В данной статье я хочу рассказать об организации собственного приватного хостинга для размещения своих сайтов. В качестве основы будет выделенный сервер под гипервизор с разделением функционала на виртуальные машины.

Содержание:

Цели статьи

1. Рассказать о схеме построения приватного хостинга на базе выделенного сервера или серверов.
2. Подробно описать роль каждой виртуальной машины в составе хостинга.

3. Поделиться своим реальным опытом на тему настройки и сопровождения веб серверов.
4. Наполнить статью ссылками на свои материалы по заданной теме, чтобы можно было на практике повторить все описанное ниже.

Введение

Данная статья будет чисто теоретической с тематическими ссылками на мои статьи, если таковые будут присутствовать в контексте повествования. Я решил поделиться своим видением и опытом в построении приватного хостинга для размещения одного или нескольких сайтов в рамках одной команды или владельца. То есть система в описанном виде не будет предназначена для публичного доступа к сервисам. Только личное использование закрытого круга лиц.

Приватный хостинг в минимальной конфигурации будет состоять из следующих компонентов, которые будут разделены виртуальными машинами:

- Frontend в виде проксирующего nginx.
- Backend сервер в связке с nginx + php-fpm либо Bitrixenv для сайтов на bitrix.
- Zabbix сервер для мониторинга.
- Elk Stack для хранения и анализа логов web сервера.

Это минимальное количество структурных единиц, которые я использую. В зависимости от задач и проектов, их может быть больше. Например, можно отдельно вынести функционал сервера баз данных. Если у вас используются сильно разные сайты, можно сделать несколько backend серверов, чтобы не разводить зоопарк на одном. К примеру, bitrix сайты я всегда разворачиваю отдельно.

Часто необходим свой почтовый сервер. Его тоже рекомендую разворачивать отдельно, причем желательно на другом ip адресе, чтобы иметь возможность эффективно противостоять ddos атакам. Если используется взаимодействие с удаленной инфраструктурой, можно настроить шлюз в отдельной виртуальной машине и использовать ее в качестве vpn сервера или клиента. В простейшем случае роль шлюза выполняет сам гипервизор.

Так же, в случае необходимости, я отдельно разворачиваю виртуальные машины с Youtrack, Teamcity, Onlyoffice, Gitlab. Можно компактно разместить все необходимое для работы небольшой команды разработчиков.

Изначально все настраивается на одном гипервизоре, но структура проекта позволяет без проблем его масштабировать на несколько серверов, либо выполнить переезд в облако. Несмотря на то, что сейчас активно развиваются кластеры (Kubernetes) и облачное размещение, предложенная мной схема полностью актуальна и востребована. Основные причины актуальности:

1. В Kubernetes имеет смысл размещать только те проекты, которые изначально построены на основе микросервисов. WordPress и Bitrix в k8s размещать неудобно.

2. Размещение на собственных серверах дает прогнозируемую производительность и максимальное выгодное соотношение производительности к стоимости. То есть это банально дешевле облаков раза в 2-3.

Итак, разберем теперь отдельно каждую виртуальную машину. В качестве базовых систем я использую Centos 8, но это не принципиально. Используйте ту систему, что вам больше нравится. Это может быть и Debian, и Ubuntu.

Гипервизор для своего хостинга

В подавляющем большинстве случаев я использую виртуализацию на базе Proxmox. Выделенный сервер арендую у Selectel. В зависимости от бюджета и потребностей, это может быть простенький сервер из двух дисков на десктопном железе, либо полноценный сервер с железным рейд контроллером и постоянной ip-kvm панелью.

Если сервер без рейд контроллера, то используется софтовый рейд mdadm. Установка и настройка proxmox на софтовом рейде у меня подробно описана. У Selectel удобно выполнены базовые шаблоны для установки. Если я заказываю сервер, то сразу выбираю в качестве системы Debian на raid1. Установщик автоматически накатывает систему на mdadm. Остается только установить гипервизор. Самому разбивать диски и собирать mdadm не придется.

В простейшем случае в качестве шлюза используется хостовая система гипервизора. Iptables и Nat настраиваются на ней. Если у вас будут использоваться несколько выделенных серверов и их придется объединять по vpn через интернет, то я настраиваю шлюз в виде отдельной виртуальной машины. Можно и на самом гипервизоре, но я не люблю смешивать функционал. К тому же, когда у вас шлюз в отдельной виртуальной машине, его проще забэкапить и развернуть в другом месте. То есть в целом переезд проекта будет проще и быстрее.

Если у вас только один внешний IP адрес, то на гипервизоре используются 2 сетевых интерфейса:

1. Реальная сетевая карта с настроенным внешним IP адресом.
2. Виртуальный бридж, обычно vmbr0 для сети виртуальных машин, а так же для их связи с самим гипервизором.

Если используются несколько IP адресов, которые нужно будет назначать разным виртуальным машинам, то сетевых интерфейса будет 3:

1. Реальная сетевая карта без настроек IP адреса.
2. Бридж vmbr1, в который будет включен сетевой интерфейс, в который приходит линк с внешними ip адресами. На этом бридже будет настроен

реальный IP адрес самого гипервизора, если нет отдельного шлюза. Этот же бридж будет подключен к тем виртуальным машинам, где нужен внешний ip адрес.
3. Бридж vmbr0 для виртуальной сети виртуальных машин.

Подробно с примерами вопрос сетевых настроек гипервизора я рассматриваю в статье про настройку proxmox, ссылку на которую дал в начале.

С гипервизором разобрались, переходим к следующему элементу web хостинга — frontend серверу.

Frontend сервер nginx

В качестве frontend сервера выступает Nginx, работающий в режиме proxy_pass. В контексте данного раздела я рекомендую 2 статьи на тему настройки Nginx — настройка проксирования и базовая настройка nginx.

В данном случае не обязательно использовать именно Nginx. В некоторых ситуациях более актуальным будет взять HAProxy. Но в общем случае подойдет Nginx. На frontend сервер будут поступать все внешние http запросы. На него будут проброшены порты 80 и 443. Расскажу, для чего использовать отдельный frontend сервер. Ведь можно обойтись и без него.

1. Удобство эксплуатации и обновления. К примеру, я хочу попробовать какую-то новую настройку — изменение ядра, модуля nginx и т.д. Я делаю копию frontend, настраиваю на нем все, что нужно и переключаю на него трафик. Если все в порядке, то оставляю в работе, либо переношу настройки на основной сервер. Если возникают какие-то проблемы, я просто переключаю проброс порта на старый сервер и все сразу начинает работать как прежде. В общем случае, frontend сервер очень маленький (20-30 гб под систему и логи).
2. На frontend удобно работать с внешними запросами — контролировать, обрабатывать, блокировать, предотвращая доступ к бэкенду с данными. Например, можно настроить ipset и блокировать отдельные страны. Можно выстроить простейшую защиту от ddos. Можно настроить отдельный модуль ModSecurity для Nginx. Когда у вас frontend отдельно, вам проще настраивать и обновлять компоненты, не боясь нарушить работу сайтов.
3. Frontend сервер аккумулирует все логи с внешних запросов и передает в ELK. Далее я покажу, как их удобно в автоматическом режиме сразу все передавать на хранение, а потом удобно анализировать.
4. В общем случае с отдельным frontend сервером безопаснее. Все внешние запросы приходят к нему. Многие зловреды (как и легальные компоненты и плагины) не понимают, как работать на бэкенде, который стоит за frontend. Это как плюс так и минус в некоторых случаях, так как усложняется эксплуатация. Особенно хлопотно с битриксом, так как у него много сервисов и проверок, которые напрямую стучатся по внешнему ip домена и не понимают, что делать, когда попадают на фронтенд. Но это все решаемо.

На frontend настраиваются бесплатные сертификаты от Let’s Encrypt. Так как весь трафик от фронта до бэка крутится в рамках одного гипервизора, смысла передавать его в шифрованном виде нет. Так что на backend он идет по http.

С frontend я отправляю в ELK логи запросов только к php скриптам. Именно они представляют для меня полезную информацию, так как позволяют оценить скорость работы сайта. Информация от отдачи статики лично мне не нужна. В общем случае, я ее не собираю, а включаю отдельно по мере необходимости. Это позволяет не раздувать лог файлы. Чем меньше их объем, тем удобнее и быстрее с ними работать.

Backend сервер nginx, apache, php

Про бэкенд сервер рассказывать особо нечего. Он настраивается в зависимости от потребностей проекта. В общем случае для php сайтов это будет либо настройка nginx + php-fpm, либо apache + php. Как я уже говорил, бэкендов может быть несколько. Если вы web студия или какое-то агенство, которое само хостит сайты клиентов, то у вас может быть как классический web сервер php, так и bitrixenv для размещения битрикс сайтов. А они сейчас очень популярны. Почти все интернет магазины, с которыми я работал, были на битриксе. Плюс коробки с bitrix24 иногда покупают. Если сотрудничаете с малым или среднем бизнесом, без битрикса скорее всего не обойтись. Я его хоть и не люблю, но работать приходится.

В общем случае на backend я не настраиваю ssl, но бывают исключения или различные ошибки. Вот примеры таких ошибок в работе типовых php сайтов:

- Ошибка WordPress
- Ошибка phpmyadmin

У Битрикса тоже есть похожие ошибки, но я их не зафиксировал в статьях.

Я каждый сайт размещаю в отдельной директории, например */mnt/web/sites/site.ru*. В этой директории уже свои поддиректории www, logs, php_sessions и т.д. Владелец каждого сайта — отдельный системный пользователь. От этого пользователя работает php-fpm пул, который обслуживает только этот сайт. Для каждого пользователя настроен sftp доступ к конкретному сайту. Каждый сайт имеет доступ только к своей базе mysql или postgresql.

При такой схеме получается практически полная изоляция сайтов. Они крутятся только в своих песочницах. Плюс, легко организовать доступ к отдельному сайту в случае необходимости. Это можно было бы заменить контейнерами для полной изоляции, но я считаю, что в таком кейсе приватного хостинга это лишняя сущность, хотя понимание, как это можно организовать с помощью docker у меня есть. Но он все же для других случаев.

Мониторинг сайтов и серверов

Для мониторинга виртуальных машин и сервисов нашего хостинга я всегда использую Zabbix. У меня накопилось огромное количество статей по нему практически на все случаи, с которыми я сталкиваюсь. В общем случае я настраиваю:

1. Мониторинг mdadm или железного контроллера. По последним, к сожалению, у меня нет статей, но в целом проблем с настройкой не возникает. У меня всегда гуглились подходящие решения. Если у сервера есть idrac, ilo, ipmi, можно с них брать нужные данные.
2. Если есть доступ к смарту дисков, то настраиваю мониторинг smart. Очень рекомендую это делать, чтобы в случае выхода из строя какого-то диска, у вас была полная информация о нем, чтобы передать ее в службу технической поддержки для замены.
3. Мониторинг подключений по ssh. Мне сразу приходит уведомление, если кто-то подключается к серверу по ssh. Если доступ есть не только у меня, то обязательно это настраиваю. Сильно упрощает жизнь и готовит к проблемам :) Если доступ только у меня, то это небольшая защита и возможность быстро среагировать на несанкционированный доступ, хотя в реальности у меня ни разу такого не было.
4. Мониторинг веб сервера, в данном случае frontend и backend. Иногда мониторинг бэка не делаю. Реально не так уж часто он нужен, хотя кажется, что полезно получать все метрики. Но лично моя практика такова, что они мне на деле чаще всего не нужны.
5. Мониторинг сайта. Это одна из самых главных метрик, так как напрямую отвечает на вопрос, все ли у нас в порядке. Если сайт не работает или не доступен, то это наивысший приоритет проблемы. Так как мониторинг у нас локальный, он не дает полную картину происходящего, нужен еще один внешний. О нем подробнее расскажу далее. Локальный мониторинг сразу определяет, к примеру, если у нас упал backend и вместо страницы сайта видим 500-ю ошибку nginx. Или что-то еще. В общем, важная штука, рекомендую внимательно отнестись к мониторингу сайта. Рекомендую к нему обращаться напрямую через внутреннюю сеть гипервизора по локальному ip фронта. Для этого надо либо в host файл виртуалки с zabbix добавить все сайты по локальному ip, либо завести свой локальный dns сервер. Обычно я это делаю, если используется отдельная виртуальная машина под шлюз.
6. Мониторинг делегирования домена и ssl сертификата. Штука не обязательная, настраивается по желанию. Если делегирование не так критично, так как регистраторы завалят напоминаниями на почту, то мониторинг ssl сертификатов рекомендую сделать. Их часто забывают продлить или возникают технические ошибки при работе с автопродлением Let’s Encrypt.
7. Я всегда настраиваю мониторинг бэкапов в том или ином виде. Он сильно зависит от конкретной ситуации, от данных, от места хранения бэкапов и т.д. Готовых решений нет, приходится импровизировать на месте. Но если не настрою мониторинг бэкапов, не могу спать спокойно. Бэкапы периодически разворачиваю вручную и проверяю. Это сильно ограничивает количество клиентов, с которыми могу сотрудничать, так как труд ручной. Но это меня много раз спасало. Так что не пренебрегаю.
8. Если есть почтовый сервер, настраиваю мониторинг postfix. За почтовым сервером рекомендую внимательно следить, особенно за очередью и количеством отправленных сообщений. Иногда учетки ящиков утекают в сеть и сервер начинает массово спамить. Если вовремя это не заметить и не остановить, можно залететь в спам листы и надолго там засесть. Это может парализовать работу того же интернет магазина, так как без почты он перестает нормально функционировать.

Основное по мониторингу перечислил. Частенько настраиваю что-то еще, в зависимости от потребностей конкретного заказчика. Если решение не типовое и нишевое, то статью не пишу по нему, хотя шаблоны себе сохраняю. Если есть какие-то критичные службы linux, можно мониторить еще и их.

Особенно удобно мониторить отклик сайта с локального сервера. Здесь нет сетевых задержек, которые возникают при работе внешнего мониторинга. Тут чистая производительность web сервера. Вкупе с внешним мониторингом получается полная и легко интерпретируемая картинка производительности веб сервера и скорости доступа к сайту. Только с двумя мониторингами — внешним и внутренним, можно адекватно оценивать и искать узкие места в работе сайта.

Хранение логов в ELK Stack

Я складываю все логи в elasticsearch. У меня есть статья про установку и настройку ELK Stack. Недавно я обновил инструкцию по установке, но скрины оставил старые. Очень хлопотно их заменять. Сам процесс установки отражен правильно, так как я регулярно пользуюсь своей статьей. У меня есть несколько примеров того, как можно анализировать логи различных сервисов.

В контексте данной статьи по настройке приватного хостинга нас будет интересовать сбор web логов и их анализ:

- Dashboard для логов Nginx в Kibana+Elasticsearch
- Мониторинг производительности бэкенда с помощью ELK Stack

Статьи немного устарели в том плане, что в процессе эксплуатации мои дашборды изменились, но принцип тот же. Главное его понять, а дальше уже не будет проблем делать так, как удобно лично вам. Например, я не настраиваю GEO карты. В реальности они мне не нужны. Так, для красоты только. Ниже пример моего актуального дашборда для этого сайта.

По дашборду я сразу получаю актуальную информацию о состоянии сайта — информация о средней скорости ответа на php запросы и карта распределения ответов по шкале. Почти все запросы укладываются в интервалы до 300 мс, что считаю хорошим результатом. Напоминаю, что это информация только о php запросах. На сайте настроено кэширование, так что большинство страниц уходят к посетителю значительно быстрее напрямую через nginx, минуя обработчик php.

Тут же можно сделать выборку по медленным запросам, по запросам с определенных ip адресов, посмотреть запросы с различными кодами ошибок и т. д. В общем, без такого дашборда я ощущаю себя слепым. Я не понимаю, как понять, что с сайтом все в порядке, или наоборот узнать, какие у него проблемы, если у тебя нет под рукой подобной информации. Сайт может начать сыпать пятисотыми ошибками, а тебе надо как-то вручную грепать access log и пытаться понять, проблема единичная или масштабная. А тут все под рукой.

Я так привык в ELK, что на сервера почти не хожу. Все логи собираю в нем (обязательно системные) и там же просматриваю. Плюс к этому мониторинг и управление через ansible, но об этом позже. Ходить на сервера по ssh практически нет необходимости.

Такой подход очень хорошо масштабируется, поэтому я его и использую, хотя на моих масштабах это и не так актуально, но тем не менее, хочется все делать правильно с заделом на будущее. У меня был проект, который начался с одного сервера и нескольких докер контейнеров, а закончился примерно сотней виртуалок. Я очень пожалел, что с самого начала не начал автоматизировать процессы. Просто не был готов к этому. Не было опыта и понимания. Все росло постепенно и каждый раз вручную сделать было быстрее. Но в какой-то момент я стал просто зашиваться. Повезло, что проект в итоге усох, но не по моей вине :)

На фронте у меня логи всех сайтов складываются в одну директорию */var/log/nginx* и оттуда единым шаблоном уходят в filebeat, а с него в logstash и далее в elasticsearch в один общий индекс, который бьется по дням. Раньше я каждый сайт отправлял в отдельный индекс, но со временем понял, что это не удобно. Так приходится для каждого индекса создавать свои визуализации и дашборды. Когда сайтов много это хлопотно, хотя и можно автоматизировать, но большого смысла нет на моих масштабах.

Сейчас я собираю все в один индекс, делаю единый dashboard и в нем уже с помощью фильтров просматриваю данные по разным сайтам. Я вывел в лог nginx информацию об имени виртуального домена. Это удобно и быстро настраивается. Для каждого нового сайта не надо вообще ничего делать. Filebeat автоматом забирает его логи. С помощью фильтра в Kibana просматривается информация в логах.

Почтовый сервер и шлюз

На настройке почтового сервера и шлюза для нужд хостинга подробно останавливаться не хочется. Статья и так очень большая получается. К тому же тут особых нюансов нет. Нужен обычный шлюз и обычный почтовый сервер. Можно взять готовую сборку, к примеру, **iredmail** или **zimbra**. Я иногда использую первый. Но чаще всего настраиваю все сам на базе postfix + dovecot + postfixadmin + roundcube. Так же рекомендую свою статью-рассуждение на тему, какой почтовый сервер выбрать.

В качестве шлюза так же можно взять либо готовую сборку, к примеру, на базе **pfsense**, **clearos** или чего-то подобного, либо настроить все самому. Я обычно настраиваю сам, так как не люблю зоопарк из систем. Мне нравится единообразие. Так удобнее управлять инфраструктурой, поэтому все собираю сам. Вот пример настройки шлюза на centos 7. Статья давно написана, на полностью актуальна. Ничего принципиально не поменялось.

При необходимости на шлюзе настраивается либо openvpn сервер, либо vpn клиент, если сервер уже есть. Если используется отдельный шлюз, я в него прокидываю реальный ip адрес, убираю его с самого гипервизора и делаю шлюзом по-умолчанию для гипервизора саму виртуалку со шлюзом. Главное не забыть ее поставить в автозагрузку, иначе после ребута гипервизора потеряете к нему доступ.

Если используются несколько дедиков в проекте, не объединенных в локальную сеть в рамках ЦОД, я их объединяю с помощью openvpn через интернет.

Backup сайтов

Для бэкапа я предпочитаю арендовать виртуальные машины в ruvds. К ним можно подключить большие диски — от 500 Гб и больше.

Подробно о том, как правильно делать бэкапы, я рассуждал отдельно. Конкретная реализация зависит от типа и частоты бэкапов, а так же глубины хранения. В общем случае достаточно будет обычных дампов баз данных и копии файлов сайта, скопированных с помощью rsync.

Если собираетесь делать бэкапы виртуальных машин, то настраивайте nfs сервер, подключайте его по vpn к гипервизору и настраивайте как отдельное хранилище для бэкапов. Штатным средством proxmox делайте регулярные бэкапы. В качестве nfs сервера отлично подходят виртуалки ruvds, так как там полноценная операционная система.

На этой же виртуалке ruvds я настраиваю простенький внешний мониторинг Zabbix для удаленным наблюдением за сайтами и физическими серверами по внешним ip адресам.

Так же я настраиваю дублирование бэкапов на s3 подобные хранилища с помощью **rclone**. Например, в selectel. Похожие хранилища есть у mail.ru и yandex.cloud. Облачные сервисы mail.ru я в настоящее время тестирую. Возможно будут статьи по этому поводу.

Я обычно делаю 3 контейнера в s3 с именами day, week, month и настраиваю правила хранения в них:

- day — 14 дней
- week — 31 день
- month — бессрочно

Следить за ротацией не надо, хранилище само удаляет старые архивы. У меня пока открыт вопрос по мониторингу бэкапов в s3 и их автоматическому разворачиванию для проверки. Пока не прорабатывал этот вопрос.

Управление хостингом

Самый главный вопрос при создании своего хостинга — как им управлять. Именно этот вопрос решают все готовые панели, но при этом добавляют массу новых проблем. В общем случае я не рекомендую использовать панели управления хостингом. Сам их терпеть не могу, так как намучался с ними. Там вечно какие-то проблемы, если тебе надо хоть немного кастомизировать настройки. А уж как они умирают при обновлении. Сколько я этих проблем повидал.

Сейчас просто не работаю с панелями вообще. Если кто-то предлагает взять на поддержку сервер с панелью управления хостингом (ispmanager, plex, vestacp), сразу отказываюсь. Я не могу гарантировать стабильную работу сайтов на них.

Раньше я писал bash скрипты для быстрого добавления сайта или настройки системы. Когда сайтов не много, создавал их руками. Сейчас я все стараюсь делать с помощью **ansible**. В паблик пока не готов выдать свои наработки. Их оформить отдельный труд. Нужно для начала хотя бы обзорную статью про ansible написать. Черновик уже год как написан, но не хватает времени оформить в полноценную статью.

Перечислю основные моменты, которые нужно автоматизировать:

- При добавлении виртуальной машины базовые настройки системы, подключение к мониторингу.
- При создании сайта — создание конфига nginx на фронте и бэке, запрос на сертификат, создание директорий и системного пользователя для сайта, создание php-fpm пула, настройка ротации логов, создание базы данных и пользователя, доступ пользователя по sftp.
- Поправить скрипт создания бэкапов. Создаю их bash скриптом. Есть мысли тоже через ansible делать, но пока не прорабатывал тему.
- Добавление мониторинга сайта в Zabbix. Пока не реализовал. Надо делать через api заббикса. Руки не дошли. С автоматизацией веб проверок в заббиксе пока все плохо.

Это основное. Может что-то забыл. Первое время все делал топорно единой yaml лапшой файла плейбука. Последнее время все переделываю с использованием шаблонов, ролей, задач, хендлеров, инвентаря и т.д. Стараюсь использовать ansible правильно.

В чем большое преимущество подобной автоматизации, помимо того, что можно быстро создать сущности на большом количестве объектов, когда у тебя этих объектов мало? В том, что настроив и проверив один раз какую-то процедуру, ее можно смело запускать без боязни где-то ошибиться. Она всегда отработает так, как ее настроили.

Например, при настройке firewall я всегда опасаюсь применять изменения в правилах. Вдруг где-то ошибся и что-то пойдет не так. Отрубится какой-то сервис или еще хуже — потеряю доступ к серверу. А когда у тебя есть шаблон и готовый плейбук по добавлению правил или заливки готового набора правил из шаблона, тебе достаточно проверить только переменные, чтобы в них не было ошибок. Далее прогнать на всякий случай плейбук в режиме

check, а потом применить, если все в порядке.

В такой последовательности меньше шансов получить ошибку. Это облегчает управление даже тогда, когда автоматизация не дает существенной прибавки к производительности из-за малого масштаба информационной системы. Зачастую, настройка и отладка плейбуков занимает больше времени, чем ручная работа. Но это дает задел на будущий рост и опыт. Даже если роста не будет в этом проекте, наработки пригодятся в другом.

Плейбуки храню в git. Приучил себя вообще все хранить в git. Первое время было не привычно и казалось, что не удобно. Но в итоге привык и оценил удобство репозиториев.

Базовые рекомендации по настройке своего хостинга

Потихоньку заканчиваю статью по настройке своего хостинга. Все основное я уже описал. Перечислю отдельно некоторые важные моменты.

1. Закрывайте с помощью firewall все не публичные доступы — ssh, панель управления proxmox, phpmyadmin и т.д. Я долгое время не делал этого, так как боялся не получить доступ в экстренной ситуации, когда я не смогу зайти через доверенный ip адрес. На практике это не было никогда проблемой. Обычно разрешаю к подключению свой статический ip адрес и 2 vpn сервера тоже со статичными внешними адресами.
2. Настраивайте мониторинг всех критичных вещей — состояние рейда, актуальность бэкапов, подключения по ssh к серверам и т.д. Не делайте много уведомлений — только реально важные вещи. Если будет спам от мониторинга, толку от него становится мало. Делайте повторяющиеся уведомления для некритичных событий, которые часто откладываешь и забываешь исправить.
3. Если занимаетесь работами по ускорению работы сайтов, можно в отдельной виртуальной машине настроить локальную версию сервиса webpagetest для объективной оценки скорости сайта без внешних сетевых задержек.
4. Когда обновляете систему на виртуальной машине, сделайте ее snapshot. После установки, если все в порядке, не забудьте его удалить. Это простое правило может очень сильно выручить в случае нештатной ситуации. Кстати, обновления я всегда делаю вручную, не через ansible или какие-то еще средства автоматизации. Если нет полного дублирования функционала с автоматическим переключением на работающий сервис, предпочитаю вручную контролировать установку обновлений. К сожалению, не такая уж и редкость, когда обновление что-то ломает. Пример — elk stack или bitrixenv. Если обновление прошло без ошибок — это удача. Обычно что-то ломается.

Заключение

> Не понравилась статья и хочешь научить меня администрировать? Пожалуйста, я люблю учиться. Комментарии в твоем распоряжении. Расскажи, как сделать правильно!

На этом у меня все. Постарался собрать и описать самый что ни на есть реальный опыт по настройке веб серверов в составе своего приватного хостинга. Буду рад советам, комментариям и замечаниям по существу.

Если у вас есть желание получить себе подобный сервер или набор серверов, пишите примерное тех задание на почту. Адрес есть в разделе Контакты.

Помогла статья? Есть возможность отблагодарить автора