Не тратим время на Docker: контейнеры systemd-nspawn

Новоселов Михаил Евгеньевич, АО «НТЦ ИТ РОСА», г. Москва, ул. Марксистская, д. 3, стр. 7, m.novosyolov@rosa.ru

Обычно словосочетание «контейнер в Linux» ассоциируется с программным обеспечением Docker.

Контейнеризация позволяет:

- минимизировать нагрузку: поскольку контейнеры используют ядро ОС хоста, они занимают гораздо меньше места и требуют значительно меньших ресурсов для своего функционирования;
- минимизировать сроки развертывания: контейнеры создаются и уничтожаются почти мгновенно, что делает их хорошо подходящими для краткосрочных рабочих нагрузок и кратковременных тестов;
- обеспечить совместимость: образы Docker стандартизированы и переносимы практически на любую систему Linux, что облегчает миграцию и использование нескольких сред одновременно,
- достичь высокой производительности в сочетании с высокой плотностью размещения рабочих нагрузок на узле: контейнеры работают производительнее, чем виртуальные машины, так как непосредственно взаимодействуют с системой хоста, обходя необходимость виртуального уровня гипервизора.

Среди причин распространенности именно Docker в качестве средства контейнеризации можно выделить:

- автоматизация жизненного цикла: более-менее воспроизводимая сборка контейнера по Dockerfile и переносимость результата;
- отсутствие необходимости проектировать сложную инфраструктуру;
- возможность взять и запустить готовый контейнер, не вникая в его устройство и не задумываясь о воспроизводимости его сборки;
- большое сообщество: готовые интеграции со множеством другого программного обеспечения, Docker’у обучают на большинстве онлайн курсов — легко найти хотя бы немного знакомых с ним людей.

Однако у Docker есть серьезные недостатки, например:

- высокий порог входа: чтобы не просто копировать и вставлять команду запуска готового контейнера, а хотя бы приблизительно понимать, как его собрать, как устроено хранение данных, каков набор технологий для запуска контейнера и т. д., нужно изучить большой объем информации;
- Docker предназначен только для запуска одного процесса в контейнере, а не запуска всей ОС целиком, как в виртуальной машине — состоящие из множества компонентов задачи приходится разбивать на несколько контейнеров, даже если это неудобно и не повышает защищенность;
- по умолчанию изолированные в контейнере процессы запускаются от root, user namespaces не используются, в случае нарушения изоляции можно получить root-доступ к хосту изнутри контейнера.

Инструмент **systemd-nspawn** позволяет:

- работать с контейнером как с виртуальной машиной, запуская несколько сервисов в одном контейнере (например, MySQL и веб-сервер);
- работать с контейнером как с обычной ОС и обычными файлами, не переусложняя образами, сложной системой хранения и т. п.;

- изолировать с использованием пользовательских пространств имен (user namespaces): root в контейнере не является root на хосте, став root в контейнере не станешь root на хосте;
- организовать резервное копирование контейнеров как обычных файлов: тарболлы, squashfs, rsync, снимки BTRFS и др.;
- использовать контейнеры и понимать, что делаешь, не изучив Docker и огромный пласт технологий.

В докладе будет рассмотрен systemd-nspawn как инструмент контейнеризации рабочих нагрузок, будут приведены примеры и опыт его использования. Отдельно будут затронуты вопросы обеспечения должной изоляции и защиты при использовании таких контейнеров.