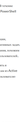
Шпаргалка по командам PowerShell



Что такое PowerShell?

PowerShell – это кроссплатформенная **Python-подобная командная строка** (CLI) на основе .net framework, которая позволяет автоматизировать выполнение задач, а также управлять конфигурациями компьютеров, серверов и приложений. Поддерживает работу с файлами, ключами реестра, сетевыми протоколами, WMI, COM, ADSI, ADO, REST API, JSON, CSV, XML, Excel, Word, Active Directory, Azure, Exchange, SQL Server, Oracle, Sharepoint, Office 365 и многими другими системами. Также поддерживает выполнение скриптов на удаленных компьютерах.

Стандартные PowerShell скрипты распространяются в сценариях исполняемых файлов, использующих или Cmdlet в C#/.Net, которые компилируются в исполняемый скриптовый файл PowerShell. Поставщики PowerShell позволяют получить доступ к данным, которые в противном случае не являются доступными из оболочки, такие как переменные, псевдонимы, функции, сценарии, сертификаты и реестр. Поставщики могут также предоставлять доступ к данным, которые в противном случае доступны из других приложений, таких как Active Directory. Поставщики могут также предоставлять доступ к данным, которые в противном случае доступны из оболочек, таких как Windows Management Instrumentation (WMI) и Component Object Model (COM).

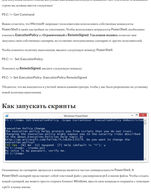
Команды PowerShell

Все 23 основных команды PowerShell

Рус.команда	Англ.Cmdlet
Get-Location	get-location
Get-Content	get-content
Добавить контент	add-content
Set-Content	set-content
Copy-Item	copy-item, cp
Удалить объект	remove-item
Move-Item	move-item
Get-Item	get-item
New-Item	new-item
Test-Path	test-path
Set-Item	set-item
Get-ChildItem	get-childitem, ls
Show-Command	show-command
Clear-Host	clear-host, cls
Запуск процесса	start-process
Приостановить Job	stop-job
Wait-Job	wait-job
Receive-Job	receive-job
Get-Job	get-job
Write-Output	write-output
Write-Host	write-host
Sort-Object	sort-object
Select-Object	select-object
Group-Object	group-object
Measure-Object	measure-object
Compare-Object	compare-object
Group-Object	group-object
Measure-Object	measure-object
Compare-Object	compare-object
Select-Object	select-object
Sort-Object	sort-object
Write-Host	write-host
Write-Output	write-output
Clear-Host	clear-host, cls
Show-Command	show-command
Get-ChildItem	get-childitem, ls
Get-Item	get-item
Set-Item	set-item
Test-Path	test-path
New-Item	new-item
Move-Item	move-item
Remove-Item	remove-item
Copy-Item	copy-item, cp
Set-Content	set-content
Get-Content	get-content
Set-Location	set-location

Если для русской версии Windows PowerShell не установлен язык, тогда вместо **Рус.команда** будет **Англ.Cmdlet**.

Шпаргалка PowerShell (PDF)



Как использовать PowerShell

PowerShell позволяет выполнять задачи и автоматизировать большинство операций, которые вы можете сделать с помощью графического и командного интерфейсов сети. Вместо того, чтобы собирать информацию о сетевых компьютерах, отправлять им сообщения, перезагружать их, устанавливать на них приложения, вы можете сделать все это с помощью PowerShell, не покидая своего рабочего места.

Существует несколько вариантов запуска PowerShell, но наиболее простой способ для начинающих – это запустить PowerShell из меню «Пуск». Вы можете также использовать ярлык PowerShell, который, скорее всего, уже есть на вашем рабочем столе. Если вы используете Windows 7 или более ранние версии Windows, вы можете запустить PowerShell из меню «Все программы».

Если вы хотите использовать расширенные возможности PowerShell, вы можете запустить его с правами администратора. Для этого щелкните правой кнопкой мыши на ярлыке PowerShell и выберите «Запуск от имени администратора». Вы можете использовать PowerShell для управления компьютерами, которые могут быть расположены в вашем домене, в рабочей группе или в ненадежной среде. Вы можете использовать PowerShell для запуска отдельных команд или для запуска сценариев, которые могут быть написаны вами или другими людьми. Вы можете использовать PowerShell для запуска как 32-, так и 64-битных версий исполняемых файлов, в зависимости от вашей среды и ваших потребностей.

Сценарии PowerShell могут быть запущены с ярлыка, аналогичного ярлыку PowerShell, с использованием параметров, которые могут быть переданы в сценарий при его выполнении.

PowerShell против командной строки

Если вы знакомы с командной строкой, вы можете задать себе вопрос, в чем разница между PowerShell и командной строкой. Ответ прост: PowerShell – это гораздо более мощный инструмент, чем командная строка. Хотя PowerShell может выполнять все те же задачи, что и командная строка, PowerShell предоставляет вам гораздо более мощный и гибкий язык сценариев, который позволяет вам автоматизировать выполнение задач, которые в противном случае потребовали бы ручного вмешательства.

Загрузка PowerShell

Microsoft регулярно выпускает обновления для PowerShell, поэтому вам необходимо загрузить последнюю версию, которая, скорее всего, будет иметь больше функций, чем версия, которая поставляется с вашим компьютером. Microsoft также выпустила версии PowerShell для Windows XP, Vista и Server 2003, которые позволяют использовать PowerShell для управления этими системами. Microsoft также выпустила версию PowerShell для Linux и Mac OS X, которая позволяет использовать PowerShell для управления этими системами.

Как запустить командлеты

В вас есть ключ, значит – это лицензионная копия Windows. Вы можете использовать в командную строку, а также вы можете использовать PowerShell. Чтобы использовать PowerShell, необходимо запустить оболочку PowerShell. Вы можете запустить оболочку PowerShell, выполнив следующие шаги:

- **Get-Location** – Показать текущий каталог.
- **Set-Location** – Задать рабочий каталог.
- **Move-item** – Переместить объект или файл.
- **Remove-item** – Удалить файл или объект.
- **Rename** – задать Переименовать существующий файл.
- **Copy-item** – Скопировать объект.

Для получения полного списка доступных вам команд нужно выполнить команду Get-Command. В командной строке PowerShell вы можете ввести:

```
PS C:> Get-Command
```

Получите **Get-ExecutionPolicy**

Получить **Set-ExecutionPolicy** неограниченную политику

```
PS C:> Set-ExecutionPolicy RemoteSigned
```

Подсказка, что вы выполнили в новой оболочке PowerShell, чтобы у вас была разрешена установка сценариев PowerShell.

Как запустить скрипты

Основными исполнителями скриптов в новой оболочке являются такие утилиты-оболочки, как PowerShell. Скрипты PowerShell могут быть запущены из оболочки PowerShell, а также из командной строки. Если вы хотите запустить сценарий, который вы только что написали, вам нужно открыть оболочку Windows, или же вы можете использовать оболочку PowerShell и запустить сценарий оттуда.

Чтобы запустить скрипт, введите его имя в окне ввода файла и нажмите Enter:

```
PS C:> Get-Location
```

Как только вы запустите, выбранный вами скрипт будет запущен.

Запуск сценария или собственных скриптов PowerShell Запуск в оболочке PowerShell сценариев, которые вы пишете, или сценариев, которые вы получаете от кого-то еще, может быть несколько сложнее, чем запуск сценариев из оболочки Windows. Вместо запуска сценариев из оболочки Windows, вы можете использовать оболочку, которая позволяет выполнять сценарии, которые могут быть написаны вами, другими пользователями или Microsoft.

Чтобы запустить сценарий PowerShell, вам нужно сначала открыть оболочку PowerShell, а затем ввести имя сценария, который вы хотите запустить, в приглашении PowerShell. Вы можете также использовать оболочку PowerShell для запуска сценариев, которые могут быть написаны другими пользователями или Microsoft.

Резервное копирование базы данных SQL

Многие люди используют PowerShell для ускоренного копирования баз данных SQL. Резервное копирование баз данных SQL – это важная задача, которая должна выполняться регулярно, чтобы гарантировать, что ваши данные будут в безопасности в случае сбоя сервера или повреждения данных. Вы можете использовать PowerShell для автоматизации процесса резервного копирования баз данных SQL, что может сэкономить вам время и деньги.

```
PS C:> Backup-SqlDatabase -ServerInstance (Localhost) -Database (Database1)
PS C:> Backup-SqlDatabase -ServerInstance (Localhost) -Database (Database1)
-BackupFile c:\sql
```

Это создаст резервный файл выбранной базы данных.

Основные команды PowerShell

Вы можете использовать большое количество различных команд PowerShell, которые можно использовать для выполнения различных задач. Вот некоторые из наиболее распространенных команд, которые вы можете использовать:

Получить помощь

Если вы когда-нибудь хотели узнать больше о любом конкретном командлете, вам нужно ввести: get-help. Например, если вы хотите узнать больше о командлете Get-Process, вы можете ввести: get-help get-process. Вы можете также использовать get-help, чтобы получить общую информацию о PowerShell, а также о таких вещах, как синтаксис, или вы можете ввести, что вам нужно сделать, и PowerShell предложит вам несколько вариантов.

Например, если вы хотите посмотреть, для чего предназначен Get-Process, введите:

```
PS C:> Get-Help Get-Process -Detailed
PS C:> Get-Help Get-Process -Examples
```

Как для получения дополнительной информации, так и для получения примеров, вам нужно ввести:

```
PS C:> Get-Help Get-Process -Full
```

Если вы ищете что-то, что вы не можете вспомнить, например, какую операцию выполняет командлет, попробуйте:

- **Get-command** – Получить все доступные команды.
- **Get-member** – Показать свойства и методы для объекта.
- **Get-module** – Показать все доступные модули.
- **Get-psdrive** – Показать все доступные диски.

```
PS C:> Get-Command
```

Чтобы получить полный выбранной вами команды:

```
PS C:> Get-Command -Verb Get
```

Для получения дополнительной информации о PowerShell, вы можете использовать:

```
PS C:> Get-Help about_*
```

Как для получения дополнительной информации, так и для получения примеров, вам нужно ввести:

```
PS C:> Get-Help about_* -Detailed
PS C:> Get-Help about_* -Examples
```

Как для получения дополнительной информации, так и для получения примеров, вам нужно ввести:

```
PS C:> Get-Help about_* -Full
```

- **Get-Help about_Variables** – Отображение информации о переменных.
- **Get-Help about_Variable_Provider** – Отображение информации о переменных.
- **Get-Help about_Variable_Cmdlets** – Отображение информации о переменных.
- **Get-Help about_Variable_Syntax** – Отображение информации о переменных.

```
PS C:> Get-Help about_Variables
```

Get-Service

Одной из наиболее важных задач, которые вы можете выполнить, должна проследить за выполнением ваших служб. Вы можете использовать PowerShell для отслеживания ваших служб. Вы можете использовать Get-Service для отслеживания ваших служб. Вы можете также использовать Get-ExecutionPolicy команду для отслеживания ваших служб.

Например, если вы набрали PS C:> Get-Service, вы получите полный список всех служб на вашем компьютере.

Вы можете также использовать Format-List Командлет, который отобразит все свойства каждой службы: PS C:> Get-Service | Format-List *. Чтобы получить все свойства одной службы, вы можете использовать свойство name, например: PS C:> Get-Service -name Spooler | Format-List *.

Вы можете также использовать свойство status:

```
PS C:> Get-Service | Where-Object {$_.status -eq "Running"}
```

Чтобы получить службы, которые начинаются с «Net»:

```
PS C:> Get-Service -name Net*
```

Чтобы получить службы, которые начинаются с «Net» и находятся в состоянии «Остановлено»:

```
PS C:> Get-Service -name Net* -exclude Net* | Where-Object {$_.status -eq "Stopped"}
```

Чтобы получить службы, которые начинаются с «Net» и находятся в состоянии «Остановлено», а также отсортированы по имени:

```
PS C:> Get-Service -name Net* -exclude Net* | Where-Object {$_.status -eq "Stopped"}
| Sort-Object -property name
```

Чтобы получить службы, которые начинаются с «Net» и находятся в состоянии «Остановлено», а также отсортированы по имени, а также отображаются только имена и статусы служб:

```
PS C:> Get-Service -name Net* -exclude Net* | Where-Object {$_.status -eq "Stopped"}
| Sort-Object -property name | Format-Table name,status
```

Экспорт CSV в Get-Cpuset

Вы можете использовать Get-Cpuset для экспорта данных в формат CSV. Чтобы экспортировать данные, вам нужно ввести PowerShell-cmdlet Export-Csv, за которым следует файл CSV, экспортируемый из объекта. Например, если вы хотите экспортировать все свойства всех процессов, вы можете ввести следующую команду в объект процессов и экспортировать данные в файл CSV:

```
PS C:> Get-Cpuset -Process | Export-Csv -Path c:\Process.csv
```

Чтобы использовать эту команду, вы должны набрать:

```
PS C:> Get-Cpuset -Process | Export-Csv -Path c:\Process.csv
```

Вместо вывода, что вы видите на экране, вы можете использовать Export-CSV. Что касается вывода, вы можете использовать Export-Csv для отправки вывода в файл. Вы можете также использовать Export-Csv для отправки вывода в файл, который вы можете использовать в Excel. Вы можете также использовать Export-Csv для отправки вывода в файл, который вы можете использовать в Excel, или вы можете

PowerShell: мощный интерфейс командной строки

Хотя переход на PowerShell может показаться довольно сложным, его интерфейс командной строки работает практически так же, как и любой другой. У него могут быть свои уникальные командлеты, но множество онлайн-ресурсов может помочь вам в решении любых административных задач. Чтобы максимально использовать возможности PowerShell, вам просто нужно привыкнуть к множеству доступных вам команд..

Будучи новым пользователем, легко напасть на 200 с лишним командлетов PowerShell. Убедитесь, что вы начинаете с интерфейса командной строки, прежде чем перейти к полноценному графическому интерфейсу. Независимо от того, являетесь ли вы новичком в PowerShell или интерфейсах командной строки, в Интернете доступно более чем достаточно информации, чтобы помочь вам максимально эффективно использовать этот мощный инструмент.