Контейнеризация

Мотивация

Программист - устройство в которое пихаешь кофе а на выходе получается код (**текст программы**)

Если программист хочет поделиться с кем-нибудь своей программой он должен передать код кому-то у кого есть компьютер способный этот код понимать и выполнять. Если до интернета носителем программного кода были перфокарты, магнитные ленты и магнитные диски, то с появлением интернета текст программы стали распространять через почтовую рассылку, а позже публиковать исходный код на специальных интернет ресурсах - репозиториях.

Для того что бы воспользоваться исходным кодом, как правило требуется его **запустить** (собрать) в специальном (сборочном) **окружении** в котором есть доступ ко всем зависимостям программы. По этой причине часто можно увидеть разные способы распространения программ, в виде исходного кода или в виде бинарника собранного под конкретную операционную систему.

В ряде случае еще более удобный способ доставки программного продукта до конечного пользователя - это создание сервиса который основную часть логики исполняет где-то на **сервере**, а конечный пользователь взаимодействует с **клиентской частью** полученной ввиде вебстраницы в его браузере, или мобильном приложении в телефоне.

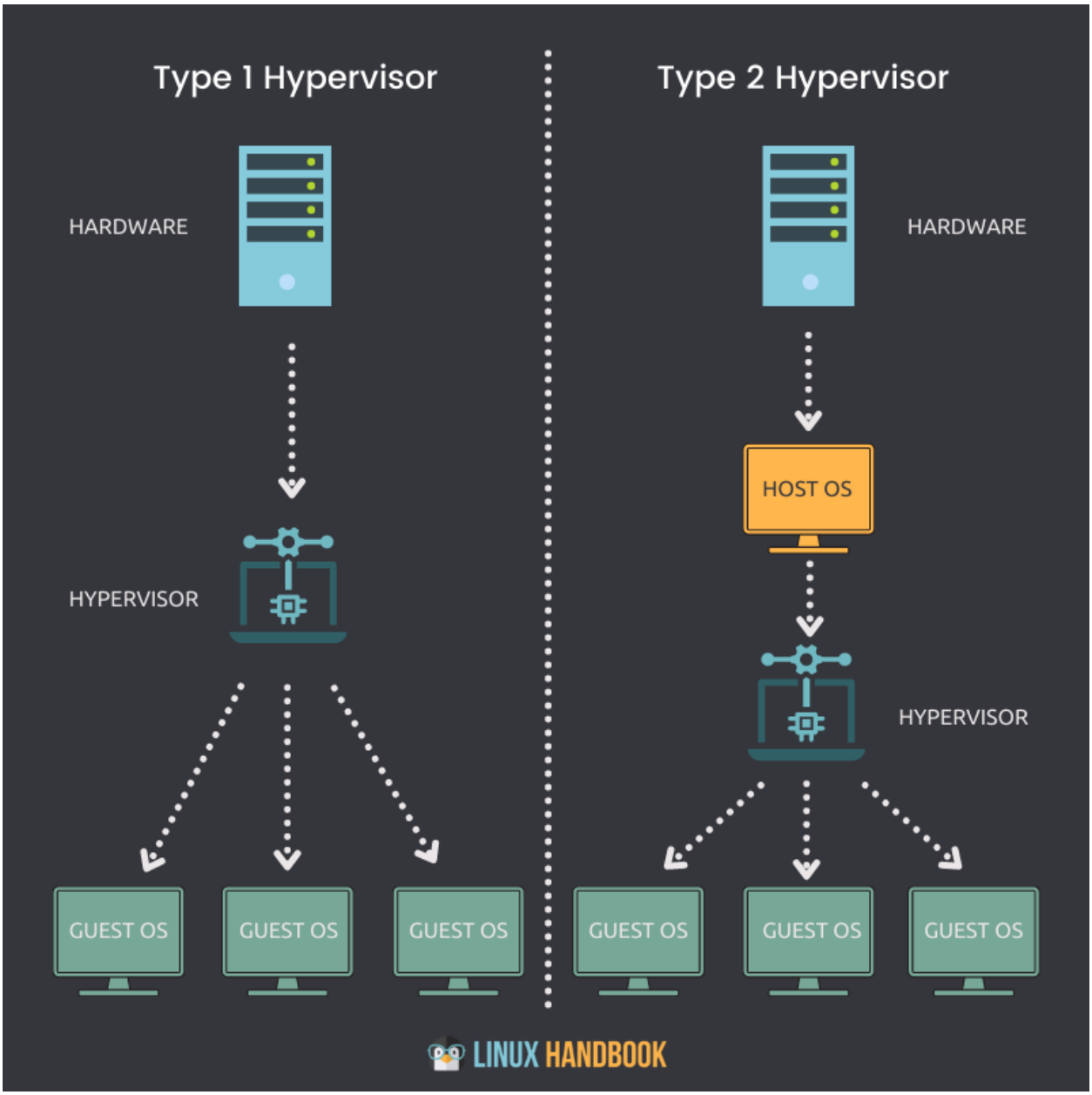
Для того что бы поддерживать серверную часть необходим системный администратор, который установит необходимую операционную систему на сервер, настроит окружение, поставит все зависимости нужных версий, и аккуратно будет обновлять все необходимые компоненты по мере развития продукта. Если в какой-то момент обнаружится что свежее прилетевшее обновление все сломало, или какое-то железо вышло из строя, то системному администратору придется в ручную откатывать все измнения и всё исправлять.

Иногда нужно что бы сервис продолжал работать во время обновления, и тогда хорошо бы запустить двойника сервера на другой машине, сначала накатить на нем все обновления, убедиться что ничего не упало, подключить к нему часть аудитории чтобы протестировать его работоспособность, и только потом

переключить на него всю нагрузку. А может быть нужно повысить пропускную способность сервиса, и добавить еще серверов. Но они нужны не всегда, а только когда большая нагрузка, и вообще-то их дорого содержать.

А что если неиспользуемые мощности арендавать кому-нибудь еще кому нужны свои сервера, но так что бы они там не напортачили, и что бы легко было переключить сервера обратно, и не приходилось переустанавливать зависимости. Желательно что бы всю необходимую информацию для работы можно было представить в ввиде файла и при необходимости скопировать на другую машину и запустить там.

Техническая реализация



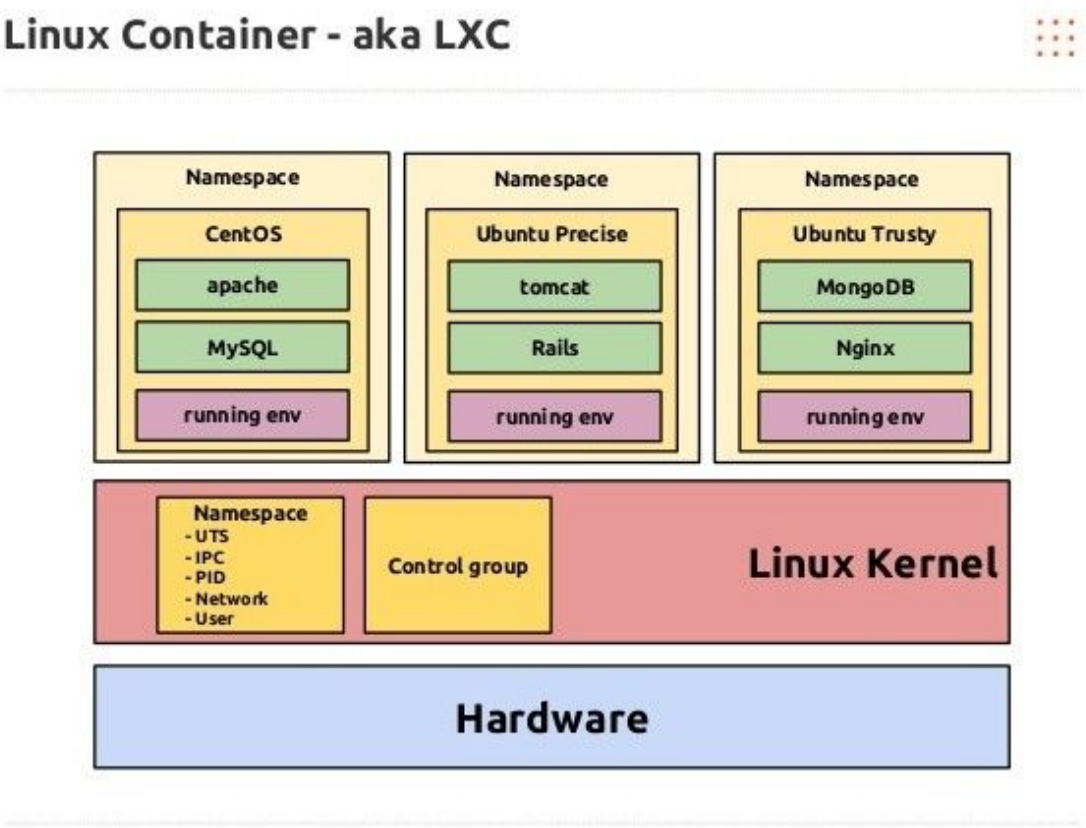
Примерно к этому моменту (~2000 гг) UNIX- подобные системы начинают поддерживать виртуализацию и появляются гипервизоры второго типа.

Гипервизоры были придуманы (~1960гг) для разделения процессорного времени и ресурсов компьютера между пользователями, для тестирования новых операционных систем и аппаратных концепций.

Гипервизор - программная оболочка позволяющая одновременное, параллельное выполнение нескольких **операционных систем** на одном и том же хост-компьютере. Гипервизор эмулирует все физические устройства необходимые

для работы гостевой системы, при это находясь физически на одной машине гостевые операционные системы как правило полностью изолированы друг от друга и могут взаимодействовать только через общие внешние устройства или сетевые протоколы.

Такой подход позволяет останавливать и запускать гостевые системы без потери всех запущенных процессов и их контекста.



Основная функция операционой системы - абстрагирование запускаемых в ней приложений от реальных физических устройств с помощью системных вызовов (API), которые на основе внутренней логики операционной системы обеспечивают доступ к реальным или виртуальным устройствам.

Кроме использования гипервизора в операционных системах семейства UNIX есть еще способ разделения ресурсов компьютера - это использование разных пространств имён. Такой подход позволяет запускать процессы с одинаковыми идентификаторами изолированно друг от друга. Для переключения пространства имён необходимо выполнять операцию **chroot** и изменить **корневой каталог**

системы. После выполнения этой команды сеанс внутри **командной оболочки** будет вести себя так словно запущена другая операционная система, поскольку эта операция меняет только контекст и не затрагивает **ядро операционной системы,** мы таким образом можем переключиться в любую операционную систему совместимую с текущим активным ядром. Этот изолированный экземпляр операционной системы со своим окружением назвают **контейнером**. Использование команды chroot самый старый (~1980гг) но не единственный способ управления пространством имён, их в настоящее время существует большое количество и наибольшее распростронение получил **docker**.

Переключение пространства имён в отличии от полноценной гостевой ОСи не дает такой свободы действий как отдельная виртуальная машина, однако это гораздо более легковесный способ, что позволяет одновременно разворачивать больше контейнеров и делать это быстрее.

На практие как правило применяется комбинация этих технологий, в сочетании с оркестраторами - программными комплексами для автоматического запуска виртаульных машин, поднятия между ними соединений а так же разворачивания и запуска контейнеров, их репликации при необходимости увеличения пропускной способности и остановки неиспользуемых компонентов для экономии ресурсов.