Инструкция по установке ПО «Boot»

Содержание

1. О документе

Документ предназначен для технических специалистов, которые хотят установить полнофункциональную версию ПО «Boot».

2. Быстрый старт

Этот раздел описывает установку Boot с ограничением в 1 процессор. Подробную информацию о том как установить лицензионный ключ, позволяющий использовать больше процессоров можно получить в разделе Лицензионный ключ.

2.1. Установка в Kubernetes

2.1.1. Системные требования

1. Работающий кластер Kubernetes.
2. Установленный `kubectl` клиент с настроенным доступом к кластеру
3. Пара ключей ssh, которые обычно хранятся в директории ~/.ssh/ под именами `id_rsa` (приватный ключ) и `id_rsa.pub` (публичный ключ). Если ключей нет, необходимо их сгенерировать командой `ssh-keygen`. Для установки Boot потребуется только публичный ключ, который обычно выглядит так:

```
$ cat ~/.ssh/id_rsa.pub
```

```
ssh-rsa AAAAB3N... some-comment
```

4. При установке Boot в кластер Kubernetes, запущенный на рабочей станции, при помощи инструмента minikube - кластер Kubernetes нужно запускать одной из следующих команд:

   Запуск Minikube на Linux

   ```
   $ minikube start --driver kvm2
   ```

   Запуск Minikube на MacOS

   ```
   $ minikube start --driver=hyperkit
   ```

   Запуск Minikube на Windows

   ```
   $ DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V # Enable Hyper-V
   $ minikube start --driver=hyperv
   ```

2.1.2. Установка Boot при помощи Helm

ВАЖНО: Helm чарты (пакеты) являются рекомендуемым инструментом установки Boot. Дальнейшие шаги предполагают установленный Helm 3. Более старые версии не поддерживаются.

Мы предоставляем готовые Helm чарты, что позволяет установить Boot с помощью Helm простой при помощи нескольких команд:

1. Создайте файл `values.yaml` с конфигурацией Helm чарта:

   ```
   boot:
     jumphost:
       authorizedKeys:
         - ssh-rsa AAAAB3N... some-comment # Вставьте содержимое хотя бы одного
   публичного ключа в данный раздел
   ```

2. Добавьте репозиторий с Helm чартами Aerokube charts:

   ```
   $ helm repo add aerokube https://charts.aerokube.ru/
   $ helm repo update
   ```

3. Для вывода списка доступных версий Boot используйте команду:

   ```
   $ helm search repo aerokube --versions
   ```

4. Создайте Kubernetes неймспейс:

```
$ kubectl create namespace boot
```

5. Установите или обновите Boot командой:

```
$ helm upgrade --install -f values.yaml -n boot boot aerokube/boot
```

6. Helm чарт для Boot содержит различные параметры, которые можно посмотреть командой:

```
$ helm show values aerokube/boot
```

   Для изменения одного из этих параметров - переопределите его в файле `values.yaml` и выполните команду установки Helm чарта еще раз.

7. Выясните IP адрес балансировщика нагрузки Boot. Как правило, адрес балансировщика может быть получен из сервиса Boot:

   Как узнать IP адрес сервиса Boot

```
$ kubectl get svc boot -n boot
NAME   TYPE           CLUSTER-IP       EXTERNAL-IP    PORT(S)    AGE
boot   LoadBalancer   10.107.117.163   192.168.64.5   2222/TCP   15d
```

   IP адрес указан в колонке `EXTERNAL-IP`. При использовании Minikube IP адрес необходимо добавить вручную, с помощью вывода команды `minikube ip`:

   Добавление IP адреса в сервис Boot вручную

```
$ kubectl patch svc boot -n boot --patch "{\"spec\":{\"externalIPs\":[\"$(minikube
ip)\"]}}"
```

   На Windows вывод команды `minikube ip` необходимо подставить вручную, поскольку выражение `$()` может не сработать.

8. Удостоверьтесь, что доменное имя указывает на этот IP адрес. При использовании Minikube - просто отредактируйте файл `hosts`:

```
$ sudo echo "$(minikube ip) boot.aerokube.local" >> /etc/hosts
```

   На Windows вам необходимо отредактировать файл `hosts` вручную.

9. Сконфигурируйте SSH клиент для использования Boot. Для этого добавьте следующие параметры в файл `~/.ssh/config`:

```
$ cat ~/.ssh/config
# ... другие записи
Host bootjump
    IdentityFile ~/.ssh/id_rsa
    HostName boot.aerokube.local
    Port 2222
Host *.vm.boot.svc.cluster.local
    IdentityFile ~/.ssh/id_rsa
    ProxyJump jump@bootjump
```

10. Создайте вашу первую виртуальную машину. Для этого создайте YAML файл с описанием структуры виртуальной машины:

```
$ cat ~/vm.yaml
apiVersion: boot.aerokube.com/v1
kind: VirtualMachine
metadata:
  name: my-vm
  namespace: boot
spec:
  os: ubuntu
  authorizedKeys:
    - ssh-rsa AAAAB3N... some-comment # The same public key contents here too
```

Создайте виртуальную машину используя этот файл:

```
$ kubectl create -f vm.yaml
```

11. Проверьте доступность виртуальной машины по SSH:

```
$ ssh root@my-vm.vm.boot.svc.cluster.local
# Здесь идет некоторый приветственный текст...
root@my-vm:~# # Теперь у вас есть root доступ к виртуальное машине
```

== Архитектура

2.2. Компоненты Boot

Компоненты Boot

[boot components]

Кластер Boot состоит из нескольких компонентов:

1. Одна или несколько реплик `Boot`. Они запускают/останавливают виртуальные машины и следят за доступом по SSH к созданным машинам. `Boot` обычно доступен по какому

либо из SSH портов, например `2222`.

2. Одна или несколько реплик `Boot UI` (эта фунциональность в данный момент ещё находится в разработке). `Boot UI` собирает информацию от Boot и визуализирует её. Обычно интерфейс доступен на HTTP порту `8080`.
3. Объектов виртуальных машин и соответствующих им запущенных подов.

2.3. Содержание пода Boot

Каждый под Boot содержит несколько контейнеров для выполнения специфичных задач.

Table 1. Контейнеры внутри пода Boot

Имя	**Назначение**
boot	Стартует и останавливает виртуальные машины
jumphost	Стандартный SSH сервер с защищенной конфигурацией работающий как SSH шлюз (SSH сервер в режиме jump host)
reloader	Во время обновления Boot в течение заданного периода следит за SSH соединениями к виртуальным машинам, чтобы закрыть или переоткрыть их после завершения обновления

2.4. Поддержка нескольких неймспейсов

Boot может запускать виртуальные машины на любом количестве неймспейсов в Kubernetes. По умолчанию сам Boot и все запущенные виртуальные машины работают в одном неймспейсе. В этом нет проблемы, если Boot используется только одной командой либо если нет необходимости ограничивать доступные вычистительные ресурсы виртуальных машин для разных пользователей Boot.

Boot настроенный для работы с несколькими неймспейсами

[multiple-namespaces-mode]

Можно настроить Boot таким образом, чтобы он был запущен в одном неймспейсе и управлял виртуальными машинами в одном или нескольких соседних неймспейсах. Необходимость в нескольких неймспейсах обычно имеется тогда, когда нужно контролировать/ограничивать вычислительные ресурсы, запускать виртуальные машины с разными релизами операционных систем либо вследствие особенностей настроек сетевого доступа (network policies) для каждой команды. Например, виртуальные машины одного пользователя будут работать в неймспейсе одной команды сотрудников, а виртуальные машины второго пользователя будут работать в неймспейсе второй команды. Информацию о том как сконфигурировать несколько неймспейсов можно получить по ссылке.

3. Необходимые права доступа

Boot не требует широких прав доступа в Kubernetes. Обычно ему достаточно настроек доступа Kubernetes по умолчанию. Boot может запускать виртуальные машины на любом количестве неймспейсов Kubernetes. Все необходимые права и роли автоматически создаются Helm чартом. Следующая таблица показывает, какие права доступа Boot имеет в каждом в неймспейсе:

Table 2. Необходимые права доступа

Роль	**Назначение**
Чтение (**get**), подписка (**watch**), вывод списка (**list**) и редактирование (**patch**) ресурсов самого Boot в группе `boot.aerokube.com`	Эти ресурсы хранят конфигурацию Boot и доступны во всем кластере, поэтому требуется добавлять ClusterRole (роль на весь кластер Kubernetes).

Необходимые права для пользователя:

Table 3. Права пользователя

Роль	**Назначение**
Чтение (**get**), подписка (**watch**), вывод списка (**list**), создание (**create**), удаление (**delete**), обновление (**update**) и редактирование (**patch**) подов	Используется для управления виртуальными машинами
Чтение (**get**), подписка (**watch**), вывод списка (**list**), создание(**create**), удаление (**delete**), обновление (**update**) и редактирование (**patch**) config maps	Используется для передачи списка пользователей и групп на виртуальные машины