Функциональное описание Q.PorteR

Exported on Jul 08, 2019

Porter это платформа для разработки, развертывания и запуска приложений в контейнерах. Porter позволяет отделить приложение от инфраструктуры, что даёт возможность ускорить цикл разработки и развертывания приложений.

Использование linux-контейнеров для простого развертывания приложений не ново, и называется контейнеризация.

Контейнеризация приобретает всё большую популярность потому что она обеспечивает:

- Гибкость - даже самые сложные приложения могут быть упакованы;
- Легковесность - контейнеры разделяют ресурсы хоста;
- Взаимозаменяемость - вы можете разворачивать обновления "на лету";
- Мобильность - вы можете собирать приложения локально, развертывать в облаке и запускать где угодно;
- Масштабируемость - Количество копий приложения увеличивается автоматически при необходимости;

Porter использует образы для запуска контейнеров. Образ - это исполняемый пакет, который содержит все необходимое для запуска приложения - библиотеки, переменные окружения и конфигурационные файлы.

Контейнер - это запущенный экземпляр образа, который разворачивается в памяти после запуска.

Контейнер запускается как нативное приложение, без дополнительной нагрузки гипервизора, что позволяет более полно использовать ресурсы хоста.

Таким образом Porter позволяет с легкостью организовать цикл разработки. Разработчики используют локальные контейнеры с приложениями, затем передают эти контейнеры на тестовые стенды, откуда они могут быть переданы на рабочие стенды.

Основывая свою систему на контейнерах, платформа позволяет легко переносить полезную нагрузку. Контейнеры могут работать как на локальной машине, так и на виртуальной машине или в облаке.

1 Архитектура Porter

Porter использует клиент-серверную архитектуру. Porter клиент общается с Porter демоном, который создает, запускает и распределяет контейнеры. Клиент и сервер могут работать как на одной машине, так и на разных. Это означает, что используя RESTFul API, клиент может подключаться к удаленным хостам с развернутыми контейнерами.

1.1 Porter демон

Porter демон (porterd) слушает запросы Porter API и управляет объектами Porter, такими как образы, контейнеры, сети и тома. Демон также может взаимодействовать с другими демонами для управления сервисами Porter.

1.2 Porter клиент

Porter клиент используется для взаимодействия пользователей с Porter. Когда вы выполняете команды, такие как porter run, клиент отправляет эти команды porterd, который непосредственно выполняет их. Клиенты Porter используют Porter API и могут взаимодействовать более чем с одним демоном.

1.3 Porter репозиторий

Porter репозитории содержат Porter образы. Вы можете использовать своё приватный или общедоступный репозиторий. Когда вы запускаете команды porter run или porter pull, требуемые образы выкачиваются из настроенного репозитория. Когда вы запускаете команду porter push, ваш образ загружается в настроенный репозиторий.

2 Porter объекты

Когда вы используете Porter, вы создаете и используете образы, контейнеры, сети, тома, плагины и другие объекты. В этой секции дается краткий обзор этих объектов.

2.1 Образы

Образ - это read-only шаблон с инструкциями по созданию Porter контейнера. Часто образы базируются на других образах с некоторой дополнительной настройкой. Например, вы можете построить образ, который базируется на операционной системе Ubuntu, но дополнительно устанавливает Web-сервер Apache и ваше приложение, а также делает настройки, необходимые для работы вашего приложения. Вы можете создавать свои собственные образы, или вы можете использовать образы, созданные другими разработчиками и опубликованные в общедоступном хранилище.

Для того, чтобы создать собственный образ, вы создаете Dockerfile, в котором используется простой синтаксис для описания шагов, необходимых для создания и выполнения образа. Каждая инструкция в Dockerfile создает слой в образе. Когда вы меняете Dockerfile и перестраиваете образ, изменяются только те слои, которые были изменены. Это позволяет делать образы легковесными, маленькими и быстрыми по сравнению с другими технологиями виртуализации.

2.2 Контейнеры

Каждый контейнер создается из образа, и содержит всё, что требуется приложению для работы. Вы можете создавать, запускать, останавливать, перемещать или удалять контейнеры, используя Porter API или клиента Porter. Вы можете настроить для контейнера одну или несколько сетей, подключить тома, или даже создать новый образ на основе текущего состояния контейнера.

По умолчанию, контейнеры относительно хорошо изолированы от других контейнеров и от хост-машины. Вы можете настраивать уровень изоляции сети, томов и других подсистем.

Контейнер определяется образом, из которого он был создан, а также опциями конфигурации, которые вы указали при его создании или запуске. Когда контейнер удаляется, все изменения в его состоянии, которые не сохранены во внешнем по отношению к контейнеру постоянном хранилище данных, безвозвратно удаляются.

2.2.1 Пример выполнения команды porter run

Следующая команда запускает контейнер с ОС Ubuntu, подключает текущую сессию терминала и запускает /bin/bash

$ docker run -i -t ubuntu /bin/bash

Когда вы выполняете эту команду, происходит следующее:

- Если у вас нет локального образа, Porter получает его из настроенного репозитория, также как если бы выполнили отдельную команду porter pull
- Porter создает новый контейнер, также как если бы выполнили отдельную команду porter container create
- Porter создает для контейнера слой файловой системы для чтения-записи. Это позволит работающему контейнеру создавать и изменять файлы и каталоги в своей локальной файловой системе.
- Porter создает сетевой интерфейс для подключения контейнера к сети по умолчанию, так как не было указано никаких сетевых опций. Это включает присвоение IP адреса контейнеру. По умолчанию, контейнер может подключаться к внешним сетям, используя сетевое соединение хост-машины.

- Porter запускает контейнер и выполняет bin/bash. Поскольку контейнер выполняется интерактивно и подключен к вашему терминалу (благодаря флагам -i и -t), вы можете вводить команды, используя клавиатуру и видеть вывод на своём терминале.
- Когда вы дадите команду exit для завершения /bin/bash, контейнер остановиться, но не будет удален. Вы можете запустить его снова или удалить его.

2.3 Сервисы

Сервисы позволяют масштабировать контейнеры, используя множество демонов Porter, которые работают сообща на нескольких хостах. Демоны взаимодействуют друг с другом используя Porter API. Сервис позволяет задать
требуемое состояние, например, число экземпляров приложений, которые должны быть доступны в любой момент времени. По умолчанию выполняется балансировка нагрузки на всех рабочих узлах. Для клиента Porter сервис выглядит как единое приложение.