

Автоматизация развертывания Kubernetes-кластеров на базе Ubuntu ОС в Rancher на инфраструктуре VMWare vSphere

Н.Б. Лазарева

Тихоокеанский государственный университет, Хабаровск

Аннотация: В статье рассмотрен механизм быстрого создания и обслуживания кластеров Kubernetes без низкоуровневых операций с существенным сокращением временных и трудовых затрат с применением автоматизации на базе продуктов Rancher, VMWare vSphere и Ubuntu.

Ключевые слова: Kubernetes, Ubuntu, Rancher, Docker, кластер, контейнеризация, автоматизация.

В современных ИТ-инфраструктурах все чаще используются Kubernetes-кластеры, как средство оркестрации контейнеров [1]. Несмотря на все преимущества использования Kubernetes, нельзя не отметить нетривиальное развертывание и дальнейшую поддержку таких кластеров. Эти процедуры несут определенные риски и требуют высокой квалификации ИТ-специалиста. Также нужно отметить, что Kubernetes-кластеры существуют в количестве N экземпляров, могут нести разнообразную нагрузку, иметь разную конфигурацию, назначение и т.д. Логичным выводом из вышесказанного будет признание необходимости найти способ автоматизировать и упростить работы по созданию и поддержке Kubernetes-кластеров. Современный рынок ИТ-решений предлагает несколько продуктов, позволяющих реализовать вышесказанное. Одним из таких продуктов является Rancher [2].

Rancher – универсальный механизм, позволяющий производить оркестрацию Kubernetes-кластеров. К его возможностям можно отнести следующее:

- создание кластеров Kubernetes из заранее подготовленных шаблонов;

-интеграция развертывания с основными продуктами инфраструктурного уровня (облачные Azure EKS [3], Amazon AKS [4], Google GKE, а также VMWare vSphere, DigitalOcean [5] и другие);

- горизонтальное масштабирование кластеров Kubernetes;
- мониторинг кластеров и всей нагрузки внутри них;
- редактирование всех объектов уровня Kubernetes;
- предоставление доступа к кластерам;
- обновление кластеров Kubernetes до более новых версий.

Очевидно, что Rancher покрывает основные потребности оркестрации Kubernetes-кластеров. Сам Rancher представляет собой небольшой Kubernetes-кластер, который является основным и управляющим для всех остальных. Для развертывания Rancher необходима инфраструктура, и в нашем случае это будет VMWare vSphere – популярная платформа для создания виртуальных машин, сетей, систем хранения и т.д.

Хотя схема интеграции Rancher и VMWare vSphere описана в документации, есть ряд нюансов, которые стоит учитывать при данной организации инфраструктуры. Кроме того, в качестве ОС для виртуальных машин будет использована актуальная на момент написания данного материала Ubuntu 22.04 LTS [6]. Ее использование также подразумевает определенные шаги подготовки. Разберем итоговую схему, которую стремимся достичь (рис.1). Нижний уровень представляет VMWare vSphere. Не является принципиальным, как именно это организовано на низком уровне.

Для реализации схемы понадобится система хранения. Пусть в данном случае средствами VMWare vSphere развернут так называемый vSAN [7] – виртуальное отказоустойчивое распределенное хранилище, на котором будут создаваться виртуальные жесткие диски виртуальных машин. Поверх VMWare vSphere и vSAN нужно создать виртуальные машины (далее VM) на

базе ОС Ubuntu 22.04 LTS. Три VM предназначены для развертывания кластера Rancher, остальные будут создаваться автоматически при развертывании новых кластеров Kubernetes.

Начать построение схемы нужно с создания трех VM для кластера Rancher (будет использована версия Rancher 2.6.9). В качестве ОС используем стандартный образ Ubuntu 22.04 TLS.

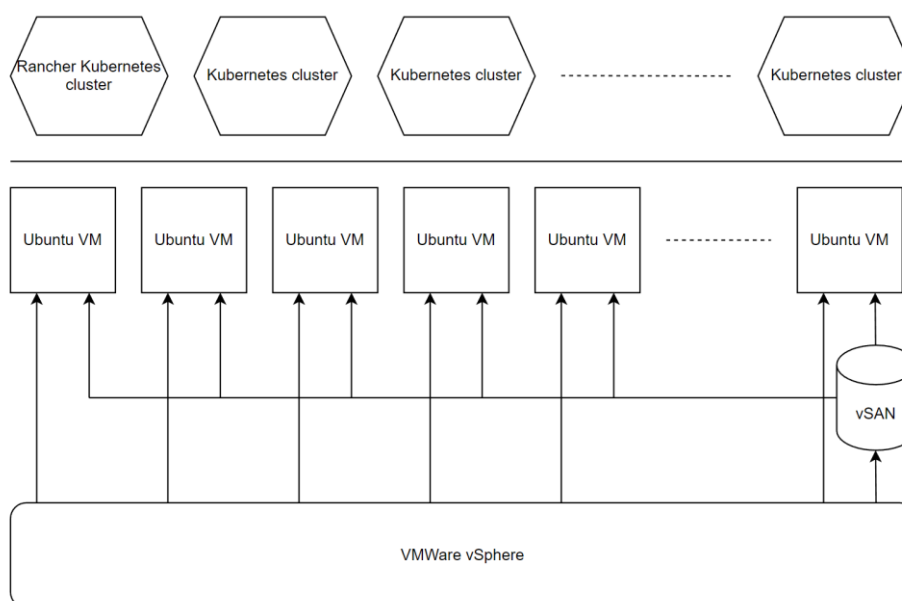


Рис. 1 – Итоговая схема

Установка и настройка в целом стандартная и зависит от ИТ-инфраструктуры, обратить внимание стоит на следующие моменты: стоит использовать статические IP-адреса; IPv6 выключить принудительно в случае, если он не используется; рекомендуется выключить Swap. Потребуется стандартные утилиты: kubectl [8], helm3, rke версии 1.4, docker [9].

Теперь, когда VM готовы, приступим к развертыванию кластера Rancher.

Нужно создать пользователя, от имени которого будет выполняться весь процесс. Пользователь должен быть добавлен в группу docker. После этого генерируются ключи для доступа ко всем трем VM:

```
ssh-keygen  
cat ~/.ssh/id_rsa.pub
```

Сгенерированный ключ нужно добавить на все VM:

```
nano ~/.ssh/authorized_keys
```

Создадим конфигурационный файл, который определяет облик будущего Rancher-кластера:

```
mkdir /root/.kube  
nano /root/.kube/rancher-cluster.yml
```

Добавим в файл следующее содержимое:

```
nodes:  
- address: rancher-01.domain.local  
  user: service  
  role: [controlplane, worker, etcd]  
- address: rancher-02.domain.local  
  user: service  
  role: [controlplane, worker, etcd]  
- address: rancher-03.domain.local  
  user: service  
  role: [controlplane, worker, etcd]  
services:  
  etcd:  
    snapshot: true  
    creation: 6h  
    retention: 24h  
  ingress:  
    provider: nginx  
    options:  
      use-forwarded-headers: "true"  
  network:  
    plugin: calico  
ssh_key_path: /home/service/.ssh/id_rsa
```

Как видно из данного описания, развертывание будет выполняться на всех трех VM с использованием созданного пользователя и его ключа [10]. Важно – в качестве сетевого плагина нужно использовать calico, так как стандартный плагин canal имеет проблемы при работе в ОС Ubuntu 22.04 LTS.

С помощью утилиты `rke` запускаем создание кластера Rancher:

```
rke up --config /root/.kube/rancher-cluster.yml
```

По завершению выполнения этой команды на трех ВМ будет развернут пустой кластер Kubernetes, в который нужно остановить полезную нагрузку – сам Rancher. Для этого используем официальный helm-чарт:

```
helm repo add rancher-stable https://releases.rancher.com/server-charts/stable  
helm fetch rancher-stable/rancher --version=2.6.9  
helm install rancher rancher-stable/rancher \  
  --namespace cattle-system \  
  --set hostname=rancher.domain.local \  
  --set bootstrapPassword=xxxxxxx \  
  --set ingress.tls.source=secret \  
  --wait \  
  --timeout 10m0s \  
  --create-namespace
```

По завершению работы этих команд у нас будет развернут Rancher. По заранее подготовленному адресу `rancher.domain.local` должен быть доступен веб-интерфейс.

Теперь с помощью уже развернутого Rancher можно создавать новые кластера Kubernetes. Однако, для этого нужно подготовить ряд необходимых вещей:

- пользователя в VMWare vSphere, под которым будет осуществляться авторизация и создаваться ВМ;
- специальный образ ОС Ubuntu;
- шаблоны для нод будущих кластеров в Rancher.

Создание пользователя тривиально, поэтому остановимся на образе ОС Ubuntu. В силу особенностей работы `cloud-init` в данной ОС, мы не можем взять тот же образ, что использовался для развертывания ВМ для Rancher. Поэтому нужно использовать `cloud-image` версию Ubuntu.

Перейдем к созданию шаблонов нод. Шаблон ноды – это описание ВМ, которая будет создана как часть будущего Kubernetes-кластера и выполнять в нем определенную роль. Мы создадим три шаблона – для ролей `control plane`,

worker и etcd. Начнем с шаблона роли control plane (рис.2). В шаблоне нужно выбрать заранее созданного пользователя для интеграции с VMWare VSphere, раздел vSAN, на котором будут создаваться виртуальные диски будущих VM, количество ресурсов процессора, объемы оперативной памяти и дискового пространства, а также образ cloud-image Ubuntu.

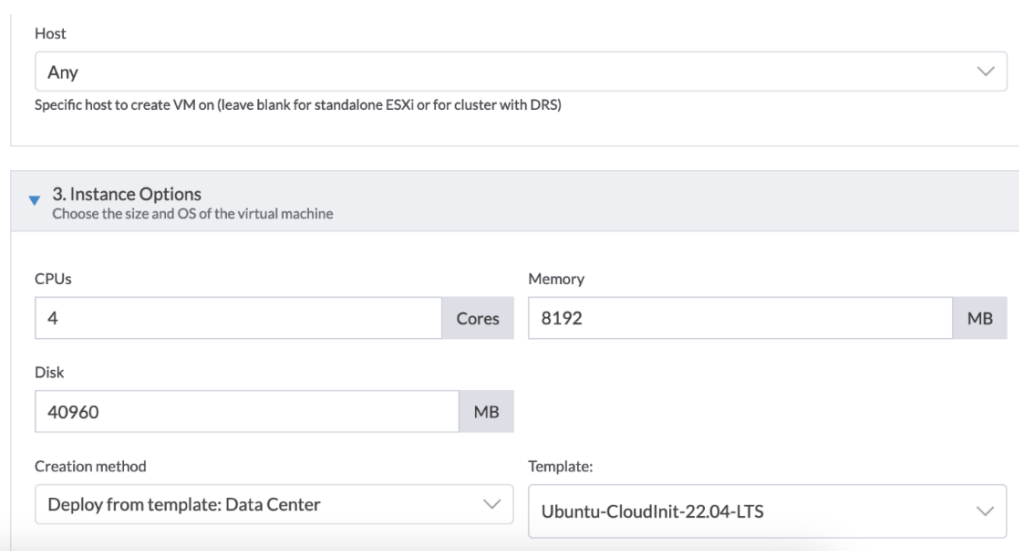


Рис. 2. – Пример конфигурации шаблона для роли control plane

Для того, чтобы определенным образом подготовить VM сразу после создания так, как нужно, можно использовать cloud-init скрипт. Пример скрипта рассмотрим далее:

```
#cloud-config
users:
- name: ubuntu
  inactive: true
- name: cluster
  lock_passwd: false
  groups: users,docker
  sudo: ALL=(ALL) NOPASSWD:ALL
  primary_group: cluster
  passwd: somehash
  shell: /bin/bash
timezone: Europe/Moscow
ntp:
  enabled: true
pools:
- 0.ru.pool.ntp.org
```

```
write_files:
#Sysctl settings
- path: /etc/sysctl.d/20-extra.conf
permissions: "0644"
owner: root
content: |
vm.max_map_count = 262144
#CPU
#Decrease processes migration from one cpu to another
kernel.sched_migration_cost_ns = 5000000
#Do not group processes to cpu's by tty
kernel.sched_autogroup_enabled = 0
#General networking
#The maximum number of connections that can be queued for acceptance by process
net.core.somaxconn = 8192
#Network read/write buffers kernel limits
#64 MB, twice bigger than tcp_mem, suppose to use in non-tcp workload
net.core.rmem_max = 67108864
#64 MB, twice bigger than tcp_mem, suppose to use in non-tcp workload
net.core.wmem_max = 67108864
#TCP
#Make sure we are not running out of ports
#Not touching it because k8s needs ports 30000-32767 for node port. Default '32768 -
60999' will be fine.
#net.ipv4.ip_local_port_range = 2000 65000
## congestion control algorithm reno - default , Hamilton TCP - advanced
net.ipv4.tcp_congestion_control = htcp
##4Kib Pages. Memory for TCP Sockets 4 KiB * 466920 =~ 2GiB RAM. Usually
calculated at boot time. Have to increase
net.ipv4.tcp_mem = 116730 311280 466920
## TCP read/write buffers kernel limits "min default max". Settings for 10g network =
#(bytes)32MB
net.ipv4.tcp_rmem = 4096 87380 33554432
#(bytes)32MB
net.ipv4.tcp_wmem = 4096 65536 33554432
# The length of time an orphaned (no longer referenced by any application) connection
will remain in the FIN_WAIT_2 state. Default - 60.
net.ipv4.tcp_fin_timeout = 10
#keepalive settings
net.ipv4.tcp_keepalive_time = 300
net.ipv4.tcp_keepalive_probes = 6
net.ipv4.tcp_keepalive_intvl = 30
- path: /etc/ssh/sshd_config
content: |
# CLOUD CONFIG MANAGED
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding no
PrintMotd no
```

```
AcceptEnv LANG LC_*  
Subsystem sftp /usr/lib/openssh/sftp-server  
PasswordAuthentication yes
```

```
#Runcmd runs only at first boot  
runcmd:  
#Apply appended sysctl.d config. Need only at first boot, because file is going to be saved  
and auto-applied further  
- "/sbin/sysctl --system"
```

В данном примере отключается пользователь по умолчанию (ubuntu), создается новый пользователь (cluster) с заранее подготовленным хэшем пароля. Также устанавливаются настройки временной зоны и синхронизации времени, что является важным для нагрузки внутри кластеров Kubernetes. Кроме этого, создается файл со специальными настройками ядра, что пригодится при больших нагрузках, и файл конфигурации ssh для возможности удаленного подключения к ВМ.

Теперь, когда шаблон готов, можно на его основе создать шаблоны для ролей worker и etcd. Они могут отличаться количеством выделенных ресурсов, наполнением cloud-init скрипта и т.д. В данной статье создание этих шаблонов описывать не будем.

Для создания кластеров все подготовлено, создадим тестовый кластер на основе созданных шаблонов ролей.

Cluster Name * Add a Description

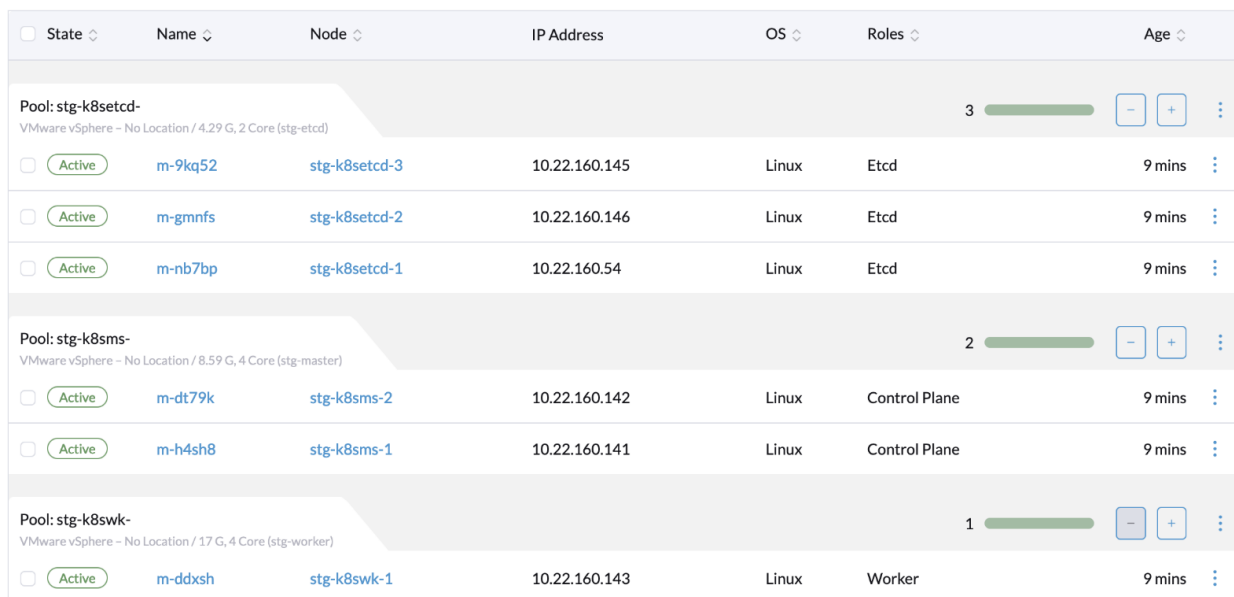
staging

Name Prefix	Count	Template	Auto Replace	Drain Before Delete	etcd	Control Plane	Worker	Taints
stg-k8sms-	2	stg-master +	0 minutes	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Taints -
stg-k8setcd-	3	stg-etcd +	0 minutes	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Taints -
stg-k8swk-	1	stg-worker +	0 minutes	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Taints -

Number of nodes required: ✔ 1, 3, or 5 ✔ 1 or more ✔ 1 or more

Рис. 3. – Создание кластера Kubernetes на основе шаблонов ролей

Как показано на рис.3, кластер будет развернут в количестве двух нод с ролью control plane, трех нод с роль etcd и одной ноды с ролью worker. Таким образом созданный кластер в минимальной конфигурации будет отказоустойчивым, количество нод с ролью worker далее нужно будет наращивать пропорционально создаваемой нагрузке. Также нужно выбрать версию Kubernetes и версию сетевого плагина – как и в случае с кластером Rancher нужно изменить выбор с canal на calico. Спустя некоторое время кластер будет развернут (рис.4).



State	Name	Node	IP Address	OS	Roles	Age
Pool: stg-k8setcd- VMware vSphere - No Location / 4.29 G, 2 Core (stg-etcd)						3
Active	m-9kq52	stg-k8setcd-3	10.22.160.145	Linux	Etcd	9 mins
Active	m-gmnfs	stg-k8setcd-2	10.22.160.146	Linux	Etcd	9 mins
Active	m-nb7bp	stg-k8setcd-1	10.22.160.54	Linux	Etcd	9 mins
Pool: stg-k8sms- VMware vSphere - No Location / 8.59 G, 4 Core (stg-master)						2
Active	m-dt79k	stg-k8sms-2	10.22.160.142	Linux	Control Plane	9 mins
Active	m-h4sh8	stg-k8sms-1	10.22.160.141	Linux	Control Plane	9 mins
Pool: stg-k8swk- VMware vSphere - No Location / 17 G, 4 Core (stg-worker)						1
Active	m-ddxsh	stg-k8swk-1	10.22.160.143	Linux	Worker	9 mins

Рис. 4 – Развернутый кластер Kubernetes

Rancher автоматически создает необходимое количество ВМ, устанавливает на них Kubernetes в той роли, которая описана в шаблоне, а также выполняет cloud-init скрипт. Данный кластер можно горизонтально масштабировать нажатием кнопок «-» и «+».

Таким образом данная схема позволяет в дальнейшем быстро создавать и обслуживать кластера Kubernetes из веб-интерфейса Rancher, не прибегая к низкоуровневым операциям и существенно сокращая время, необходимое на оркестрацию кластеров. Созданные кластеры автоматически добавляются в

систему мониторинга Rancher, поэтому можно просматривать графики нагрузки на всех уровнях. Легко решается вопрос обновления кластеров – это так же можно делать из веб-интерфейса Rancher.

Литература

1. Евстратов В.В. Оркестрация контейнеров на примере Kubernetes // Молодой учёный, 2020, №51(341). URL: moluch.ru/archive/341/76636/
2. Документация Rancher. URL: ranchermanager.docs.rancher.com (дата обращения: 20/03/2023).
3. Burns B., Beda J., Hightower K., Lachlan Evenson L. Kubernetes: Up and Running: Dive into the Future of Infrastructure. – O'Reilly Media, 2022. – P. 354.
4. Арундел Дж., Домингус Дж. Kubernetes для DevOps: развертывание, запуск и масштабирование в облаке. – Питер, 2020. — 384 с.
5. Beyer B., Jones C., Petoff J. Site Reliability Engineering. How Google Runs Production Systems. – O'Reilly Media, Incorporated, 2016. – P. 522.
6. Релиз Ubuntu 22.04 LTS. URL: ubuntu.com/blog/tag/22-04-lts (дата обращения: 20/03/2023).
7. Официальная документация VMWare vSAN. URL: docs.vmware.com/en/VMware-vSAN/index.html (дата обращения: 20/03/2023).
8. Джиджи С. Осваиваем Kubernetes. Оркестрация контейнерных архитектур. — СПб.: Питер, 2019. — 400 с.
9. Моуэт Э. Использование Docker. – ДМК-Пресс, 2017. – 354 с.
10. Лазарева Н. Б., Ловцова Н. Н. Автоматизация получения доступа к базам данных для компонентов в среде Kubernetes предприятия // Инженерный вестник Дона, 2021, №3. URL: ivdon.ru/ru/magazine/archive/n3y2021/6844/.

References

1. Evstratov V.V. Molodoj uchyonyj, 2020, №51 (341). URL: moluch.ru/archive/341/76636/
2. Oficial'naya dokumentaciya Rancher [Rancher Documentation]. URL: ranchermanager.docs.rancher.com (accessed 20/03/2023).
3. Burns B., Beda J., Hightower K., Lachlan Evenson L. Kubernetes: Up and Running: Dive into the Future of Infrastructure. O'Reilly Media, 2022. P. 354.
4. Arundel Dzh., Domingus Dzh. Kubernetes dlya DevOps: razvertyvanie, zapusk i masshtabirovanie v oblake [Kubernetes for DevOps: deployment, launch and scaling in the cloud]. Piter, 2020. 384 p.
5. Beyer B., Jones C., Petoff J. Site Reliability Engineering. How Google Runs Production Systems. O'Reilly Media, Incorporated, 2016. P. 522.
6. Reliz Ubuntu 22.04 LTS [Ubuntu release 22.04 LTS]. URL: ubuntu.com/blog/tag/22-04-lts (accessed 20/03/2023).
7. Oficial'naya dokumentaciya VMWare vSAN [Official documentation of VMware vSAN]. URL: docs.vmware.com/en/VMware-vSAN/index.html (accessed 20/03/2023).
8. Dzhidzhi S. Osvaivaem Kubernetes. Orkestraciya kontejneryx arxitektur [Orchestration of container architectures]. SPb.: Piter, 2019. 400 p.
9. Moue't E`. Ispol'zovanie Docker [Using Docker]. DMK-Press, 2017. 354 p.
10. Lazareva N.B., Lovczova N.N. Inzhenernyj vestnik Dona, 2021, №3. URL: ivdon.ru/ru/magazine/archive/n3y2021/6844/.