            STATUS   ROLES     AGE     VERSION
kub-ingress-1   Ready    ingress   5m29s   v1.15.3
kub-master-1    Ready    master    8m17s   v1.15.3
kub-master-2    Ready    master    6m19s   v1.15.3
kub-master-3    Ready    master    6m49s   v1.15.3
kub-node-1      Ready    <none>    5m29s   v1.15.3
kub-node-2      Ready    <none>    5m29s   v1.15.3
[root@kub-master-1 kubespray]#
```


Вы увидите список всех нод и их роли. Я не знаю по какой причине, но при разворачивании кластера для этой статьи, у меня рабочие ноды не получили роли node. Исправить это можно командами.

```
# kubectl label node kub-node-1 node-role.kubernetes.io/node=
# kubectl label node kub-node-2 node-role.kubernetes.io/node=
# kubectl label node kub-ingress-1 node-role.kubernetes.io/node=
```

Для сервера ingress смотрите сами, хотите вы на него дополнительно вешать роль node или оставите только в качестве ingress контроллера. В продакшене лучше оставить его отдельно. Если у вас тестовый кластер, то можете объединить эти роли на одном сервере.

Если же вы захотите убрать какую-то роль, то команда будет такой.

```
# kubectl label node kub-ingress-1 node-role.kubernetes.io/node-
```

Мы убрали роль node на сервере kub-ingress-1. Проверяем снова состояние кластера.

```
# kubectl get nodes
```

```
# kubectl cluster-info
```

Посмотреть подробную информацию о ноде можно командой.

```
# kubectl describe node kub-master-1
```

```
 OS Image:                   CentOS Linux 7 (Core)
 Operating System:           linux
 Architecture:               amd64
 Container Runtime Version:  docker://18.9.7
 Kubelet Version:            v1.15.3
 Kube-Proxy Version:         v1.15.3
PodCIDR:                     10.233.64.0/24
Non-terminated Pods:         (6 in total)
  Namespace                  Name                                CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                  ----                                ------------  ----------  ---------------  -------------  ---
  kube-system                kube-apiserver-kub-master-1         250m (13%)    0 (0%)      0 (0%)           0 (0%)         9m35s
  kube-system                kube-controller-manager-kub-master-1  200m (11%)  0 (0%)      0 (0%)           0 (0%)         9m39s
  kube-system                kube-flannel-6tqmg                  150m (8%)     300m (16%)  64M (1%)         500M (14%)     6m32s
  kube-system                kube-proxy-ld95b                    0 (0%)        0 (0%)      0 (0%)           0 (0%)         7m17s
  kube-system                kube-scheduler-kub-master-1         100m (5%)     0 (0%)      0 (0%)           0 (0%)         9m27s
  kube-system                nodelocaldns-s9mvz                  100m (5%)     0 (0%)      70Mi (2%)        170Mi (5%)     5m18s
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests         Limits
  --------           --------         ------
  cpu                800m (44%)       300m (16%)
  memory             137400320 (4%)   678257920 (20%)
  ephemeral-storage  0 (0%)           0 (0%)
Events:
  Type    Reason                   Age                From                     Message
  ----    ------                   ----               ----                     -------
  Normal  NodeHasSufficientMemory  10m (x8 over 10m)  kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientMemory
  Normal  NodeHasNoDiskPressure    10m (x8 over 10m)  kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasNoDiskPressure
  Normal  NodeHasSufficientPID     10m (x7 over 10m)  kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientPID
  Normal  Starting                 10m                kube-proxy, kub-master-1 Starting kube-proxy.
  Normal  Starting                 8m12s              kubelet, kub-master-1    Starting kubelet.
  Normal  NodeHasSufficientMemory  8m12s              kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientMemory
  Normal  NodeHasNoDiskPressure    8m12s              kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasNoDiskPressure
  Normal  NodeHasSufficientPID     8m12s              kubelet, kub-master-1    Node kub-master-1 status is now: NodeHasSufficientPID
  Normal  NodeAllocatableEnforced  8m11s              kubelet, kub-master-1    Updated Node Allocatable limit across pods
  Normal  Starting                 7m28s              kube-proxy, kub-master-1 Starting kube-proxy.
  Normal  NodeReady                6m31s              kubelet, kub-master-1    Node kub-master-1 status is now: NodeReady
[root@kub-master-1 kubespray]#
```

Рекомендую запомнить эту команду. Она очень пригодится в процессе эксплуатации кластера и дебага. Особое внимание на раздел Events. Именно он будет очень полезен при разборе ошибок на нодах.

Посмотрим список всех запущенных подов.

```
# kubectl get pod -A
```

```
[root@kub-master-1 kubespray]# kubectl get pod -A
NAMESPACE       NAME                                       READY   STATUS    RESTARTS   AGE
ingress-nginx   ingress-nginx-controller-twwpb             1/1     Running   0          7m13s
kube-system     coredns-74c9d4d795-7p814                   1/1     Running   0          6m48s
kube-system     coredns-74c9d4d795-clszw                   1/1     Running   0          7m1s
kube-system     dns-autoscaler-576b576b74-m6qgp            1/1     Running   0          6m56s
kube-system     kube-apiserver-kub-master-1                1/1     Running   0          11m
kube-system     kube-apiserver-kub-master-2                1/1     Running   0          9m34s
kube-system     kube-apiserver-kub-master-3                1/1     Running   0          10m
kube-system     kube-controller-manager-kub-master-1       1/1     Running   0          11m
kube-system     kube-controller-manager-kub-master-2       1/1     Running   0          9m34s
kube-system     kube-controller-manager-kub-master-3       1/1     Running   0          10m
kube-system     kube-flannel-6tqmg                         2/2     Running   0          8m8s
kube-system     kube-flannel-6wcsm                         2/2     Running   0          8m8s
kube-system     kube-flannel-hjrng                         2/2     Running   0          8m8s
kube-system     kube-flannel-jxv6g                         2/2     Running   0          8m8s
kube-system     kube-flannel-mjg94                         2/2     Running   0          8m8s
kube-system     kube-flannel-v6z7k                         2/2     Running   0          8m8s
kube-system     kube-proxy-2mh5p                           1/1     Running   0          8m53s
kube-system     kube-proxy-4lwcc                           1/1     Running   0          8m53s
kube-system     kube-proxy-ld95b                           1/1     Running   0          8m53s
kube-system     kube-proxy-vxw7d                           1/1     Running   0          8m53s
kube-system     kube-proxy-w6vxs                           1/1     Running   0          8m53s
kube-system     kube-proxy-x5q88                           1/1     Running   0          8m53s
kube-system     kube-scheduler-kub-master-1                1/1     Running   0          11m
kube-system     kube-scheduler-kub-master-2                1/1     Running   0          9m34s
kube-system     kube-scheduler-kub-master-3                1/1     Running   0          10m
kube-system     kubernetes-dashboard-7c547b4c64-6qfdm      1/1     Running   0          6m51s
kube-system     nginx-proxy-kub-ingress-1                  1/1     Running   0          9m28s
kube-system     nginx-proxy-kub-node-1                     1/1     Running   0          9m28s
kube-system     nginx-proxy-kub-node-2                     1/1     Running   0          9m28s
kube-system     nodelocaldns-b68px                         1/1     Running   0          6m54s
kube-system     nodelocaldns-mxvk7                         1/1     Running   0          6m54s
kube-system     nodelocaldns-pcw8t                         1/1     Running   0          6m54s
kube-system     nodelocaldns-s9mvz                         1/1     Running   0          6m54s
kube-system     nodelocaldns-t6hnz                         1/1     Running   0          6m54s
kube-system     nodelocaldns-xpf7t                         1/1     Running   0          6m54s
[root@kub-master-1 kubespray]#
```

Они все должны быть в состоянии Running. Если это не так, то у вас какие-то ошибки, с которыми надо разбираться. В общем случае ошибок быть не должно, если вы все сделали правильно на моменте подготовки инвентаря.

На этом непосредственно установка кластера Kubernetes закончена. Он готов к эксплуатации. Если вы только знакомитесь с ним, то, думаю, вам совсем не понятно, что делать дальше и как его эксплуатировать. Об этом будут мои последующие статьи. Следите за обновлениями.

Проблема с сертификатами

Сразу обращаю внимание на очень важный момент. Необходимо тем или иным способом настроить мониторинг сертификатов, которые установил и настроил kubespray для обмена информацией мастеров. Сертификатов много и у них срок действия 1 год. Пока сертификаты не просрочились, их относительно легко обновлять. Если упустить этот момент, то все становится сложнее.

Я до конца не понял и не проработал вопрос обновления сертификатов, но это нужно будет сделать. Пока просто покажу, как за ними можно следить.

Сертификат **api-server**, порт 6443

```
# echo -n | openssl s_client -connect localhost:6443 2>&1 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' |
openssl x509 -text -noout | grep Not
            Not Before: Sep 18 19:32:42 2019 GMT
            Not After : Sep 17 19:32:42 2020 GMT
```

Сертификат **controller manager**, порт 10257.

```
# echo -n | openssl s_client -connect localhost:10257 2>&1 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' |
openssl x509 -text -noout | grep Not
            Not Before: Sep 18 18:35:36 2019 GMT
            Not After : Sep 17 18:35:36 2020 GMT
```

Сертификат **scheduler**, порт 10259.

```
# echo -n | openssl s_client -connect localhost:10259 2>&1 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' |
openssl x509 -text -noout | grep Not
            Not Before: Sep 18 18:35:35 2019 GMT
            Not After : Sep 17 18:35:35 2020 GMT
```

Это все разные сертификаты и они выпущены на год. Их надо будет не забыть обновить. А вот сертификат для **etcd**. Он выпущен на 100 лет.

```
# echo -n | openssl s_client -connect localhost:2379 2>&1 | sed -ne '/-BEGIN CERTIFICATE-/,/-END CERTIFICATE-/p' |
openssl x509 -text -noout | grep Not
            Not Before: Sep 18 19:28:50 2019 GMT
            Not After : Aug 25 19:28:50 2119 GMT
```

Я не понял, почему этого не сделали для всех сервисов, а оставили такой головняк на потом. Если у кого-то есть информация о том, как корректно и безпроблемно обновлять сертификаты, прошу поделиться. Я пока видел только вариант с полуручным обновлением и раскидыванием этих сертификатов по мастерам.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

На этом начальную статью по Kubernetes заканчиваю. На выходе у нас получился рабочий кластер из трех мастер нод, двух рабочих нод и ingress контроллера. В последующих статьях я расскажу об основных сущностях kubernetes, как деплоить приложения в кластер с помощью Helm, как добавлять различные стореджи, как мониторить кластер и т.д. Да и в целом, хочу много о чем написать, но не знаю, как со временем будет.

В планах и git, и ansible, и prometeus, и teamcity, и кластер elasticsearch. К сожалению, доход с сайта не оправдывает временных затрат на написание

статей, поэтому приходится писать их либо редко, либо поверхностно. Основное время уходит на текущие задачи по настройке и сопровождению.