Утилита rsync

rsync (англ. Remote Synchronization) - программа для UNIX- подобных систем, которая выполняет синхронизацию файлов и каталогов в двух местах с минимизированием трафика, используя кодировку данных при необходимости. Важным отличием rsync от многих других программ/протоколов является то, что зеркалирование осуществляется одним потоком в каждом направлении (а не по одному или несколько потоков на каждый файл). rsync может копировать или отображать содержимое каталога и копировать файлы, опционально используя сжатие и рекурсию.

Демон rsyncd, реализующий протокол rsync, по умолчанию использует TCP порт 873.

Применение. rsync была создана как замена для rcp и scp. Одним из первых применений rsync стало зеркалирование или резервное копирование клиентских Unix- систем на центральный Unix- сервер с использованием rsync/ssh и обычной учётной записи Unix. С планировщиком задач, таким как cron, возможно организовать автоматизированное основанное на rsync зеркалирование по защищённому криптографически каналу между многими компьютерами и центральным сервером.

> **Обратите внимание**
> rsync должен быть установлен на обоих узлах, локальном и удалённом.

Как работает (алгоритм) rsync

Утилита rsync использует алгоритм, разработанный австралийским программистом Эндрю Триджеллом, для эффективной передачи структур (например, файлов) по коммуникационным соединениям в том случае, когда принимающий компьютер уже имеет отличающуюся версию этой структуры. Принимающий компьютер разделяет свою

копию файла на неперекрывающиеся куски фиксированного размера S, и вычисляет контрольную сумму для каждого куска: MD4-хеш и более слабый rolling checksum, и отправляет их серверу, с которым синхронизируется. Сервер, с которым синхронизируются, вычисляет контрольные суммы для каждого кусочка размера S в своей версии файла, в том числе перекрывающиеся куски. Это может быть эффективно подсчитано ввиду особого свойства rolling checksum: если rolling checksum байт от n до n+S-1 равняется R, то rolling checksum байт от n+1 до n+S может быть посчитана исходя из R, байта n и байта n+S без необходимости учитывать байты, лежащие внутри этого интервала. Таким образом, если уже подсчитана rolling checksum байт 1-25, то для подсчета rolling checksum байт 2-26 используется предыдущая контрольная сумма и байты 1 и 26.

> ⚠ Rsync находит файлы, которые нужно отправить, используя "quick check" алгоритм (алгоритм используется по умолчанию), ищутся файлы, которые изменились в размере, или в дате последней модификации.

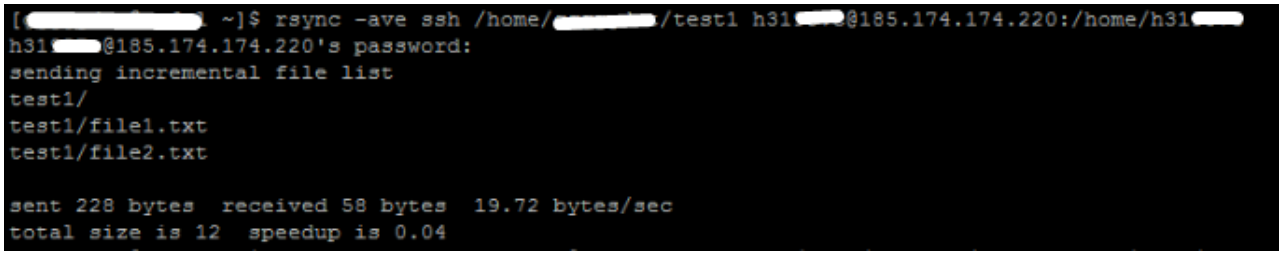
Синтаксис Rsync

Копирование данных с локального сервера на удаленный:

```
rsync ключи исходный_каталог пользователь@хост:/каталог_назначения
```

Где:

- исходный_каталог -файл или путь к каталогу, который нужно скопировать;
- пользователь - имя пользователя на удаленном сервере;
- хост - IP адрес или хост удаленного сервера;
- каталог_назначения - путь к месту, куда будет выполняться копирование.



Ключи запуска rsync

Вы можете отключить одну или несколько опций (-no-OPTION), подставив перед названием опции префикс "no-". Не ко всем опциям может быть применён этот префикс: только к опциям, которые вытекают из других опций (например -no-D, -no-perms) или имеют разные обязательства в различных обстоятельствах (например -no-whole-file, -no-blocking-io, -no-dirs). Вы можете указать длинные или короткие опции после префикса (например -no-R или -no-relative). Например: если вы хотите использовать опцию -a (-archive) но не хотите -o (-owner), вместо

превращения -a в -rlptgD, вы можете указать -a -no-o (или -a -no-owner). Порядок расположения опций важен: если вы укажете -no-r -a, опция -r всё равно будет включена, необходимо указывать -a -no-r. Обратите внимание, что побочный эффект опции -files-from НЕ позиционный, в то время как она влияет на состояние по умолчанию в нескольких опциях и слегка меняет смысл опции -a (смотрите опцию -files-from для получения более подробной информации)

- **-n, -dry-run** Отладочный режим. В этом случае, rsync не будет менять или удалять файлы, но покажет весь ход работы.
- **-q, -quiet** Опция уменьшает количество выводимой информации во время трансфера, значительно подавляется количество сообщений от сервера. Опция полезна, когда rsync запускается по крону.
- **-v, -verbose** Опция увеличивает количество выводимой информации во время трансфера. По умолчанию rsync ничего не выводит. Одна -v опция даст вам информацию о том, какие файлы переносятся и короткое заключение в конце. Две опции -v дадут информацию о том, какие файлы переносятся, какие не требуют обновления и немного больше информации в конце. Более чем две опции -v используются при отладке rsync. Формат вывода файлов по умолчанию задан, как -out-format "%n%L", он показывает только имена файлов и если объект является ссылкой, то на что он ссылается. На первом уровне отладки (одна -v) не показывается смена атрибутов файла. Если вы попросите детализированный список изменившихся атрибутов(указать опцию -itemize-changes, либо добавить "%i" к -out-format), то вывод (в клиенте) увеличится до упоминания всех пунктов, которые изменились. Смотрите опцию -out-format для получения более подробной информации.
- **-progress** показать % выполнения во время передачи
- **-stats** Указывает выводить подробную статистику по передаче файлов, позволяя Вам оценить, насколько эффективен алгоритм rsync относительно Ваших данных.
- **-x, -one-file-system** Требует не переходить границ файловой системы при рекурсивном копировании. Это полезно при необходимости копирования только одной файловой системы.
- **-a, -archive** задает архивный режим работы утилиты. Равносильно -rlptgoD, где

```
-r, --recursive — рекурсивный режим;
-l, --links — пересоздание symlinks, это значит, что символическ
ие ссылки будут так же переноситься;
-p, --perms — перенос прав;
-t, --times — передача времени модификации и его обновление на у
даленной системе. Этот ключ должен быть установлен для точной с
инхронизации;
-g, --group — установить группу конечного файла таким же, как и
у исходного;
-o, --owner — установить владельца конечного файла таким же, как
и у исходного;
-D, - same as --devices --specials — установить тип файла устройст
ва и файла специального типа
таким же, как у исходного.
```

Это быстрый способ сказать, что вы хотите рекурсивную обработку и сохранить практически всё (добавьте опцию -H если нужно сохранять жесткие ссылки). Единственным исключением из вышеупомянутой равносильности, это когда указан -files-from , в случае которого опция -r не работает. Обратите внимание, что опция -a не сохраняет жёсткие ссылки, потому что поиск множественно-символьных файлов, требует много затрат. Необходимо отдельно использовать опцию -H.

- **-H, -hard-links** сохранять жесткие ссылки. Жесткая ссылка позволяет иметь один и тот же файл в нескольких местах (ссылка не является отдельной копией файла, это тот же файл, который отображается в двух разных местах).
- **-r, -recursive** рекурсивно входить в подкаталоги
- **-l, -links** копировать символьные ссылки как символьные ссылки
- **-p, -perms** сохранять разрешения
- **-z, -compress** с этим параметром rsync сжимает все передаваемые данные файлов. Это полезно на медленных линиях. Метод сжатия, используемый при этом, тот же, что реализует gzip. Заметьте, что при этом обычно достигается лучший коэффициент сжатия, чем может быть достигнут, используя сжатие программ удаленной оболочки или сжатие транспортного уровня, т.к. в процесс сжатия вовлекается вся информация, посылаемая в соответствующих блоках данных.

```
     --compress-level=NUM    explicitly set compression level
     --skip-compress=LIST    skip compressing files with suffix
in LIST
```

- **-b, -backup** делать бэкапы

```
     --backup-dir=DIR        делать бэкапы в указанную директор
ию
     --suffix=SUFFIX         суффикс бэкапов (по умолчанию ~ )
```

- **-numeric-ids** Вместо имен групп и пользователей посылаются их числовые id и ставятся в соответствие друг другу на обоих концах. По умолчанию rsync использует имена групп и пользователей для определения владельца файлов. Специальные uid 0 и gid 0 никогда не отображаются через имена пользователей/групп, даже если не указана -numeric-ids. Если исходная система работает в ограниченном chroot-окружении или если пользователь или группа не существуют на приемной стороне, то используются исходные числовые id.
- **-c, -checksum** Меняет способ проверки на изменившиеся файлы. Без этой опции, rsync использует "quick chech" алгоритм (установлен по умолчанию), который проверяет различие в размере и времени модификации файлов. Эта опция меняет алгоритм на сравнивание по 128-разрядным контрольным суммам MD4 для каждого файла, который соответствует размеру. Составление контрольных сумм означает, что обе стороны будут тратить много дискового ввода/вывода читая все данные в передаваемых файлах(и это перед любым чтением, которое будет для передачи изменившихся файлов), так что это может значительно замедлить работу. Посылающая сторона генерирует контрольные суммы, в то время как она делает сканирование файловой системы, что составляет список доступных файлов. Получатель генерирует контрольные суммы, когда он просматривает на изменившиеся файлы, и проверяет контрольные суммы любого файла, который имеет такой же размер, как и у соответствующего посылаемого файла: файлы с изменившемся размером или изменившейся контрольной суммой выбираются для передачи. Обратите внимание, rsync всегда проверяет, что каждый переданный файл был правильно восстановлен на принимающей стороне, делает он это проверкой всей контрольной суммы файла, что была сгенерирована во время передачи файла.
- **-e**, -rsh=COMMAND -можно указать любую удалённую оболочку (ssh, rsh, remsh), либо задать переменную окружения RSYNC_RSH.
- -delete удалять файлы, которых нет на передающей стороне
- -delete-before получатель удаляет перед передачей (по умолчанию)

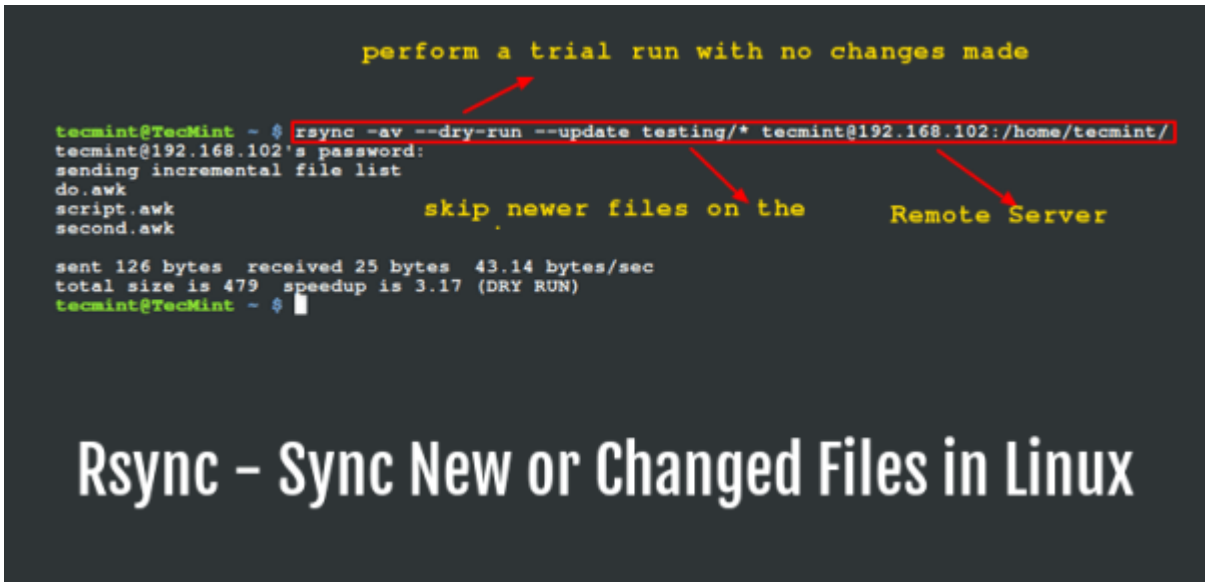
> -delete отличается от -delete-after тем, что удаление производится вначале, а не на завершающей стадии процесса бэкапа. -delete-after работает быстрее, так как не требует лишней стадии обхода списка файлов, но требует использования опции -force для обработки таких ситуаций как удаление файла и появление директории с тем же именем;

- -delete-after означает, что удалить файлы нужно, только после окончания синхронизации.
- -force принудительно удалять даже не пустые каталоги
- -partial По умолчанию rsync будет удалять все частично-переданные файлы в случае прерывания передачи. В некоторых случаях более желательно сохранять такие файлы. Используя параметр -partial, можно указать rsync сохранять

частично-переданный файл, что может ускорить передачу всего файла при последовательном повторении таких передач.

- **-t, -times** Указывает передавать время модификации файлов и обновлять им соответствующий атрибут на приемной стороне. Обратите внимание, что если этот параметр не установлен, то становится неэффективной оптимизация передачи по исключению не изменявшихся по времени файлов; другими словами, пропуск -t или -a будет означать, что следующая передача будет осуществляться с параметром -I, для всех файлов будут сравниваться контрольные суммы, а в журнале появятся соответствующие сообщения о них, даже если они не изменялись.
- **-g, -group** сохранять группу
- -P – данная опция включает в себя опцию -partial и -progress.

Шаблоны исключения



Шаблоны исключения и включения, будучи заданными, позволяют гибко выделять, какие файлы должны передаваться, а какие должны быть пропущены.

rsync строит упорядоченный список на основе параметров –include/–exclude, указанных в командной строке. Rsync проверяет каждое имя файла или каталога относительно соответствия каждому шаблону включения/исключения. В действие вступает первое же совпадение. Если совпавший шаблон исключающий, то соответствующий файл пропускается. Если шаблон включающий, то не пропускается. При отсутствии подходящих шаблонов к имени файла он также не пропускается.

Имена файлов проверяемых по таким шаблонам задаются относительно каталога назначения, "верхнего каталога", так что шаблоны необязательно должны включать элементы исходного или конечного каталогов. Единственный случай, когда шаблоны будут проверяться относительно абсолютных путей к файлам или каталогам, - это случай, когда исходным путем является корневой каталог файловой системы.

Заметьте, что при использовании параметра -r (он подразумевается при -a) каждая отдельная часть каждого пути просматривается сверху вглубь, так что шаблоны включения/исключения применяются рекурсивно к каждой такой части.

Также обратите внимание, что параметры –include и –exclude допускают только по одному шаблону каждый. Для добавления множества шаблонов используйте параметры –include-from и –exclude-from или множество параметров –include и –exclude соответственно.

Шаблоны могут быть заданы в нескольких формах. Правила для них таковы:

- Если шаблон начинается с /, то он проверяется на соответствие с началом имени файла, в противном случае - относительно завершающей части имени. Это эквивалент начального ^ в регулярных выражениях. Таким образом, "/foo" должен соответствовать файлу с именем "foo" в вершине передаваемого дерева. С другой стороны, "foo" должен соответствовать любому файлу "foo" везде в пределах дерева каталогов, потому что алгоритм применяется рекурсивно сверху вглубь. Он работает так, как если бы каждая составная часть пути была завершением имени файла. Начальный / не превращает шаблон в абсолютный путь.
- Если шаблон заканчивается на /, то он соответствует только каталогу, не файлу, не ссылке и не устройству.
- Если шаблон содержит символы подстановки из набора *?[, то при проверке соответствия используются правила подстановки командной оболочки для имен файлов. В противном случае используется просто соответствие строке.
- В соответствия двойной звёздочки входит слэш, в то время как для одиночной звездочки * на слэше совпадение заканчивается.
- Если шаблон содержит слэш (не считая завершающего слэша) или "**" , то такой шаблон проверяется на соответствие полному имени файла, включая любые вышестоящие каталоги. Если шаблон не содержит / или "**", то такой шаблон проверяется относительно завершающей части имени файла. Еще раз, помните, что алгоритм применяется рекурсивно, так что "полным именем файла" может оказаться в действительности любая часть пути в глубине иерархии от начального каталога.
- Если шаблон начинается на "+ " (плюс с последующим пробелом), то это всегда рассматривается как включающий шаблон, даже если указан как часть параметра исключения. Сама часть "+ " не учитывается при проверке на соответствие.
- Если шаблон начинается на "- " (минус с последующим пробелом), то это всегда рассматривается как исключающий шаблон, даже если он фигурирует как часть параметра включения. Сама часть "- " не учитывается при проверке на соответствие.
- Если шаблоном указан одиночный восклицательный знак ! , то текущий список включения/исключения сбрасывается удалением всех ранее определённых шаблонов.

Правила +/- наиболее полезны в списках, читаемых из файла, позволяя Вам иметь один общий список, который содержит как исключающие, так и включающие шаблоны.

Если Вы завершите исключающий список указанием –exclude '*', то обратите внимание, что из-за рекурсивного характера алгоритм остановится на родительских каталогах и не будет пробовать файлы внутри них за исключением тех случаев, когда Вы явно укажете включать родительские каталоги файлов, которые Вы хотите включить. Для включения всех каталогов используйте –include '*/' до –exclude '*'.

Несколько примеров включения/исключения:

```
    --exclude "*.o" исключить все имена файлов, соответствующие *.o
    --exclude "/foo" исключить файл с именем foo в верхнем каталоге
    --exclude "foo/" исключить любой каталог с именем foo
    --exclude "/foo/*/bar" исключить любой файл с именем bar в двух ур
каталога "foo" в вершине дерева
    --exclude "/foo/**/bar" исключить любой файл с именем bar в двух и
по иерархии от верхнего каталога "foo"
    --include "*/" --include "*.c" --exclude "*" включить только катал
    --include "foo/" --include "foo/bar.c" --exclude "*" включит толь
```

Переменные окружения

- CVSIGNORE Переменная окружения CVSIGNORE дополняет исключающие шаблоны из файла .cvsignore. За деталями см. параметр –cvs-exclude.
- RSYNC_RSH Переменная окружения RSYNC_RSH позволяет переопределить программу удаленной оболочки, используемую rsync'ом как транспорт. Параметры командной строки для оболочки указываются после имени программы, также как и для параметра -e.
- RSYNC_PROXY Переменная окружения RSYNC_PROXY позволяет указать rsync-клиенту использовать web-прокси для подключения к rsync-серверу. Вы должны указать прокси в виде пары hostname:port.
- RSYNC_PASSWORD Установка значения RSYNC_PASSWORD позволяет осуществить rsync-соединение к rsync-серверу без вмешательства пользователя для ввода пароля. Заметьте, что это не тоже самое, что и пароль для транспортной оболочки, например, ssh.
- USER или LOGNAME Переменные окружения USER или LOGNAME используются для определения пользователя по умолчанию, имя которого предоставляется rsync-серверу для аутентификации.
- HOME Переменная HOME используется для поиска пользовательского файла .cvsignore.

Примеры использование rsync

Если на конце исходной директории стоит «/», то это означает копирование содержимого директории; отсутствие слеша означает копирование директории и ее содержимого.

Скрипт бэкап при помощи rsync

Простой скрипт наглядно демонстрирующий возможности утилиты rsync синхронизировать локальные директории на удаленный сервер при помощи протокола SSH.

Алгоритм скрипта: в текстовом файле backup.lst в каждой новой строчке указываете путь к директории которую нужно скопировать/синхронизировать. В конце слеш не ставим, пустая строки удаляем. Например:

```
cat backup.lst
/home/darkfire/pictures
/home/darkfire/documents
```

В скрипте считывается файл backup.lst и его содержимое передается в цикл while оболочки bash. Для каждой строчки файла backup.lst запускается выполнение rsync для копирования/синхронизации директории на удаленный сервер, при помощи протокола SSH. Используемые ключи: e для указания использования SSH; a включение архивного режима; c включает проверки передаваемых файлов MD4; q уменьшает выводимую информацию (удобно если скрипт будет запускаться при помощи cron); delete удалять файлы, которых нет на передающей стороне (синхронизация); force принудительно удалять не пустые каталоги.

```
#!/bin/bash

# полный путь к вашему списку каталогов
BACKUP_LST=/home/darkfire/scripts/backup.lst

cat ${BACKUP_LST} | while read Res; do
rsync -e ssh -acq --delete --force $Res root@xxx.xxx.xxx.xxx:/root/bakd
ir2
done
```

Скрипт можно сделать более удобным не используя напрямую SSH, а подключив удалённую директорию на локальный сервер, при помощи технологии fuse: sftp (sshfs), curlftpfs.

Синхронизация локальных директории

Рекурсивно синхронизируется содержимое папки dir_a с директорией dir_b. Файлы передаются в "archive" режиме, который обеспечивают, что при трансфере будут сохранены символьные ссылки, файлы устройств, атрибуты, права, полномочия доступа и т.д. Использовано сжатие, для уменьшения размера передаваемых данных. В директории dir_b удаляются файлы отсутствующие в источнике (dir_a).

```
rsync -avz --delete /src/dir_a/ /data/dir_b
```

Синхронизация на удаленный сервер с использованием нестандартного порта SSH:

```
rsync --rsh='ssh -p1004' -acq --delete --force /src/dir_a/ backup@backu
p.example.com:dir_b
```

Перенос локальной директории

Для переноса локальной директории в другое место можно использовать rsync, например так

```
rsync -aHx --numeric-ids --progress --stats /var/www/ /tmp/www/
```

Где a архивный режим и рекурсия; H сохранять жесткие ссылки; x запрет на переходит границ файловой системы при рекурсивном копировании; numeric-ids вместо имен групп и пользователей посылаются их числовые id; progress показать % выполнения: stats выводить подробную статистику по передаче файлов.

Как запустить Rsync в несколько потоков

Как известно, Rsync работает в 1 поток. Часто это является узким местом в скорости передачи файлов. Ниже предоставляю решения для запуска Rsync в несколько потоков.

Если нужно перенести много директорий, то лучше запускать rsync в несколько потоков. В примере ниже вы должны находиться в директории /var/www/. Ключ P указывает количество потоков, в данном случае 5 потоков.

```
ls  | xargs -I '{}' -P 5 -n1 rsync -aHx --numeric-ids --progress --stats /
var/www/'{}'/ 10.10.10.10:/var/www/'{}'/
```

Второй вариант многопоточности rsync. У нас имеются директория с вложениями как фалов, так и еще поддиректориями со своими файлами.

```
find . -type d | xargs -I '{}' -P 5 -n1 rsync -a --delete /home/old/Foto/'{}'/ /home/old/CloudDisk/'{}'/
```

Разберем данный пример:

- find . -type d выводит имена всех директорий в текущем каталоге.

find выводит названия директории с точкой (например ./snap). Соответственно в rsync вы увидите путь такого вида data/www/./example.com/wp-content Не нужно волноваться, /./ несет смысл текущего каталога, но по сути оно эквивалентно просто /, так как знак точки означает "остаться в текущем каталоге".

- xargs -I '{}' xargs запускает для каждого файла команду rsync. Ключ -P 5 утилиты xargs задает количество одновременно запущенных команд (потоков rsync), то есть одновременно будут запущена передача 5-ти файлов. Каждый файл передается отдельно.

Ещё пример:

```
ls -p | grep -v / | xargs -I '{}' -P 5 -n1 rsync -av --exclude="lost+found" --no-links  /home/myuser/files/'{}'
```

Заключение

Надеюсь эта практическая информация будет полезна многим. Примеры в статье взяты из повседневного опыта использования rsync для бэкапа/синхронизации/копирования данных между VPS.

10 Самых Популярных Статей

1. 7 способов сравнения файлов по содержимому в Windows или Linux
2. Что такое страны tier 1,2,3 и как правильно выбрать ГЕО для рекламной кампании
3. Настройка, использование GitLab CI/CD
4. Китайский VPN Shadowsocks простая установка и настройка

5. Настройка и использование сервера OpenVPN в Linux
6. PostgreSQL: создать БД, пользователя, таблицу, установить права
7. top, htop, atop определение загрузки ОС (Load average, LA)
8. Использование rsync в примерах
9. my.cnf примеры конфигурации MySQL, MariaDB
10. dig проверка DNS сервера

10 Самых Популярных Обзоров

1. ТОП 4 лучших антидетект браузеров в 2024 (Бесплатные & Платные)
2. Обзор и отзывы о Namecheap в 2024 году
3. ТОП 4 Лучших Игровых Хостингов в 2024
4. Обзор браузера Dolphin {anty}
5. ТОП 3 Проверенных VPN, Прокси, Хостинг VPS Турция в 2024
6. Что такое Оффшорный хостинг, абузоустойчивый хостинг, DMCA игнорируется | ТОП 4 провайдеров в 2024
7. Обзор и отзывы AstroProxy в 2024 году
8. Обзор и отзывы о PQ Hosting в 2024 году
9. Проверенные VPS / VDS хостинг провайдеры
10. Обзор и отзывы о AÉZA (Аеза) в 2024 году: преимущества и недостатки