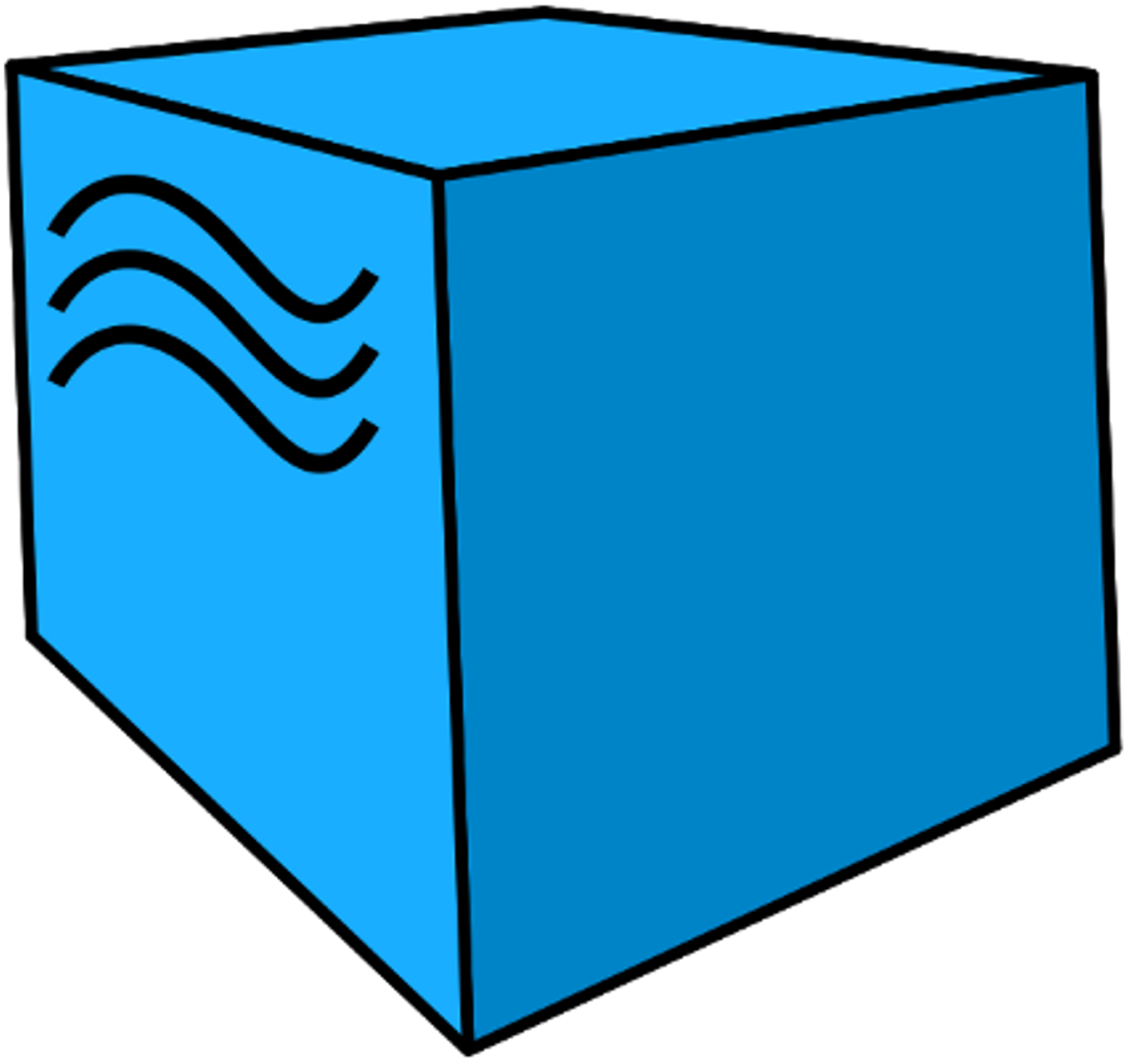
Aerokube Moon



https://aerokube.ru/

Инструкция по установке ПО «Moon»

Содержание

1. О документе

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

2. Начало работы

Данный раздел описывает установку Moon с ограничением в четыре параллельных браузерных сессии. Подробная информация о том как установить лицензионный ключ, позволяющий снять это ограничение описан в разделе установка лицениионного ключа.

2.1. Установка в Kubernetes

2.1.1. Системные требования

1. Работающий кластер Kubernetes
2. Установленная утилита `kubectl` с настроенным доступом к кластеру
3. Если вы запускаете кластер Kubernetes на виртуальных машинах мы рекомендуем создавать виртуальные машины с насколько возможно большим количеством процессоров. Это позволит избежать проблем с фрагментацией памяти и нехваткой ресурсов. К примеру, если у вас имеется 24 процессора мы рекомендуем создать 3 виртуальные машины с 8 процессорами, а не 12 виртуальных машин по 2 процессора.

4. Если вы запускаете Moon в кластере, развернутом на рабочей станции при помощи инструмента minikube - ознакомьтесь с разделом Вариант 3: у вас Minikube.

2.1.2. Вариант 1: Установка с помощью Helm

Установка с помощью Helm является рекомендуемым способом установки Moon. Мы предполагаем, что используется версия Helm 3. Более старые версии не поддерживаются.

Мы предоставляем готовые Helm чарты, поэтому установка Moon с помощью Helm делается очень просто:

1. Добавьте репозиторий Aerokube, содержащий Helm чарты:

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

2. Для уточнения доступных версий используйте команду:

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

3. Создайте неймспейс:

```
$ kubectl create namespace moon
```

4. Установите или обновите Moon командой:

```
$ helm upgrade --install -n moon moon aerokube/moon2
```

5. Helm чарт для Moon содержит различные параметры конфигурации, которые можно посмотреть командой:

```
$ helm show values aerokube/moon2
```

   Для изменения каких-либо параметров используйте аргумент `--set`:

```
$ helm upgrade --install --set=moon.enabled.resources=false -n moon moon
aerokube/moon2
```

6. По умолчанию, развернутому объекту Ingress присваивается имя хоста `moon.aerokube.local`. Вы можете его изменить командой:

```
$ helm upgrade --install -n moon moon aerokube/moon2 --set
ingress.host=moon.example.com
```

Откройте ссылку http://moon.example.com/ в браузере и вы увидите интерфейс пользователя. В коде Selenium тестов используйте ссылку http://moon.example.com/wd/hub.

7. По умолчанию Moon запускается в режиме HTTP-only. Для включения TLS шифрования, то есть HTTPS, вам необходимо предоставить TLS сертификат и приватный ключ:

```
$ helm upgrade --install -n moon moon aerokube/moon2 --set
ingress.host=moon.example.com --set-file ingress.tlsCert=server.crt --set-file
ingress.tlsKey=server.key
```

Обычно сертификат и приватный ключ предоставляются либо сторонними поставщиками (центрами сертификации) либо отделом безопасности в вашей организации. Для создания тестовой пары сертификат+приватный ключ используйте следующие команды:

```
# Создание CA ключа и сертификата
$ openssl req -x509 -sha256 -newkey rsa:4096 -keyout ca.key -out ca.crt -days 356
-nodes -subj '/CN=My Cert Authority'
# Создание серверного ключа, создание сертификата и подписание сертификата
$ openssl req -new -newkey rsa:4096 -keyout server.key -out server.csr -nodes -subj
'/CN=moon.aerokube.local'
$ openssl x509 -req -sha256 -days 365 -in server.csr -CA ca.crt -CAkey ca.key
-set_serial 01 -out server.crt
```

При использовании таких тестовых сертификатов вам понадобится явно, вручную разрешить браузеру открывать Moon интерфейс.

8. Если Moon устанавливается на машину с архитектурой процессора ARM64 (например, Mac M1 и подобные CPU или облачные ноды Kubernetes с архитектурой ARM64), то выбор доступных браузеров ограничен. Selenium будет работать с Chromium, Firefox или Safari (другие браузеры не имеют версий, совместимых с Linux ARM64). Playwright, Cypress и Puppeteer не будут работать вообще. Для того, чтобы использовать браузеры, совместимые с ARM64, вам необходимо настроить файл `values.yaml` следующим образом:

Использование образов с браузерами, совместимых с ARM64

```
browsers:
  default:
    playwright: {}
    cypress: {}
    devtools: {}
    selenium:
      MicrosoftEdge: null
```

```
      opera: null
      chrome:
        default: 124.0.6367.60-1
        repository: quay.io/browser/chromium
      firefox:
        default: 125.0.3-1
        repository: quay.io/browser/firefox
      safari:
        default: 613.1.6.1
        repository: quay.io/browser/webkit
```

2.1.3. Вариант 2: Установка с помощью Minikube

Каждый браузер по умолчанию требует 1 процессор и 2 гигабайта памяти. В вашем кластере Minikube мы рекомендуем как минимум 4 процессора и 8 гигабайт памяти. При меньшем количестве процессоров поды могут не стартовать из-за нехватки вычислительных ресурсов. Мы не рекомендуем использовать Docker драйвер для Minikube.

1. Запуск Minikube на Linux

```
$ minikube start --cpus=4 --memory=8G --disk-size=20G --driver kvm2
```

Запуск Minikube на MacOS и процессором архитектуры x86

```
$ minikube start --cpus=4 --memory=8G --disk-size=20G --driver=hyperkit
```

Запуск Minikube на MacOS и процессором архитектуры ARM64 (M1 и подобные процессоры)

```
$ brew install qemu
$ brew install socket_vmnet
$ brew tap homebrew/services
$ HOMEBREW=$(which brew) && sudo ${HOMEBREW} services start
socket_vmnet
$ minikube start --cpus=4 --memory=8G --disk-size=20G --driver qemu
--network socket_vmnet
```

Запуск Minikube на Windows

```
$ DISM /Online /Enable-Feature /All /FeatureName:Microsoft-Hyper-V #
Enable Hyper-V
$ minikube start --cpus=4 --memory=8G --disk-size=20G --driver=hyperv
```

1. Включите поддержку Ingress в Minikube:

```
$ minikube addons enable ingress
```

Данная команда может не работать на некоторых версиях Minikube для Mac M1 и подобных процессоров.

2. Сам процесс установки выполняется с помощью Helm и был описан выше.
3. Настройка доступа в Moon по сети:
   a. **Вариант 1.** Используйте команду `minikube ip` для обновления настроек Moon.
      i. Обновление с помощью вывода команды `minikube ip`:

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

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

      ii. Добавьте `moon.aerokube.local` в файл `/etc/hosts`:

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

      На Windows вам может понадобиться обновить файл вручную.

   b. **Вариант 2.** Используйте туннель minikube. Этот вариант возможен только при использовании minikube с драйвером Docker.
      i. Добавьте `moon.aerokube.local` в `/etc/hosts`:

      ```
      $ sudo echo '127.0.0.1 moon.aerokube.local' >> /etc/hosts
      ```

      ii. Запустите туннель Minikube в отдельном терминале, введите пароль, когда потребуется:

      ```
      $ minikube tunnel
      ```

4. Откройте ссылку http://moon.example.com/ в браузере, вы увидите интерфейс пользователя. В коде Selenium тестов используйте ссылку http://moon.example.com/wd/hub.

2.2. Установка в Openshift

1. Системные требования:
   - Работающий кластер Openshift версии 4.x

- Установленный `oc` клиент с настроенным доступом к кластеру. Установка была протестирована на пользователе с правами администратора кластера Openshift.

2. Создайте проект для Moon (аналог неймспейса в Kubernetes):

```
$ oc new-project moon
```

   В следующих шагах мы предполагаем, что созданный проект называется `moon`.

3. Добавьте репозиторий с Helm чартами:

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

4. Установите или обновите Moon командой:

```
$ helm upgrade --install --set ingress.openshift=true -n moon moon aerokube/moon2
```

   Флаг `-n moon` указывает на проект, созданный на предыдущем этапе.

5. Отредактируйте идентификаторы пользователя и группы конфигурация объекта для совпадения со значениями, разрешенными политиками Openshift (например, установите идентификатор `1000650000`, конкретное значение зависит от конфигурации Openshift):

```
$ oc edit config.moon.aerokube.com default -n moon
```

Для тестирования на локальной машине вы можете использовать Openshift Local. В этом случае вам необходимо дополнительно передать имя хоста для Ingress следующим образом:

```
$ helm upgrade --install --set ingress.openshift=true --set
ingress.host=moon.apps-crc.testing -n moon moon aerokube/moon2
```

После запуска подов Moon добавьте `moon.apps-crc.testing` в `/etc/hosts`:

```
$ sudo echo '127.0.0.1 moon.apps-crc.testing' >> /etc/hosts
```

2.3. Рекомендации по Работе с Helm

2.3.1. Values-файлы Helm

Как было сказано выше, использование Helm чарта является рекомендуемым способом установки Moon в Kubernetes и Openshift. Наш Helm чарт поставляется с рекомендуемыми значениями настроек по-умолчанию, которые будут работать "из коробки" в большинстве случаев. Тем не менее, бывают случаи, когда необходимо установить Moon с заранее включенными продвинутыми функциями и иметь возможность легко воспроизводить данную конфигурацию позднее. Например, по-умолчанию в Moon отключена видеозапись браузерных сессий, но она может требоваться в ваших сценариях тестирования. Аналогично, вы можете захотеть запускать браузеры в нескольких неймспейсах или использовать дополнительные доверенные TLS сертификаты.

Хотя все эти возможность могут быть включены ручным редактированием соответствующих объектов Moon (конфигурационного объекта, набора браузеров, набора устройств) с помощью команды `kubectl edit`, все эти изменения будут перезаписаны Helm при следующей выкладке Helm чарта. Helm предоставляет лучший способ для хранения воспроизводимой конфигурации выкладки - values-файлы. Это работает следующим образом:

1. Каждый Helm чарт содержит файл `values.yaml` с рекомендуемыми значениями параметров по-умолчанию. Этот файл распространяется вместе с другими частями чарта. Например, актуальный файл `values.yaml` для Moon 2.x хранится здесь.
2. Если требуется поменять некоторые значения по-умолчанию на пользовательские значения, вы создаете локальный файл `values.yaml`, содержащий только значения, которые вы хотите переопределить. Например, в Helm чарте Moon число реплик Moon может быть задано, используя поле `deployment.replicas` в `values.yaml`. При этом ваш файл `values.yaml` будет выглядеть так:

   ```
   deployment:
     replicas: 3
   ```

   Такой файл следует хранить под предпочитаемой вами системой контроля версий. Такой подход гарантирует, что вы никогда не потеряете пользовательские настройки для установки Moon.

3. Для того, чтобы применить свой файл `values.yaml` при установке (переустановке) Moon - передайте полный путь до `values.yaml` в команде Helm:

   ```
   $ helm upgrade --install -f путь/до/values.yaml -n moon moon aerokube/moon2
   ```

   Переопределение параметров `values.yaml` аналогично, но более удобно, чем передача тех же самых ключей через флаг `--set`, например:

   ```
   $ helm upgrade --install --set deployment.replicas=3 -n moon moon aerokube/moon2
   ```

2.3.2. Настройка Пользовательских Ресурсов Moon

В Moon 2.x все настройка хранятся как встроенные объекты Kubernetes, называемые пользовательскими ресурсами:

- Квоты - хранят конфигурацию для одного пользователя Moon.
- Конфигурационные объекты - хранят различные конфигурационные опции.
- Наборы браузеров - хранят настройки запуска браузеров.
- Наборы устройств - предоставляют информацию о списке доступных для мобильной эмуляции мобильных устройств.
- Лицензии - хранят установленные лицензионные ключи Moon.

Наш Helm чарт имеет отдельный ключ в `values.yaml`, соответствующий каждому из этих ресурсов:

Пользовательские ресурсы Moon могут быть также настроены через файл values.yaml

```
quota: # Хранит список квот
  quota1:
    # Параметры квоты Quota1
  quota2:
    # Параметры квоты Quota2

configs: # Хранит список конфигурационных объектов
  config1:
    # Параметры конфигурационного объекта Config1
  config2:
    # Параметры конфигурационного объекта Config2

browsers: # Хранит список наборов браузеров
  browsersset1:
    # Параметры набора браузера Browsersset1
  browsersset2:
    # Параметры набора браузера Browsersset2

license: # Хранит пользовательский лицензионный ключ для Moon (в данный момент Helm
чарт поддерживает только один лицензионный ключ)
```

Как вы видите, для всех пользовательских ресурсов кроме лицензий вы можете определить несколько разных объектов: несколько конфигурационных объектов, несколько квот и несколько наборов браузеров. Все поля, которые вы выставляете в спецификации пользовательского ресурса преобразуются "как есть" в соответствующий ресурс Kubernetes. Например, пусть в `values.yaml` определен набор браузеров:

Пример определения набора браузеров в values.yaml

```
browsers:
  my-browserset:
```

```
    selenium:
      firefox:
        repository: quay.io/browser/firefox
      chrome:
        repository: quay.io/browser/chrome
```

Когда вы применяете Helm чарт с таким определением в файле values.yaml, будет автоматически создан такой ресурс Kubernetes типа BrowserSet (т.е. набор браузеров) Kubernetes:

Эквивалентные объект типа "набор браузеров" в Kubernetes

```
apiVersion: moon.aerokube.com/v1
kind: BrowserSet
metadata:
  name: my-browserset
  namespace: moon # Здесь будет имя неймспейса, куда установлен Moon
spec:
  selenium:
    firefox:
      repository: quay.io/browser/firefox
    chrome:
      repository: quay.io/browser/chrome
```

Если вы создаете несколько объектов в списке наборов браузеров в `values.yaml`, то будет создано соответствующее количество пользовательских ресурсов Kubernetes.