



# PowerShell Commands

Большая база заметок **PowerShell** на русском языке.

## Навигация:

- [Cheat-Sheet](#)
- [Help](#)
- [Object
  - \[Variable\]\(#\)
  - \[ENV\]\(#\)
  - \[History\]\(#\)
  - \[Clipboard\]\(#\)
  - \[Write-Host\]\(#\)
  - \[Write-Progress\]\(#\)
  - \[for\]\(#\)
  - \[Array\]\(#\)
  - \[Index\]\(#\)
  - \[HashTable\]\(#\)
  - \[Collections/List\]\(#\)
  - \[PSCustomObject\]\(#\)
  - \[Add and Remove Property\]\(#\)
  - \[Add Method\]\(#\)
  - \[Class\]\(#\)
  - \[Pipeline\]\(#\)
  - \[Select-Object\]\(#\)
  - \[Expression\]\(#\)
  - \[Select-String\]\(#\)
  - \[Format-List/Format-Table\]\(#\)
  - \[Measure-Object\]\(#\)
  - \[Compare-Object\]\(#\)
  - \[Where-Object \\(?\\)\]\(#\)
  - \[Sort-Object\]\(#\)
  - \[Last/First\]\(#\)
  - \[Group-Object\]\(#\)
  - \[Property\]\(#\)
  - \[Method\]\(#\)](#)
- [Error
  - \[ExecutionStatus\]\(#\)
  - \[Measure-Command\]\(#\)](#)
- [DateTime](#)
- [TimeSpan
  - \[Format\]\(#\)
  - \[Timer\]\(#\)](#)
- [Regex
  - \[Якори\]\(#\)
  - \[Группы захвата\]\(#\)
  - \[Matches\]\(#\)
  - \[Форматирование\]\(#\)
  - \[Условный оператор\]\(#\)
  - \[Логические операторы сравнения\]\(#\)
  - \[Pipeline Operators\]\(#\)](#)

- [Invocation Operator](#)
- [DataType](#)
  - [Math](#)
  - [Round](#)
  - [ToString](#)
  - [Char](#)
- [Function](#)
  - [Switch function](#)
  - [Switch param](#)
  - [psm1 \(module file and parameters\)](#)
  - [psd1 \(module description file\)](#)
- [Bit](#)
  - [Bit Convertor](#)
- [Cycle](#)
  - [Foreach](#)
  - [ForEach-Object \(%\)](#)
  - [While](#)
  - [Try-Catch-Finally](#)
- [Files](#)
  - [Clear-env-Temp-14-days](#)
  - [System.IO.File](#)
  - [Read/Write Bytes](#)
- [Archive](#)
  - [Microsoft.PowerShell.Archive](#)
  - [System.IO.Compression.FileSystem](#)
  - [WinRAR](#)
- [Handle](#)
- [Console-Menu](#)
- [Credential](#)
  - [Out-Gridview](#)
  - [Out-File](#)
  - [Get-Content \(gc/cat/type\)](#)
  - [AES Key](#)
- [WinEvent](#)
  - [Filter XPath/Hashtable](#)
  - [Reboot](#)
  - [Logon](#)
  - [EventLog](#)
- [Firewall](#)
  - [Firewall-Manager](#)
- [Defender](#)
- [DISM](#)
- [Scheduled](#)
- [Network](#)
  - [ping](#)
  - [dhcp](#)
  - [port](#)
  - [netstat](#)
  - [nslookup](#)
  - [ipconfig](#)
  - [Adapter](#)
  - [DNSClient](#)
  - [DNSCache](#)
  - [Binding](#)
  - [TCPSetting](#)
  - [hostname](#)
  - [arp](#)

- Network Adapter Statistics

- SpeedTest

- iPerf

- Install

- Env-Update-Exec-Path

- iPerf-GUI

- iPerf-Docker

- Server

- Client

- Output

- PS-iPerf

- RDP

- IPBan

- shutdown

- LocalAccounts

- SMB

- Get-Acl

- NTFSSecurity

- Storage

- iSCSI

- WSUS

- WindowsUpdate

- PSWindowsUpdate

- UpdateServices

- PoshWSUS

- ActiveDirectory

- RSAT

- Import-Module AD

- ADSI

- LDAP

- LAPS

- Recycle Bin

- thumbnailPhoto

- ADDomainController

- ADComputer

- ADUser

- ADGroupMember

- ADReplication

- repadmin

- dcdiag

- ntdsutil

- GPO

- ServerManager

- PSWA

- WSB (Windows Server Backup)

- RDS

- DNSServer

- DHCPServer

- DFS

- StorageReplica

- PS2EXE

- NSSM

- Jobs

- ThreadJob

- PoshRSJob

- Invoke-Parallel

- ForEach-Object-Parallel

- [SMTP](#)
  - [Net.Mail](#)
  - [SMTP over OpenSSL](#)
  - [Swaks](#)
- [VMWare/PowerCLI](#)
- [Hyper-V](#)
  - [VMConnect via RDCMan](#)
- [Azure](#)
  - [Manage-VM](#)
  - [Manage-Disk](#)
- [Exchange/EMShell](#)
  - [Roles](#)
  - [MessageTrackingLog](#)
  - [Mailbox](#)
  - [MoveRequest](#)
  - [Archive](#)
  - [Quota](#)
  - [Database](#)
  - [MailboxRepairRequest](#)
  - [eseutil](#)
  - [Dial Tone Recovery](#)
  - [Recovery database \(RDB\)](#)
  - [Transport](#)
  - [Connector](#)
  - [PST](#)
  - [DistributionGroup](#)
  - [Search](#)
  - [AuditLog](#)
  - [Test](#)
  - [Queue](#)
  - [Defrag](#)
  - [DAG](#)
  - [Index](#)
- [NAS](#)
  - [TrueNAS](#)
  - [Synology](#)
- [Veeam](#)
  - [Get-VBRCommand](#)
  - [Veeam-REStat](#)
- [REST API](#)
  - [Methods](#)
  - [Download Image](#)
  - [Token](#)
  - [GET](#)
  - [Cookie](#)
- [Pode](#)
- [Selenium](#)
  - [Selenium modules](#)
- [IE](#)
- [COM](#)
  - [Wscript.Shell.SendKeys](#)
  - [Wscript.Shell.Popup](#)
  - [WScript.Network](#)
  - [Shell.Application](#)
  - [Outlook](#)
  - [Microsoft.Update](#)
- [.NET](#)

- Match
- GeneratePassword
- SoundPlayer
- Static Class
- Clicker
- Audio
- NetSessionEnum
- CopyFile
- ShowWindowAsync
- GetAsyncKeyState
- Console API
- Drawing
- ObjectEvent
- Sockets
  - UDP-Socket
  - Test-NetUDPConnection
  - TCP-Socket
  - WakeOnLan
  - HTTPListener
  - WebClient
  - HttpClient
  - Certificate
- Base64
  - UTF8
  - Unicode
  - Image
- Excel
  - Excel.Application.Open
  - ImportExcel
- CSV
  - ConvertFrom-String
  - ConvertFrom-StringData
- XML
  - Get-CredToXML
  - XmlWriter
  - CreateElement
- JSON
- YAML
- TOML
- Markdown
  - Convert Markdown to HTML
  - PSMarkdown
  - ConvertFrom-MarkdownV2
- ConvertTo-Markdown
- HTML
  - ConvertFrom-Html
  - ConvertTo-Html
  - PSWriteHTML
  - HtmlReport
- HtmlAgilityPack
- KeePass
- SQLite
  - Database password
- MySQL
  - DATA TYPE
  - DATABASE
  - USER

- TABLE
- COLUMN
- INSERT
- SELECT
- WHERE
- DELETE
- UPDATE
- CHECK
- DUMP
- innodb\_force\_recovery
- MySQL Connector NET
- Add-ADUser
- Get-ADUser

- MSSQL
  - System.Data.SqlClient
  - SqlCommand INSERT
  - SSMS INSERT
  - T-SQL
  - Тип резервной копии
  - Модели восстановления
  - Системные БД
  - Регламентные операции

- InfluxDB
  - Install Windows
  - Install Ubuntu
  - API
  - Chronograf
  - Grafana
  - CLI Client
  - USERS
  - DATABASE
  - MEASUREMENT
  - SELECT/WHERE
  - REGEX
  - GROUP BY tag\_key
  - Functions(field\_key)
  - POLICY

- InfluxDB-api
  - API POST
  - API GET
  - Endpoints
  - PingTo-InfluxDB
  - PerformanceTo-InfluxDB
  - Service
  - PSIInfluxDB

- Telegraf
- Elasticsearch
- CData
  - ADO.NET Assembly
  - UPDATE
  - INSERT
  - DELETE
- ODBC
- PostgreSQL
  - ODBC Driver
  - npgsql
- WMI

- [WMI/CIM](#)
- [NLA](#)
- [Regedit](#)
- [Performance](#)
- [SNMP](#)
  - [Setup SNMP Service](#)
  - [Setting SNMP Service via Regedit](#)
  - [snmpwalk](#)
  - [SNMP Modules](#)
  - [Lexim.SharpSnmpLib](#)
  - [Walk](#)
- [Zabbix](#)
  - [Zabbix Agent](#)
  - [Zabbix Sender](#)
  - [Zabbix Get](#)
  - [UserParameter](#)
  - [Include Plugins](#)
  - [Zabbix API](#)
- [pki](#)
- [OpenSSL](#)
- [OpenVPN](#)
  - [server.ovpn](#)
  - [client.ovpn](#)
  - [Client](#)
- [Route](#)
- [NAT](#)
- [WireGuard](#)
- [VpnClient](#)
- [ProxyClient](#)
- [netsh](#)
  - [Reverse Proxy](#)
  - [Wlan](#)
  - [Firewall](#)
- [OpenSSH](#)
  - [PSRemoting over SSH](#)
- [WinRM](#)
  - [WinRM Configuration](#)
  - [Kerberos](#)
- [PackageManagement](#)
  - [winget](#)
  - [jlang](#)
  - [Scoop](#)
  - [Chocolatey](#)
- [NuGet](#)
- [Modules](#)
  - [Get-Query](#)
  - [Console-Translate](#)
  - [Console-Download](#)
  - [PSEverything](#)
  - [HardwareMonitor](#)
  - [CrystalDisk-Cli](#)
  - [PS-Pi-Hole](#)
  - [Check-Host](#)
  - [PSDomainTest](#)
  - [WinAPI](#)
  - [pSyslog](#)
  - [Syslog source message](#)

- Syslog type message
- Atlassian
  - Bitbucket
  - Jira
  - Confluence
- Pester
- DSC
- PSAppDeployToolkit
  - Install-DeployToolkit
  - Deploy-Notepad-Plus-Plus
  - Uninstall-Notepad-Plus-Plus
  - Deploy-WinSCP
- Load Testing
  - Apache Benchmark
  - Locust
- LLM
- OpenAI
- Mock
- OpenRouter
- LM Studio
- Ollama
- GigaChat
  - Windows
  - Linux
- YandexGPT
- SuperAGI
- Replicate
- Google API
  - Google Translate
  - Google Search
  - Google Search via RapidAPI
  - Google Filter
- Media API
- IMDb
- MoviesDatabase
- TMDB
- OMDb
- ivi
- Kinopoisk
  - kinopoisk.dev
  - UrlCode
  - KinopoiskApiUnofficial
- Kinobox
- VideoCDN
- Torrent
  - Jackett
  - Torrent-API-py
  - Plex
  - Jellyfin
- Telegram
  - Button
  - Send-ToTelegramFile
- Discord
  - Send to Discord
  - Read from Discord
  - HttpClient
  - Button

- [Discord.Net.Webhook](#)
- [Discord.Net.WebSocket](#)
- [oh-my-posh](#)
  - [themes-performance](#)
- [Windows-Terminal](#)
  - [Terminal-Icons](#)
  - [Custom Actions](#)
- [Pandoc](#)
  - [Convert Excel to Markdown](#)
- [FFmpeg](#)
- [HandBrake](#)
- [ImageMagick](#)
- [YouTube](#)

# Cheat-Sheet

## PowerShell Commands >

### Переменные

\$variable = 123	Объявить переменную
\$variable.GetType().Name	Узнать тип данных (Int32)
\$variable.ToString()	Изменить тип данных (из числа в строку)
[string]\$variable = 123	Задать тип данных
\$Global:var = "global"	Глобальное объявление (действует за пределами функции)
\$env:HOMEPATH / \$home / ~	Переменная окружения
Get-ChildItem variable:	Список всех переменных

### Массивы

\$arr = @()	Объявить пустой массив (array)
\$arr = "a","b","c"	Массив из строк (string)
1,2,3,4,5 / 1..5	Массив из цифр (int) / срез
\$arr[0]	Получить первое значение массива
\$arr[1..2]	С второго по третий индекс
\$arr[-1]	Последний элемент массива
\$arr[-1..-2]	Первый и второй с конца
\$arr[3..2] + \$arr[1..0]	Объединить два массива в один
\$arr.Count	Количество элементов в массиве
\$arr[0].Length	Количество символов элемента
\$arr[(\$arr.Count-1)..0]	Пересобрать массив с конца
\$arr += 6	Добавить в массив новый элемент
\$arr[0] += 9	Добавить (текст) или прибавить (число) к первому элементу массива

### Ассоциативный массив (хэш-таблицы) и функции

\$hash = @{a=1;b="test"}	Инициализировать
\$hash.Keys	Список всех ключей
\$hash.a = 2	Изменить значение
\$hash.a / \$hash["a"]	Отобразить по ключу
\$hash = \$hash + @{c=3}	Добавить новый ключ
\$hash.Remove("c")	Удалить ключ
function test (\$param) { \$param }	Создать функцию
test -param "input to param"	Вызвать функцию

### Справка

Get-Command g*	Поиск команды по имени
Get-Help gc	Синтаксис команды
Get-History	Вывести историю команд текущей сессии

### Управление данными

1..10   ForEach-Object {\$_.+10}	Цикл потока
\$x = 1; while (\$x -lt 10) {\$x; \$x++}	Цикл while
for (\$i=0; \$i -lt 10; \$i++) {\$i}	Цикл for
foreach (\$file in gci C:\) {\$file.name}	Цикл foreach
\$x = 5 if (\$x -lt 5) {"\$x меньше 5"} elseif (\$x -gt 5) {"\$x больше 5"} else {"\$x равно 5"}	Условие

### Директории и файлы

Get-Location	Текущая директория (pwd)
Set-Location	Сменить директорию (sl/cd)
Get-Content	Вывести содержимое файла (gc/cat)
Get-ChildItem	Содержимое папки (gci/ls)
Test-Path	Проверить путь
Out-File	Передать в файл
Out-Null	Передать в \$null
New-Item	Создай File/Directory
Get-ItemProp	Свойства
Copy-Item	Копировать (cp)
Move-Item	Переместить (mi/mv)
Rename-Item	Переименовать
Clear-Item	Удалить содержимое
Remove-Item	Удалить (ri/rm)

### Операторы присваивания, сравнения и массива

=,+,-,*,/,%=,+=,-,	Операторы переменной / массива
-and / -or / -not (!)	Операторы условий: и / или / нет
-eq (-ceq) / -ne (-nce)	Логическое равно / не равно (рег.)
-gt / -ge	Больше / больше или равно
-lt / -le	Меньше / меньше или равно
@(1,2,3) -contains 3	Проверка присутствия в массиве
-like / -notlike "*Get*"	Точное равенство (* - wildcard)
-match / -notmatch	Приблизительное совпадение
"n: 77" -match "n: (7+)"	Поиск с применением групп ()
\$matches[1]	Вывести второй результат поиска
"12321" -split "2"	Разбить строку на массив
@(10,20) -join ":"	Объединить массив в строку

### Регулярные выражения (Regex)

-replace	Замена (мама на папу): "мама" -replace "м","п"
\s / \n	Пробел / новая строка
\d / \D	Любая цифра / не цифра
\w / \W	Любая буква, цифра, или знак подчёркивания
\s / \S	Пробел, табуляция, перенос строки
\b / \B	Граница слова ("test tester" -replace "\btest\b")
^ / \$	Начало / конец ("she" -replace "\$","ll")
{2} / {2,4}	Найти два / два или 4 повторения ({2} и более)
[а-яА-Я-9]	Диапазон символов для поиска

### Объекты

Get-Date   Get-Member	Список свойств и методов объекта
\$([Get-Date]).Date / Get-Date   Select-Object Date	Получить свойство объекта по его имени
[DateTime]::UtcNow	Статическое свойство (время в формате UTC 0)
\$([Get-Date]).AddHours(-3)	Использовать метод (вычесть 3 часа)
\$obj = [PSCustomObject]@{a=1;b=2}	Создать объект
\$obj   Add-Member NoteProperty Arr @(1,2,3)	Добавить метод (новое свойство)

## Help

Get-Verb действия/глаголы, утвержденные для использования в командлетах

Get-Command \*Language\* поиск команды по имени

(Get-Command Get-Language).Module узнать к какому модулю принадлежит команда

Get-Command Get-Content | fl Module,DLL узнать принадлежность команды к модулю и dll

Get-Command -Module LanguagePackManagement отобразить список команд указанного модуля

(Get-Module LanguagePackManagement).ExportedCommands.Values отобразить список команд указанного модуля

Get-Language | Get-Member отобразить список методов команды (действия), объекты вывода и Event (события объектов: Click)

(Get-Help Get-Service).Aliases узнать псевдонимом команды

Get-Alias gsv узнать имя команды по псевдониму

Get-Help Get-Service СИНТАКСИС

Get-Help Get-Service -Parameter \* описание всех параметров

```
Get-Help Get-Service -Online
Get-Help Get-Service -ShowWindow описание параметров в GUI с фильтрацией
Show-Command вывести список команд в GUI
Show-Command Get-Service список параметров команды в GUI
Invoke-Expression iex принимает текст для выполнения в консоли как команды
$PSVersionTable текущая версия PowerShell
Set-ExecutionPolicy Unrestricted
Get-ExecutionPolicy
$Metadata = New-Object System.Management.Automation.CommandMetaData (Get-Command Get-Service) получить информацию о командлете
[System.Management.Automation.ProxyCommand]::Create($Metadata) исходный код функции
```

## Object

### Variable

```
$var = Read-Host "Enter" ручной ввод
$pass = Read-Host "Enter Password" -AsSecureString скрывать набор
$global:path = "\\path" задать глобальную переменную, например в функции
$using:srv использовать переменную текущей сессии в Invoke-сессии
Get-Variable отобразить все переменные
ls variable:/ отобразить все переменные
Get-Variable *srv* найти переменную по имени
Get-Variable -Scope Global отобразить все глобальные переменные
Get-Variable Error последняя команда с ошибкой
Remove-Variable -Name * очистить все переменные
$LASTEXITCODE содержит код вывода последней запущенной программы, например ping. Если код возврата положительный (True), то
$LastExitCode = 0
```

### Env

```
Get-ChildItem Env: отобразить все переменные окружения
$env:PSModulePath директории импорта модулей
$env:UserProfile
$env:ComputerName
$env:Username
$env:UserDNSDomain
$env:LogonServer
([DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest()).Name
[Environment]::GetFolderPath('ApplicationData')
```

### History

```
Get-History история команд текущей сессии
(Get-History)[-1].Duration.TotalSeconds время выполнения последней команды
(Get-PSReadlineOption).HistorySavePath путь к сохраненному файлу с 4096 последних команд (из модуля PSReadLine)
Get-Content (Get-PSReadlineOption).HistorySavePath | Select-String Get поиск по содержимому файла (GREP)
Set-PSReadlineOption -MaximumHistoryCount 10000 изменить количество сохраняемых команд в файл
Get-PSReadlineOption | select MaximumHistoryCount
Set-PSReadlineOption -HistorySaveStyle SaveNothing отключить ведение журнала
F2 переключиться с InlineView на ListView
```

### Clipboard

```
Set-Clipboard $srv скопировать в буфер обмена
Get-Clipboard вставить
```

## Write-Host

```
Write-Host -BackgroundColor Green "Test:" -NoNewline # изменить цвет фона и запретить перенос строки
Write-Host " True" -ForegroundColor Green # данная строка будет печататься продолжая предыдущую с новыми параметрами цвета (фон по умолчанию, изм
Write-Error "False"
Write-Warning "False"
```

## Write-Progress

```
foreach ($n in 1..100) {
    Write-Progress -Activity "Test Progress" -PercentComplete $n
    Start-Sleep -Milliseconds 100
}
```

## for

```
for ($i = 0; $i -le 100; $i+=10) {
    Write-Progress -Activity "Test Progress" -PercentComplete $i
    Start-Sleep -Seconds 1
}
```

## Array

```
$srv = @("server-01", "server-02") создать массив
$srv += @("server-03") добавить в массив новый элемент
$srv.Count отобразить кол-во элементов в массиве
Out-String построчный вывод
```

## Index

```
$srv[0] вывести первое значение элемента массива
$srv[0] = Name замена элемента в массиве
$srv[0].Length узнать кол-во символов первого значения в массиве
$srv[10..100] срез
```

```
$array = "a","b","c","d"
$num = 0
foreach ($a in $array) {
    $num += 1
    $index = [array]::IndexOf($array, $a) # узнать номер индекса по значению
    $array[$index] = $num # пересобрать исходный массив
}
```

## HashTable

```
$hashtable = @{ # Создать (инициализировать)
    "User" = $env:USERNAME;
    "Server" = $env:COMPUTERNAME
}

$hashtable += @{ # Добавить ключи
    "Profile" = $PROFILE;
    "PowerShell_Home_Dir" = $PSHOME
}

$hashtable.Keys СПИСОК всех ключей
$hashtable["User"] получить значение (Values) по ключу
```

```
$hashtable["User"] = "Test" изменить  
$hashtable.Remove("User") удалить ключ
```

## Collections/List

```
$Collections = New-Object System.Collections.Generic.List[System.Object]  
$Collections.Add([PSCustomObject]@{  
    User = $env:username;  
    Server = $env:computername  
})
```

## PSCustomObject

```
$CustomObject = [PSCustomObject][ordered]@{  
    User = $env:username;  
    Server = $env:computername  
}
```

## Add and Remove Property

```
$CustomObject | Add-Member -MemberType NoteProperty -Name Arr -Value @(1,2,3) добавить свойство/стобец  
$CustomObject.Arr = @(1,3,5) изменить содержимое  
$CustomObject.PsObject.Properties.Remove('User') удалить Property
```

## Add Method

```
$ScriptBlock = {Get-Service}  
$CustomObject | Add-Member -Name "TestMethod" -MemberType ScriptMethod -Value $ScriptBlock  
$CustomObject | Get-Member  
$CustomObject.TestMethod()
```

## Class

```
Class CustomClass {  
    [string]$User  
    [string]$Server  
    Start([bool]$Param1) {  
        If ($Param1) {  
            Write-Host "Start Function"  
        }  
    }  
}  
  
$Class = New-Object -TypeName CustomClass  
$Class.User = $env:username  
$Class.Server = $env:computername  
$Class.Start(1)
```

## Pipeline

```
$CustomObject | Add-Member -MemberType NoteProperty -Name "Type" -Value "user" -Force добавление объекта вывода NoteProperty  
$CustomObject | Add-Member -MemberType NoteProperty -Name "User" -Value "admin" -Force изменение содержимого для сущности объекта User  
ping $srv | Out-Null перенаправить результат вывода в Out-Null
```

## Select-Object

```
Get-Process | Select-Object -Property * отобразить все доступные объекты вывода  
Get-Process | select -Unique "Name" удалить повторяющиеся значения в массиве
```

```
Get-Process | select -ExpandProperty ProcessName преобразовать из объекта-коллекции в массив (вывести содержимое без наименовав столбца)
(Get-Process | ? Name -match iperf).Modules список используемых модулей процессом
```

## Expression

```
Get-Process | Sort-Object -Descending CPU | select -first 10 ProcessName, # сортировка по CPU, вывести первых 10 значений (-first)
@{Name="ProcessorTime";
    Expression={$_.TotalProcessorTime -replace "\.\d+"} # затрачено процессорного времени в минутах
},
@{Name="Memory";
    Expression={[string]([int]($_.WS / 1024kb))+"MB"} # делим байты на КБ (1mb)
},
@{Label="RunTime";
    Expression={((Get-Date) - $_.StartTime) -replace "\.\d+"} # вычесть из текущего времени - время запуска, и удалить milisec
}
```

## Select-String

```
$ipconfig | Select-String IPv4 -replace ".+": " | Where-Object {$_.Match -match "^172."} узнать только IP
$Current_IP = Get-Content $RDCMan_RDG_PATH | Select-String $RDCMan_Display_Name -Context 0,1 получить две строки
$Current_IP = $Current_IP.Context.DisplayPostContext[0] -replace ".+<name>|</name>" забрать только вторую строку и удалить тэги
```

## Format-List/Format-Table

```
Get-Process | fl ProcessName, StartTime
Get-Process | ft ProcessName, StartTime -AutoSize автоматическая группировка размера столбцов
```

## Measure-Object

```
Get-Process | Measure | select Count кол-во объектов
Get-Process | Measure -Line -Word -Character кол-во строк, слов и Char объектов
Get-Process | Measure-Object PM -sum | Select-Object Count,@{Name="MEM_MB"; Expression={[int]($_.Sum/1mb)}} кол-во процессов и общий объем
занятой памяти в МБайт
```

## Compare-Object

```
Compare-Object -ReferenceObject (Get-Content -Path .\file1.txt) -DifferenceObject (Get-Content -Path .\file2.txt) сравнение двух файлов
$group1 = Get-ADGroupMember -Identity "Domain Admins"
$group2 = Get-ADGroupMember -Identity "Enterprise Admins"
Compare-Object -ReferenceObject $group1 -DifferenceObject $group2 -IncludeEqual сравнение двух объектов
== нет изменений
<= есть изменения в group1 => 'есть изменения в group2
```

## Where-Object (?)

```
Get-Process | Where-Object {$_.ProcessName -match "zabbix"} фильтрация/поиск процессов по имени свойства объекта
Get-Process | where CPU -gt 10 | Sort-Object -Descending CPU вывести объекты, где значения CPU больше 10
Get-Process | where WS -gt 200MB отобразить процессы где WS выше 200МБ
Get-Service | where Name -match "zabbix" поиск службы
Get-Service -ComputerName $srv | Where {$_.Name -match "WinRM"} | Restart-Service перезапустить службу на удаленном компьютере
(Get-Service).DisplayName вывести значения свойства массива
netstat -an | where {$_.Match 443}
netstat -an | ?{$_.Match 443}
(netstat -an) -match 443
```

## Sort-Object

```
Get-Process | Sort-Object -Descending CPU | ft обратная (-Descending) сортировка по CPU
Get-Process | Sort-Object -Descending cpu,ws сортировка по двум свойствам
```

```
$path[-1..-10] обратная сборка массива без сортировки  
$arr = @(1..20); $arr[$($arr.Count - 1)..0] пересобрать массив с конца
```

## Last/First

```
Get-Process | Sort-Object -Descending CPU | select -First 10 вывести первых 10 объектов  
Get-Process | Sort-Object -Descending CPU | select -Last 10 вывести последних 10 объектов
```

## Group-Object

```
$Groups = Get-CimInstance -Class Win32_PnPDriver |  
Select-Object DriverProviderName, FriendlyName, Description, DriverVersion, DriverDate |  
Group-Object DriverProviderName, FriendlyName, Description, DriverVersion, DriverDate  
$(foreach ($Group in $Groups) {  
    $Group.Group[0]  
}) | Format-Table
```

## Property

```
$srv.Count кол-во элементов в массиве  
$srv.Length содержит количество символов строки переменной [string] или количество значений (строк) объекта  
$srv.Chars(2) отобразить 3-й символ в строке  
$srv[2] отобразить 3-ю строку в массиве
```

## Method

```
$srv = "127.0.0.1"  
$srv.Insert(0,"https://") добавить значение перед первым символом  
$srv.Substring(4) удалить (из всего массива) первые 4 символа  
$srv.Remove(3) удалить из всего массива все после 3 символа  
$string = "123" создать строку  
$int = [convert]::ToInt32($string) преобразовать строку в тип данных число  
[string]::Concat($text,$num) объединить переменные в одну строку  
[string]::Join(":",$text,$num) объединить используя разделитель  
[string]::Compare($text,$num,$true) выдает 0 при совпадении или 1-1 при несовпадении, true(без учета регистра) или false (с учетом регистра)  
[string]::Equals($text,$num) производит сравнение двух строк и выдает true при их совпадении или false при несовпадении  
[string]::IsNullOrEmpty($text) проверяет наличие строки, если строка пуста true, если нет false  
[string]::IsNullOrWhiteSpace($text2) проверяет на наличие только символов пробел, табуляция или символ новой строки
```

## Error

```
$Error выводит все ошибки текущего сеанса  
$Error[0].InvocationInfo развернутый отчет об ошибке  
$Error.clear()  
$LASTEXITCODE результат выполнения последней команды (0 - успех)  
exit 1 код завершения, который возвращается $LASTEXITCODE
```

## ExecutionStatus

```
$(Get-History)[-1] | Select-Object @{  
    Name="RunTime"; Expression={$_.EndExecutionTime - $_.StartExecutionTime}  
},ExecutionStatus,CommandLine # посчитать время работы последней [-1] (или Select-Object -Last 1) выполненной команды и ее узнать статус
```

## Measure-Command

```
$(Measure-Command {ping ya.ru}).TotalSeconds получить время выполнения в секундах
```

## DateTime

```
[DateTime]::UtcNow время в формате UTC 0
$(Get-Date).AddHours(-3) вычесть три часа из текущего времени
$date = $(Get-Date -Format "dd/MM/yyyy HH:mm:ss") изменить формат отображения времени
$date = Get-Date -f "dd/MM/yyyy" получаем тип данных [string] (Date.GetType().Name)
$date = "19.05.2024"
[DateTime]$date = Get-Date "$date" преобразовать строку подходящую под формат даты в тип данных [DateTime]
$beforeDate = Get-Date "12.05.2024"
[int32]$days=$($date - $beforeDate).Days посчитать разницу в днях
"5/7/07" -as [DateTime] преобразовать входные данные в тип данных [DateTime]
```

## TimeSpan

```
New-TimeSpan -Start $(Get-Date) -End $($($Get-Date).AddMinutes(+1)) получить разницу во времени
$timeZone = (Get-TimeZone).BaseUtcOffset.TotalMinutes получить разницу в минутах от текущего часового пояса относительно UTC 0
$unixTime = (New-TimeSpan -Start (Get-Date "01/01/1970") -End ((Get-Date).AddMinutes(-$tz))).TotalSeconds вычесть минуты для получения UTC 0
$timeStamp = ([string]$unixTime -replace "\..+") + "00000000" получить текущий TimeStamp
```

## Format

```
HH # Часы в 24-часовом формате (00 до 23)
hh # Часы в 12-часовом формате (01 до 12)
mm # Минуты (00 до 59)
ss # Секунды (00 до 59)
tt # Десигнатор (AM/PM)
fff # Миллисекунды (000 до 999)
d # День месяца без ведущего нуля (1-31)
dd # День месяца с ведущим нулём (01-31)
ddd # Сокращённое название дня недели (например, "Пн")
dddd # Полное название дня недели (например, "Понедельник")
M # Номер месяца без ведущего нуля (1-12)
MM # Номер месяца с ведущим нулём (01-12)
MMM # Сокращённое название месяца (например, "янв")
MMMM # Полное название месяца (например, "Январь")
y # Год без века (0-99)
yy # Год без века с ведущим нулём (00-99)
yyyy # Год с веком (например, 2024)
g # Период или эра (например, "н.э.")
```

## Timer

```
$start_time = Get-Date зафиксировать время до выполнения команды
$end_time = Get-Date зафиксировать время по завершению
$time = $end_time - $start_time высчитать время работы скрипта
$min = $time.minutes
$sec = $time.seconds
Write-Host "$min минут $sec секунд"

$timer = [System.Diagnostics.Stopwatch]::StartNew() запустить таймер
$timer.IsRunning статус работы таймера
$timer.Elapsed.TotalSeconds отобразить время с момента запуска (в секундах)
$timer.Stop() остановить таймер
```

# Regex

```
.      # Обозначает любой символ, кроме новой строки
\     # Экранирует любой специальный символ (метасимвол). Используется, если нужно указать конкретный символ, вместо специального ({ } [ ] / \
\A (^) # Начало строки
\Z ($) # Конец строки
\n    # Новая строка
\s    # Пробел (эквивалент " "), табуляция, перенос строки
\S    # Не пробел
\d    # Число от 0 до 9 (20-07-2022 эквивалент: "\d\d-\d\d-\d\d\d\d")
\D    # Обозначает любой символ, кроме числа (цифры). Удаления всех символов, кроме цифр: [int]$("123 test" -replace "\D")
\w    # Любая буква латиницы, цифра, или знак подчёркивания (от "a" до "z" и от "A" до "Z" или число от 0 до 9)
\W    # Не латиница, не цифра, не подчёркивание
\b    # Граница слова. Применяется когда нужно выделить, что искомые символы являются словом, а не частью другого слова
\B    # Не граница слова
\A    # Начало текста
\Z    # Конец текста
+   # Повторяется 1 и более раз (\s+)
|   # Или. Соединяет несколько вариантов
()  # В круглые скобки заключаются все комбинации с "или" и поиск начала и конца строк
[]  # поиск совпадения любой буквы, например, [A-z0-9] от A до z и цифры от 0 до 9 ("192.168.1.1" -match "192.1[6-7][0-9]")
[^ ] # Исключает из поиска символы указанные в квадратных скобках
{ } # Квантификатор в фигурных скобках, указывает количество повторений символа слева на право (от 1 до 25 раз)
\d{2} # Найти две цифры
\d{2,4} # Найти две или четыре
{4,} # Найти четыре и более
```

## Якори

```
^ или \A определяет начало строки ( $url -replace '^','https:' - добавить текст в начало строки)
$ или \Z обозначают конец строки ( $ip -replace "\d{1,3}$","0" )
(?=text) поиск слова слева. Пишем слева на право от искомого (ищет только целые словосочетания - "Server:\s(.{1,30})\s(?\=$username)" )
(?<=text) поиск слова справа ( $in_time -replace ".+(?<=Last)" - удалить все до Last )
(?!text) не совпадает со словом слева
(?<!text) не совпадает со словом справа

$test = "string"
$test -replace ".{1}$" удалить любое кол-во символов в конце строки
$test -replace "^.{1}" удалить любое кол-во символов в начале строки
```

## Группы захвата

```
$date = '12.31.2021'
$date -replace '^(\d{2}).(\d{2})','$2.$1' поменять местами
$1 содержимое первой группы в скобках
$2 содержимое второй группы

-replace "1","2" замена элементов в индексах массива (везде где присутствует 1, заменить на 2), для удаления используется только первое значение
-split " " преобразовать строку в массив, разделителем указан пробел, которой удаляется ($url.Split("/")[-1])
-join " " преобразовать массив (коллекцию) в единую строку (string), добавить разделителем пробел

@(1,2,3) -contains 3 проверить, что элемент справа содержится в массиве слева
@(1,2) -notcontains 3 проверить, что элемент справа не содержится в массиве слева

-like *txt* поиск по маскам wildcard, выводит значение на экран
-match txt поиска по шаблонам, проверка на соответствие содержимого текста
-match "zabbix|rpc" условия, для поиска по нескольким словам
-NotMatch проверка на отсутствие вхождения \
```

## Matches

```
"num: 777" -match "num: ([0-9]+)" | Out-Null  
$Matches[1] ВЫВОДИМ ТОЛЬКО НОМЕР  
  
$ip = "192.168.10.1"  
$ip -match "(\\.\\d{1,3})\\.\\d{1,2}" True  
$Matches отобразить все подходящие переменные последнего поиска, которые входят и не входят в группы ()  
  
$String = "09/14/2017 12:00:27 - mtbill_post_201709141058.txt 7577_Delivered: OK"  
$String -Match ".*(?=\\.txt)" | Out-Null  
$Matches[0][-4..-1] -Join ""  
  
$string.Substring($string.IndexOf(".txt")-4, 4) 2-й вариант (IndexOf)
```

## Форматирование

```
[string]::Format("{1} {0}", "Index0", "Index1")  
"{1} {0}" -f "Index0", "Index1"  
"{0:###-##-##}" -f 1234567 записать число в другом формате (#)  
"{0:0000}" -f 123 вывести число в формате не меньше 4 знаков (0123)  
"{0:P0}" -f (220/1000) посчитать в процентах (P)  
"{0:P}" -f (512MB/1GB) сколько % составляет 512Мб от 1Гб  
"{0:0.0%}" -f 0.123 умножить на 100%
```

  

```
$gp = Get-Process | sort cpu -Descending | select -First 10  
foreach ($p in $gp) {  
    "{0} - {1:N2}" -f $p.ProcessName, $p.CPU # округлить  
}
```

## Условный оператор

```
$rh = Read-Host  
if ($rh -eq 1) {  
    ipconfig  
} elseif (  
    $rh -eq 2  
) {  
    getmac  
} else {  
    hostname  
}
```

Если условие if () является истинным (*True*), выполнить действия. Если условие if () является ложным (*False*), выполнить действие не обязательного оператора else

Условие ElseIf идёт после условия if для проверки дополнительных условий перед выполнение оператора else. Оператор, который первый вернет *True*, отменит выполнение следующих дополнительных условий. Если передать переменную \$using:operator, то будет проверяться \$True/\$False \

```
if ($(Test-NetConnection $srv -Port 80).TcpTestSucceeded) {  
    "Opened port"  
} else {  
    "Closed port"  
}
```

## Логические операторы сравнения

- eq равно (equal)
- ceq учитывать регистр
- ne не равно (not equal)

```

-ne не равно учитывая регистр
-gt больше (greater)
-ge больше или равно
-lt меньше (less)
-le меньше или равно
-in проверить на наличие (5 -in @(1,2,3,4,5))
-NOT логическое НЕТ !(Test-Path $path)
-and логическое И
-or логическое ИЛИ \

```

```

if (((($1 -eq 1) -and ($2 -eq 2)) -or ($1 -ne 3)) {
    $true
} else {
    $false
} # два условия: (если $1 = 1 И $2 = 2) ИЛИ $1 не равно 3 вернуть $true

```

## Pipeline Operators

```

Write-Output "First" && Write-Output "Second" две успешные команды выполняются
Write-Error "Bad" && Write-Output "Second" первая команда завершается ошибкой, из-за чего вторая команда не выполняется
Write-Error "Bad" || Write-Output "Second" первая команда завершается ошибкой, поэтому выполняется вторая команда
Write-Output "First" || Write-Output "Second" первая команда выполнена успешно, поэтому вторая команда не выполняется

```

## Invocation Operator

```

$addr = "8.8.8.8"
$ping = "ping"
& $ping $addr запускает текст как команду

& $ping $addr & запустить команду в фоне
(Get-Job)[-1] | Receive-Job -Keep

```

## Data Type

```

$rv.GetType() узнать тип данных
$rv -is [string] проверка на соответствие типа данных
$rv -isnot [System.Object] проверка на несоответствие
[Object] массив (BaseType:System.Array)
[DateTime] формат времени (BaseType:System.ValueType)
[Bool]/[Boolean] логическое значение (True/False) или 1/0 (1 бит) наличие/отсутствие напряжения
[Byte] 8-битное (1 байт) целое число без знака (0..255)
[Int16] 16-битное знаковое целое число от -32767 до 32767 (тип данных WORD 0..65535)
[Int] 32-битное (4 байта) знаковое целое число от -2147483648 до 2147483647 (DWORD)
[Int64] 64-битное от -9223372036854775808 до 9223372036854775808 (LWORD)
[Decimal] 128-битное десятичное значение от -79228162514264337593543950335 до 79228162514264337593543950335
[Single] число с плавающей запятой (32-разрядное)
[Double] число с плавающей запятой с двойной точностью (64-разрядное)
[String] неизменяемая строка символов Юникода фиксированной длины (BaseType:System.Object)

```

## Math

```

[math] | Get-Member -Static
[math]::Pow(2,4) 2 в 4 степени
[math]::Truncate(1.8) грубое округление, удаляет дробную часть
[math]::Ceiling(1.8) округляет число в большую сторону до ближайшего целого значения
[math]::Floor(-1.8) округляет число в меньшую сторону
[math]::Min(33,22) возвращает наименьшее значение двух значений
[math]::Max(33,22) возвращает наибольшее значение двух значений

```

## Round

```
[double]::Round(87.5, 0) 88 (нечетное), в .NET по умолчанию используется округление в средней точке ToEven, где *.5 значения округляются до ближайшего четного целого числа  
[double]::Round(88.5, 0) 88 (четное)  
[double]::Round(88.5, 0, 1) 89 (округлять в большую сторону)  
[double]::Round(1234.56789, 2) округлить до 2 символов после запятой
```

## ToString

```
(4164539/1MB).ToString("0.00") разделить на дважды на 1024/1024 и округлить до 3,97
```

## Char

```
[Char] символ Юникода (16-разрядный)  
$char = $srv.ToCharArray() разбить строку [string] на массив [System.Array] из букв
```

## Function

### Switch function

```
$MMM = Get-Date -UFormat "%m"  
switch($MMM) {  
    "01" {$Month = 'Jan'}  
    "02" {$Month = 'Feb'}  
    "03" {$Month = 'Mar'}  
    "04" {$Month = 'Apr'}  
    "05" {$Month = 'May'}  
    "06" {$Month = 'Jun'}  
    "07" {$Month = 'Jul'}  
    "08" {$Month = 'Aug'}  
    "09" {$Month = 'Sep'}  
    "10" {$Month = 'Oct'}  
    "11" {$Month = 'Nov'}  
    "12" {$Month = 'Dec'}  
}
```

### Switch param

```
Function fun-switch (  
    [switch]$param  
) {  
    If ($param) {"yes"} else {"no"}  
}  
  
fun-switch -param
```

## psm1 (module file and parameters)

```
function Get-Function {
    <#
    .SYNOPSIS
Описание
.DESCRIPTION
Описание
.LINK
https://github.com/Lifailon/PS-Commands
#>
param (
    [Parameter(Mandatory,ValueFromPipeline)][string]$Text,
    [ValidateSet("Test1","Test2")][string]$Provider = "Test1",
    [ValidateRange(1,3)][int]$Number = 2
)
Write-Host Param Text: $Text
Write-Host Param Provider: $Provider
Write-Host Param Number: $Number
}

Get-Function -Text Text1
Get-Function -Text Text2 -Provider Test2 -Number 3
```

## psd1 (module description file)

```
@{
    RootModule      = "Get-Function.psm1"
    ModuleVersion   = "0.1"
    Author         = "Lifailon"
    CompanyName    = "Open Source Community"
    Copyright       = "Apache-2.0"
    Description     = "Function example"
    PowerShellVersion = "7.2"
    PrivateData     = @{
        PSData = @{
            Tags      = @("Function","Example")
            ProjectUri = "https://github.com/Lifailon/PS-Commands"
            LicenseUri = "https://github.com/Lifailon/Console-Translate/blob/rsa/LICENSE"
            ReleaseNotes = "Second release"
        }
    }
}
```

# Bit

Двоичное	Десятичное
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15
1 0000	16

Двоичное	Десятичное	Номер разряда
1	1	0
10	2	1
100	4	2
1000	8	3
1 0000	16	4
10 0000	32	5
100 0000	64	6
1000 0000	128	7
1 0000 0000	256	8

Из двоичного => десятичное (1-й вариант по таблице)

$$1001\ 0011 = 1000\ 0000 + 1\ 0000 + 10 + 1 = 128 + 16 + 2 + 1 = 147$$

2-й вариант

$$7654\ 3210 \text{ (разряды двоичного выражения)} = (1*2^7) + (0*2^6) + (0*2^5) + (1*2^4) + (0*2^3) + (0*2^2) + (1*2^1) + (1*2^0) = 147 \\ [\mathit{math}]\mathit{::}\mathit{Pow}(2,7) + [\mathit{math}]\mathit{::}\mathit{Pow}(2,4) + [\mathit{math}]\mathit{::}\mathit{Pow}(2,1) + [\mathit{math}]\mathit{::}\mathit{Pow}(2,0) = 147` \text{ исключить } 0 \text{ и сложить степень}$$

Из десятичного => двоичное (1-й вариант по таблице)

347 вычесть ближайшие 256 = 91 (+ 1 0000 0000 забрать двоичный остаток)

$$91 - 64 = 27 \text{ ближайшее 16 (+ 100 0000)}$$

$$27 - 16 = 11 \text{ ближайшее 8 (+ 1 0000)}$$

$$11 - 8 = 3 \text{ ближайшее 2 (+ 1000)}$$

$$3 - 2 = 1 \text{ (+ 10)}$$

$$1 - 1 = 0 \text{ (+ 1)}$$

1 0101 1011

2-й вариант

Последовательное деления числа на 2, предварительно забирая остаток для получения четного числа в меньшую сторону

$$347 - 346 = \text{oстаток } 1, (347-1)/2 = 173$$

$$173 - 172 = \text{oстаток } 1, (172-1)/2 = 86$$

$$86 - 86 = \text{oстаток } 0, 86/2 = 43$$

$$43 - 42 = \text{oстаток } 1, (43-1)/2 = 21$$

$$21 - 20 = \text{oстаток } 1, (21-1)/2 = 10$$

$$10 - 10 = \text{oстаток } 0, 10/2 = 5$$

$$5 - 4 = \text{oстаток } 1, (5-1)/2 = 2$$

$$2 - 2 = \text{oстаток } 0, 2/2 = 1$$

$$1 - 1 = \text{oстаток } 1, (1-1)/2 = 0$$

Результат деления записывается снизу вверх

## Bit Convertor

```
function ConvertTo-Bit {
    param (
        [Int]$int
    )
    [array]$bits = @()
    $test = $true
    while ($test -eq $true) {
        if (($int/2).GetType() -match [double]) {
            $int = ($int-1)/2
            [array]$bits += 1
        }
        elseif (($int/2).GetType() -match [int]) {
            $int = $int/2
            [array]$bits += 0
        }
        if ($int -eq 0) {
            $test = $false
        }
    }
    $bits = $bits[-1..-999]
    ([string]($bits)) -replace "\s"
}
```

ConvertTo-Bit 347

```
function ConvertFrom-Bit {
    param (
        $bit
    )
    [int]$int = 0
    $bits = $bit.ToString().ToCharArray()
    $index = ($bits.Count)-1
    foreach ($b in $bits) {
        if ($b -notlike 0) {
            $int += [math]::Pow(2,$index)
        }
        $index -= 1
    }
    $int
}
```

ConvertFrom-Bit 10010011

Get-Process pwsh | fl ProcessorAffinity привязка процесса к ядрам, представляет из себя битовую маску (bitmask), где каждому биту соответствует ядро процессора. Если для ядра отмечено сходство (affinity), то бит выставляется в 1, если нет — то в 0. Например, если выбраны все 16 ядер, то это 1111 1111 1111 1111 или 65535.

```
$(Get-Process pwsh).ProcessorAffinity = 15 0000000000001111 присвоить 4 первых ядра
$(Get-Process pwsh).ProcessorAffinity = 61440 1111000000000000 присвоить 4 последних ядра
$(Get-Process pwsh).ProcessorAffinity = (ConvertFrom-Bit 1111000000000000)
```

## Cycle

### Foreach

```
$list = 100..110 создать массив из цифр от 100 до 110
foreach ($srv in $list) {ping 192.168.3.$srv -n 1 -w 50} $rv хранит текущий элемент из list и повторяет команду до последнего элемента в массиве
$foreach.Current текущий элемент в цикле
```

`$foreach.Reset()` обнуляет итерацию, перебор начнется заново, что приводит к бесконечному циклу  
`$foreach.MoveNext()` переход к следующему элементу в цикле

## ForEach-Object (%)

```
100..110 | %{
    ping -n 1 -w 50 192.168.3.$_ > $null
    if ($LastExitCode -eq 0) {
        Write-Host "192.168.3.$_" -ForegroundColor green
    } else {
        Write-Host "192.168.3.$_" -ForegroundColor Red
    }
}

% передать цикл через конвеер (ForEach-Object)
$_ переменная цикла и конвеера (ParseError: KaTeX parse error: Expected '}', got 'EOF' at end of input: ...ering | where {.InstalledOn.ToString() -match
"2022"} | %{$_.HotFixID.Substring(2)})` gwmi создает массив, вывод команды передается where для поиска подходящих под критерии объектов. По
конвееру передается в цикл для удаления первых (2) символов методом Substring из всех объектов HotFixID.
```

## While

```
$srv = "yandex.ru"
$out2 = "Есть пинг"
$out3 = "Нет пинга"
$out = $false # предварительно сбросить переменную, While проверяет условие до запуска цикла
while ($out -eq $false) { # пока условие является $true, цикл будет повторяться
    $out = ping -n 1 -w 50 $srv
    if ($out -match "ttl") {$out = $true; $out2} else {$out = $false; $out3; sleep 1}
}

while ($True) { # запустить бесконечный цикл
    $result = ping yandex.ru -n 1 -w 50
    if ($result -match "TTL") { # условие, при котором будет выполнен break
        Write-Host "Сайт доступен"
        break # остановит цикл
    } else {
        Write-Host "Сайт недоступен"; sleep 1
    }
}
```

## Try-Catch-Finally

```
Try {$out = pping 192.168.3.1}
Catch {Write-Warning "$($error[0])"} # выводит в случае ошибки (вместо ошибки)
finally {$out = "End"} # выполняется в конце в любом случае
```

## Files

```
Get-Content $home/desktop\test.txt -Wait аналог tail
Test-Path $path проверить доступность пути
Get-FileHash -Algorithm SHA256 "$path" узнать хэш файла по алгоритму sha256
Get-ChildItem $path -Filter *.txt -Recurse отобразить содержимое каталога (Alias: ls/gci/dir) и дочерних каталогов (-Recurse) и отфильтровать
вывод
Get-Location отобразить текущие месторасположение (Alias: pwd/gl)
Set-Location $path перемещение по каталогам (Alias: cd/sl)
Invoke-Item $path открыть файл (Alias: ii/start)
Get-ItemProperty $env:userprofile\Documents\dns-list.txt | select FullName,Directory,Name,BaseName,Extension свойства файла
Get-ItemProperty -Path $path\* | select FullName,CreationTime,LastWriteTime свойства файлов содержимого директории, дата их создания и
последнего изменения
```

```

New-Item -Path "C:\test\" -ItemType "Directory" создать директорию (Alias: mkdir/md)
New-Item -Path "C:\test\file.txt" -ItemType "File" -Value "добавить текст в файл" создать файл
"test" > "C:\test\file.txt" заменить содержимое
"test" >> "C:\test\file.txt" добавить строку в файл
New-Item -Path "C:\test\test\file.txt" -Force Ключ используется для создания отсутствующих в пути директорий или перезаписи файла если он уже существует
Move-Item перемещение объектов (Alias: mv/move)
Remove-Item "$path\" -Recurse удаление всех файлов внутри каталога, без запроса подтверждения (Alias: rm/del)
Remove-Item $path -Recurse -Include "*.txt","*.temp" -Exclude "log.txt" удалить все файлы с расширением txt и temp ([Array]), кроме log.txt
Rename-Item "C:\test\*.*" "*.jpg" переименовать файлы по маске (Alias: ren)
Copy-Item копирование файлов и каталогов (Alias: cp/copy)
Copy-Item -Path "\server-01\test" -Destination "C:\" -Recurse копировать директорию с ее содержимым (-Recurse)
Copy-Item -Path "C:\*.txt" -Destination "C:\test\" знак " в конце Destination используется для переноса папки внутрь указанной, отсутствие, что это новое имя директории
Copy-Item -Path "C:\*" -Destination "C:\test\" -Include '*.txt','*.jpg' копировать объекты с указанным расширением (Include)
Copy-Item -Path "C:\*" -Destination "C:\test\" -Exclude '*.jpeg' копировать объекты, за исключением файлов с расширением (Exclude)
$log = Copy-Item "C:\*.txt" "C:\test\" -PassThru вывести результат копирования (логирование) в переменную, можно забирать строки с помощью индексов $log[0].FullName

Unblock-File "script.ps1" разблокирует файлы скриптов PowerShell скачанных из Интернета, чтобы их можно было запустить, даже если политика выполнения PowerShell в режиме RemoteSigned

```

## Clear-env-Temp-14-days

```

$ls = Get-Item $env:TEMP\*.tmp # считать все файлы с указанным расширением
$date = (Get-Date).AddDays(-14)
foreach ($l in $ls) {
    if ($l.LastWriteTime -le $date) {
        $l.FullName
        Remove-Item $l.FullName -Recurse
    }
}

```

## System.IO.File

```

$file = [System.IO.File]::Create("$home\Desktop\test.txt") создать файл
$file.Close() закрыть файл
[System.IO.File]::ReadAllLines("$home\Desktop\test.txt") прочитать файл
$file = New-Object System.IO.StreamReader("$home\Desktop\test.txt") файл будет занят процессом PowerShell
$file | Get-Member
$file.ReadLine() построчный вывод
$file.ReadToEnd() прочитать файл целиком

```

## Read/Write Bytes

```

$file = [io.file]::ReadAllBytes("$home\Desktop\powershell.jpg") метод открывает двоичный файл, считывает его в массив байт и закрывает файл
[io.file]::WriteAllBytes("$home\Desktop\tloztotk-2.jpg",$file) сохранить байты в файл (можно использовать для выгрузки двоичных файлов из БД)

```

## Archive

### Microsoft.PowerShell.Archive

```

Compress-Archive -Path $srcPath -DestinationPath "$($srcPath).zip" -CompressionLevel Optimal архивировать (по исходному пути и названию с добавлением расширения)
Expand-Archive -Path $zip разархивировать
Expand-Archive -Path $zip -DestinationPath $dstPath указать путь извлечения
Expand-Archive -Path $zip -OutputPath $dstPath

```

## System.IO.Compression.FileSystem

```
function Expand-ArchiveFile {
    param (
        # Путь к архиву
        $Path,
        # Путь, куда извлечь файл
        $DestinationPath,
        # Имя файла, который нужно извлечь
        $FileName
    )
    Add-Type -AssemblyName System.IO.Compression.FileSystem
    try {
        # Открыть архив для чтения
        $zipArchive = [System.IO.Compression.ZipFile]::OpenRead($Path)
        # Найти файл в архиве
        $fileEntry = $zipArchive.Entries | Where-Object { $_.FullName -eq $fileName }
        if ($fileEntry) {
            # Создание потока для чтения содержимого файла
            $stream = $fileEntry.Open()
            # Создание файла и запись в него данных из потока
            $DestinationPathFileName = "$DestinationPath\$FileName"
            $fileStream = [System.IO.File]::Create($DestinationPathFileName)
            $stream.CopyTo($fileStream)
            # Закрытие потоков
            $stream.Close()
            $fileStream.Close()
        } else {
            Write-Output "Файл $fileName не найден в архиве"
        }
    } catch {
        Write-Error "Ошибка при извлечении файла из архива"
    } finally {
        # Закрыть архив
        $zipArchive.Dispose()
    }
}
```

## WinRAR

```
function Expand-ArchivePassword {
    param (
        $Path,
        $Password
    )
    $winrar = "C:\Program Files\WinRAR\WinRAR.exe"
    & $winrar x -p"$Password" $Path
}

cd "$home\Downloads"
Expand-ArchivePassword archive.rar qwe123
```

## Handle

```
$url = "https://download.sysinternals.com/files/Handle.zip"
Invoke-RestMethod $url -OutFile "$env:TEMP\handle.zip"
Expand-ArchiveFile -Path "$env:TEMP\handle.zip" -DestinationPath "$home\Documents" -FileName "handle.exe" извлекаем выбранный файл из архива
Remove-Item "$env:TEMP\handle.zip"
$handle = "$home\Documents\handle.exe"
$test = New-Object System.IO.StreamReader("$home\desktop\test.txt") занять файл текущим процессом pwsh (pid) 'SearchProcess = & ParseError:
```

KaTeX parse error: Undefined control sequence: \Users at position 11: handle "C:\Users\Lifealon\Desktop\SearchProcess.PID` завершить процесс, который удерживал файл

## Console-Menu

```
# Импортируем модуль PS-Menu в текущую сессию из репозитория GitHub
$module = "https://raw.githubusercontent.com/chrisseroka/ps-menu/master/ps-menu.psm1"
Invoke-Expression $(Invoke-RestMethod $module)
```

Пример навигации по директориям в системе используя меню:

```
function ls-menu {
    param (
        $startDir = "C:\"
    )
    clear
    # Проверяем, что мы не находимся в root директории (исключить возврат назад)
    if ([System.IO.Path]::GetPathRoot($startDir) -eq $startDir) {
        $select = menu @(
            $($Get-ChildItem $startDir).name
        )
    }
    else {
        $select = menu @(
            ("..") + $($Get-ChildItem $startDir).name
        )
    }
    # Если выбрали возврат назад, то забираем только путь у стартовой директории
    if ($select -eq "..") {
        $backPath = [System.IO.Path]::GetDirectoryName($startDir)
        ls-menu $backPath
    }
    else {
        # Проверяем, что выбрали директорию
        if ($(Test-Path "$startDir\$select" -PathType Container)) {
            # Если выбрали директорию, к стартовому пути добавляем выбранное имя директории
            ls-menu "$startDir\$select"
        }
        else {
            ls-menu $startDir
        }
    }
}

ls-menu
ls-menu $home
ls-menu "D:\\"
```

## Credential

```
$Cred = Get-Credential сохраняет креды в переменные $Cred.Username и $Cred.Password
$cRed.GetNetworkCredential().password извлечь пароль
cmdkey /generic:"TERMSRV/$srv" /user:"$username" /pass:"$password" добавить указанные креды аудентификации на терминальный сервер для подключения без пароля
mstsc /admin /v:$srv авторизоваться
cmdkey /delete:"TERMSRV/$srv" удалить добавленные креды аудентификации из системы
rundll32.exe keymgr.dll,KRShowKeyMgr хранилище Stored User Names and Password
Get-Service VaultSvc служба для работы Credential Manager
Install-Module CredentialManager установить модуль управления Credential Manager к хранилищу PasswordVault из PowerShell
[System.Net.ServicePointManager]::SecurityProtocol = [System.Net.SecurityProtocolType]'Tls11,Tls12' для установки модуля
```

```

Get-StoredCredential ПОЛУЧИТЬ учетные данные из хранилища Windows Vault
Get-StrongPassword генератор пароля
New-StoredCredential -UserName test -Password "123456" добавить учетную запись
Remove-StoredCredential удалить учетную запись
$Cred = Get-StoredCredential | where {$_.username -match "admin"}
$pass = $cred.password
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($pass)
[System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)

```

## Out-Gridview

```
Get-Service -cn $srv | Out-GridView -Title "Service $srv" -OutputMode Single -PassThru | Restart-Service перезапустить выбранную службу
```

## Out-File

Read-Host -AsSecureString | ConvertFrom-SecureString | Out-File "\$env:UserProfile\Desktop\password.txt" писать в файл. Преобразовать пароль в формат SecureString с использованием шифрования Windows Data Protection API (DPAPI)

## Get-Content (gc/cat/type)

\$password = gc "\$env:UserProfile\Desktop\password.txt" | ConvertTo-SecureString читать хэш пароля из файла с помощью ключей, хранящихся в профиле текущего пользователя, который невозможно прочитать на другом компьютере

## AES Key

```

$AESKey = New-Object Byte[] 32
[Security.Cryptography.RNGCryptoServiceProvider]::Create().GetBytes($AESKey)
$AESKey | Out-File "C:\password.key"
$cred.Password | ConvertFrom-SecureString -Key (Get-Content "C:\password.key") | Set-Content "C:\password.txt" сохранить пароль в файл используя внешний ключ
$pass = Get-Content "C:\password.txt" | ConvertTo-SecureString -Key (Get-Content "\\Server\Share\password.key") расшифровать пароль на втором компьютере

```

## WinEvent

```

Get-WinEvent -ListLog * отобразить все доступные журналы логов
Get-WinEvent -ListLog * | where RecordCount -ne 0 | where RecordCount -ne $null | sort -Descending RecordCount отобразить не пустые журналы с сортировкой по кол-ву записей
Get-WinEvent -ListProvider * | ft отобразить всех провайдеров приложений
Get-WinEvent -ListProvider GroupPolicy найти в какой журнал LogLinks {Application} пишутся логи приложения
Get-WinEvent -ListProvider *smb*
Get-WinEvent -ListLog * | where logname -match SMB | sort -Descending RecordCount найти все журналы по имени
Get-WinEvent -LogName "Microsoft-Windows-SmbClient/Connectivity"
Get-WinEvent -ListProvider *firewall*

```

## Filter XPath/Hashtable

```

Get-WinEvent -FilterHashtable @{LogName="Security";ID=4624} найти логи по ID в журнале Security
Get-WinEvent -FilterHashtable @{LogName="System";Level=2} найти все записи ошибки (1 - критический, 3 - предупреждение, 4 - сведения)
Get-WinEvent -FilterHashtable @{LogName="System";Level=2;ProviderName="Service Control Manager"} отфильтровать по имени провайдера
([xml](Get-WinEvent -FilterHashtable @{LogName="Security";ID=4688} -MaxEvents 1).ToXml()).Event.EventData.Data отобразить все свойства, хранимые в EventData (Message)
Get-WinEvent -FilterHashtable @{logname="security";ID=4688} -MaxEvents 1 | select timecreated,$_.Properties[5].value} отфильтровать время события и имя запущенного процесса

```

```

$query = '
<QueryList>
    <Query Id="0" Path="Security">
        <Select Path="Security">
            *[System[EventID=4688]] and
            *[EventData[Data[@Name="NewProcessName"]="C:\Windows\System32\autochk.exe" or Data[@Name="NewProcessName"]="C:\Windows\System32\servi
        </Select>
    </Query>
</QueryList>
'

Get-WinEvent -LogName Security -FilterXPath $query

```

## Reboot

```

$query = '
<QueryList>
    <Query Id="0" Path="System">
        <Select Path="System">
            *[[
                System[
                    EventID=41 or
                    EventID=1074 or
                    EventID=1076 or
                    EventID=6005 or
                    EventID=6006 or
                    EventID=6008 or
                    EventID=6009 or
                    EventID=6013
                ]
            ]]
        </Select>
    </Query>
</QueryList>
'

Get-WinEvent -LogName System -FilterXPath $query

```

41` Система была перезагружена без корректного завершения работы.  
1074` Система была корректно выключена пользователем или процессом.  
1076` Следует за Event ID 6008 и означает, что первый пользователь (с правом выключения системы) подключившийся к серверу после неожиданной перезагрузки.  
6005` Запуск «Журнала событий Windows» (Event Log). Указывает на включение системы.  
6006` Остановка «Журнала событий Windows». Указывает на выключение системы.  
6008` Предыдущее выключение системы было неожиданным.  
6009` Версия операционной системы, зафиксированная при загрузке системы.  
6013` Время работы системы (system uptime) в секундах.

## Logon

```

$srvc = "localhost"
$filterXPath = '<QueryList><Query Id="0"><Select>*[System[EventID=21]]</Select></Query></QueryList>'
$RDPAuths = Get-WinEvent -ComputerName $srvc -LogName "Microsoft-Windows-TerminalServices-LocalSessionManager/Operational" -FilterXPath $filterXPath
$xml[]]$xml = $RDPAuths | Foreach {$_.ToXml()}
$EventData = Foreach ($event in $xml.Event) {
    New-Object PSObject -Property @{
        "Connection Time" = (Get-Date ($event.System.TimeCreated.SystemTime) -Format 'yyyy-MM-dd hh:mm K')
        "User Name" = $event.UserData.EventXML.User
        "User ID" = $event.UserData.EventXML.SessionID
        "User Address" = $event.UserData.EventXML.Address
        "Event ID" = $event.System.EventID
    }
}
$EventData | ft

```

## EventLog

```
Get-EventLog -List отобразить все корневые журналы логов и их размер  
Clear-EventLog Application очистить логи указанного журнала  
Get-EventLog -LogName Security -InstanceId 4624 найти логи по ID в журнале Security
```

## Firewall

```
$days = 5  
$obj = @()  
$fw = Get-WinEvent "Microsoft-Windows-Windows Firewall With Advanced Security/Firewall"  
foreach ($temp_fw in $fw) {  
    if ($temp_fw.id -eq 2097) { # 2004  
        $type = "Added Rule"  
    }  
    elseif ($temp_fw.id -eq 2006) {  
        $type = "Deleted Rule"  
    }  
    $port = $temp_fw.Properties[7] | select -ExpandProperty value  
    $name = $temp_fw.Properties[1] | select -ExpandProperty value  
    $obj += [PSCustomObject]@{  
        Time = $temp_fw.TimeCreated;  
        Type = $type;  
        Port = $port;  
        Name = $name}  
}  
$obj | Where-Object time -gt (Get-Date).AddDays(-$days)  
  
New-NetFirewallRule -Profile Any -DisplayName "Open Port 135 RPC" -Direction Inbound -Protocol TCP -LocalPort 135 открыть in-порт  
Get-NetFirewallRule | where DisplayName -match kms | select * найти правило по имени  
Get-NetFirewallPortFilter | where LocalPort -like 80 найти действующие правила по номеру порта  
  
Get-NetFirewallRule -Enabled True -Direction Inbound | select -Property DisplayName,  
@{Name='Protocol';Expression={({_| Get-NetFirewallPortFilter).Protocol}},  
@{Name='LocalPort';Expression={({_| Get-NetFirewallPortFilter).LocalPort}},  
@{Name='RemotePort';Expression={({_| Get-NetFirewallPortFilter).RemotePort}},  
@{Name='RemoteAddress';Expression={({_| Get-NetFirewallAddressFilter).RemoteAddress}},  
Enabled,Profile
```

## Firewall-Manager

```
Install-Module Firewall-Manager  
Export-FirewallRules -Name * -CSVFile $home\documents\fw.csv -Inbound -Outbound -Enabled -Disabled -Allow -Block (фильтр правил для экспорта)  
Import-FirewallRules -CSVFile $home\documents\fw.csv
```

## Defender

```
Import-Module Defender  
Get-Command -Module Defender  
Get-MpComputerStatus  
(Get-MpComputerStatus).AntivirusEnabled статус работы антивируса  
  
$session = NewCimSession -ComputerName hostname подключиться к удаленному компьютеру, используется WinRM  
Get-MpComputerStatus -CimSession $session | fl fullscan* узнать дату последнего сканирования на удаленном компьютере  
  
Get-MpPreference настройки  
(Get-MpPreference).ScanPurgeItemsAfterDelay время хранения записей журнала защитника в днях  
Set-MpPreference -ScanPurgeItemsAfterDelay 30 изменить время хранения  
ls "C:\ProgramData\Microsoft\Windows Defender\Scans\History"
```

```

Get-MpPreference | select disable* отобразить статус всех видов проверок/сканирований
Set-MpPreference -DisableRealtimeMonitoring $true отключить защиту Defender в реальном времени (использовать только ручное сканирование)
Set-MpPreference -DisableRemovableDriveScanning $false включить сканирование USB накопителей
Get-MpPreference | select excl* отобразить список всех исключений
(Get-MpPreference).ExclusionPath
Add-MpPreference -ExclusionPath C:\install добавить директорию в исключение
Remove-MpPreference -ExclusionPath C:\install удалить из исключения
New-ItemProperty -Path "HKLM:\SOFTWARE\Policies\Microsoft\Windows Defender" -Name DisableAntiSpyware -Value 1 -PropertyType DWORD -Force
полностью отключить Windows Defender

Set-MpPreference -SignatureDefinitionUpdateFileSharesSources \\FileShare1\Updates для обновления из сетевой папки нужно предварительно
скачать файлы с сигнатурами баз с сайта https://www.microsoft.com/security/portal/definitions/adl.aspx и поместить в сетевой каталог
Update-MpSignature -UpdateSource FileShares изменить источник обновлений (MicrosoftUpdateServer – сервера обновлений MS в интернете,
InternalDefinitionUpdateServer — внутренний WSUS сервер)
Update-MpSignature обновить сигнатурные базы

Start-MpScan -ScanType QuickScan быстрая проверка или FullScan
Start-MpScan -ScanType FullScan -AsJob
Set-MpPreference -RemediationScheduleDay 1-7 выбрать дни, начиная с воскресенья или 0 каждый день, 8 - сбросить
Set-MpPreference -ScanScheduleQuickScanTime 14:00:00
Start-MpScan -ScanType CustomScan -ScanPath "C:\Program Files" сканировать выбранную директорию

Get-MpThreat история угроз и тип угрозы (ThreatName: HackTool/Trojan)
Get-MpThreatCatalog список известных видов угроз
Get-MpThreatDetection история защиты (активных и прошлые) и ID угрозы
Get-MpThreat -ThreatID 2147760253

ls "C:\ProgramData\Microsoft\Windows Defender\Quarantine\" директория хранения файлов в карантине
cd "C:\Program Files\Windows Defender\
.\MpCmdRun.exe -restore -name $ThreatName восстановить файл из карантина
.\MpCmdRun.exe -restore -filepath $path_file

```

## DISM

```

Get-Command -Module Dism -Name *Driver*
Export-WindowsDriver -Online -Destination C:\Users\Lifailon\Documents\Drivers\ Извлечение драйверов из текущей системы
(C:\Windows\System32\DriverStore\FileRepository), выгружает список файлов, которые необходимы для установки драйвера (dll,sys,exe) в
соответствии со списком файлов, указанных в секции [CopyFiles] inf-файла драйвера.
Export-WindowsDriver -Path C:\win_image -Destination C:\drivers извлечь драйвера из оффлайн образа Windows, смонтированного в каталог
c:\win_image
$BackupDrivers = Export-WindowsDriver -Online -Destination C:\Drivers
$BackupDrivers | ft Driver,ClassName,ProviderName,Date,Version,ClassDescription список драйверов в объектном представлении
$BackupDrivers | where classname -match printer
pnputil.exe /add-driver C:\drivers\*.inf /subdirs /install установить все (параметр subdirs) драйвера из указанной папки (включая вложенные)

sfc /scannow проверить целостность системных файлов с помощью утилиты SFC (System File Checker), в случае поиска ошибок, попробует
восстановить их оригинальные копии из хранилища системных компонентов Windows (каталог C:\Windows\WinSxS). Вывод работы логируется в
C:\Windows\Logs\CBS с тегом SR
Get-ComputerInfo | select * подробная информация о системе (WindowsVersion,WindowsEditionId,Bios)
Get-WindowsImage -ImagePath E:\sources\install.wim список доступных версий в образе
Repair-WindowsImage -Online -ScanHealth
Repair-WindowsImage -Online -RestoreHealth восстановление хранилища системных компонентов
Repair-WindowsImage -Online -RestoreHealth -Source E:\sources\install.wim:3 -LimitAccess восстановление в оффлайн режиме из образа по
номеру индекса

```

## Scheduled

```
$Trigger = New-ScheduledTaskTrigger -At 01:00am -Daily 1:00 ночи
$Trigger = New-ScheduledTaskTrigger -AtLogon запуск при входе пользователя в систему
$Trigger = New-ScheduledTaskTrigger -AtStartup при запуске системы
$user = "NT AUTHORITY\SYSTEM"
$action = New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "$home\Documents\dns-change-tray-1.3.ps1"
$action = New-ScheduledTaskAction -Execute "PowerShell.exe" -Argument "-NoProfile -NoLogo -NonInteractive -ExecutionPolicy Unrestricted -WindowStyle
Register-ScheduledTask -TaskName "dns-change-tray-startup" -Trigger $trigger -User $user -Action $action -RunLevel Highest -Force

Get-ScheduledTask | ? state -ne Disabled СПИСОК всех активных заданий
Start-ScheduledTask dns-change-tray-startup запустить задание немедленно
Get-ScheduledTask dns-change-tray-startup | Disable-ScheduledTask отключить задание
Get-ScheduledTask dns-change-tray-startup | Enable-ScheduledTask включить задание
Unregister-ScheduledTask dns-change-tray-startup удалить задание
Export-ScheduledTask dns-change-tray-startup | Out-File $home\Desktop\Task-Export-Startup.xml экспортовать задание в xml
Register-ScheduledTask -Xml (Get-Content $home\Desktop\Task-Export-Startup.xml | Out-String) -TaskName "dns-change-tray-startup"
```

## Network

### ping

```
Test-Connection -Count 1 $srv1, $srv2 отправить icmp-пакет двум хостам
Test-Connection $srv -ErrorAction SilentlyContinue не выводить ошибок, если хост не отвечает
Test-Connection -Source $srv1 -ComputerName $srv2 Пинг с удаленного компьютера
```

```
function Test-PingNetwork {
    param (
        [Parameter(Mandatory,ValueFromPipeline)][string[]]$Network,
        [ValidateRange(100,10000)][int]$Timeout = 100
    )
    $ping = New-Object System.Net.NetworkInformation.Ping
    $Network = $Network -replace "0$"
    $net = @()
    foreach ($r in @(1..254)) {
        $net += "$network$r"
    }
    foreach ($n in $net) {
        $ping.Send($n, $timeout) | select @{Name="Address"; Expression={$n -replace ".+\."}}, Status
    }
}
```

```
Test-PingNetwork -Network 192.168.3.0
Test-PingNetwork -Network 192.168.3.0 -Timeout 1000
```

```
Get-CimInstance -Class Win32_PingStatus -Filter "Address='127.0.0.1'"
Get-CimInstance -Class Win32_PingStatus -Filter "Address='127.0.0.1'" | Format-Table -Property Address, ResponseTime, StatusCode -AutoSize 0 - успех
'127.0.0.1','8.8.8.8' | ForEach-Object -Process {Get-CimInstance -Class Win32_PingStatus -Filter ("Address='$$_") | Select-Object -Property Address
$ips = 1..254 | ForEach-Object -Process {'192.168.1.' + $$_} сформировать массив из ip-адресов подсети
```

### dhcp

```
Get-CimInstance -Class Win32_NetworkAdapterConfiguration -Filter "DHCPEnabled=$true" отобразить адAPTERы с включенным DHCP
$qql = 'SELECT * from Win32_NetworkAdapterConfiguration WHERE IPEnabled=True and DHCPEnabled=False'
Invoke-CimMethod -MethodName ReleaseDHCPLease -Query $qql включение DHCP на всех адAPTERах
Invoke-CimMethod -ClassName Win32_NetworkAdapterConfiguration -MethodName ReleaseDHCPLeaseAll отменить аренду адресов DHCP на всех адAPTERах
Invoke-CimMethod -ClassName Win32_NetworkAdapterConfiguration -MethodName RenewDHCPLeaseAll обновить аренду адресов DHCP на всех адAPTERах
```

## **port**

```
tnc $srv -p 5985
tnc $srv -CommonTCPPort WINRM HTTP,RDP,SMB
tnc ya.ru -TraceRoute -Hops 2 TTL=2
tnc ya.ru -DiagnoseRouting маршрутизация до хоста, куда (DestinationPrefix: 0.0.0.0/0) через (NextHop: 192.168.1.254)
```

## **netstat**

```
netstat -anop tcp -n/-f/-b
Get-NetTCPConnection -State Established,Listen | ? LocalPort -Match 3389
Get-NetTCPConnection -State Established,Listen | ? RemotePort -Match 22
Get-NetUDPEndpoint | ? LocalPort -Match 514 netstat -ap udp`
```

## **nslookup**

```
nslookup ya.ru 1.1.1.1 с указанием DNS сервера
nslookup -type=any ya.ru указать тип записи
Resolve-DnsName ya.ru -Type MX ALL,ANY,A,NS,SRV,CNAME,PTR,TXT(spf)
[System.Net.Dns]::GetHostEntry("ya.ru")
```

## **ipconfig**

```
Get-NetIPConfiguration
Get-NetIPConfiguration -InterfaceIndex 14 -Detailed
```

## **Adapter**

```
Get-NetAdapter
Set-NetIPInterface -InterfaceIndex 14 -Dhcp Disabled отключить DHCP
Get-NetAdapter -InterfaceIndex 14 | New-NetIPAddress -IPAddress 192.168.3.99 -DefaultGateway 192.168.3.1 -PrefixLength 24 задать/добавить
статический IP-адрес
Set-NetIPAddress -InterfaceIndex 14 -IPAddress 192.168.3.98 изменить IP-адрес на адаптере
Remove-NetIPAddress -InterfaceIndex 14 -IPAddress 192.168.3.99 удалить IP-адрес на адаптере
Set-NetIPInterface -InterfaceIndex 14 -Dhcp Enabled включить DHCP
```

## **DNSClient**

```
Get-DNSClientServerAddress список интерфейсов и настроенные на них адреса DNS сервера
Set-DNSClientServerAddress -InterfaceIndex 14 -ServerAddresses 8.8.8.8 изменить адрес DNS сервера на указанного интерфейсе
```

## **DNSCache**

```
Get-DnsClientCache отобразить кэшированные записи клиента DNS
Clear-DnsClientCache очистить кэш
```

## **Binding**

```
Get-NetAdapterBinding -Name Ethernet -IncludeHidden -AllBindings
Get-NetAdapterBinding -Name "Беспроводная сеть" -DisplayName "IP версии 6 (TCP/IPv6)" | Set-NetAdapterBinding -Enabled $false отключить IPv6 на
адаптере
```

## **TCPSetting**

```
Get-NetTCPSetting
Set-NetTCPSetting -SettingName DatacenterCustom,Datacenter -CongestionProvider DCTCP настраивает провайдера управления перегрузкой
(Congestion Control Provider) на DCTCP (Data Center TCP) для профилей TCP с именами DatacenterCustom и Datacenter
Set-NetTCPSetting -SettingName DatacenterCustom,Datacenter -CwndRestart True включает функцию перезапуска окна перегрузки (Congestion Window
Restart, CwndRestart) для указанных профилей TCP. Это означает, что после периода идле (когда нет передачи данных) TCP окно перегрузки
будет сбрасываться
```

```
Set-NetTCPSetting -SettingName DatacenterCustom,Datacenter -ForceWS Disabled отключает принудительное масштабирование окна (Forced Window Scaling) для указанных профилей TCP. Масштабирование окна — это механизм, который позволяет увеличивать размер окна перегрузки TCP, чтобы улучшить производительность передачи данных по сети с высокой пропускной способностью и большой задержкой
```

## hostname

```
$env:computername  
hostname.exe  
(Get-CIMInstance CIM_ComputerSystem).Name  
(New-Object -ComObject WScript.Network).ComputerName  
[System.Environment]::MachineName  
[System.Net.Dns]::GetHostName()
```

## arp

```
ipconfig /all | Select-String "физ" grep  
Get-NetNeighbor -AddressFamily IPv4
```

```
function Get-ARP {  
    Param (  
        $proxy,  
        $search  
    )  
    if (!$proxy) {  
        $arp = arp -a  
    }  
    if ($proxy) {  
        $arp = icm $proxy { arp -a }  
    }  
    $mac = $arp[3..260]  
    $mac = $mac -replace "^\\s\\s"  
    $mac = $mac -replace "\\s{1,50}", "  
$mac_coll = New-Object System.Collections.Generic.List[System.Object]  
foreach ($m in $mac) {  
    $smac = $m -split " "  
    $mac_coll.Add([PSCustomObject]@{  
        IP      = $smac[0];  
        MAC    = $smac[1];  
        Type   = $smac[2]  
    })  
}  
if ($search) {  
    if ($search -NotMatch "\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}") {  
        #$ns = nslookup $search  
        #$ns = $ns[-2]  
        #global:ns = $ns -replace "Address:\\s{1,10}"  
        $rdns = Resolve-DnsName $search -ErrorAction Ignore  
        $ns = $rdns.IPAddress  
        if ($ns -eq $null) {  
            return  
        }  
    }  
    else {  
        $ns = $search  
    }  
    $mac_coll = $mac_coll | ? ip -Match $ns  
}  
$mac_coll  
}  
  
Get-ARP -search 192.168.3.100  
Get-ARP -search 192.168.3.100 -proxy dc-01
```

## Network Adapter Statistics

```
netstat -se  
Get-NetAdapterStatistics
```

## SpeedTest

```
function Get-SpeedTestOokla {  
    param (  
        $Server = 3682,  
        [switch]$List  
    )  
    $path = "$env:TEMP\speedtest.exe"  
    $testPath = Test-Path "$env:TEMP\speedtest.exe"  
    if ($testPath -eq $false) {  
        Invoke-RestMethod https://install.speedtest.net/app/cli/ookla-speedtest-1.2.0-win64.zip -OutFile "$env:TEMP\speedtest.zip"  
        Expand-Archive "$env:TEMP\speedtest.zip" -DestinationPath $env:TEMP  
    }  
    if ($List) {  
        (& $path -L -f json | ConvertFrom-Json).servers  
    } else {  
        $test = & $path -s $Server -f json  
        $Collections = New-Object System.Collections.Generic.List[System.Object]  
        $Collections.Add(  
            [PSCustomObject]@{  
                Date = $($test | ConvertFrom-Json).timestamp  
                Url = $($test | ConvertFrom-Json).result.url  
                Download = [double]::Round($($test | ConvertFrom-Json).download.bandwidth / 1mb * 8), 2  
                Upload = [double]::Round($($test | ConvertFrom-Json).upload.bandwidth / 1mb * 8), 2  
                Ping = $($test | ConvertFrom-Json).ping.latency  
                Internal_IP = $($test | ConvertFrom-Json).interface.internalIp  
                External_IP = $($test | ConvertFrom-Json).interface.externalIp  
                Server = $($test | ConvertFrom-Json).server  
            }  
        )  
        $Collections  
    }  
}
```

## iPerf

### Install

```
$url = $($Invoke-RestMethod https://api.github.com/repos/ar5ian/iperf3-win-builds/releases/latest).assets | Where-Object name -match "win64.zip"  
Invoke-RestMethod $url -OutFile $home\Downloads\iperf.zip  
New-Item "$home\Documents\iperf3" -Type Directory | Out-Null  
Expand-Archive -Path "$home\Downloads\iperf.zip" -OutputPath "$home\Documents\iperf3"  
Remove-Item "$home\Downloads\iperf*" -Force -Recurse  
  
& "$home\Documents\iperf3\iperf3.exe" -h
```

### Env-Update-Exec-Path

```
$EnvPath = [Environment]::GetEnvironmentVariable("Path", [EnvironmentVariableTarget]::Machine)  
$EnvPath -split ";"  
$iperfPath = "$home\Documents\iperf3\"  
$EnvAddPath = $EnvPath + ";" + $iperfPath  
[Environment]::SetEnvironmentVariable("Path", $EnvAddPath, [EnvironmentVariableTarget]::Machine)  
$([Environment]::GetEnvironmentVariable("Path", [EnvironmentVariableTarget]::Machine)) -split ";"
```

```
iperf3 -h
```

## iPerf-GUI

```
Invoke-RestMethod "https://github.com/Lifailon/iPerf-GUI/raw/rsa/iPerf-GUI-Install.exe" -OutFile "$home\Downloads\iPerf-GUI-Install.exe" скачать установочную версию собранную с помощью WinRAR
```

```
Start-Process -FilePath "$home\Downloads\iPerf-GUI-Install.exe" -ArgumentList "/S" -NoNewWindow -Wait установить в тихом режиме
```

## iPerf-Docker

```
echo '  
FROM alpine:latest  
RUN apk update && apk add --no-cache iperf3  
ENV PORT=5201  
EXPOSE $PORT  
CMD ["sh", "-c", "iperf3 -s -p $PORT"]  
' > Dockerfile  
  
docker build -t iperf3-alpine-server .  
docker run -d -p 5201:5201 --name iperf3-alpine-server iperf3-alpine-server
```

## Server

```
iperf3 -s запуск сервера  
iperf3 -s -D запустить сервер в фоновом режиме как службу (--daemon)  
Get-NetTCPConnection -State Established,Listen | ? LocalPort -Match 5201 проверить, что порт сервера слушает  
Get-Process -Id $(Get-NetTCPConnection -State Established,Listen | ? LocalPort -Match 5201).OwningProcess получить процесс по порту  
Get-Process iperf3 | Stop-Process остановить процесс  
iperf3 -s -D --logfile "$home\Documents\iperf3\iperf3.log" перенаправить вывод в лог файл  
iperf3 -s -p 5211 указать порт, на котором будет слушать сервер или отправлять запросы клиент  
iperf3 -s -p 5211 -f M изменить формат выводимых данных (измерять в байтах а не в битах, доступные значения: K,M,G,T)  
iperf3 -s -p 5211 -f M -J вывод в формате json  
iperf3 -s -p 5211 -f M -v вывод подробной информации
```

## Client

```
iperf3 -c 192.168.3.100 -p 5211 подключение к серверу (по умолчанию проверяется отдача на сервер с клиента)  
iperf3 -c 192.168.3.100 -p 5211 -R обратный тест, проверка скачивания с сервера (--reverse, сервер отправляет данные клиенту)  
iperf3 -c 192.168.3.100 -p 5211 -R -P 2 количество одновременных потоков ([SUM] - суммарная скорость нескольких потоков)  
iperf3 -c 192.168.3.100 -p 5211 -R -4 использовать только IPv4  
iperf3 -c 192.168.3.100 -p 5211 -R -u использовать UDP вместо TCP  
iperf3 -c 192.168.3.100 -p 5211 -R -u -b 2mb установить битрейт в 2.00 Mbits/sec для UDP (по умолчанию 1 Мбит/сек, для TCP не ограничено)  
iperf3 -c 192.168.3.100 -p 5211 -R -t 30 время одного теста в секундах (по умолчанию 10 секунд)  
iperf3 -c 192.168.3.100 -p 5211 -R -n 1gb указать объем данных для проверки (применяется вместо времени -t)  
iperf3 -c 192.168.3.100 -p 5211 -R --get-server-output вывести вывод сервера на клиенте
```

## Output

```
sender upload (скорость передачи на удаленный сервер)  
receiver download (скорость скачивания с удаленного сервера)  
Interval общее время сканирования  
Transfer кол-во переданных и полученных МБайт  
Bandwidth скорость передачи (измеряется в Мбит/с)
```

## PS-iPerf

```
Install-Module ps-ipperf -Repository NuGet  
Import-Module PS-iPerf  
Start-iPerfServer -Port 5211 запустить сервер
```

```
Get-iPerfServer статус работы сервера
Stop-iPerfServer остановить сервер
Connect-iPerfServer -Server 192.168.3.100 -Port 5211 -MBytes 500 -Download подключиться к серверу и скачать 500 МБайт
$SpeedTest = Connect-iPerfServer -Server 192.168.3.100 -Port 5211 -MBytes 500 -LogWrite передать 500 МБайт на сервер (вести запись в лог-файл)
$SpeedTest.Intervals метрики измерений
Get-iPerfLog прочитать лог-файл
```

## RDP

```
Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name "PortNumber" отобразить номер текущего RDP порта
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name "PortNumber" -Value "3390" изменить RDP-порт
$(Get-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Terminal Server\" -Name "fDenyTSConnections").fDenyTSConnections если 0, то включен
Set-ItemProperty -Path "HKLM:\System\CurrentControlSet\Control\Terminal Server\" -Name "fDenyTSConnections" -Value 0 включить RDP
reg add "HKLM\SYSTEM\CurrentControlSet\Control\Terminal Server" /v fDenyTSConnections /t REG_DWORD /d 0 /f
(gcim -Class Win32_TerminalServiceSetting -Namespace root\CMV2\TerminalServices).SetAllowTSConnections(0) включить RDP (для Windows Server)
Get-Service TermService | Restart-Service -Force перезапустить rdp-службу
New-NetFirewallRule -Profile Any -DisplayName "RDP 3390" -Direction Inbound -Protocol TCP -LocalPort 3390 открыть RDP-порт
```

## IPBan

```
auditpol /get /category:* отобразить все политики аудита
auditpol /get /category:Вход/выход отобразить локальные политики аудита для Входа и Выхода из системы
auditpol /set /subcategory:"Вход в систему" /success:enable /failure:enable включить локальные политики - Аудит входа в систему
auditpol /set /subcategory:"Выход из системы" /success:enable /failure:enable

$url = $($([Invoke-RestMethod https://api.github.com/repos/DigitalRuby/IPBan/releases/latest].assets | Where-Object name -match ".+win.+x64+").brow
получить ссылку для загрузки последней версии
$version = $($([Invoke-RestMethod https://api.github.com/repos/DigitalRuby/IPBan/releases/latest].tag_name) получить номер последней версии
$path = "$home\Documents\ipban-$version" путь для установки
Invoke-RestMethod $url -OutFile "$home\Downloads\IPBan-$version.zip" скачать дистрибутив
Expand-Archive "$home\Downloads\ipban-$version.zip" -DestinationPath $path разархивировать в путь для установки
Remove-Item "$home\Downloads\ipban-$version.zip" удалить дистрибутив
sc create IPBan type=own start=delayed-auto binPath="$path\DigitalRuby.IPBan.exe" DisplayName=IPBan создать службу
Get-Service IPBan статус службы
$conf = $(Get-Content "$path\ipban.config") читаем конфигурацию
$conf = $conf -replace '<add key="Whitelist" value="">','<add key="Whitelist" value="192.168.3.0/24">' добавить в белый лист домашнюю сеть для исключения
$conf = $conf -replace '<add key="ProcessInternalIPAddresses" value="false"/>','<add key="ProcessInternalIPAddresses" value="true"/>' включить обработку локальных (внутренних) ip-адресов
$conf = $conf -replace '<add key="FailedLoginAttemptsBeforeBanUserNameWhitelist" value="20"/>','<add key="FailedLoginAttemptsBeforeBanUserNameWhite указать количество попыток подключения до блокировки
$conf = $conf -replace '<add key="ExpireTime" value="01:00:00:00"/>','<add key="ExpireTime" value="00:01:00:00"/>' задать время блокировки 1 час
$conf > "$path\ipban.config" обновить конфигурацию
Get-Service IPBan | Start-Service запустить службу

Get-NetFirewallRule | Where-Object DisplayName -Match "IPBan" | ForEach-Object {
    $Name = $_.DisplayName
    Get-NetFirewallAddressFilter -AssociatedNetFirewallRule $_ | Select-Object @{Name="Name"; Expression={$Name}},LocalIP,RemoteIP
} # отобразить область применения правил Брандмауэра для IPBan

Get-Content -Wait "$path\logfile.txt" читать лог
Get-Service IPBan | Stop-Service остановить службу
sc delete IPBan удалить службу
```

## shutdown

```
shutdown /r /o перезагрузка в безопасный режим
shutdown /s /t 600 /c "Power off after 10 minutes" ВЫКЛЮЧЕНИЕ
shutdown /s /f принудительное закрытие приложений
shutdown /a ОТМЕНА
shutdown /r /t 0 /m \\192.168.3.100
Restart-Computer -ComputerName 192.168.3.100 -Protocol WSMAN через WinRM
Restart-Computer -ComputerName 192.168.3.100 -Force через WMI
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\PolicyManager\default\Start\HideShutdown" -Name "value" -Value 1 скрыть кнопку выключения
Set-ItemProperty -Path "HKLM:\SOFTWARE\Microsoft\PolicyManager\default\Start\HideRestart" -Name "value" -Value 1 скрыть кнопку перезагрузки
```

```
function Start-Shutdown {
    <#
    .SYNOPSIS
    Module for shutdown and restart the computer at a specified time
    .DESCRIPTION
    Example:
    # Start-Shutdown -Time "18:00"
    # Start-Shutdown -Restart -Time "18:00"
    # Start-Shutdown -Cancel
    .LINK
    https://github.com/Lifailon/PS-Commands
    #>
    param(
        [string]$Time,
        [switch]$Restart,
        [switch]$Cancel
    )
    if ($time) {
        $currentDateTime = Get-Date
        $shutdownTime = Get-Date $Time
        if ($shutdownTime -lt $currentDateTime) {
            $shutdownTime = $shutdownTime.AddDays(1)
        }
        $timeUntilShutdown = $shutdownTime - $currentDateTime
        $secondsUntilShutdown = [math]::Round($timeUntilShutdown.TotalSeconds)
    }
    if ($Cancel) {
        Start-Process -FilePath "shutdown.exe" -ArgumentList "/a"
    } elseif ($Restart) {
        Write-Host "The computer will restart after $($timeUntilShutdown.Hours) hours and $($timeUntilShutdown.Minutes) minutes."
        Start-Process -FilePath "shutdown.exe" -ArgumentList "/r", "/f", "/t", "$secondsUntilShutdown"
    } else {
        Write-Host "The computer will shutdown after $($timeUntilShutdown.Hours) hours and $($timeUntilShutdown.Minutes) minutes."
        Start-Process -FilePath "shutdown.exe" -ArgumentList "/s", "/f", "/t", "$secondsUntilShutdown"
    }
}
```

## LocalAccounts

```
Get-Command -Module Microsoft.PowerShell.LocalAccounts
Get-LocalUser СПИСОК пользователей
Get-LocalGroup СПИСОК групп
New-LocalUser "1C" -Password $Password -FullName "1C Domain" создать пользователя
Set-LocalUser -Password $Password 1C ИЗМЕНИТЬ пароль
Add-LocalGroupMember -Group "Administrators" -Member "1C" добавить в группу Администраторов
Get-LocalGroupMember "Administrators" члены группы
```

```

@("vproxy-01","vproxy-02","vproxy-03") | %{
    icm $_ {Add-LocalGroupMember -Group "Administrators" -Member "support4"}
    icm $_ {Get-LocalGroupMember "Administrators"}
}

```

## SMB

```

Get-SmbServerConfiguration
Set-SmbServerConfiguration -EnableSMB1Protocol $false -Force ОТКЛЮЧИТЬ протокол SMB v1
Get-WindowsFeature | Where-Object {$_.name -eq "FS-SMB1"} | ft Name,Installstate модуль ServerManager, проверить установлен ли компонент SMB1
Install-WindowsFeature FS-SMB1 установить SMB1
Uninstall-WindowsFeature -Name FS-SMB1 -Remove удалить SMB1 клиента (понадобится перезагрузка)
Get-WindowsOptionalFeature -Online модуль DISM, для работы с компонентами Windows
Disable-WindowsOptionalFeature -Online -FeatureName SMB1Protocol -Remove удалить SMB1
Set-SmbServerConfiguration -AuditSmb1Access $true ВКЛЮЧИТЬ аудит SMB1
Get-SmbConnection СПИСОК активных сессий и используемая версия SMB (Dialect)
Get-SmbOpenFile | select ClientUserName,ClientComputerName,Path,SessionID СПИСОК открытых файлов
Get-SmbShare список сетевых папок
New-SmbShare -Name xl-share -Path E:\test создать новую общую сетевую папку (расшарить)
-EncryptData $True ВКЛЮЧИТЬ шифрование SMB
-Description ИМЯ в сетевом окружении
-ReadAccess "domain\username" доступ на чтение
-ChangeAccess доступ на запись
-FullAccess ПОЛНЫЙ доступ
-NoAccess ALL нет прав
-FolderEnumerationMode [AccessBased | Unrestricted] позволяет скрыть в сетевой папке объекты, на которых у пользователя нет доступа с помощью Access-Based Enumeration (ABE)
Get-SmbShare xl-share | Set-SmbShare -FolderEnumerationMode AccessBased ключить ABE для всех расшаренных папок
Remove-SmbShare xl-share -force удалить сетевой доступ (шару)
Get-SmbShareAccess xl-share вывести список доступов безопасности к шаре
Revoke-SmbShareAccess xl-share -AccountName Everyone -Force удалить группу из списка доступов
Grant-SmbShareAccess -Name xl-share -AccountName "domain\XL-Share" -AccessRight Change -force изменить/добавить разрешения на запись (Full,Read)
Grant-SmbShareAccess -Name xl-share -AccountName "все" -AccessRight Change -force
Block-SmbShareAccess -Name xl-share -AccountName "domain\noAccess" -Force принудительный запрет
New-SmbMapping -LocalPath X: -RemotePath \\$srv\xl-share -UserName support4 -Password password -Persistent $true ПОДКЛЮЧИТЬ сетевой диск
-Persistent Восстановление соединения после отключения компьютера или сети
-SaveCredential позволяет сохранить учетные данные пользователя для подключения в диспетчер учетных данных Windows Credential Manager
Stop-Process -Name "explorer" | Start-Process -FilePath "C:\Windows\explorer.exe" перезапустить процесс для отображения в проводнике
Get-SmbMapping список подключенных сетевых дисков
Remove-SmbMapping X: -force ОТМОНТИРОВАТЬ сетевой диск
$CIMSession = New-CIMSession -Computername $srv создать сеанс CIM (аутентификация на SMB)
Get-SmbOpenFile -CIMSession $CIMSession | select ClientUserName,ClientComputerName,Path | Out-GridView -PassThru | Close-SmbOpenFile -CIMSession $C закрыть файлы (открыть к ним сетевой доступ)

```

## Get-Acl

```

(Get-Acl \\$srv\xl-share).access доступ ACL на уровне NTFS
Get-Acl C:\Drivers | Set-Acl C:\Distr скопировать NTFS разрешения с одной папки и применить их на другую

```

## NTFSSecurity

```

Install-Module -Name NTFSSecurity -force
Get-Item "\\$srv\xl-share" | Get-NTFSAccess
Add-NTFSAccess -Path "\\$srv\xl-share" -Account "domain\xl-share" -AccessRights Fullcontrol -PassThru добавить
Remove-NTFSAccess -Path "\\$srv\xl-share" -Account "domain\xl-share" -AccessRights FullControl -PassThru удалить

```

```
Get-ChildItem -Path "\$srv\xl-share" -Recurse -Force | Clear-NTFSAccess удалить все разрешения, без удаления унаследованных разрешений  
Get-ChildItem -Path "\$srv\xl-share" -Recurse -Force | Enable-NTFSAccessInheritance включить NTFS наследование для всех объектов в каталоге
```

## Storage

```
Get-Command -Module Storage  
Get-Disk СПИСОК ЛОГИЧЕСКИХ ДИСКОВ  
Get-Partition отобразить разделы на всех дисках  
Get-Volume СПИСОК ЛОГИЧЕСКИХ РАЗДЕЛОВ  
Get-PhysicalDisk СПИСОК ФИЗИЧЕСКИХ ДИСКОВ  
Initialize-Disk 1 -PartitionStyle MBR инициализировать диск  
New-Partition -DriveLetter D -DiskNumber 1 -Size 500gb создать раздел (выделить все место -UseMaximumSize)  
Format-Volume -DriveLetter D -FileSystem NTFS -NewFileSystemLabel Disk-D форматировать раздел  
Set-Partition -DriveLetter D -IsActive $True сделать активным  
Remove-Partition -DriveLetter D -DiskNumber 1 удалить раздел  
Clear-Disk -Number 1 -RemoveData очистить диск  
Repair-Volume -driveletter C -Scan Check disk  
Repair-Volume -driveletter C -SpotFix  
Repair-Volume -driverletter C -Scan -CimSession $CIMSession
```

## iSCSI

```
New-IscsiVirtualDisk -Path D:\iSCSIVirtualDisks\iSCSI2.vhdx -Size 20GB создать динамический vhdx-диск (для фиксированного размера -UseFixed)  
New-IscsiServerTarget -TargetName iscsi-target-2 -InitiatorIds "IQN:iqn.1991-05.com.microsoft:srv3.contoso.com" создать Target  
Get-IscsiServerTarget | fl TargetName, LunMappings  
Connect-IscsiTarget -NodeAddress "iqn.1995-05.com.microsoft:srv2-iscsi-target-2-target" -IsPersistent $true подключиться инициатором к таргету  
Get-IscsiTarget | fl  
Disconnect-IscsiTarget -NodeAddress "iqn.1995-05.com.microsoft:srv2-iscsi-target-2-target" -Confirm:$false отключиться
```

## WSUS

```
Get-Hotfix | Sort-Object -Descending InstalledOn список установленных обновлений (информация из simv2)  
Get-Hotfix -Description "Security update"  
Get-CimInstance Win32_QuickFixEngineering  
Get-Service uhssvc служба Microsoft Health Update Tools, которая отвечает за предоставление обновлений
```

## WindowsUpdate

```
Get-Command -Module WindowsUpdate  
Get-WindowsUpdateLog формирует отчет в $home\AppData\Local\Temp\WindowsUpdateLog в формате csv
```

## PSWindowsUpdate

```
Install-Module -Name PSWindowsUpdate -Scope CurrentUser  
Import-Module PSWindowsUpdate  
Get-Command -Module PSWindowsUpdate  
Get-WindowsUpdate СПИСОК обновлений для скачать и установить с сервера WSUS или Microsoft Update  
Get-WindowsUpdate -Download загрузить все обновления  
Get-WindowsUpdate -Install установить все обновления  
Install-WindowsUpdate -MicrosoftUpdate -AcceptAll -IgnoreReboot установить все обновления без перезагрузки  
Get-WindowsUpdate -KBArticleID KB2267602, KB4533002 -Install  
Get-WindowsUpdate -KBArticleID KB2538243 -Hide скрыть обновления, что бы они никогда не устанавливались  
Get-WindowsUpdate -IsHidden отобразить скрытые обновления (Hide-WindowsUpdate)  
Remove-WindowsUpdate -KBArticleID KB4011634 -NoRestart удалить обновление  
Uninstall-WindowsUpdate удалить обновление  
Add-WUServiceManager регистрация сервера обновления (Windows Update Service Manager)  
Enable-WURemoting включить правила Windows Defender, разрешающие удаленное использование командлета PSWindowsUpdate  
Get-WUApiVersion версия Windows Update Agent
```

```
Get-WUHistory список всех установленных обновлений (история обновлений)
Get-WUHistory | Where-Object {$_.Title -match "KB4517389"} поиск обновления
Get-WULastResults даты последнего поиска и установки обновлений
Get-WURebootStatus проверить, нужна ли перезагрузка для применения конкретного обновления
Get-WUServiceManager выводит источники обновлений
Get-WUInstallerStatus статус службы Windows Installer
Remove-WUServiceManager отключить Windows Update Service Manager
```

## UpdateServices

```
Install-WindowsFeature -Name UpdateServices-RSAT установить роль UpdateServices
$updateScope = New-Object Microsoft.UpdateServices.Administration.UpdateScope
[enum]::GetValues([Microsoft.UpdateServices.Administration.ApprovedStates]) список утвержденных состояний
[enum]::GetValues([Microsoft.UpdateServices.Administration.UpdateInstallationStates]) статус установки (неизвестный, непригодный, не установлен, скачено, установлен, неуспешный, установлен и ожидает перезагрузки, все)
$updateScope.ApprovedStates = [Microsoft.UpdateServices.Administration.ApprovedStates]"NotApproved" выставляем статус не утвержденных обновлений на сервере WSUS
$updateScope.IncludedInstallationStates = [Microsoft.UpdateServices.Administration.UpdateInstallationStates]"NotInstalled" выставляем статус не установленных обновлений
$updateScope.IncludedInstallationStates = [Microsoft.UpdateServices.Administration.UpdateInstallationStates]"NotInstalled,Downloaded" обновления загружены, но не установлены
```

## PoshWSUS

```
Install-Module -Name PoshWSUS
Get-Command -Module PoshWSUS нужны права администратора
Add-PSWSUSClientToGroup добавление клиента в группу
Approve-PSWSUSUpdate утверждение обновления
Connect-PSWSUSDatabaseServer подключение к серверу базы данных WSUS
Connect-PSWSUSServer подключение к серверу WSUS
Deny-PSWSUSUpdate отклонение обновления
Disconnect-PSWSUSServer отключение от сервера WSUS
Export-PSWSUSMetaData экспорт метаданных WSUS
Get-PoshWSUSSyncUpdateCategories получение категорий обновлений для синхронизации
Get-PoshWSUSSyncUpdateClassifications получение классификаций обновлений для синхронизации
Get-PSWSUSCategory получение категории WSUS
Get-PSWSUSChildServer получение дочернего сервера WSUS
Get-PSWSUSClassification получение классификации WSUS
Get-PSWSUSClient получение информации о клиенте
Get-PSWSUSClientGroupMembership получение групповой принадлежности клиента
Get-PSWSUSClientPerUpdate получение информации о клиентах по обновлениям
Get-PSWSUSClientsInGroup получение клиентов в группе
Get-PSWSUSCommand получение информации о командах WSUS
Get-PSWSUSConfig получение конфигурации WSUS
Get-PSWSUSConfigEnabledUpdateLanguages получение включенных языков обновлений
Get-PSWSUSConfigProxyServer получение конфигурации прокси-сервера
Get-PSWSUSConfigSupportedUpdateLanguages получение поддерживаемых языков обновлений
Get-PSWSUSConfigSyncSchedule получение расписания синхронизации
Get-PSWSUSConfigSyncUpdateCategories получение категорий обновлений для синхронизации
Get-PSWSUSConfigSyncUpdateClassifications получение классификаций обновлений для синхронизации
Get-PSWSUSConfigUpdateFiles получение конфигурации файлов обновлений
Get-PSWSUSConfigUpdateSource получение источника обновлений
Get-PSWSUSConfiguration получение полной конфигурации WSUS
Get-PSWSUSContentDownloadProgress Получение прогресса загрузки контента
Get-PSWSUSCurrentUserRole получение роли текущего пользователя
Get-PSWSUSDatabaseConfig получение конфигурации базы данных
Get-PSWSUSDownstreamServer получение конфигурации нижестоящего сервера
Get-PSWSUSEmailConfig Получение конфигурации электронной почты
```

Get-PSWSUSEnabledUpdateLanguages получение включенных языков обновлений  
Get-PSWSUSEvent получение событий WSUS  
Get-PSWSUSGroup получение информации о группе  
Get-PSWSUSInstallableItem получение информации об установленных элементах  
Get-PSWSUSInstallApprovalRule получение правил утверждения установки  
Get-PSWSUSProxyServer получение информации о прокси-сервере  
Get-PSWSUSServer получение информации о сервере WSUS  
Get-PSWSUSStatus получение статуса WSUS  
Get-PSWSUSSubscription получение подписки WSUS  
Get-PSWSUSSupportedUpdateLanguages получение поддерживаемых языков обновлений  
Get-PSWSUSSyncEvent получение событий синхронизации  
Get-PSWSUSSyncHistory получение истории синхронизаций  
Get-PSWSUSSyncProgress получение прогресса синхронизации  
Get-PSWSUSSyncSchedule получение расписания синхронизации  
Get-PSWSUSUpdate получение информации об обновлении  
Get-PSWSUSUpdateApproval получение информации об утверждении обновлений  
Get-PSWSUSUpdateCategory получение категории обновлений  
Get-PSWSUSUpdateClassification получение классификации обновлений  
Get-PSWSUSUpdateFile получение файлов обновлений  
Get-PSWSUSUpdateFiles получение списка файлов обновлений  
Get-PSWSUSUpdatePerClient получение информации об обновлениях по клиентам  
Get-PSWSUSUpdateSource получение источника обновлений  
Get-PSWSUSUpdateSummary получение сводки по обновлениям  
Get-PSWSUSUpdateSummaryPerClient получение сводки по обновлениям для каждого клиента  
Get-PSWSUSUpdateSummaryPerGroup получение сводки по обновлениям для каждой группы  
Import-PSWSUSMetaData импорт метаданных WSUS  
New-PSWSUSComputerScope создание области охвата компьютеров  
New-PSWSUSGroup создание новой группы  
New-PSWSUSInstallApprovalRule создание правила утверждения установки  
New-PSWSUSUpdateScope создание области охвата обновлений  
Remove-PSWSUSClient удаление клиента  
Remove-PSWSUSClientFromGroup удаление клиента из группы  
Remove-PSWSUSGroup удаление группы  
Remove-PSWSUSInstallApprovalRule удаление правила утверждения установки  
Remove-PSWSUSUpdate удаление обновления  
Reset-PSWSUSContent сброс контента WSUS  
Resume-PSWSUSDownload возобновление загрузки  
Resume-PSWSUSUpdateDownload возобновление загрузки обновлений  
Set-PoshWsusClassification установка классификации WSUS  
Set-PoshWsusProduct установка продукта WSUS  
Set-PSWSUSConfigEnabledUpdateLanguages установка включенных языков обновлений  
Set-PSWSUSConfigProduct установка продукта конфигурации WSUS  
Set-PSWSusConfigProxyServer установка прокси-сервера конфигурации WSUS  
Set-PSWSUSConfigSyncSchedule установка расписания синхронизации  
Set-PSWSUSConfigTargetingMode установка режима таргетирования  
Set-PSWSUSConfigUpdateClassification установка классификации обновлений  
Set-PSWSUSConfigUpdateFiles установка файлов обновлений  
Set-PSWSUSConfigUpdateSource установка источника обновлений  
Set-PSWSUSEmailConfig установка конфигурации электронной почты  
Set-PSWSUSEnabledUpdateLanguages установка включенных языков обновлений  
Set-PSWSUSInstallApprovalRule установка правила утверждения установки  
Set-PSWSUSProxyServer установка прокси-сервера  
Set-PSWSUSSyncSchedule установка расписания синхронизации  
Set-PSWSUSTargetingMode установка режима таргетирования  
Set-PSWSUSUpdateFiles установка файлов обновлений  
Set-PSWSUSUpdateSource установка источника обновлений  
Start-PSWSUSCleanup запуск очистки WSUS

```
Start-PSWSUSInstallApprovalRule запуск правила утверждения установки
Start-PSWSUSSync запуск синхронизации
Stop-PSWSUSDownload остановка загрузки
Stop-PSWSUSSync остановка синхронизации
Stop-PSWSUSUpdateDownload остановка загрузки обновлений
Test-PSWSUSDatabaseServerConnection тестирование подключения к серверу базы данных
```

## ActiveDirectory

### RSAT

Remote Server Administration Tools

```
DISM.exe /Online /add-capability /CapabilityName:Rsat.ActiveDirectory.DS-LDS.Tools~~~0.0.1.0 /CapabilityName:Rsat.GroupPolicy.Management.Tools~~~
Add-WindowsCapability -online -Name Rsat.Dns.Tools~~~0.0.1.0
Add-WindowsCapability -online -Name Rsat.DHCP.Tools~~~0.0.1.0
Add-WindowsCapability -online -Name Rsat.FileServices.Tools~~~0.0.1.0
Add-WindowsCapability -online -Name Rsat.Wsus.Tools~~~0.0.1.0
Add-WindowsCapability -online -Name Rsat.CertificateServices.Tools~~~0.0.1.0
Add-WindowsCapability -online -Name Rsat.RemoteDesktop.Services.Tools~~~0.0.1.0
Get-WindowsCapability -Name RSAT* -Online | Select-Object -Property DisplayName, State отобразить список установленных компонентов
```

### Import-Module AD

```
$session = New-PSSession -ComputerName $srv -Credential cred 'Export – PSsession – SessionSession -Module ActiveDirectory -OutputModule
ActiveDirectory экспортировать модуль из удаленной сессии (например, с DC) \ Remove-PSSession -Session $Session \ Import-Module
ActiveDirectory \ Get-Command -Module ActiveDirectory'
```

### ADSI

Active Directory Service Interface

```
$d0 = $env:userdnsdomain
$d0 = $d0 -split "."
$d1 = $d0[0]
$d2 = $d0[1]
$group = [ADSI]"LDAP://OU=Domain Controllers,DC=$d1,DC=$d2"
$group | select *

$Local_User = [ADSI]"WinNT://./Администратор,user"
$Local_User | Get-Member
$Local_User.Description
$Local_User.LastLogin Время последней авторизации локального пользователя
```

### LDAP

Lightweight Directory Access Protocol

```
$ldapsearcher = New-Object System.DirectoryServices.DirectorySearcher
$ldapsearcher.SearchRoot = "LDAP://OU=Domain Controllers,DC=$d1,DC=$d2"
$ldapsearcher.Filter = "(objectclass=computer)"
$dc = $ldapsearcher.FindAll().path

$usr = $env:username СПИСОК групп текущего пользователя
$ldapsearcher = New-Object System.DirectoryServices.DirectorySearcher
$ldapsearcher.Filter = "(&(objectCategory=User)(samAccountName=$usr))"
$usrfind = $ldapsearcher.FindOne()
$groups = $usrfind.properties.memberof -replace "(,OU=.)"
$groups = $groups -replace "(CN=)"
```

DC (Domain Component) - компонент доменного имени

OU (Organizational Unit) - организационные подразделения (type), используются для упорядочения объектов

Container - так же используется для упорядочения объектов, контейнеры в отличии от подразделений не могут быть переименованы, удалены, созданы или связаны с объектом групповой политики (Computers, Domain Controllers, Users)

DN (Distinguished Name) — уникальное имя объекта и местоположение в лесу AD. В DN описывается содержимое атрибутов в дереве (путь навигации), требуемое для доступа к конкретной записи или ее поиска

CN (Common Name) - общее имя

```
(Get-ADObject (Get-ADRootDSE).DefaultNamingContext -Properties wellKnownObjects).wellKnownObjects отобразить отобразить контейнеры по умолчанию
```

```
redircmp OU=Client Computers,DC=root,DC=domain,DC=local ИЗМЕНИТЬ контейнер компьютеров по умолчанию
```

```
redirusr ИЗМЕНИТЬ контейнер пользователей по умолчанию
```

## LAPS

Local Admin Password Management

```
Import-module AdmPwd.ps импортировать модуль
```

```
Get-AdmPwdPassword -ComputerName NAME посмотреть пароль
```

```
Reset-AdmPwdPassword -ComputerName NAME изменить пароль
```

```
Get-ADComputer -Filter * -SearchBase "DC=$d1,DC=$d2" | Get-AdmPwdPassword -ComputerName {$_.Name} | select ComputerName,Password,ExpirationTimestamp
```

```
Get-ADComputer -Identity $srv | Get-AdmPwdPassword -ComputerName {$_.Name} | select ComputerName,Password,ExpirationTimestamp
```

## Recycle Bin

Удаленные объекты хранятся в корзине AD в течении времени захоронения (определяется в атрибуте домена msDS-deletedObjectLifetime), заданном для леса. По умолчанию это 180 дней. Если данный срок прошел, объект все еще остается в контейнере Deleted Objects, но большинство его атрибутов и связей очищаются (Recycled Object). После истечения периода tombstoneLifetime (по умолчанию также 180 дней, но можно увеличить) объект полностью удаляется из AD автоматическим процессом очистки.

```
Get-ADForest domain.local отобразить уровень работы леса
```

```
Set-ADForestMode -Identity domain.local -ForestMode Windows2008R2Forest -force увеличить уровень работы леса
```

```
Enable-ADOptionalFeature -Identity "CN=Recycle Bin Feature,CN=Optional Features,CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration,DC=включить корзину
```

```
Get-ADOptionalFeature "Recycle Bin Feature" | select-object name,EnabledScopes если значение EnabledScopes не пустое, значит в домене корзина Active Directory включена
```

```
Get-ADObject -Filter 'Name -like "*tnas*" -IncludeDeletedObjects найти удаленную (Deleted: True) УЗ (ObjectClass: user) в AD
```

```
Get-ADObject -Filter 'Name -like "*tnas*" -IncludeDeletedObjects -Properties * | select-object Name, sAMAccountName, LastKnownParent, memberOf, IsD проверить значение атрибута IsDeleted, контейнер, в котором находился пользователе перед удалением (LastKnownParent) и список групп, в которых он состоял
```

```
Get-ADObject -filter {Deleted -eq $True -and ObjectClass -eq "user"} -includeDeletedObjects вывести список удаленных пользователей
```

```
Restore-ADObject -Identity "3dc33c7c-b912-4a19-b1b7-415c1395a34e" восстановить по значению атрибута ObjectGUID
```

```
Get-ADObject -Filter 'SAMAccountName -eq "tnas-01' -IncludeDeletedObjects | Restore-ADObject восстановить по SAMAccountName
```

```
Get-ADObject -Filter {Deleted -eq $True -and ObjectClass -eq 'group' -and Name -like '*Allow*'} -IncludeDeletedObjects | Restore-ADObject -Verbose восстановить группу или компьютер
```

## thumbnailPhoto

```
$photo = [byte[]](Get-Content C:\Install\adm.jpg -Encoding byte) преобразовать файл картинки в массив байтов (jpeg/bmp файл, размером фото до 100 Кб и разрешением 96?96)
```

```
Set-ADUser support4 -Replace @{thumbnailPhoto=$photo} задать значение атрибута thumbnailPhoto
```

## ADDomainController

```
Get-ADDominController выводит информацию о текущем контроллере домена (LogonServer), который используется данным компьютером для аутентификации (DC выбирается при загрузке в соответствии с топологией сайтов AD)
```

```
Get-ADDominController -Discover -Service PrimaryDC найти контроллер с ролью PDC в домене
```

```
Get-ADDominController -Filter * | ft HostName,IPv4Address,Name,Site,OperatingSystem,IsGlobalCatalog список все DC, принадлежность к сайту, версии ОС и GC
```

При загрузке ОС служба NetLogon делает DNS запрос со списком контроллеров домена (к SRV записи \_ldap.\_tcp.dc.\_msdcs.domain), DNS возвращает список DC в домене с записью Service Location (SRV). Клиент делает LDAP запрос к DC для определения сайта AD по своему IP адресу. Клиент через DNS запрашивает список контроллеров домена в сайте (в разделе \_tcp.sitename.\_sites...).

USN (Update Sequence Numbers) - счетчик номера последовательного обновления, который существует у каждого объекта AD. При репликации контроллеры обмениваются значениями USN, объект с более низким USN будет при репликации перезаписан объектом с более высоким USN. Находится в свойствах - Object (включить View - Advanced Features). Каждый контроллер домена содержит отдельный счетчик USN, который начинает отсчет в момент запуска процесса Dcpromo и продолжает увеличивать значения в течение всего времени существования контроллера домена. Значение счетчика USN увеличивается каждый раз, когда на контроллере домена происходит транзакция, это операции создания, обновления или удаления объекта.

```
Get-ADDomainController -Filter * | % { отобразить USN объекта на всех DC в домене \ Get-ADUser -Server $_.HostName -Identity support4 -Properties uSNChanged | select SamAccountName,uSNChanged \ }
```

dcpromo /forceremoval принудительно выполнит понижение в роли контроллера домена до уровня рядового сервера. После понижения роли выполняется удаление всех ссылок в домене на этот контроллер. Далее производит включение сервера в состав домена, и выполнение обратного процесса, т.е. повышение сервера до уровня контроллера домена.

## ADComputer

```
nlttest /DSGETDC:$env:userdnsdomain узнать на каком DC аутентифицирован хост (Logon Server)
nlttest /SC_RESET:$env:userdnsdomain\srvc-dc2.$env:userdnsdomain переключить компьютер на другой контроллер домена AD вручную (The command completed successfully)
Get-ADComputer -Identity $env:computername -Properties PasswordLastSet время последней смены пароля на сервере
Test-ComputerSecureChannel -verbose проверить доверительные отношения с доменом (соответствует ли локальный пароль компьютера паролю, хранящемуся в AD)
Reset-ComputerMachinePassword -Credential domain\admin принудительно обновить пароль
Netdom ResetPWD /Server:dc-01 /UserD:domain\admin /PasswordD:* сбросить хэш пароля компьютера в домене (перезагрузка не требуется)
Search-ADAccount -AccountDisabled -ComputersOnly | select Name,LastLogonDate,Enabled отобразить все отключенные компьютеры

Get-ADComputer -Filter * -Properties * | select name СПИСОК всех компьютеров в домене (Filter), вывести все свойства (Properties)
Get-ADComputer -Identity $rv -Properties * | ft Name,LastLogonDate,PasswordLastSet,ms-Mcs-AdmPwd -Autosize конкретного компьютера в AD (Identity)
Get-ADComputer -SearchBase "OU=Domain Controllers,DC=$d1,DC=$d2" -Filter * -Properties * | ft Name, LastLogonDate, distinguishedName -Autosize поиск в базе по DN (SearchBase)

(Get-ADComputer -Filter {enabled -eq "true"}).count получить общее количество активных (незаблокированных) компьютеров
(Get-ADComputer -Filter {enabled -eq "true" -and OperatingSystem -like "*Windows Server 2016*").count кол-во активных компьютеров с ОС WS 2016

Get-ADComputer -Filter * -Properties * | select @{Label="Ping Status"; Expression={
$ping = ping -n 1 -w 50 $_.Name
if ($ping -match "TTL") {"Online"} else {"Offline"}
}},
@{Label="Status"; Expression={
if ($.Enabled -eq "True") {$_.Enabled -replace "True","Active"} else {$_.Enabled -replace "False","Blocked"}
}}, Name, IPv4Address, OperatingSystem, @{Label="UserOwner"; Expression={$_.ManagedBy -replace "(CN=|,.)+"}}
},Created | Out-GridView
```

## ADUser

```
Get-ADUser -Identity support4 -Properties * СПИСОК всех атрибутов
Get-ADUser -Identity support4 -Properties DistinguishedName, EmailAddress, Description путь DN, email и описание
Get-ADUser -Filter {(Enabled -eq "True") -and (mail -ne "null")} -Properties mail | ft Name,mail СПИСОК активных пользователей и есть почтовый ящик
Get-ADUser -Filter {SamAccountName -like "*"} | Measure-Object посчитать кол-во всех аккаунтов (Count)
Get-ADUser -Filter * -Properties WhenCreated | sort WhenCreated | ft Name, whenCreated дата создания
Get-ADUser -Identity support4 -property LockedOut | select samaccountName,Name,Enabled,Lockedout
Enabled=True учетная запись включена - да
Lockedout=False учетная запись заблокирована (например, политикой паролей) - нет
```

```

Get-ADUser -Identity support4 | Unlock-ADAccount разблокировать учетную запись
Disable-ADAccount -Identity support4 отключить учетную запись
Enable-ADAccount -Identity support4 включить учетную запись
Search-ADAccount -LockedOut найти все заблокированные учетные записи
Search-ADAccount -AccountDisabled | select Name,LastLogonDate,Enabled отобразить все отключенные учетные записи с временем последнего входа

Get-ADUser -Identity support4 -Properties PasswordLastSet,PasswordExpired,PasswordNeverExpires
PasswordLastSet время последней смены пароля
PasswordExpired=False пароль истек - нет
PasswordNeverExpires=True срок действия пароля не истекает - да
Set-ADAccountPassword support4 -Reset -NewPassword (ConvertTo-SecureString -AsPlainText "password" -Force -Verbose) изменить пароль учетной записи
Set-ADUser -Identity support4 -ChangePasswordAtLogon $True смена пароля при следующем входе в систему

$day = (Get-Date).AddDays(-90)
Get-ADUser -filter { (passwordlastset -le $day) } | ft пользователи, которые не меняли свой пароль больше 90 дней

$day = (Get-Date).AddDays(-30)
Get-ADUser -filter { (Created -ge $day) } -Property Created | select Name,Created Новые пользователи за 30 дней

$day = (Get-Date).AddDays(-360)
Get-ADUser -Filter { (LastLogonTimestamp -le $day) } -Property LastLogonTimestamp | select Name, SamAccountName, @{n='LastLogonTimestamp';e={[DateTime] пользователи, которые не логинились больше 360 дней. Репликация атрибута LastLogonTimestamp составляет от 9 до 14 дней.

| Disable-ADAccount $_.SamAccountName заблокировать
-WhatIf отобразить вывод без применения изменений

```

## ADGroupMember

```

(Get-ADUser -Identity support4 -Properties MemberOf).MemberOf список групп в которых состоит пользователь
Get-ADGroupMember -Identity "Domain Admins" | Select Name, SamAccountName список пользователей в группе
Add-ADGroupMember -Identity "Domain Admins" -Members support5 добавить в группу
Remove-ADGroupMember -Identity "Domain Admins" -Members support5 -force удалить из группы
Get-ADGroup -filter * | where {!($_.Get-ADGroupMember)} | Select Name отобразить список пустых групп (-Not)

```

## ADReplication

```

Get-Command -Module ActiveDirectory -Name *Replication* список всех командлетов модуля
Get-ADReplicationFailure -Target dc-01 список ошибок репликации с партнерами
Get-ADReplicationFailure -Target $env:userdnsdomain -Scope Domain
Get-ADReplicationPartnerMetadata -Target dc-01 | select Partner, LastReplicationAttempt, LastReplicationSuccess, LastReplicationResult, LastChangeUsn
время последней и время успешной репликации с партнерами
Get-ADReplicationUpToDateNumberVectorTable -Target dc-01 Update Sequence Number (USN) увеличивается каждый раз, когда на контроллере домена происходит транзакция (операции создания, обновления или удаления объекта), при репликации DC обмениваются значениями USN, объект с более низким USN при репликации будет перезаписан высоким USN.

```

## repadmin

```

repadmin /replicsummary отображает время последней репликации на всех DC по направлению (Source и Destination) и их состояние с учетом партнеров
repadmin /showrep1 $srv отображает всех партнеров по репликации и их статус для всех разделов Naming Contexts (DC=ForestDnsZones, DC=DomainDnsZones, CN=Schema, CN=Configuration)
repadmin /replicate $srv2 $srv1 DC=domain,DC=local выполнить репликацию с srv1 на srv2 только указанный раздела домена
repadmin /SyncAll /AdeP запустить межсайтовую исходящую репликацию всех разделов от текущего сервера со всеми партнерами по репликации /A выполнить для всех разделов NC
/d в сообщениях идентифицировать серверы по DN (вместо GUID DNS - глобальным уникальным идентификаторам)
/e межсайтовая синхронизация (по умолчанию синхронизирует только с DC текущего сайта)
/p извещать об изменениях с этого сервера (по умолчанию: опрашивать об изменениях)
repadmin /Queue $srv отображает кол-во запросов входящей репликации (очередь), которое необходимо обработать (причиной может быть

```

большое кол-во партнеров или формирование 1000 объектов скриптом)

repadmin /showbackup \* узнать дату последнего Backup

Error: 1722 сервер rpc недоступен (ошибка отката репликации). Проверить имя домена в настройках сетевого адаптера, первым должен идти адрес DNS-сервера другого контроллера домена, вторым свой адрес.

Get-Service -ComputerName \$srv | select name,status | ? name -like "RpcSs"

Get-Service -ComputerName \$srv -Name RpcSs -RequiredServices зависимые службы

Зависимые службы RPC:

"Служба сведений о подключенных сетях" - должен быть включен отложенный запуск. Если служба срабатывает до "службы списка сетей", может падать связь с доменом (netlogon)

"Центр распространения ключей Kerberos"

"DNS-сервер"

nslookup \$srv

tnc \$srv -p 135

repadmin /retry повторить попытку привязки к целевому DC, если была ошибка 1722 или 1753 (RPC недоступен)

repadmin /showrepl \$srv

Last attempt @ 2022-07-15 10:46:01 завершена с ошибкой, результат 8456 (0x2108) при проверки showrepl этого партнера, его ошибка: 8457 (0x2109)

Last success @ 2022-07-11 02:29:46 последний успех

Когда репликация автоматически отключена, ОС записывает в DSA - not writable одно из четырех значений:

Path: HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Services\NTDS\Parameters

Dsa Not Writable

#define DSA\_WRITABLE\_GEN 1 версия леса несовместима с ОС

#define DSA\_WRITABLE\_NO\_SPACE 2 на диске, где размещена база данных Active Directory или файлы журналов (логи), недостаточно свободного места

#define DSA\_WRITABLE\_USNROLLBCK 4 откат USN произошел из-за неправильного отката базы данных Active Directory во времени (восстановление из снапшота)

#define DSA\_WRITABLE\_CORRUPT\_UTDV 8 вектор актуальности поврежден на локальном контроллере домена

## dcdiag

dcdiag /s:<DomainController> [/n:<NamingContext>] [[/u:<domain\user>] [/p:<password>]] [{/a|/e}{/q|/v}] [/f:<LogFile>] [/ferr:<ErrorLog>] [/test:<t

dcdiag /Test:replications /s:dc-01 отображает ошибки репликации

dcdiag /Test:DNS /e /v /q тест DNS

/a проверка всех серверов данного сайта

/e проверка всех серверов предприятия

/q выводить только сообщения об ошибках

/v выводить подробную информацию

/fix автоматически исправляет ошибки

/test:

NetLogons проверка наличие прав на выполнение репликации

Connectivity проверяет регистрацию DNS для каждого контроллера домена, отправляет тестовый эхо-пакет на каждый контроллер домена и проверяет подключение по протоколам LDAP и RPC к каждому контроллеру домена

Services проверяет работоспособность всех служб, необходимых для работы контроллера домена, на указанном контроллере домена

SystemLog проверяет наличие ошибок в журналах контроллера домена

FRSEvent проверяет ошибки репликации в работе службы репликации файлов, что может означать наличие проблем в репликации SYSVOL и, таким образом, целостности копий объектов групповых политик

FSMOCHECK не проверяет роли хозяев операций, а вместо этого запрашивает сервер глобального каталога, первый контроллер домена, предпочтительный сервер времени, сервер времени и центр распространения ключей (контроллер домена может подключиться к KDC, PDC, серверу глобального каталога)

KnowsOfRoleHolders проверяет возможность подключения контроллеров домена ко всем пяти хозяевам операций (ролями FSMO)

MachineAccount проверяет правильность регистрации учетной записи целевого компьютера и правильность объявлений служб этого компьютера (корректность доверительных отношений с доменом). Если обнаружена ошибка, ее можно исправить с помощью утилиты dcdiag, указав параметры /fixmachineaccount или /createmachineaccount

Advertising проверяет, правильно ли контроллер домена сообщает о себе и о своей роли хозяина операций. Этот тест завершится неудачно, если служба NetLogon не запущена

CheckSDRefDom проверяет правильность доменов ссылок дескрипторов безопасности для каждого раздела каталогов программ

CrossRefValidation проверяет правильность перекрестных ссылок для доменов  
RSSysvol проверяет состояние готовности для FRS SYSVOL  
Intersite проверяет наличие ошибок, которые могут помешать нормальной репликации между сайтами. Компания Microsoft предупреждает, что иногда результаты этого теста могут оказаться неточными  
KCCEvent проверяет безошибочность создания объектов соединений для репликации между сайтами  
NCSCDesc проверяет правильность разрешений для репликации в дескрипторах безопасности для заголовков контекста именования  
ObjectsReplicated проверяет правильность репликации агента сервера каталогов и объектов учетных записей компьютеров  
OutboundSecureChannels проверяется наличие безопасных каналов между всеми контроллерами домена в интересующем домене  
Replications проверяет возможность репликации между контроллерами домена и сообщает обо всех ошибках при репликации  
RidManager проверяет работоспособность и доступность хозяина относительных идентификаторов  
VerifyEnterpriseReferences проверяет действительность системных ссылок службы репликации файлов для всех объектов на всех контроллерах домена в лесу  
VerifyReferences проверяет действительность системных ссылок службы репликации файлов для всех объектов на указанном контроллере домена  
VerifyReplicas проверяет действительность всех разделов каталога приложения на всех серверах, принимающих участие в репликации

## ntdsutil

Перенос БД AD (ntds.dit):

```
Get-Acl C:\Windows\NTDS | Set-Acl D:\AD-DB скопировать NTFS разрешения на новый каталог
Stop-Service -ComputerName dc -name NTDS остановить службу Active Directory Domain Services
ntdsutil запустить утилиту ntdsutil
activate instance NTDS выбрать активный экземпляр базы AD
files перейдем в контекст files, в котором возможно выполнение операция с файлами базы ntds.dit
move DB to D:\AD-DB\ перенести базу AD в новый каталог (предварительно нужно его создать)
info проверить, что БД находится в новом каталоге
move logs to D:\AD-DB\ переместим в тот же каталог файлы с журналами транзакций
quit
Start-Service -ComputerName dc -name NTDS
```

Сброс пароля DSRM (режим восстановления служб каталогов):

```
ntdsutil
set dsrm password
reset password on server NULL
новый пароль
подтверждение пароля
quit
quit
```

Синхронизовать с паролем УЗ в AD:

```
ntdsutil
set dsrm password
sync from domain account dsrmadmin
quit
quit
```

Ошибка 0x00002e2 при загрузке ОС.

Загрузиться в режиме восстановления WinRE (Windows Recovery Environment) - Startup Settings - Restart - DSRM (Directory Services Restore Mode) reagentc /boottore shutdown /f /r /o /t 0 перезагрузка в режиме WinRE - ОС на базе WinPE (Windows Preinstallation Environment), образ winre.wim находится на скрытом разделе System Restore

На контроллере домена единственная локальная учетная запись — администратор DSRM. Пароль создается при установке роли контроллера домена ADDS на сервере (SafeModeAdministratorPassword).

```
ntdsutil
activate instance ntds
Files
Info
integrity проверить целостность БД
Ошибка: Failed to open DIT for AD DS/LDS instance NTDS. Error -2147418113
```

```
mkdir c:\ntds_bak
xcopy c:\Windows\NTDS\*.* c:\ntds_bak backup содержимого каталога с БД
esentutl /g c:\windows\ntds\ntds.dit проверим целостность файла
Вывод: Integrity check completed. Database is CORRUPTED ошибка, база AD повреждена
esentutl /p c:\windows\ntds\ntds.dit исправить ошибки
Вывод: Operation completed successfully in xx seconds. нет ошибок
esentutl /g c:\windows\ntds\ntds.dit проверим целостность файла
Выполнить анализ семантики базы с помощью ntdsutil:
ntdsutil
activate instance ntds
semantic database analysis
go
go fixup исправить семантические ошибки
Сжать файл БД:
activate instance ntds
files
compact to C:\Windows\NTDS\TEMP
copy C:\Windows\NTDS\TEMP\ntds.dit C:\Windows\NTDS\ntds.dit заменить оригиналный файл ntds.dit
Del C:\Windows\NTDS\*.log удалить все лог файлы из каталога NTDS
```

## GPO

```
Get-Command -Module GroupPolicy
Get-GPO -Domain domain.local -All | ft
Get-GPO -Name LAPS
[xml](Get-GPReport LAPS -ReportType Xml)
Get-GPPermission -Name LAPS -All
Get-GPO LAPS | New-GPLink -Target "ou=servers,dc=domain,dc=local"
Set-GPLink -Name LAPS -Target "ou=servers,dc=domain,dc=local" -LinkEnabled No
Backup-GPO -Name LAPS -Path "$home\Desktop"
Backup-GPO -All -Path "$home\Desktop"
Restore-GPO -Name LAPS -Path C:\Backup\GPOs\
```

## ServerManager

```
Get-Command *WindowsFeature* source module ServerManager
Get-WindowsFeature -ComputerName "localhost"
Get-WindowsFeature | where Installed -eq $True список установленных ролей и компонентов
Get-WindowsFeature | where FeatureType -eq "Role" отсортировать по списку ролей
Install-WindowsFeature -Name DNS установить роль
Get-Command *DNS*
Get-DnsServerSetting -ALL
Uninstall-WindowsFeature -Name DNS удалить роль
```

## PSWA

```
Install-WindowsFeature -Name WindowsPowerShellWebAccess -IncludeManagementTools
Install-PswaWebApplication -UseTestCertificate Создать веб-приложение /pswa
Add-PswaAuthorizationRule -UserGroupName "$domain\Domain Admins" -ComputerName * -ConfigurationName * -RuleName "For Admins" добавить права
авторизации
```

## WSB (Windows Server Backup)

При создании backup DC через WSB, создается копия состояния системы (System State), куда попадает база AD (NTDS.DIT), объекты групповых политик, содержимое каталога SYSVOL, реестр, метаданные IIS, база AD CS, и другие системные файлы и ресурсы. Резервная копия создается через службу теневого копирования VSS.

```
Get-WindowsFeature Windows-Server-Backup ПРОВЕРИТЬ установлена ли роль  
Add-WindowsFeature Windows-Server-Backup -Includeallsubfeature УСТАНОВИТЬ роль
```

```
$path="\\$srv\bak-dc\dc-03\"  
[string]$TargetUNC=$path+(get-date -f 'yyyy-MM-dd')  
if ((Test-Path -Path $path) -eq $true) {New-Item -Path $TargetUNC -ItemType directory} # если путь доступен, создать новую директорию по дате  
$WBadmin_cmd = "wbadm.in.exe START BACKUP -backupTarget:$TargetUNC -systemState -noverify -vssCopy -quiet"  
# $WBadmin_cmd = "wbadm in start backup -backuptarget:$path -include:C:\Windows\NTDS\ntds.dit -quiet" # Backup DB NTDS  
Invoke-Expression $WBadmin_cmd
```

## RDS

```
Get-Command -Module RemoteDesktop  
Get-RDServer -ConnectionBroker $broker СПИСОК всех серверов в ферме, указывается полное доменное имя при обращение к серверу с ролью  
RDCB  
Get-RDRemoteDesktop -ConnectionBroker $broker СПИСОК коллекций  
(Get-RDLicenseConfiguration -ConnectionBroker $broker | select *).LicenseServer СПИСОК серверов с ролью RDL  
Get-RDUserSession -ConnectionBroker $broker СПИСОК всех активных пользователей  
Disconnect-RDUser -HostServer $srv -UnifiedSessionID $id -Force ОТКЛЮЧИТЬ сессию пользователя  
Get-RDAvailableApp -ConnectionBroker $broker -CollectionName C03 СПИСОК установленного ПО на серверах в коллекции  
(Get-RDSessionCollectionConfiguration -ConnectionBroker $broker -CollectionName C03 | select *).CustomRdpProperty use redirection server name:i:1  
Get-RDConnectionBrokerHighAvailability
```

## DNSServer

```
Get-Command -Module DnsServer  
Show-DnsServerCache отобразить весь кэш DNS-сервера  
Show-DnsServerCache | where HostName -match ru  
Clear-DnsServerCache  
Get-DnsServerCache  
Get-DnsServerDiagnostics  
  
$zone = icm $srv {Get-DnsServerZone} | select ZoneName,ZoneType,DynamicUpdate,ReplicationScope,SecureSecondaries,  
DirectoryPartitionName | Out-GridView -Title "DNS Server: $srv" -PassThru  
$zone_name = $zone.ZoneName  
if ($zone_name -ne $null) {  
    icm $srv {  
        Get-DnsServerResourceRecord -ZoneName $using:zone_name | sort RecordType | select RecordType,HostName, @{  
            Label="IPAddress"; Expression={$_ . RecordData.IPv4Address.ToString}} ,TimeToLive, Timestamp  
    } | select RecordType,HostName,IPAddress,TimeToLive, Timestamp | Out-GridView -Title "DNS Server: $srv"  
}  
  
Sync-DnsServerZone -passthru СИНХРОНИЗИРОВАТЬ зоны с другими DC в домене  
Remove-DnsServerZone -Name domain.local УДАЛИТЬ зону  
Get-DnsServerResourceRecord -ZoneName domain.local -RRType A ВЫВЕСТИ все A-записи в указанной зоне  
Add-DnsServerResourceRecordA -Name new-host-name -IPv4Address 192.168.1.100 -ZoneName domain.local -TimeToLive 01:00:00 -CreatePtr СОЗДАТЬ A-  
запись и PTR для нее  
Remove-DnsServerResourceRecord -ZoneName domain.local -RRType A -Name new-host-name -Force УДАЛИТЬ A-запись
```

```

$DNSServer = "DC-01"
$DNSFZone = "domain.com"
$DataFile = "C:\Scripts\DNS-Create-A-Records-from-File.csv"
# cat $DataFile
# "HostName;IP"
# "server-01;192.168.1.10"
$DNSRR = [WmiClass]"\"$DNSServer\root\MicrosoftDNS:MicrosoftDNS_ResourceRecord"
$ConvFile = $DataFile + "_unicode"
Get-Content $DataFile | Set-Content $ConvFile -Encoding Unicode
Import-Csv $ConvFile -Delimiter ";" | ForEach-Object {
    $FQDN = $_.HostName + "." + $DNSFZone
    $IP = $_.HostIP
    $TextA = "$FQDN IN A $IP"
    [Void]$DNSRR.CreateInstanceFromTextRepresentation($DNSServer,$DNSFZone,$TextA)
}

```

## DHCPServer

```

Get-Command -Module DhcpServer

$mac = icm $srv -ScriptBlock {Get-DhcpServerv4Scope | Get-DhcpServerv4Lease} | select AddressState,
HostName,IPAddress,ClientId,DnsRegistration,DnsRR,ScopeId,ServerIP | Out-GridView -Title "HDCP Server: $srv" -PassThru
(New-Object -ComObject Wscript.Shell).Popup($mac.ClientId,0,$mac.Hostname,64)

Add-DhcpServerv4Reservation -ScopeId 192.168.1.0 -IPAddress 192.168.1.10 -ClientId 00-50-56-C0-00-08 -Description "new reservation"

```

## DFS

dfsutil /root:\\domain.sys\public /export:C:\export-dfs.txt экспорт конфигурации namespace root  
 dfsutil /AddFtRoot /Server:\\\$srv /Share:public на новой машине предварительно создать корень на основе домена  
 dfsutil /root:\\domain.sys\public /import:C:\export-dfs.txt /<verify /set Import (перед импортом данных в существующий корень DFS, утилита создает резервную копию конфигурации корня в текущем каталоге, из которого запускается утилита dfsutil)  
 /verify выводит изменения, которые будут внесены в процессе импорта, без применения  
 /set меняет целевое пространство имен путем полной перезаписи и замены на конфигурацию пространства имен из импортируемого файла  
 /merge импортирует конфигурацию пространства имен в дополнение к существующей конфигурации для слияния, параметры из файла конфигурации будут иметь больший приоритет, чем существующие параметры пространства имен

Export-DfsrClone экспортирует клонированную базу данных репликации DFS и параметры конфигурации тома

Get-DfsrCloneState получает состояние операции клонирования базы данных

Import-DfsrClone импортирует клонированную базу данных репликации DFS и параметры конфигурации тома

```

net use x: \\$srv1\public\* примонтировать диск
Get-DfsrFileHash x:\* | Out-File C:\\$rv1.txt забрать hash всех файлов диска в файл (файлы с одинаковыми хешами всегда являются точными
копиями друг друга)
net use x: /d отмонтировать
net use x: \\$rv2\public\*
Get-DfsrFileHash x:\* | Out-File C:\\$rv2.txt
net use x: /d
Compare-Object -ReferenceObject (Get-Content C:\\$rv1.txt) -DifferenceObject (Get-Content C:\\$rv2.txt) -IncludeEqual сравнивать содержимое
файлов

```

Get-DfsrBacklog -DestinationComputerName "fs-06" -SourceComputerName "fs-05" -GroupName "folder-rep" -FolderName "folder" -Verbose получает список ожидающих обновлений файлов между двумя партнерами репликации DFS

Get-DfsrConnection отображает группы репликации, участников и статус

Get-DfsReplicatedFolder отображает имя и полный путь к папкам репликации в системе DFS

Get-DfsrState -ComputerName fs-06 -Verbose состояние репликации DFS для члена группы

Get-DfsReplicationGroup отображает группы репликации и их статус

Add-DfsrConnection создает соединение между членами группы репликации

Add-DfsrMember добавляет компьютеры в группу репликации  
ConvertFrom-DfsrGuid преобразует идентификаторы GUID в понятные имена в заданной группе репликации  
Get-DfsrConnectionSchedule получает расписание соединений между членами группы репликации  
Get-DfsrGroupSchedule извлекает расписание группы репликации  
Get-DfsrIdRecord получает записи ID для реплицированных файлов или папок из базы данных репликации DFS  
Get-DfsrMember получает компьютеры в группе репликации  
Get-DfsrMembership получает параметры членства для членов групп репликации  
Get-DfsrPreservedFiles получает список файлов и папок, ранее сохраненных репликацией DFS  
Get-DfsrServiceConfiguration получает параметры службы репликации DFS для членов группы  
Grant-DfsrDelegation предоставляет разрешения участникам безопасности для группы репликации  
Revoke-DfsrDelegation отменяет разрешения участников безопасности для группы репликации  
New-DfsReplicationGroup создает группу репликации  
New-DfsReplicatedFolder создает реплицированную папку в группе репликации  
Remove-DfsrConnection удаляет соединение между членами группы репликации  
Remove-DfsReplicatedFolder удаляет реплицированную папку из группы репликации  
Remove-DfsReplicationGroup удаляет группу репликации  
Remove-DfsrMember удаляет компьютеры из группы репликации  
Restore-DfsrPreservedFiles восстанавливает файлы и папки, ранее сохраненные репликацией DFS  
Set-DfsrConnection изменяет параметры соединения между членами группы репликации  
Set-DfsrConnectionSchedule изменяет параметры расписания соединений между членами группы репликации  
Set-DfsReplicatedFolder изменяет настройки реплицированной папки  
Set-DfsReplicationGroup изменяет группу репликации  
Set-DfsrGroupSchedule изменяет расписание группы репликации  
Set-DfsrMember изменяет информацию о компьютере-участнике в группе репликации  
Set-DfsrMembership настраивает параметры членства для членов группы репликации  
Set-DfsrServiceConfiguration изменяет параметры службы репликации DFS  
Sync-DfsReplicationGroup синхронизирует репликацию между компьютерами независимо от расписания  
Suspend-DfsReplicationGroup приостанавливает репликацию между компьютерами независимо от расписания  
Update-DfsrConfigurationFromAD инициирует обновление службы репликации DFS  
Write-DfsrHealthReport создает отчет о работоспособности репликации DFS  
Write-DfsrPropagationReport создает отчеты для тестовых файлов распространения в группе репликации  
Start-DfsrPropagationTest создает тестовый файл распространения в реплицированной папке

## StorageReplica

```
Install-WindowsFeature Storage-Replica -IncludeManagementTools -Restart
Get-Command -Module StorageReplica
Test-SRTopology проверить соответствует ли сервер и канал связи технологии Storage Replica
New-SRPartnership -SourceComputerName srv-01 -SourceRGName srv-01-rep-group-01 -SourceVolumeName D: -SourceLogVolumeName L: -DestinationComputerName
Get-Counter -Counter "\Storage Replica Statistics(*)"
Get-WinEvent -ProviderName Microsoft-Windows-StorageReplica -max 10
Set-SRPartnership -ReplicationMode Asynchronous переключить режим репликации на асинхронный
Set-SRPartnership -NewSourceComputerName srv-02 -SourceRGName srv-02-rep-group-01 -DestinationComputerName srv-01 -DestinationRGName srv-01-rep-group-01
изменить вручную направление репликации данных, переведя вторичную копию в онлайн режим (при выходе из строя основного сервера)
Get-SRGroup Информация о состоянии группы репликации
Get-SRPartnerShip информация о направлении репликации
(Get-SRGroup).Replicas | Select-Object numofbytesremaining проверить длину очереди копирования
Get-SRPartnership | Remove-SRPartnership удалить репликацию на основном сервере
Get-SRGroup | Remove-SRGroup удалить репликацию на обоих серверах
```

## PS2EXE

```
Install-Module ps2exe -Repository PSGallery
Get-Module -ListAvailable список всех модулей
-noConsole использовать GUI, без окна консоли powershell
-noOutput выполнение в фоне
```

```

-noError без вывода ошибок
-requireAdmin при запуске запросить права администратора
-credentialGUI вывод диалогового окна для ввода учетных данных
Invoke-ps2exe -inputFile "$home\Desktop\WinEvent-Viewer-1.1.ps1" -outputFile "$home\Desktop\WEV-1.1.exe" -iconFile "$home\Desktop\log_48px.ico" -ti

```

## NSSM

```

$powershell_Path = (Get-Command powershell).Source
$NSSM_Path = (Get-Command "C:\WinPerf-Agent\nssm-2.24.exe").Source
$Script_Path = "C:\WinPerf-Agent\WinPerf-Agent-1.1.ps1"
$Service_Name = "WinPerf-Agent"

& $NSSM_Path install $Service_Name $powershell_Path -ExecutionPolicy Bypass -NoProfile -f $Script_Path создать Service
& $NSSM_Path start $Service_Name запустить
& $NSSM_Path status $Service_Name статус
$Service_Name | Restart-Service перезапустить
$Service_Name | Get-Service статус
$Service_Name | Stop-Service остановить
& $NSSM_Path set $Service_Name description "Check performance CPU and report email" изменить описание
& $NSSM_Path remove $Service_Name удалить

```

## Jobs

```

Get-Job получение списка задач
Start-Job запуск процесса
Stop-Job остановка процесса
Suspend-Job Приостановка работы процесса
Resume-Job Восстановление работы процесса
Wait-Job Ожидание вывода команды
Receive-Job Получение результатов выполненного процесса
Remove-Job удалить задачу

```

```

function Start-PingJob ($Network) {
    $RNetwork = $Network -replace "\.\d{1,3}\$","."
    foreach ($4 in 1..254) {
        $ip = $RNetwork+$4
        # Создаем задания, забираем 3-ю строку вывода и добавляем к выводу ip-адрес
        (Start-Job {"$using:ip : "+(ping -n 1 -w 50 $using:ip)[2]}) | Out-Null
    }
    while ($True) {
        $status_job = $(Get-Job).State[-1] # забираем статус последнего задания (задания выполняются по очереди сверху вниз)
        if ($status_job -like "Completed") { # проверяем задание на выполнение
            $ping_out = Get-Job | Receive-Job # если выполнено, забираем вывод всех заданий
            Get-Job | Remove-Job -Force # удаляем задания
            break # завершаем цикл
        }
    }
    $ping_out
}

Start-PingJob -Network 192.168.3.0
$(Measure-Command {Start-PingJob -Network 192.168.3.0}).TotalSeconds 60 Seconds

```

## ThreadJob

```

Install-Module -Name ThreadJob
Get-Module ThreadJob -list
Start-ThreadJob {ping ya.ru} | Out-Null создать фоновую задачу
Get-Job | Receive-Job -Keep отобразить и не удалять вывод

```

```
(Get-Job).HasMoreData если False, то вывод команды удален  
(Get-Job)[-1].Output отобразить вывод последней задачи
```

```
function Start-PingThreadJob ($Network) {  
    $RNetwork = $Network -replace "\.\d{1,3}\$","."  
    foreach ($4 in 1..254) {  
        $ip = $RNetwork+$4  
        $(Start-ThreadJob {  
            "$using:ip : " + $(ping -n 1 -w 50 $using:ip)[2]  
        }) | Out-Null  
    }  
    while ($True) {  
        $status_job = $(Get-Job).State[-1]  
        if ($status_job -like "Completed") {  
            $ping_out = Get-Job | Receive-Job  
            Get-Job | Remove-Job -Force  
            break  
        }  
    }  
    $ping_out  
}
```

```
Start-PingThreadJob -Network 192.168.3.0  
$(Measure-Command {Start-PingThread -Network 192.168.3.0}).TotalSeconds 24 Seconds
```

## PoshRSJob

```
Install-Module -Name PoshRSJob
```

```
function Start-PingRSJob ($Network) {  
    $RNetwork = $Network -replace "\.\d{1,3}\$","."  
    foreach ($4 in 1..254) {  
        $ip = $RNetwork+$4  
        $(Start-RSJob {  
            "$using:ip : " + $(ping -n 1 -w 50 $using:ip)[2]  
        }) | Out-Null  
    }  
    while ($True) {  
        $status_job = $(Get-RSJob).State -notcontains "Running" # проверяем, что массив не содержит активных заданий  
        if ($status_job) {  
            $ping_out = Get-RSJob | Receive-RSJob  
            Get-RSJob | Remove-RSJob  
            break  
        }  
    }  
    $ping_out  
}
```

```
Start-PingRSJob -Network 192.168.3.0  
$(Measure-Command {Start-PingRSJob -Network 192.168.3.0}).TotalSeconds 10 Seconds
```

## Invoke-Parallel

```
# Import function from GitHub to current session  
$module = "https://raw.githubusercontent.com/RamblingCookieMonster/Invoke-Parallel/master/Invoke-Parallel/Invoke-Parallel.ps1"  
Invoke-Expression $(Invoke-WebRequest $module)
```

```
Get-Help Invoke-Parallel -Full
```

```

function Start-PingInvokeParallel ($Network) {
    $RNetwork = $Network -replace "\.\d{1,3}\$","."
    1..254 | ForEach-Object {$srvList += @($RNetwork+$_)}
    Invoke-Parallel -InputObject $srvList -ScriptBlock {
        "$_ : " + $(ping -n 1 -w 50 $_)[2]
    }
}

Start-PingInvokeParallel -Network 192.168.3.0
$(Get-History)[-1].Duration.TotalSeconds 7 seconds

$array_main = 1..10 | ForEach-Object {"192.168.3.$_"}
Invoke-Parallel -InputObject $($0..($array_main.Count-1)) -ScriptBlock {
    Foreach ($n in 1..100) {
        Start-Sleep -Milliseconds 100
        Write-Progress -Activity $($array_main[$_]) -PercentComplete $n -id $_
    }
} -ImportVariables

```

## ForEach-Object-Parallel

```

function Start-PingParallel ($Network) {
    $RNetwork = $Network -replace "\.\d{1,3}\$","."
    1..254 | ForEach-Object -Parallel {
        "$using:RNetwork.$_ : " + $(ping -n 1 -w 50 "$using:RNetwork$_")[2]
    } -ThrottleLimit 254
}

```

```

Start-PingParallel -Network 192.168.3.0
$(Get-History)[-1].Duration.TotalSeconds 2 seconds

```

```

function Start-TestConnectParallel (
    $Network,
    [switch]$Csv
) {
    $RNetwork = $Network -replace "\.\d{1,3}\$","."
    if ($csv) {
        "Address,Status,Latency"
    }
    1..254 | ForEach-Object -Parallel {
        $test = Test-Connection "$using:RNetwork$_" -Count 1 -TimeoutSeconds 1
        if ($using:csv) {
            "$($using:RNetwork)$_, $($test.Status), $($test.Latency)"
        } else {
            $test
        }
    } -ThrottleLimit 254
}

```

```

Start-TestConnectParallel -Network 192.168.3.0 -Csv | ConvertFrom-Csv
$(Get-History)[-1].Duration.TotalSeconds 3 seconds

```

```

$array_main = 1..10 | ForEach-Object {"192.168.3.$_"}
$($0..($array_main.Count-1)) | ForEach-Object -Parallel {
    Foreach ($n in 1..100) {
        Start-Sleep -Milliseconds 100
        $array_temp = $using$array_main
        Write-Progress -Activity $($array_temp[$_]) -PercentComplete $n -id $_
    }
} -ThrottleLimit $($array_main.Count)

```

# SMTP

## Net.Mail

```
function Send-SMTP {
    param (
        [Parameter(Mandatory = $True)]$mess
    )
    $srv_smtp = "smtp.yandex.ru"
    $port = "587"
    $from = "login1@yandex.ru"
    $to = "login2@yandex.ru"
    $user = "login1"
    $pass = "password"
    $subject = "PowerShell"
    $Message = New-Object System.Net.Mail.MailMessage
    $Message.From = $from
    $Message.To.Add($to)
    $Message.Subject = $subject
    $Message.IsBodyHTML = $true
    $Message.Body = "<h1> $mess </h1>"
    $smtp = New-Object Net.Mail.SmtpClient($srv_smtp, $port)
    $smtp.EnableSSL = $true
    $smtp.Credentials = New-Object System.Net.NetworkCredential($user, $pass);
    $smtp.Send($Message)
}

Send-SMTP "This is a test email from PowerShell"
```

## SMTP over OpenSSL

```
# Получить логин и пароль в формате Base64
echo -n "fromUserName@yandex.ru" | base64 # ZnJvbVVzZXJOYW1lQHlhbmr1eC5ydQ==
echo -n "app-password" | base64 # YXBwLXBhc3N3b3Jk
# Подключаемся к серверу через OpenSSL и авторизуемся
openssl s_client -connect smtp.yandex.ru:465 -crlf -quiet
# 220 Ok
EHLO hv-dev-101
AUTH LOGIN
ZnJvbVVzZXJOYW1lQHlhbmr1eC5ydQ==
YXBwLXBhc3N3b3Jk
# 235 Authentication successful
# Отправляем письмо
MAIL FROM:<fromUserName@yandex.ru>
RCPT TO:<toUserName@yandex.ru>
DATA
Subject: OpenSSL
# Отделить тему от тела письма пустой строкой
This is a test email from OpenSSL
.
# 250 Ok
```

## Swaks

[Swaks](#) - SMTP клиент на Perl

```

swaks --from fromUserName@yandex.ru \
--to toUserName@yandex.ru \
--server smtp.yandex.ru \
--port 587 \
--auth LOGIN \
--auth-user fromUserName@yandex.ru \
--auth-password "app-password" \
--tls \
--header "Subject: Test Subject" \
--body "This is the body test from swaks"

```

## VMWare/PowerCLI

```

Install-Module -Name VMware.PowerCLI # -AllowClobber установить модуль (PackageProvider: nuget)
Get-Module -ListAvailable VMware* | Select Name,Version
Import-Module VMware.VimAutomation.Core ИМПОРТИРОВАТЬ в сессию
Get-PSPowerShell | format-list Name,PSSnapIn,ModuleName СПИСОК оснасток Windows PowerShell

Get-PowerCLIConfiguration конфигурация подключения
Set-PowerCLIConfiguration -Scope AllUsers -InvalidCertificateAction ignore -confirm:$false если используется самоподписанный сертификат,
изменить значение параметра InvalidCertificateAction с Unset на Ignore/Warn
Set-PowerCLIConfiguration -Scope AllUsers -ParticipateInCeip $false отключить уведомление сбора данных через VMware Customer Experience
Improvement Program (CEIP)

Read-Host -AsSecureString | ConvertFrom-SecureString | Out-File "$home\Documents\vcsa_password.txt" зашифровать пароль и сохранить в файл
$esxi = "vcsa.domain.local"
$user = "administrator@vsphere.local"
$pass = Get-Content "$home\Documents\vcsa_password.txt" | ConvertTo-SecureString прочитать пароль
$pass = "password"
$Cred = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $user , $pass
Connect-VIServer $esxi -User $Cred.Username -Password $Cred.GetNetworkCredential().password подключиться, используя PSCredential (
$Cred) 'Connect – VIServer $esxi -User $user – Password $pass' подключиться, используя логин и пароль

Get-Command -Module *vmware* отобразить список команд модуля VMware
Get-Command -Module *vmware* -name *get*iscsi* найти команды модуля VMware, связанные с iSCSI
Get-IscsiHbaTarget получить цели iSCSI HBA (Host Bus Adapter)
Get-Datacenter получить список数据中心 в инфраструктуре VMware
Get-Cluster список кластеров
Get-VMHost список хостов виртуальных машин (ESXi хостов)
Get-VMHost | Select-Object Name,Model,ProcessorType,MaxEVCMode,NumCpu,CpuTotalMhz,CpuUsageMhz,MemoryTotalGB,MemoryUsageGB получить информацию
о хостах виртуальных машин, включая имя, модель, тип процессора и использование ресурсов
Get-VMHostDisk | Select-Object VMHost,ScsiLun>TotalSectors получить информацию о дисках хостов виртуальных машин, включая хост, SCSI LUN
и общее количество секторов

Get-Datastore список всех хранилищ данных
Get-Datastore TNAS-vmfs-4tb-01 получить информацию о хранилище данных с именем TNAS-vmfs-4tb-01
Get-Datastore TNAS-vmfs-4tb-01 | get-vm получить список виртуальных машин, которые используют хранилище данных
Get-Datastore -RelatedObject vm-01 получить информацию о хранилищах данных, связанных с виртуальной машиной
$(Get-Datastore TNAS-vmfs-4tb-01).ExtensionData.Info.GetType() получить тип информации о хранилище данных
$(Get-Datastore TNAS-vmfs-4tb-01).ExtensionData.Info.Vmfs.Extent получить информацию о том, на каких дисках находится хранилище данных

Get-Command -Module *vmware* -name *disk* найти команды модуля VMware, связанные с дисками
Get-VM vm-01 | Get-Datastore получить информацию о хранилищах данных, используемых виртуальной машиной
Get-VM vm-01 | Get-HardDisk получить информацию о подключенных дисках к виртуальной машине
Get-VM | Get-HardDisk | select Parent,Name,CapacityGB,StorageFormat,FileName | ft получить информацию о дисках всех виртуальных машин,
включая родительскую ВМ, имя, емкость, формат и имя файла
Copy-HardDisk скопировать жесткий диск виртуальной машины
Get-VM | Get-Snapshot список всех снимков виртуальных машин
Get-VM | where {$_.Powerstate -eq "PoweredOn"} список всех включенных виртуальных машин

```

```
Get-VMHost esxi-05 | Get-VM | where {$_._Powerstate -eq "PoweredOff"} | Move-VM -Destination (Get-VMHost esxi-06) переместить все выключенные виртуальные машины с хоста esxi-05 на хост esxi-06
```

```
Get-VM | select Name,VMHost,PowerState,NumCpu,MemoryGB,  
@{Name="UsedSpaceGB"; Expression={[int32]($_.UsedSpaceGB)}},  
@{Name="ProvisionedSpaceGB"; Expression={[int32]($_.ProvisionedSpaceGB)}},  
CreateDate,CpuHotAddEnabled,MemoryHotAddEnabled,CpuHotRemoveEnabled,Notes
```

```
Get-VMGuest vm-01 | Update-Tools обновить VMware Tools на виртуальной машине vm-01
```

```
Get-VMGuest vm-01 | select OSFullName,IPAddress,HostName,State,Disks,Nics,ToolsVersion получить информацию о гостевой операционной системе виртуальной машины vm-01 (имя ОС, IP-адрес, имя хоста, состояние, диски, сетевые адAPTERы и версию VMware Tools)
```

```
Get-VMGuest * | select -ExpandProperty IPAddress получить IP-адреса всех гостевых ОС виртуальных машин
```

```
Restart-VMGuest -vm vm-01 -Confirm:$False перезагрузить гостевую ОС виртуальной машины без запроса подтверждения
```

```
Start-VM -vm vm-01 -Confirm:$False ВКЛЮЧИТЬ виртуальную машину без запроса подтверждения
```

```
Shutdown-VMGuest -vm vm-01 -Confirm:$false ВЫКЛЮЧИТЬ
```

```
New-VM -Name vm-01 -VMHost esxi-06 -ResourcePool Production -DiskGB 60 -DiskStorageFormat Thin -Datastore TNAS-vmfs-4tb-01 создать новую виртуальную машину vm-01 на хосте esxi-06 в пуле ресурсов Production с диском 60 ГБ в формате Thin и размещением на хранилище данных TNAS-vmfs-4tb-01
```

```
Get-VM vm-01 | Copy-VMGuestFile -Source "\$srv\Install\Soft\Btest.exe" -Destination "C:\Install\" -LocalToGuest -GuestUser USER -GuestPassword PAS Скопировать файл с хоста на гостевую ОС виртуальной машины vm-01 с принудительным выполнением
```

```
Get-VM -name vm-01 | Export-VApp -Destination C:\Install -Format OVF экспортirовать виртуальную машину vm-01 в шаблон OVF (.ovf, .vmdk, .mf)  
Get-VM -name vm-01 | Export-VApp -Destination C:\Install -Format OVA экспортirовать виртуальную машину vm-01 в шаблон OVA
```

```
Get-VMHostNetworkAdapter | select VMHost,Name,Mac,IP,@{Label="Port Group"; Expression={$_._ExtensionData.Portgroup}} | ft получить информацию о сетевых адAPTERах хостов виртуальных машин, включая хост, имя, MAC-адрес, IP-адрес и портовую группу
```

```
Get-VM | Get-NetworkAdapter | select Parent,Name,Id,Type,MacAddress,ConnectionState,WakeOnLanEnabled | ft получить информацию о сетевых адAPTERах виртуальных машин, включая родительскую ВМ, имя, идентификатор, тип, MAC-адрес, состояние подключения и поддержку Wake-on-LAN
```

```
Get-Command -Module *vmware* -name *event*
```

```
Get-VIEvent -MaxSamples 1000 | where {($_._FullFormattedMessage -match "power")} | select username,CreatedTime,FullFormattedMessage получить последние 1000 событий, связанных с питанием, включая имя пользователя, время создания и полное сообщение
```

```
Get-logtype | select Key,SourceEntityId,Filename,Creator,Info получить информацию о типах логов
```

```
$(Get-Log vpxd:vpxd.log).Entries | select -Last 50 получить последние 50 записей из лога vpxd
```

```
Get-Command -Module *vmware* -name *syslog*
```

```
Set-VMHostSysLogServer -VMHost esxi-05 -SysLogServer "tcp://192.168.3.100" -SysLogServerPort 3515 установить сервер syslog сервер для хранения системных логов для хоста esxi-05
```

```
Get-VMHostSysLogServer -VMHost esxi-05
```

## Hyper-V

```
Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -Restart установить роль на Windows Server
```

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Hyper-V -All установить роль на Windows Desktop
```

```
Get-Command -Module hyper-v
```

```
Get-VMHost
```

```
New-VMSwitch -name NAT -SwitchType Internal # создать виртуальный коммутатор и адAPTER для него
```

```
Get-NetAdapter | where InterfaceDescription -match Hyper-V # список сетевых адAPTERов
```

```
New-NetNat -Name LocalNat -InternalIPInterfaceAddressPrefix "192.168.3.0/24" # задать сеть
```

```
Get-NetAdapter "vEthernet (NAT)" | New-NetIPAddress -IPAddress 192.168.3.200 -AddressFamily IPv4 -PrefixLength 24 # присвоить адрес, необходимо н
```

```
Add-NetNatStaticMapping -NatName LocalNat -Protocol TCP -ExternalIPAddress 0.0.0.0 -ExternalPort 2222 -InternalIPAddress 192.168.3.103 -InternalPort (Get-NetAdapter | where Name -match NAT).Status
```

```
Get-NetNatStaticMapping отобразить пробросы (NAT)
```

```
Get-NetNat СПИСОК сетей
```

```

Remove-NetNatStaticMapping -StaticMappingID 0 удалить проброс
Remove-NetNat -Name LocalNat удалить сеть

New-VMSwitch -Name Local -AllowManagementOS $True -NetAdapterName "Ethernet 4" -SwitchType External создать вшений (External) виртуальный
коммутатор

```

```

$VMName = "hv-dc-01"
$VM = @{
    Name = $VMName
    MemoryStartupBytes = 4Gb
    Generation = 2
    NewVHDPath = "D:\VM\$VMName\$VMName.vhdx"
    NewVHDSIZEBytes = 50Gb
    BootDevice = "VHD"
    Path = "D:\VM\$VMName"
    SwitchName = "NAT"
}

```

New-VM @VM создать виртуальную машину с параметрами

```

Set-VMDvdDrive -VMName $VMName -Path "C:\Users\Lifailon\Documents\WS-2016.iso" примонтировать образ
New-VHD -Path "D:\VM\$VMName\disk_d.vhdx" -SizeBytes 10GB создать VHDX диск
Add-VMHardDiskDrive -VMName $VMName -Path "D:\VM\$VMName\disk_d.vhdx" примонтировать диск
Get-VM -VMname $VMName | Set-VM -AutomaticStartAction Start автозапуск
Get-VM -Name $VMName | Set-VMMemory -StartupBytes 8Gb назначить стартовый размер оперативной памяти при запуске
Set-VMProcessor $VMName -Count 2 количество виртуальных процессоров (vCPU: ядер/потоков)
Set-VMProcessor $VMName -Count 2 -Maximum 4 -Reserve 50 -RelativeWeight 200 указать максимальное количество выделяемых процессоров,
резервируется 50% ресурсов процессора хоста и установить относительный вес 200 для приоритизации распределения ресурсов процессора
относительно других виртуальных машин
Get-VM -Name $VMName | Checkpoint-VM -SnapshotName "Snapshot-1" создать снапшот
Restore-VMCheckpoint -Name "Snapshot-1" -VMName $VMName -Confirm:$false восстановление из снапшота
Get-VM | Select -ExpandProperty NetworkAdapters | Select VMName,IPAddresses,Status получить IP адрес всех BM

```

## VMConnect via RDCMan

```

vmconnect.exe localhost $VMHost подключиться к виртуальной машине через VMConnect (используется в диспетчере Hyper-V)
Get-NetTCPConnection -State Established,Listen | Where-Object LocalPort -Match 2179 найти порт слушателя
Get-Process -Id (Get-NetTCPConnection -State Established,Listen | Where-Object LocalPort -Match 2179).OwningProcess найти процесс по ID
(vmms/VMConnect)
New-NetFirewallRule -Name "Hyper-V" -DisplayName "Hyper-V" -Group "Hyper-V" -Direction Inbound -Protocol TCP -LocalPort 2179 -Action Allow -Profile
открыть порт в локальном Firewall
Get-LocalGroupMember -Group "Администраторы Hyper-V"
Get-LocalGroupMember -Group "Hyper-V Administrators"
Add-LocalGroupMember -Group "Администраторы Hyper-V" -Member "lifailon" добавить пользователя в группу администраторов (для возможности
подключения)
Get-VM * | Select-Object Name,Id добавить id в RDCMan для подключения
Grant-VMConnectAccess -ComputerName plex-01 -VMName hv-devops-01 -UserName lifailon дать доступ на подключение не администратору
Get-VMConnectAccess
Revoke-VMConnectAccess -VMName hv-devops-01 -UserName lifailon забрать доступ
Error: Unknown disconnection reason 3848 - добавить ключи реестра на стороне клиента

```

```
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowFreshCredentialsDomain -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowDefaultCredentials -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowFreshCredentialsWhenNTLMOnlyDomain -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowDefaultCredentialsDomain -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowFreshCredentials -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowFreshCredentialsWhenNTLMOnly -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowSavedCredentialsWhenNTLMOnly -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowSavedCredentials -Name Hyper-V -PropertyType String
New-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Credssp\PolicyDefaults\AllowSavedCredentialsDomain -Name Hyper-V -PropertyType String
```

## Azure

```
Install-Module -Name Az -Scope CurrentUser -Repository PSGallery -Force установить все модули для работы с Azure
Get-Module *Az.* СПИСОК ВСЕХ МОДУЛЕЙ
```

```
Get-Command -Module Az.Accounts отобразить список команд модуля Az.Accounts
Connect-AzAccount подключиться у учетной записи Azure
Get-AzContext получить текущий статус подключения к Azure
Get-AzSubscription получить список подписок Azure, доступных для текущего пользователя
Set-AzContext установить контекст Azure для конкретной подписки и/или учетной записи
Disconnect-AzAccount отключиться от учетной записи Azure
```

```
Get-Command -Module Az.Compute
Get-AzVM получить список виртуальных машин в текущей подписке или группе ресурсов
Get-AzVMSize получить список доступных размеров виртуальных машин в определенном регионе
Get-AzVMImage получить список доступных образов виртуальных машин
New-AzVM создать новую виртуальную машину
Remove-AzVM удалить виртуальную машину
Start-AzVM запустить виртуальную машину
Stop-AzVM остановить виртуальную машину
Restart-AzVM перезагрузить виртуальную машину
```

```
Get-Command -Module Az.Network
Get-AzVirtualNetwork получить список виртуальных сетей в текущей подписке или группе ресурсов
New-AzVirtualNetwork создать новую виртуальную сеть
Remove-AzVirtualNetwork удалить виртуальную сеть
Get-AzNetworkInterface получить список сетевых интерфейсов
New-AzNetworkInterface создать новый сетевой интерфейс
Remove-AzNetworkInterface удалить сетевой интерфейс
```

```
Get-Command -Module Az.Storage
Get-AzStorageAccount получить список учетных записей хранилища
New-AzStorageAccount создать новую учетную запись хранилища
Remove-AzStorageAccount удалить учетную запись хранилища
Get-AzStorageContainer список контейнеров в учетной записи хранилища
New-AzStorageContainer создать новый контейнер в учетной записи хранилища
Remove-AzStorageContainer удалить контейнер
```

```
Get-Command -Module Az.ResourceManager
Get-AzResourceGroup получить список групп ресурсов в текущей подписке
New-AzResourceGroup создать новую группу ресурсов
Remove-AzResourceGroup удалить группу ресурсов
Get-AzResource получить список ресурсов
New-AzResource создать новый ресурс
Remove-AzResource удалить ресурс
```

```
Get-Command -Module Az.KeyVault
Get-AzKeyVault список хранилищ ключей
```

```
New-AzKeyVault создать новое хранилище ключей в Azure
```

```
Remove-AzKeyVault удалить хранилище ключей в Azure
```

```
Get-Command -Module Az.Identity
```

```
Get-AzADUser получить информацию о пользователях Azure Active Directory
```

```
New-AzADUser создать нового пользователя
```

```
Remove-AzADUser удалить пользователя
```

```
Get-AzADGroup получить информацию о группах
```

```
New-AzADGroup создать новую группу
```

```
Remove-AzADGroup удалить группу
```

## Manage-VM

Source: <https://learn.microsoft.com/ru-ru/azure/virtual-machines/windows/tutorial-manage-vm>

```
New-AzResourceGroup -Name "Resource-Group-01" -Location "EastUS" создать группу ресурсов (логический контейнер, в котором происходит развертывание ресурсов Azure)
```

```
Get-AzVMImageOffer -Location "EastUS" -PublisherName "MicrosoftWindowsServer" список доступных образов Windows Server для установки  
$cred = Get-Credential
```

```
New-AzVm -ResourceGroupName "Resource-Group-01" -Name "vm-01" -Location 'EastUS' -Image "MicrosoftWindowsServer:WindowsServer:2022-datacenter-azure  
создать виртуальную машину
```

```
Get-AzVM -ResourceGroupName "Resource-Group-01" -Name "vm-01" -Status | Select @n="Status"; e={$_.Statuses[1].Code} статус виртуальной  
машины
```

```
Start-AzVM -ResourceGroupName "Resource-Group-01" -Name "vm-01" запустить виртуальную машину
```

```
Stop-AzVM -ResourceGroupName "Resource-Group-01" -Name "vm-01" -Force остановить виртуальную машину
```

```
Invoke-AzVMRunCommand -ResourceGroupName "Resource-Group-01" -VMName "vm-01" -CommandId "RunPowerShellScript" -ScriptString "Install-WindowsFeature  
установить роль веб-сервера IIS
```

## Manage-Disk

Source: <https://learn.microsoft.com/ru-ru/azure/virtual-machines/windows/tutorial-manage-data-disk>

```
$diskConfig = New-AzDiskConfig -Location "EastUS" -CreateOption Empty -DiskSizeGB 512 -SkuName "Standard_LRS" создать диск на 512 ГБ
```

```
$dataDisk = New-AzDisk -ResourceGroupName "Resource-Group-01" -DiskName "disk-512" -Disk $diskConfig создание объекта диска для подготовки  
диска данных к работе
```

```
Get-AzDisk -ResourceGroupName "Resource-Group-01" -DiskName "disk-512" список дисков
```

```
$vm = Get-AzVM -ResourceGroupName "Resource-Group-01" -Name "vm-01"
```

```
Add-AzVMDataDisk -VM $vm -Name "Resource-Group-01" -CreateOption Attach -ManagedDiskId $dataDisk.Id -Lun 1 подключить диск к виртуальной  
машине
```

```
Update-AzVM -ResourceGroupName "Resource-Group-01" -VM $vm обновить конфигурацию виртуальной машины
```

```
Get-Disk | Where PartitionStyle -eq 'raw' | Initialize-Disk -PartitionStyle MBR -PassThru | New-Partition -AssignDriveLetter -UseMaximumSize | Format  
инициализировать диск в ОС (необходимо подключиться к виртуальной машине) с таблицей MBR, создать раздел и назначить все пространство  
и форматировать в файловую систему NTFS
```

## Exchange/EMShell

```
$srv_cas = "exchange-cas"
```

```
$session_exchange = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://$srv_cas/PowerShell/ -Credential $cred -  
Authentication Kerberos 'Get - PSSession' 'Import - PSSession session_exchange -DisableNameChecking' импортировать в текущую  
сессию
```

```
Get-ExchangeServer | select name,serverrole,admindisplayversion,Edition,OriginatingServer,WhenCreated,WhenChanged,DataPath | ft список всех  
серверов
```

```
Get-ImapSettings настройки IMAP
```

```
Get-ExchangeCertificate список сертификатов
```

```
Get-ExchangeCertificate -Thumbprint "5CEC8544D4743BC279E5FEA1679F79F5BD0C2B3A" | Enable-ExchangeCertificate -Services IMAP, POP, IIS, SMTP  
iisreset
```

```
Get-ClientAccessService | fl identity, *uri* настройки службы автообнаружения в Exchange 2016
```

```

Get-ClientAccessService -Identity $srv | Set-ClientAccessService -AutoDiscoverServiceInternalUri https://mail.domain.ru/Autodiscover/Autodiscover.x
изменить на внешний адрес
Get-OutlookAnywhere OA позволяет клиентам Outlook подключаться к своим почтовым ящикам за пределами локальной сети (без использования
VPN)
Get-WebServicesVirtualDirectory
Get-OwaVirtualDirectory
Get-ActiveSyncVirtualDirectory
Get-OabVirtualDirectory виртуальная директория автономной адресной книги
Get-OabVirtualDirectory -Server $srv | Set-OabVirtualDirectory -InternalUrl "https://mail.domain.ru/OAB" -ExternalUrl "https://mail.domain.ru/OAB"

```

## Roles

MS (Mailbox) - сервер с БД почтовых ящиков и общих папок, отвечает только за их размещение и не выполняет маршрутизацию никаких сообщений.

CAS (Client Access Server) - обработка клиентских подключений к почтовым ящикам, которые создаются клиентами Outlook Web Access (HTTP для Outlook Web App), Outlook Anywhere, ActiveSync (для мобильных устройств), интернет протоколы POP3 и IMAP4, MAPI для клиентов Microsoft Outlook.

Hub Transport - отвечает за маршрутизацию сообщений интернета и инфраструктурой Exchange, а также между серверами Exchange.

Сообщения всегда маршрутизируются с помощью роли транспортного сервера-концентратора, даже если почтовые ящики источника и назначения находятся в одной базе данных почтовых ящиков.

Relay - роль пограничного транспортного сервера (шлюз SMTP в периметре сети).

SCP (Service Connection Point) - запись прописывается в AD, при создание сервера CAS. Outlook запрашивает SCP, выбирает те, которые находятся в одном сайте с ним и по параметру WhenCreated – по дате создания, выбирая самый старый.

Autodiscover. Outlook выбирает в качестве сервера Client Access тот, который прописан в атрибуте RPCCClientAccessServer базы данных пользователя. Сведения о базе данных и сервере mailbox, на котором она лежит, берутся из AD.

## MessageTrackingLog

```

Get-MessageTrackingLog -ResultSize Unlimited | select Timestamp,Sender,Recipients,RecipientCount,MessageSubject,Source,EventID,ClientHostname,Serve
Get-MessageTrackingLog -Start (Get-Date).AddHours(-24) -ResultSize Unlimited | where {[string]$_.recipients -like "*@yandex.ru"} вывести
сообщения за последние 24 часа, где получателем был указанный домен
-Start "04/01/2023 09:00:00" -End "04/01/2023 18:00:00" - поиск по указанному промежутку времени
-MessageSubject "Тест" - поиск по теме письма
-Recipients "support4@domain.ru" - поиск по получателю
-Sender - поиск по отправителю
-EventID – поиск по коду события сервера (RECEIVE, SEND, FAIL, DSN, DELIVER, BADMAIL, RESOLVE, EXPAND, REDIRECT, TRANSFER,
SUBMIT, POISONMESSAGE, DEFER)
-Server – поиск на определенном транспортном сервере
-messageID – трекинг письма по его ID

```

## Mailbox

```

Get-Mailbox -Database "it2" список почтовых серверов в базе данных
Get-Mailbox -resultsize unlimited | ? Emailaddresses -like "support4" | format-list name,emailaddresses,database,servername какую БД, сервер и
smtp-адреса использует почтовый ящик
Get-Mailbox -Database $db_name -Archive отобразить архивные почтовые ящики

Get-MailboxFolderStatistics -Identity "support4" -FolderScope All | select Name,ItemsInFolder,FolderSize отобразить кол-во писем и размер в
каждой папке
Get-MailboxStatistics "support4" | select DisplayName,LastLoggedOnUserAccount,LastLogonTime,LastLogoffTime,ItemCount,TotalItemSize,DeletedItemCount
общее кол-во писем, их размер, время последнего входа и выхода, имя сервера и БД
Get-Mailbox -Server s2 | Get-MailboxStatistics | where {$_.Lastlogontime -lt (get-date).AddDays(-30)} | Sort Lastlogontime -desc | ft displayname,L
ящики, которые не использовались 30 и более дней

Enable-Mailbox -Identity support9 -Database test_base создать почтовый ящик для существующего пользователя в AD
New-Mailbox -Name $login -UserPrincipalName "$login@$domain" -Database $select_db -OrganizationalUnit $path -Password (ConvertTo-SecureString -Stri
создать новый почтовый ящик без привязки к пользователю AD
Get-MailboxDatabase -Database $db_name | Remove-MailboxDatabase удалить БД

```

```
Set-MailBox "support4" -PrimarySmtpAddress support24@domain.ru -EmailAddressPolicyEnabled $false добавить и изменить основной SMTP-адрес электронной почты для пользователя
Set-Mailbox -Identity "support4" -DeliverToMailboxAndForward $true -ForwardingSMTPAddress "username@outlook.com" включить переадресацию почты (электронная почта попадает в почтовый ящик пользователя support4 и одновременно пересыпается по адресу username@outlook.com)
```

## MoveRequest

```
Get-Mailbox -Database $db_in | New-MoveRequest -TargetDatabase $db_out переместить все почтовые ящики из одной БД в другую
New-MoveRequest -Identity $db_in -TargetDatabase $db_out переместить один почтовый ящик
Get-MoveRequest | Suspend-MoveRequest остановить запросы перемещения
Get-MoveRequest | Remove-MoveRequest удалить запросы на перемещение
Get-MoveRequest | Get-MoveRequestStatistics статус перемещения
```

Status:

Cleanup - нужно подождать

Queued - в очереди

InProgress - в процессе

Percent Complete - процент выполнения

CompletionInProgress - завершение процесса

Completed - завершено

```
Remove-MoveRequest -Identity $db_name завершить процесс перемещения (убрать статус перемещения с почтового ящика и очистить список
перемещений)
Get-MailboxDatabase | Select Name, MailboxRetention после перемещения ящиков, размер базы не изменится, полное удаление из базы
произойдет, как пройдет количество дней, выставленное в параметре MailboxRetention
Set-MailboxDatabase -MailboxRetention '0.00:00:00' -Identity $db_name изменить значение
```

## Archive

```
Enable-Mailbox -Identity $name -Archive включить архив для пользователя
Get-Mailbox $name | New-MoveRequest -ArchiveOnly -ArchiveTargetDatabase DBArch переместить архивный почтовый ящик в другую БД
Get-Mailbox $name | fl Name, Database, ArchiveDatabase место расположения БД пользователя и БД его архива
Disable-Mailbox -Identity $name -Archive отключить архив
Connect-Mailbox -Identity "8734c04e-981e-4ccf-a547-1c1ac7ebf3e2" -Archive -User $name -Database it2 подключение архива пользователя к
указанному почтовому ящику
Get-Mailbox $name | Set-Mailbox -ArchiveQuota 20GB -ArchiveWarningQuota 19GB настроить квоты хранения архива
```

## Quota

```
Get-Mailbox -Identity $mailbox | fl IssueWarningQuota, ProhibitSendQuota, ProhibitSendReceiveQuota, UseDatabaseQuotaDefaults отобразить квоты
почтового ящика
IssueWarningQuota — квота, при достижении которой Exchange отправит уведомление
ProhibitSendQuota — при достижении будет запрещена отправка
ProhibitSendReceiveQuota — при достижении будет запрещена отправка и получение
UseDatabaseQuotaDefaults — используется ли квота БД или false - индивидуальные
Set-Mailbox -Identity $mailbox -UseDatabaseQuotaDefaults $false -IssueWarningQuota "3 GB" -ProhibitSendQuota "4 GB" -ProhibitSendReceiveQuota "5 GB"
задать квоту для пользователя
```

```
Get-MailboxDatabase $db_name | fl Name, *Quota отобразить квоты наложенные на БД
Set-MailboxDatabase $db -ProhibitSendReceiveQuota "5 GB" -ProhibitSendQuota "4 GB" -IssueWarningQuota "3 GB" настроить квоты на БД
```

## Database

```
Get-MailboxDatabase -Status | select ServerName, Name, DatabaseSize СПИСОК и размер всех БД на всех MX-серверах
New-MailboxDatabase -Name it_2022 -EdbFilePath E:\Bases\it_2022\it_2022.edb -LogFolderPath G:\Logs\it_2022 -OfflineAddressBook "Default Offline Add
создать БД
Restart-Service MSExchangeIS
Get-Service | Where {$_.Name -match "exchange"} | Restart-Service
Get-MailboxDatabase -Server exch-01 список баз данных на MX-сервере
```

```
New-MoveRequest -Identity "support4" -TargetDatabase it_2022 переместить почтовый ящик в новую БД
Move-DatabasePath $db_name -EdbFilePath "F:\DB\$db_name\$db_name.edb" -LogFolderPath "E:\DB\$db_name\logs\" переместить БД и транзакционные логи на другой диск
Set-MailboxDatabase -CircularLoggingEnabled $true -Identity $db_name включить циклическое ведение журнала (Circular Logging), где последовательно пишутся 4 файла логов по 5 МБ, после чего первый лог-файл перезаписывается
Set-MailboxDatabase -CircularLoggingEnabled $false -Identity $db_name отключить циклическое ведение журнала
Get-MailboxDatabase -Server "exch-mx-01" -Status | select EdbFilePath, LogFolderPath, LogFilePrefix путь к БД, логам, имя текущего актуального лог-файла
```

## MailboxRepairRequest

```
New-MailboxRepairRequest -Database it2 -CorruptionType ProvisionedFolder, SearchFolder, AggregateCounts, FolderView запустить последовательный тест (в конкретный момент времени не доступен один почтовый ящик) и исправление ошибок на прикладном уровне
Get-MailboxRepairRequest -Database it2 прогресс выполнения
Позволяет исправить:
ProvisionedFolder – нарушения логической структуры папок
SearchFolder – ошибки в папках поиска
AggregateCounts – проверка и исправление информации о количестве элементов в папках и их размере
FolderView – неверное содержимое, отображаемое представлениями папок
```

## eseutil

При отправке/получении любого письма Exchange сначала вносит информацию в транзакционный лог, и только потом сохраняет элемент непосредственно в базу данных. Размер одного лог файла - 1 Мб. Есть три способа урезания логов: DAG, Backup на базе Volume Shadow Copy, Circular Logging.

Ручное удаление журналов транзакций:

```
cd E:\MS_Exchange_2010\MailBox\Reg_v1_MailBoxes\ перейти в каталог с логами
ls E*.chk узнать имя файла, в котором находится информация из контрольной точки фиксации журналов
eseutil /mk .\E18.chk узнать последний файл журнала, действия из которого были занесены в БД Exchange
Checkpoint: (0x561299,8,16) 561299 имя файла, который был последним зафиксирован (его информация уже в базе данных)
Находим в проводнике файл E0500561299.txt, можно удалять все файлы журналов, которые старше найденного файла
```

Восстановление БД (если две копии БД с ошибкой):

```
Get-MailboxDatabaseCopyStatus -Identity db_name\* | Format-List Name, Status, ContentIndexState
Status : FailedAndSuspended
ContentIndexState : Failed
Status : Dismounted
ContentIndexState : Failed
```

```
Get-MailboxDatabase -Server exch-mx-01 -Status | fl Name, EdbFilePath, LogFolderPath проверить расположение базы и транзакционных логов
LogFolderPath - директория логов
```

E18 - имя транзакционного лога (из него читаются остальные логи)

```
dismount-Database db_name отмонтировать БД
eseutil /mh D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb проверить базу
```

State: Dirty Shutdown - несогласованное состояние, означает, что часть транзакций не перенесена в базу, например, после того, как была осуществлена аварийная перезагрузка сервера.

```
eseutil /ml E:\MS_Exchange_2010\MailBox\db_name\E18 проверка целостности транзакционных логов, если есть логи транзакций и они не испорчены, то можно восстановить из них, из файла E18читываются все логи, должен быть статус - OK
```

Soft Recovery (мягкое восстановление) - необходимо перевести базу в состояние корректного отключения (Clear shutdown) путем записи недостающих файлов журналов транзакций в БД.

```
eseutil /R E18 /l E:\MS_Exchange_2010\MailBox\db_name\ /d D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb
eseutil /R E18 /a /i /l E:\MS_Exchange_2010\MailBox\db_name\ /d D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb если с логами что-то не так, можно попробовать восстановить базу игнорируя ошибку в логах
eseutil /mk D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb состоянию файла контрольных точек
eseutil /g D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb проверка целостности БД
eseutil /k D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb проверка контрольных сумм базы (CRC)
```

Hard Recovery - если логи содержат ошибки и база не восстанавливается, то восстанавливаем базу без логов.

```
eseutil /p D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb
```

/p - удалит поврежденные страницы, эта информация будет удалена из БД и восстановит целостность

```
eseutil /d D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb
```

 выполнить дефрагментацию (если был потерян большой объем данных, то может сильно снизиться производительность)

После выполнения команды необходимо вручную удалить все файлы с расширением log в папке MDBDATA, перед попыткой смонтировать базу данных.

```
isinteg -s "db_name.edb" -test alltests проверьте целостность базы данных
```

```
isinteg -s "server_name" -fix -test -alltests если проверка будет провалена. Выполнять команду до тех пор, пока у всех ошибок не станет статус 0 или статус не перестанет меняться, иногда необходимо 3 прохода для достижения результата.
```

```
eseutil /mh D:\MS_Exchange_2010\Mailbox\db_name\db_name.edb | Select-String -Pattern "State:","Log Required:" проверить статус
```

State: Clear shutdown - успешный статус

Log Required требуются ли файлы журналов, необходимые базе, чтобы перейти в согласованное состояние. Если база размонтирована корректно, то это значение будет равняться 0.

```
mount-Database -force db_name ПРИМОНТИРОВАТЬ БД
```

```
Get-MailboxDatabase -Status db_name | fl Mounted статус БД
```

```
New-MailboxRepairRequest -Database db_name -CorruptionType SearchFolder,AggregateCounts,ProvisionedFolder,FolderView восстановление логической целостности данных
```

После этого восстановить Index.

Если индексы не восстанавливаются, но БД монтируется, то перенести почтовые ящики в новую БД.

Восстановление БД из Backup:

1-й вариант:

1. Отмонтировать текущую БД и удалить или переименовать директорию с файлами текущей БД.
2. Восстановить в ту же директорию из Backup базу с логами.
3. Запустить мягкое восстановление БД (Soft Recovery).
4. Примониторить.

2-й вариант:

1. Отмонтировать и удалить текущую БД.
2. Восстановить БД с логами из Backup в любое место.
3. Запустить мягкое восстановление БД (Soft Recovery).
4. Создать новую БД.
5. Создать Recovery Database и смонтировать в нее восстановленную из бэкапа БД, скопировать из неё почтовые ящики в новую БД и переключить на них пользователей.
6. Если использовать Dial Tone Recovery, то так же перенести из временной БД промежуточные данные почтовых ящиков.

3-й вариант:

1. Восстановить целостность Soft Repair или Hard Recovery.
2. Создать новую БД. Указывать в свойствах: «база может быть перезаписана при восстановлении».
3. Если база была только чтооздана и еще не была подмонтирована, то эта папка будет пуста, туда перемещаем базу из Backup, которая была обработана ESEUTIL вместе со всеми файлами. Указать имя .edb такое же, которое было при создании новой базы.
4. Монтируем базу.
5. Перенацеливаем ящики со старой (Mailbox\_DB\_02), неисправной базы, на новую базу (Mailbox\_DB\_02\_02):  

```
Get-Mailbox -Database Mailbox_DB_02 | where {$_.ObjectClass -NotMatch '(SystemAttendantMailbox|ExOleDbSystemMailbox)'} | Set-Mailbox -Database
```
6. Восстановление логической целостности данных:

```
New-MailboxRepairRequest -Database "Mailbox_DB_02_02" -CorruptionType ProvisionedFolder, SearchFolder, AggregateCounts, Folderview
```

## Dial Tone Recovery

```
Get-Mailbox -Database "MailboxDB" | Set-Mailbox -Database "TempDB" перенацелить ящики с одной БД (нерабочей) на другую (пустую)
```

```
Get-Mailbox -Database TempDB отобразить почтовые ящики в БД TempDB
```

```
Restart-Service MSExchangeIS перезапустить службу Mailbox Information Store (банка данных), иначе пользователи будут по-прежнему пытаться подключиться к старой БД
```

```
iisreset
```

```
Get-Mailbox -Database "TempDB" | Set-Mailbox -Database "MailboxDB" После восстановления старой БД, нужно переключить пользователей с
```

временной БД обратно

После этого сделать слияние с временной БД с помощью Recovery.

## Recovery database (RDB)

```
New-MailboxDatabase -Recovery -Name RecoveryDB -Server $exch_mx -EdbFilePath "D:\TempDB\TempDB.edb" -LogFolderPath "D:\TempDB" для переноса
новых писем из временной БД в основную необходим только сам файл TempDB.edb со статусом Clean Shutdown, из нее необходимо создать
служебную БД (ключ -Recovery)
Mount-Database "D:\TempDB\TempDB.edb" примонтировать БД
Get-MailboxStatistics -Database RecoveryDB
New-MailboxRestoreRequest -SourceDatabase RecoveryDB -SourceStoreMailbox support -TargetMailbox support скопировать данные почтового ящика с
DisplayName: support из RecoveryDB в почтовый ящик с псевдонимом support существующей базы. По умолчанию ищет в почтовой базе
совпадающие LegacyExchangeDN либо проверяет совпадение адреса X500, если нужно восстановить данные в другой ящик, нужно указывать
ключ -AllowLegacyDNMismatch
New-MailboxRestoreRequest -SourceDatabase RecoveryDB -SourceStoreMailbox support -TargetMailbox support -TargetRootFolder "Restore" скопировать
письма в отдельную папку в ящике назначения (создается автоматически), возможно восстановить содержимое конкретной папки -IncludeFolders
"#Inbox#"
Get-MailboxRestoreRequest | Get-MailboxRestoreRequestStatistics статус запроса восстановления
Get-MailboxRestoreRequestStatistics -Identity support
Get-MailboxRestoreRequest -Status Completed | Remove-MailboxRestoreRequest удалить все успешные запросы
```

## Transport

```
Get-TransportServer $srv_cas | select MaxConcurrentMailboxDeliveries,MaxConcurrentMailboxSubmissions,MaxConnectionRatePerMinute,MaxOutboundConnecti
настройки пропускной способности транспортного сервера
MaxConcurrentMailboxDeliveries — максимальное количество одновременных потоков, которое может открыть сервер для отправки писем.
MaxConcurrentMailboxSubmissions — максимальное количество одновременных потоков, которое может открыть сервер для получения писем.
MaxConnectionRatePerMinute — максимальная возможная скорость открытия входящих соединений в минуту.
MaxOutboundConnections — максимальное возможное количество соединений, которое может открыть Exchange для отправки.
MaxPerDomainOutboundConnections — максимальное возможное количество исходящих соединений, которое может открыть Exchange для
одного удаленного домена.
PickupDirectoryMaxMessagesPerMinute — скорость внутренней обработки сообщений в минуту (распределение писем по папкам).
Set-TransportServer exchange-cas -MaxConcurrentMailboxDeliveries 21 -MaxConcurrentMailboxSubmissions 21 -MaxConnectionRatePerMinute 1201 -MaxOutbou
изменить значения
```

Get-TransportConfig | select MaxSendSize, MaxReceiveSize ограничение размера сообщения на уровне транспорта (наименьший приоритет, после
коннектора и почтового ящика).

```
New-TransportRule -Name AttachmentLimit -AttachmentSizeOver 15MB -RejectMessageReasonText "Sorry, messages with attachments over 15 MB are not acce
создать транспортное правило для проверки размера вложения
```

## Connector

```
Get-ReceiveConnector | select Name,MaxMessageSize,RemoteIPRanges,WhenChanged ограничения размера сообщения на уровне коннектора (приоритет
ниже, чем у почтового ящика)
Set-ReceiveConnector ((Get-ReceiveConnector).Identity)[-1] -MaxMessageSize 30Mb изменить размер у последнего коннектора в списке (приоритет
выше, чем у транспорта)
Get-Mailbox "support4" | select MaxSendSize, MaxReceiveSize наивысший приоритет
Set-Mailbox "support4" -MaxSendSize 30MB -MaxReceiveSize 30MB изменить размер

Set-SendConnector -Identity "ConnectorName" -Port 26 изменить порт коннектора отправки
Get-SendConnector "proxmox" | select port

Get-ReceiveConnector | select Name,MaxRecipientsPerMessage по умолчанию Exchange принимает ограниченное количество адресатов в одном
письме (200)
Set-ReceiveConnector ((Get-ReceiveConnector).Identity)[-1] -MaxRecipientsPerMessage 50 изменить значение
Set-ReceiveConnector ((Get-ReceiveConnector).Identity)[-1] -MessageRateLimit 1000 задать лимит обработки сообщений в минуту для коннектора

Get-OfflineAddressbook | Update-OfflineAddressbook обновить ОАБ
Get-ClientAccessServer | Update-FileDistributionService
```

## PST

```
New-MailboxExportRequest -Mailbox $name -filepath "\$srv\pst\$name.PST" # -ContentFilter {(Received -lt "01/01/2021") } -Priority Highest/Lower # -  
выполнить экспорт из архива пользователя  
New-MailboxExportRequest -Mailbox $name -IncludeFolders "#Inbox#" -FilePath "\$srv\pst\$name.PST" только папку входящие  
New-MailboxImportRequest -Mailbox $name "\$srv\pst\$name.PST" импорт из PST  
Get-MailboxExportRequest статус запросов  
Get-MailboxExportRequest -Status Completed | Remove-MailboxExportRequest удалить успешно завершенные запросы  
Remove-MailboxExportRequest -RequestQueue MBXDB01 -RequestGuid 25e0eaf2-6cc2-4353-b83e-5cb7b72d441f отменить экспорт
```

## DistributionGroup

```
Get-DistributionGroup список групп рассылки  
Get-DistributionGroupMember "!_Офис" список пользователей в группе  
Add-DistributionGroupMember -Identity "!_Офис" -Member "$name@$domain" добавить в группу рассылки  
Remove-DistributionGroupMember -Identity "!_Офис" -Member "$name@$domain"  
New-DistributionGroup -Name "!_Тест" -Members "$name@$domain" создать группу  
Set-DistributionGroup -Identity "support4" -HiddenFromAddressListsEnabled $true (или Set-Mailbox) скрыть из списка адресов Exchange
```

## Search

```
Search-Mailbox -Identity "support4" -SearchQuery 'Тема:"Mikrotik DOWN"' поиск писем по теме  
Search-Mailbox -Identity "support4" -SearchQuery 'Subject:"Mikrotik DOWN"'  
Search-Mailbox -Identity "support4" -SearchQuery 'attachment -like:"*.rar"'  
Search-Mailbox -Identity "support4" -SearchQuery "отправлено: < 01/01/2020" -DeleteContent -Force удаление писем по дате
```

Формат даты в зависимости от региональных настроек сервера:

```
20/07/2018  
07/20/2018  
20-Jul-2018  
20/July/2018
```

## AuditLog

```
Get-AdminAuditLogConfig настройки аудита  
Set-Mailbox -Identity "support4" -AuditOwner HardDelete добавить логирование HardDelete писем  
Set-mailbox -identity "support4" -AuditlogAglimit 120 указать время хранения  
Get-mailbox -identity "support4" | Format-list Audit* данные аудита  
Search-MailboxAuditLog -Identity "support4" -LogonTypes Delegate -ShowDetails -Start "2022-02-22 18:00" -End "2022-03-22 18:00" просмотр логов  
Search-AdminAuditLog -StartDate "02/20/2022" | ft CmdLetName,Caller,RunDate,ObjectModified -Autosize Поиск событий истории выполненных команд  
в журнале аудита Exchange
```

## Test

```
Test-ServiceHealth проверить доступность ролей сервера: почтовых ящиков, клиентского доступа, единой системы обмена сообщениями,  
транспортного сервера  
$mx_srv_list | %{Test-MapiConnectivity -Server $_} проверка подключения MX-серверов к БД  
Test-MAPIConnectivity -Database $db проверка возможности логина в базу  
Test-MAPIConnectivity -Identity $user@$domain проверка возможности логина в почтовый ящик  
Test-ComputerSecureChannel проверка работы службы AD  
Test-MailFlow результат тестового потока почты
```

## Queue

```
Get-TransportServer | % {Get-Queue -Server $_.Name} отобразить очереди на всех транспортных серверах  
Get-Queue -Identity EXCHANGE-CAS\155530 | Format-List подробная информация об очереди  
Get-Queue -Identity EXCHANGE-CAS\155530 | Get-Message -ResultSize Unlimited | Select FromAddress,Recipients отобразить список отправителей  
(FromAddress) и список получателей в очереди (Recipients)  
Get-Message -Queue EXCHANGE-CAS\155530 отобразить идентификатор сообщений в конкретной очереди (сервер\очередь\идентификатор письма)
```

```
Resume-Message EXCHANGE-CAS\155530\444010 повторить отправку письма из очереди
Retry-Queue -Filter {Status -eq "Retry"} принудительно повторить отправку всех сообщений с статусом "Повторить"
Get-Queue -Identity EXCHANGE-CAS\155530 | Get-Message -ResultSize unlimited | Remove-Message -WithNDR $False очистить очередь
Get-transportserver EXCHANGE-CAS | Select MessageExpirationTimeout отобразить время жизни сообщений в очереди (по умолчанию, 2 дня)
```

Error Exchange 452 4.3.1 Insufficient system resources - окончание свободного места на диске, на котором находятся очереди службы Exchange Hub Transport, за мониторинг отвечает компонент доступных ресурсов Back Pressure, который в том числе отслеживает свободное место на диске

Порог Medium (90%) — перестать принимать по SMTP почту от внешних отправителей (почта от MAPI клиентов при этом обрабатывается)

Порог High (99%) — обработка потока почты полностью прекращается

Решение: очистить, например логи IIS (C:\inetpub\logs\LogFiles\W3SVC1), увеличить размер диска, отключить мониторинг Back Pressure (плохой вариант) или перенести транспортные очередь на другой диск достаточного объема.

```
Get-Service | ? name -like "MSExchangeTransport" | Stop-Service остановить службу очереди
Rename-Item "C:\Program Files\Microsoft\Exchange Server\V15\TransportRoles\data\Queue" "C:\Program Files\Microsoft\Exchange Server\V15\TransportRo очистить базу очереди
```

C:\Program Files\Microsoft\Exchange Server\V15\Bin\EdgeTransport.exe.config конфигурационный файл, который содержит путь к бд с очередью (блок ключи и QueueDatabaseLoggingPath)

Для переноса БД, необходимо переместить существующие файлы базы данных Mail.que и Trn.chk (контрольные точки для отслеживания записи в логах) из исходного местоположения в новое. Переместите существующие файлы журнала транзакций Trn.log, Trntmp.log, Trn nnnn.log , Trnres00001.jrs, Trnres00002.jrs и Temp.edb из старого расположения в новое. tmp.edb — временный файл для проверки схемы самой базы, переноса не нужно.

После запуска службы транспорта удалить старую базу данных очереди и файлы журнала транзакций из старого расположения.

## Defrag

```
Get-MailboxDatabase -Status | ft Name, DatabaseSize, AvailableNewMailboxSpace
```

DatabaseSize - текущий размер базы

AvailableNewMailboxSpace - объем пустых страниц, пространство, которое можно освободить при дефрагментации

(DatabaseSize — AvailableNewMailboxSpace) x 1,1 - необходимо дополнительно иметь свободного места не менее 110% от текущего размера базы (без учета пустых страниц)

```
cd $path
```

```
Dismount-Database "$path\$db_name" отмонтировать БД
```

```
eseutil /d "$path\$db_name.edb"
```

```
Mount-Database "$path\$db" промонтировать БД
```

## DAG

### Database Availability Group

```
Install-WindowsFeature -Name Failover-Clustering -ComputerName EXCH-MX-01 основывается на технологии Windows Server Failover Cluster
```

New-DatabaseAvailabilityGroup -Name dag-01 -WitnessServer fs-05 -WitnessDirectory C:\witness\_exchange1 создать группу с указанием файлового свидетеля для кворума

Quorum - это процесс голосования, в котором для принятия решения нужно иметь большинство голосов, что бы сделать текущую копию базы данных активной.

WitnessDirectory — используется для хранения данных файлового ресурса-свидетеля.

```
Set-DatabaseAvailabilityGroup dag-01 -DatabaseAvailabilityGroupIPAdress $ip изменить ip-адрес группы
```

```
Get-DatabaseAvailabilityGroup СПИСОК всех групп
```

```
Get-DatabaseAvailabilityGroup -Identity dag-01
```

Add-DatabaseAvailabilityGroupServer -Identity dag-01 -MailboxServer EXCH-MX-01 добавить первый сервер (все БД на серверах в DAG должны храниться по одинаковому пути)

```
Add-MailboxDatabaseCopy -Identity db_name -MailboxServer EXCH-MX-04 добавить копию БД
```

Get-MailboxDatabaseCopyStatus -Identity db\_name\\* | select Name, Status, LastInspectedLogTime статус и время последнего копирования журнала транзакций

Status:

Mounted - рабочая база

Suspended - приостановлено копирование

Healthy - рабочая пассивная копия

ServiceDown - недоступна (выключен сервер)  
Dismounted - отмонтирована  
FailedAndSuspended - ошибка и приостановка копирования  
Resynchronizing - процесс синхронизация, где будет постепенно уменьшаться длина очереди  
CopyQueue Length - длина репликационной очереди копирования (0 - значит все изменения из активной базы реплицированы в пассивную копию)

Resume-MailboxDatabaseCopy -Identity db\_name\EXCH-MX-04 возобновить (Resume) или запустить копирование бд на EXCH-MX-04 (из статуса Suspended в Healthy)  
Suspend-MailboxDatabaseCopy -Identity db\_name\EXCH-MX-04 остановить копирование (в статус Suspended)  
Update-MailboxDatabaseCopy -Identity db\_name\EXCH-MX-04 -DeleteExistingFiles обновить копию БД (сделать Full Backup)  
Set-MailboxDatabaseCopy -Identity db\_name\EXCH-MX-04 -ActivationPreference 1 изменить приоритет для активации копий БД (какую использовать, 1 – самое высокое значение)  
Move-ActiveMailboxDatabase db\_name -ActivateOnServer EXCH-MX-04 -MountDialOverride:None -Confirm:\$false включить копию БД в DAG (переключиться на активную копию)  
Remove-MailboxDatabaseCopy -Identity db\_name\EXCH-MX-04 -Confirm:\$False удалить копии пассивной базы в DAG-группе (у БД должно быть отключено ведение циклического журнала)  
Remove-DatabaseAvailabilityGroupServer -Identity dag-01 -MailboxServer EXCH-MX-04 -ConfigurationOnly удалить MX сервер из группы DAG  
Import-Module FailoverClusters  
Get-ClusterNode EXCH-MX-04 | Remove-ClusterNode -Force удалить отказавший узел из Windows Failover Cluster  
Get-DatabaseAvailabilityGroup | Get-DatabaseAvailabilityGroupHealth МОНИТОРИНГ

## Index

Get-MailboxDatabaseCopyStatus \* | select name,status,ContentIndexState,ContentIndexErrorMessage,ActiveDatabaseCopy,LatestCopyBackupTime,CopyQueueLength  
узнать состояние работы индексов БД и текст ошибки, на каком сервере активная копия БД, дата последней копии и текущая очередь  
Get-MailboxDatabaseCopyStatus -Identity \$db\_name\\* | Format-List Name,ContentIndexState отобразить список всех копий конкретной БД на всех серверах, и статус их индексов, если у второго сервера статус Healthy, можно восстановить из него  
Get-MailboxDatabaseCopyStatus -Identity \$db\_name\EXCH-MX-04 | Update-MailboxDatabaseCopy -SourceServer EXCH-MX-01 -CatalogOnly восстановить БД из копии  
cd %PROGRAMFILES%\Microsoft\Exchange Server\V14\Scripts Или v15 для Exchange 2016  
.\\ResetSearchIndex.ps1 \$db\_name скрипт восстановления индекса  
Get-MailboxDatabaseCopyStatus \* | where {\$\_.ContentIndexState -eq "Failed" -or \$\_.ContentIndexState -eq "FailedAndSuspended"} отобразить у какой БД произошел сбой работы (FailedAndSuspended) или индекса (ContentIndexState)

## NAS

### TrueNAS

#### PowerTrueNas

Install-Module TrueNas  
Import-Module TrueNas  
\$(Get-Module TrueNas).ExportedCommands  
Connect-TrueNasServer -Server tnas-01 -SkipCertificateCheck  
Get-TrueNasCertificate Настройки сертификата  
Get-TrueNasSetting Настройки языка, time zone, syslog level и server, https port  
Get-TrueNasUser СПИСОК пользователей  
Get-TrueNasSystemVersion характеристики (Physical Memory, Model, Cores) и Uptime  
Get-TrueNasSystemAlert snmp для оповещений  
Get-TrueNasSystemNTP СПИСОК используемых NTP серверов  
Get-TrueNasDisk СПИСОК разделов физического диска  
Get-TrueNasInterface сетевые интерфейсы  
Get-TrueNasGlobalConfig сетевые настройки  
Get-TrueNasDnsServer Настроенные DNS-сервера  
Get-TrueNasIscsiTarget отобразить ID группы инициаторов использующих таргет, используемый portal, authentication и authen-method

```
Get-TrueNasIscsiInitiator отобразить группы инициаторов
Get-TrueNasIscsiPortal слушатель (Listen) и порт
Get-TrueNasIscsiExtent список iSCSI Target (статус работы, путь)
Get-TrueNasPool список pool (Id, Path, Status, Healthy)
Get-TrueNasVolume -Type FILESYSTEM список pool файловых систем
Get-TrueNasVolume -Type VOLUME список разделов в pool и их размер
Get-TrueNasService | ft список служб и их статус
Start-TrueNasService ssh запустить службу
Stop-TrueNasService ssh остановить службу
```

## Synology

### pSynology

```
New-SYNOSession аутентификация на Synology Diskstation и запуск нового сеанса API
Close-SYNOSession выход из сеанса API
Get-SYNOInfo получить информацию об API DiskStation
Add-SYNOFSAvorite добавить папку в избранное пользователя
Add-SYNOFSSfile загрузить файл
Clear-SYNOFSTask удалить все завершенные фоновые задачи
Clear-SYNOFSShareStatus удалить все избранное с неработающим статусом
Clear-SYNOFSSharingLink удалить все просроченные и неработающие ссылки для обмена
Get-SYNOFSArchiveCompress получить статус задачи сжатия
Get-SYNOFSArchiveContent вывести содержимое архива
Get-SYNOFSArchiveExtract получить статус задачи извлечения
Get-SYNOFSTask список всех фоновых задач, включая копирование
Get-SYNOFSCopy получить статус операции копирования
Get-SYNOFSDelItem получить статус задачи удаления
Get-SYNOFSDirSize получить статус задачи расчета размера
Get-SYNOFSAvorite список избранного пользователя
Get-SYNOFSSfile перечислить файлы в заданной папке
Get-SYNOFSSFileInfo получить информацию о файлах или файле
Get-SYNOFSSInfo получить информацию о файловой станции
Get-SYNOFSSMD5 получить статус вычислительной задачи MD5
Get-SYNOFSSearch перечислить совпадающие файлы во временной базе данных поиска
Get-SYNOFSShare список всех общих папок
Get-SYNOFSSharingLink список ссылок пользователя на общий доступ к файлам
Get-SYNOFSShareLinkInfo получить информацию о ссылке общего доступа по идентификатору ссылки общего доступа
Get-SYNOFSThumbnail получить миниатюру файла
Get-SYNOFSVirtualFolder список всех папок точек монтирования заданного типа виртуальной файловой системы
New-SYNOFSSFolder создать папку
New-SYNOFSShareLink создать одну или несколько ссылок для общего доступа по пути к файлу или папку
Remove-SYNOFSAvorite удаление избранного из избранного пользователя
Remove-SYNOFSDelItem удалить файл или папку
Remove-SYNOFSSearch удалить временные базы данных поиска
Remove-SYNOFSSharingLink удалить одну или несколько ссылок общего доступа
Rename-SYNOFSDelItem переименовываем файл или папку
Save-SYNOFSSfile загрузить файл или папку
Set-SYNOFSAvorite редактировать избранное имя
Set-SYNOFSShareLink изменить ссылку для общего доступа
Start-SYNOFSArchiveCompress запустить задание сжатия файла или папки
Start-SYNOFSArchiveExtract запустить задание распаковки архива
Start-SYNOFSCopy запустить задание копирования файлов
Start-SYNOFSDelItem запустить задание удаления файлов или папок
Start-SYNOFSDirSize рассчитать размер для одного или нескольких путей к файлам или папкам
Start-SYNOFSSMD5 получить MD5 файла
Start-SYNOFSSearch поиск файлов по заданным критериям
Stop-SYNOFSArchiveCompress остановить задачу сжатия
```

```
Stop-SYNOFSArchiveExtract остановить задачу извлечения
Stop-SYNOFSCopy остановить задачу копирования
Stop-SYNOFSDeleteItem остановить задачу удаления
Stop-SYNOFSDirSize остановить расчет размера
Stop-SYNOFSMD5 остановить вычисление MD5 файла
Stop-SYNOFSSearch остановить задачу поиска
Test-SYNOFSPermission Проверить, имеет ли вошедший в систему пользователь разрешение на запись в данную папку
Update-SYNOFSFAvorite заменить несколько выбранных папок в избранном пользователя
```

## Veeam

VeeamHub (<https://github.com/VeeamHub/powershell>)

### Get-VBRCommand

```
Set-ExecutionPolicy AllSigned
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol
установить choco
choco install veeam-backup-and-replication-console установить консоль управления (https://community.chocolatey.org/packages/veeam-backup-and-replication-console)
Get-Module Veeam.Backup.PowerShell модуль, который устанавливается вместе с клиентской консолью по умолчанию
Get-Command -Module Veeam.Backup.PowerShell
Get-Command -Module Get-VBRCommand
Connect-VBRSRver -Server $srv -Credential $cred -Port 9392
Get-VBRJob
Get-VBRCommand *get*backup*
Get-VBRComputerBackupJob
Get-VBRBackup
Get-VBRBackupRepository
Get-VBRBackupSession
Get-VBRBackupServerCertificate
Get-VBRRestorePoint
Get-VBRViProxy
```

### Veeam-REStat

```
$path = ($env:PSModulePath.Split(";")[0])+"\Veeam-REStat\Veeam-REStat.psm1"
if (!(Test-Path $path)) {
    New-Item $path -ItemType "File" -Force
}
$(iwr https://raw.githubusercontent.com/Lifailon/Veeam-REStat/rsa/Veeam-REStat/Veeam-REStat.psm1).Content | Out-File $path -Force
```

Veeam-REStat -Server srv-veeam-11 -Port 9419 при первом запуске необходимо заполнить Credential для подключения к экземпляру сервера VBR, которые сохраняются в файл с именем сервера в формате xml с применением шифрования PSCredential для последующего подключения  
Veeam-REStat -Reset сброс учетных данных для подключения к серверу VBR  
Veeam-REStat -Statistic статистика всех заданий с сортировкой по дате (выводит время начала, завершения и статус работы, процент прогресса, результат выполнений и сообщение с причиной в случае ошибки: Warning или Failed)  
Veeam-REStat -Jobs подробная статистика по всем настроенным заданиям резервного копирования: статус работы (In Active/disabled), результат последнего задания (LastResult), тип аутентификации (Standard/Linux), имя и размер виртуальной машины, тип резервного копирования (например, Incremental), дату и время последнего и следующего выполнения  
Veeam-REStat -ConfigBackup отображает статус состояния работы резервного копирования конфигурации сервера VBR, кол-во точек восстановления, дату и время последней копии  
Veeam-REStat -Repositories статистика по инвентарным данным репозиториев: тип хранилища, путь на сервере до директории хранения, общий (capacityGB), свободный (freeGB) и используемый (usedSpaceGB) размер диска под данные  
Veeam-REStat -Backup Список заданий резервного копирования, тип копирования (VM/Directory) и кол-во точек восстановления  
Veeam-REStat -Points История статистики всех точек восстановления с датой создания  
Veeam-REStat -Hosts Список физически (вручную) добавленных хостов в инфраструктуру VBR

Veeam-REStat -Proxy Список серверов с ролью Proxy  
Veeam-REStat -Users список УЗ, добавленных для подключения к серверам  
Veeam-REStat -Service выводит информацию о связанных внутренних службах, подключение к этим службам может потребоваться только для интеграции с VBR

## REST API

```
$url = "https://habr.com/ru/rss/users/Lifailon/publications/articles/?fl=ru" RSS лента публикаций на Habr
Invoke-RestMethod $url
$iwr = Invoke-WebRequest -Uri $url
$iwr | Get-Member
$iwr.Content
$iwr.StatusCode -eq 200
$iwr.Headers
$iwr.ParsedHtml | Select lastModified
$iwr.Links | fl title,innerText,href
$iwr.Images.src
```

## Methods

**GET** - Read  
**POST** - Create  
**PATCH** - Partial update/modify  
**PUT** - Update/replace  
**DELETE** - Remove

## Download Image

```
function Download-Image {
    param (
        [Parameter(Mandatory = $True)]$url
    )
    $folder = $url -replace "http.://" -replace "/", "-" -replace "-$"
    $path = "$home\Pictures\$folder"
    if (Test-Path $path) {
        Remove-Item $path -Recurse -Force
        New-Item -ItemType Directory $path > $null
    } else {
        New-Item -ItemType Directory $path > $null
    }
    $irm = Invoke-WebRequest -Uri $url
    foreach ($img in $irm.Images.src) {
        $name = $img -replace ".+/"
        Start-Job {
            Invoke-WebRequest $using:img -OutFile "$using:path\$using:name"
        } > $null
    }
    while ($True){
        $status_job = (Get-Job).State[-1]
        if ($status_job -like "Completed"){
            Get-Job | Remove-Job -Force
            break
        }
    }
    $count_all = $irm.Images.src.Count
    $count_down = (Get-Item $path\*).Count
    "Downloaded $count_down of $count_all files to $path"
}
```

Download-Image -url https://losst.pro/

## Token

```
https://veeam-11:9419/swagger/ui/index.html
$header = @{
    "x-api-version" = "1.0-rev2"
}
$body = @{
    "grant_type" = "password"
    "username" = "$login"
    "password" = "$password"
}
$post = iwr "https://veeam-11:9419/api/oauth2/token" -Method POST -Headers $header -Body $body -SkipCertificateCheck
$token = (($post.Content) -split '')[3]
```

## GET

```
$token = $token | ConvertTo-SecureString -AsPlainText -Force
$vjob = iwr "https://veeam-11:9419/api/v1/jobs" -Method GET -Headers $header -Authentication Bearer -Token $token -SkipCertificateCheck

$header = @{
    "x-api-version" = "1.0-rev1"
    "Authorization" = "Bearer $token"
}
$vjob = iwr "https://veeam-11:9419/api/v1/jobs" -Method GET -Headers $header -SkipCertificateCheck
$vjob = $vjob.Content | ConvertFrom-Json

$vjob = Invoke-RestMethod "https://veeam-11:9419/api/v1/jobs" -Method GET -Headers $header -SkipCertificateCheck
$vjob.data.virtualMachines.includes.inventoryObject
```

## Cookie

Получить hash торрент файла на сайте Кинозал

```
function Get-KinozalTorrentHash {
    param (
        [Parameter(Mandatory = $True)][string]$id,
        [Parameter(Mandatory = $True)][string]$cookies
    )
    $url = "https://kinozal.tv/get_srv_details.php?id=$($id)&action=2"
    $cookies = "uid=...+"
    $headers = @{
        "User-Agent" = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Safari/537.36"
        "Cookie" = $cookies
    }
    $result = Invoke-RestMethod -Uri $url -Headers $headers -Method Get
    $result -match "Инфо хеш: (.+)</li><li>Размер" | Out-Null
    return $Matches[1]
}

$id = 1656552
$cookies = "uid=..." получить cookie в браузере на вкладке сеть из заголовка запросов после авторизации на сайте
Get-KinozalTorrentHash $id $cookies
```

## Pode

[PowerShell Web framework](#) для создания REST API, Веб-сайтов, TCP и SMTP серверов.

```

Start-PodeServer {
    Add-PodeEndpoint -Address localhost -Port "8080" -Protocol "HTTP"
    ### Get info endpoints
    Add-PodeRoute -Path "/" -Method "GET" -ScriptBlock {
        Write-PodeJsonResponse -Value @{
            "service"="/api/service";
            "process"="/api/process"
        }
    }
    ### GET
    Add-PodeRoute -Path "/api/service" -Method "GET" -ScriptBlock {
        Write-PodeJsonResponse -Value $(
            Get-Service | Select-Object Name,@{
                Name="Status"; Expression={[string]$_.Status}
            },@{
                Name="StartTime"; Expression={[string]$_.StartTime}
            } | ConvertTo-Json
        )
    }
    Add-PodeRoute -Path "/api/process" -Method "GET" -ScriptBlock {
        Write-PodeJsonResponse -Value $($(
            Get-Process | Sort-Object -Descending CPU | Select-Object -First 15 ProcessName,
            @{$Name="ProcessorTime"; Expression=$_.TotalProcessorTime -replace "\.\d+"}),
            @{$Name="Memory"; Expression={[string]([int]($_.WS / 1024kb))+"MB"}},
            @{$Label="RunTime"; Expression={((Get-Date) - $_.StartTime) -replace "\.\d+"}}
        ))
    }
    Add-PodeRoute -Path "/api/process-html" -Method "GET" -ScriptBlock {
        Write-PodeHtmlResponse -Value $($(
            Get-Process | Sort-Object -Descending CPU | Select-Object -First 15 ProcessName,
            @{$Name="ProcessorTime"; Expression=$_.TotalProcessorTime -replace "\.\d+"}),
            @{$Name="Memory"; Expression={[string]([int]($_.WS / 1024kb))+"MB"}},
            @{$Label="RunTime"; Expression={((Get-Date) - $_.StartTime) -replace "\.\d+"}} # Auto ConvertTo-Html
        ))
    }
    ### POST
    Add-PodeRoute -Path "/api/service" -Method "POST" -ScriptBlock {
        # https://pode.readthedocs.io/en/latest/Tutorials/WebEvent/
        # $WebEvent | Out-Default
        $Value = $WebEvent.Data["ServiceName"]
        $Status = (Get-Service -Name $Value).Status
        Write-PodeJsonResponse -Value @{
            "Name"="$Value";
            "Status"="$Status";
        }
    }
}

irm http://localhost:8080/api/service -Method Get
irm http://localhost:8080/api/process -Method Get
http://localhost:8080/api/process-html использовать браузер
irm http://localhost:8080/api/service -Method Post -Body @{"ServiceName" = "AnyDesk"}

```

## Selenium

Invoke-Expression(New-Object Net.WebClient).DownloadString("https://raw.githubusercontent.com/Lifailon/Deploy-Selenium/rsa/Deploy-Selenium-Drivers")  
установка всех драйверов и Chromium соответствующий версии для драйвера

```

$path = "$home\Documents\Selenium\"
$log = "$path\ChromeDriver.log"
$ChromeDriver = "$path\ChromeDriver.exe"
$WebDriver = "$path\WebDriver.dll"
$SupportDriver = "$path\WebDriver.Support.dll"
$Chromium = (Get-ChildItem $path -Recurse | Where-Object Name -like chrome.exe).FullName
Add-Type -Path $WebDriver
Add-Type -Path $SupportDriver
try {
    $ChromeOptions = New-Object OpenQA.Selenium.Chrome.ChromeOptions # создаем объект с настройками запуска браузера
    $ChromeOptions.BinaryLocation = $Chromium # передаем путь до исполняемого файла, который отвечает за запуск браузера
    $ChromeOptions.AddArgument("start-maximized") # добавляем аргумент, который позволяет запустить браузер на весь экран
    #$ChromeOptions.AddArgument("start-minimized") # запускаем браузер в окне
    #$ChromeOptions.AddArgument("window-size=400,800") # запускаем браузер с заданными размерами окна в пикселях
    $ChromeOptions.AcceptInsecureCertificates = $True # игнорировать предупреждение на сайтах с не валидным сертификатом
    #$ChromeOptions.AddArgument("headless") # скрывать окно браузера при запуске
    $ChromeDriverService = [OpenQA.Selenium.Chrome.ChromeDriverService]::CreateDefaultService($ChromeDriver) # создаем объект настроек службы драйвера
    $ChromeDriverService.HideCommandPromptWindow = $True # отключаем весь вывод логирования драйвера в консоль (этот вывод нельзя перенаправить)
    $ChromeDriverService.LogPath = $log # указать путь до файла с журналом
    $ChromeDriverService.EnableAppendLog = $True # не перезаписывать журнал при каждом новом запуске
    #$ChromeDriverService.EnableVerboseLogging = $True # кроме INFO и ошибок, записывать DEBUG сообщения
    $Selenium = New-Object OpenQA.Selenium.Chrome.ChromeDriver($ChromeDriverService, $ChromeOptions) # инициализируем запуск с указанными настройками

    $Selenium.Navigate().GoToUrl("https://google.com") # переходим по указанной ссылке в браузере
    #$Selenium.Manage().Window.Minimize() # свернуть окно браузера после запуска и перехода по нужному url (чтобы считать страницу корректно)
    # Ищем поле для ввода текста:
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::Id('APjFqb'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::XPath('//*[@id="APjFqb"]'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::Name('q'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::XPath('//*[@name="q"]'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::ClassName('gLFyf'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::CssSelector('[jsname="yZiJbe"]'))
    $Search = $Selenium.FindElements([OpenQA.Selenium.By]::TagName('textarea')) | Where-Object ComputedAccessibleRole -eq combobox
    $Search.SendKeys("calculator online") # передаем текст выбранному элементу
    $Search.SendKeys([OpenQA.Selenium.Keys]::Enter) # нажимаем Enter для вызова функции поиска

    Start-Sleep 1
    $div = $Selenium.FindElements([OpenQA.Selenium.By]::TagName("div"))
    $2 = $div | Where-Object {($_.ComputedAccessibleRole -eq "button") -and ($_.ComputedAccessibleLabel -eq "2")}
    $2.Click()
    $2.Click()
    $plus = $div | Where-Object {($_.ComputedAccessibleRole -eq "button") -and ($_.Text -eq "+")}
    $plus.Click()
    $3 = $Selenium.FindElement([OpenQA.Selenium.By]::CssSelector('[jsname="KN1kY"]'))
    $3.Click()
    $3.Click()
    $sum = $Selenium.FindElement([OpenQA.Selenium.By]::CssSelector('[jsname="Pt8tGc"]'))
    $sum.Click()
    $result = $Selenium.FindElement([OpenQA.Selenium.By]::CssSelector('[jsname="VssY5c"]')).Text
    Write-Host "Result: $result" -ForegroundColor Green
}
finally {
    $Selenium.Close()
    $Selenium.Quit()
}

```

## Selenium modules

```
Invoke-RestMethod https://raw.githubusercontent.com/Lifailon/Selenium-Modules/rst/Modules/Get-GPT/Get-GPT.psm1 | Out-File -FilePath "$(New-Item -
```

```
Get-GPT "Исполняй роль калькулятора. Посчитай сумму чисел: 22+33"
```

```

Invoke-RestMethod https://raw.githubusercontent.com/Lifailon/Selenium-Modules/rsa/Modules/Get-Translation/Get-Translation.ps1 | Out-File -FilePath "C:\Windows\Temp\Get-Translation.ps1"

Get-Translation -Provider DeepL -Text "I translating the text"
Get-Translation -Provider DeepL -Text "Я перевожу текст"
Get-Translation -Provider Google -Text "I translating the text"
Get-Translation -Provider Google -Text "Я перевожу текст" -Language en

Invoke-RestMethod https://raw.githubusercontent.com/Lifailon/Selenium-Modules/rsa/Modules/Get-SpeedTest/Get-SpeedTest.ps1 | Out-File -FilePath "C:\Windows\Temp\Get-SpeedTest.ps1"

Get-SpeedTest -Provider Libre
Get-SpeedTest -Provider Open
Get-SpeedTest -Provider Ookla

```

## IE

```

$ie.document.IHTMLDocument3_getElementsByTagName("input") | select name получить имена всех Input Box
$ie.document.IHTMLDocument3_getElementsByTagName("button") | select innerText получить имена всех Button
$ie.Document.documentElement.innerHTML прочитать сырой Web Content (<input name="login" tabindex="100" class="input__control input__input" id="uniq32005644019429136" spellcheck="false" placeholder="Логин")
>All_Elements = $ie.document.IHTMLDocument3_getElementsByTagName("*") забрать все элементы
$Go_Button = $All_Elements | ? innerText -like "go" ПОИСК элемента по имени
$Go_Button | select ie9_tagName получить TagName (SPAN) для быстрого дальнейшего поиска
$SPAN_Elements = $ie.document.IHTMLDocument3_getElementsByTagName("SPAN")

$ie = New-Object -ComObject InternetExplorer.Application
$ie.navigate("https://yandex.ru")
$ie.visible = $true
$ie.document.IHTMLDocument3_getElementByID("login").value = "Login"
$ie.document.IHTMLDocument3_getElementByID("passwd").value = "Password"
$Button_Auth = ($ie.document.IHTMLDocument3_getElementsByTagName("button")) | ? innerText -match "Войти"
$Button_Auth.Click()
$Result = $ie.Document.documentElement.innerHTML
$ie.Quit()

```

## COM

```

$wshell = New-Object -ComObject Wscript.Shell
$wshell | Get-Member
$link = $wshell.CreateShortcut("$Home\Desktop\Yandex.lnk") создать ярлык
$link | Get-Member
$link.TargetPath = "https://yandex.ru" куда ссылается (метод TargetPath объекта link где хранится объект CreateShortcut) 'link.Save()' сохранить

Set-WinUserLanguageList -LanguageList en-us,ru -Force Изменить языковую раскладку клавиатуры

```

## Wscript.Shell.SendKeys

```

(New-Object -ComObject Wscript.shell).SendKeys([char]173) включить/выключить звук
$wshell.Exec("notepad.exe") запустить приложение
$wshell.AppActivate("Блокнот") развернуть запущенное приложение

$wshell.SendKeys("Login") текст
$wshell.SendKeys("{A 5}") напечатать букву 5 раз подряд
$wshell.SendKeys("%{TAB}") ALT+TAB
$wshell.SendKeys("^") CTRL
$wshell.SendKeys("%") ALT
$wshell.SendKeys("+") SHIFT

```

```

$wshell.SendKeys("{DOWN}") вниз
$wshell.SendKeys("{UP}") вверх
$wshell.SendKeys("{LEFT}") влево
$wshell.SendKeys("{RIGHT}") вправо
$wshell.SendKeys("{PGUP}") PAGE UP
$wshell.SendKeys("{PGDN}") PAGE DOWN
$wshell.SendKeys("{BACKSPACE}") BACKSPACE/BKSP/BS
$wshell.SendKeys("{DEL}") DEL/DELETE
$wshell.SendKeys("{INS}") INS/INSERT
$wshell.SendKeys("{PRTSC}") PRINT SCREEN
$wshell.SendKeys("{ENTER}")

$wshell.SendKeys("{ESC}")
$wshell.SendKeys("{TAB}")
$wshell.SendKeys("{END}")
$wshell.SendKeys("{HOME}")
$wshell.SendKeys("{BREAK}")
$wshell.SendKeys("{SCROLLLOCK}")
$wshell.SendKeys("{CAPSLOCK}")
$wshell.SendKeys("{NUMLOCK}")
$wshell.SendKeys("{F1}")
$wshell.SendKeys("{F12}")

$wshell.SendKeys("{+}{^}{%}{~}{()}{[]}{{}{}}")

```

```

function Get-AltTab {
    (New-Object -ComObject wscript.shell).SendKeys("%{Tab}")
    Start-Sleep $(Get-Random -Minimum 30 -Maximum 180)
    Get-AltTab
}

```

Get-AltTab

## Wscript.Shell.Popup

```

$wshell = New-Object -ComObject Wscript.Shell
$output = $wshell.Popup("Выберите действие?", 0, "Заголовок", 4)
if ($output -eq 6) {"yes"} elseif ($output -eq 7) {"no"} else {"no good"}

```

Type:

- 0 OK
- 1 OK и Отмена
- 2 Стоп, Повтор, Пропустить
- 3 Да, Нет, Отмена
- 4 Да и Нет
- 5 Повтор и Отмена
- 16 Stop
- 32 Question
- 48 Exclamation
- 64 Information

Output:

- 1 Timeout
- 1 OK
- 2 Отмена
- 3 Стоп
- 4 Повтор
- 5 Пропустить
- 6 Да
- 7 Нет

## **WScript.Network**

```
$wshell = New-Object -ComObject WScript.Network  
$wshell | Get-Member  
$wshell.UserName  
$wshell.ComputerName  
$wshell.UserDomain
```

## **Shell.Application**

```
$wshell = New-Object -ComObject Shell.Application  
$wshell | Get-Member  
$wshell.Explore("C:\")  
$wshell.Windows() | Get-Member ПОЛУЧИТЬ ДОСТУП К ОТКРЫТЫМ В ПРОВОДНИКЕ ИЛИ БРАУЗЕРЕ Internet Explorer ОКОН  
  
$shell = New-Object -Com Shell.Application  
$RecycleBin = $shell.Namespace(10)  
$RecycleBin.Items()
```

## **Outlook**

```
$outlook = New-Object -ComObject Outlook.Application  
$outlook | Get-Member  
$outlook.Version  
  
$outlook = New-Object -ComObject Outlook.Application  
$Namespace = $outlook.GetNamespace("MAPI")  
$Folder = $Namespace.GetDefaultFolder(4)` исходящие  
$Folder = $Namespace.GetDefaultFolder(6)` входящие  
$Explorer = $Folder.GetExplorer()  
$Explorer.Display()  
$outlook.Quit()
```

## **Microsoft.Update**

```
(New-Object -com 'Microsoft.Update.AutoUpdate').Settings  
(New-Object -com 'Microsoft.Update.AutoUpdate').Results  
(New-Timespan -Start ((New-Object -com 'Microsoft.Update.AutoUpdate').Results|Select -ExpandProperty LastInstallationSuccessDate) -End (Get-Date)).  
КОЛ-ВО ЧАСОВ, ПРОШЕДШИХ С ПОСЛЕДНЕЙ ДАТЫ УСТАНОВКИ ОБНОВЛЕНИЯ БЕЗОПАСНОСТИ В Windows.
```

## **.NET**

```
[System.Diagnostics.EventLog] | select Assembly,Module  
$EventLog = [System.Diagnostics.EventLog]::new("Application")  
$EventLog = New-Object -TypeName System.Diagnostics.EventLog -ArgumentList Application,192.168.3.100  
$EventLog | Get-Member -MemberType Method  
$EventLog.MaximumKilobytes МАКСИМАЛЬНЫЙ РАЗМЕР ЖУРНАЛА  
$EventLog.Entries ПРОСМОТРЕТЬ ЖУРНАЛ  
$EventLog.Clear() ОЧИСТИТЬ ЖУРНАЛ  
  
Join-Path C: Install Test  
[System.IO.Path]::Combine("C:", "Install", "Test")
```

## **Match**

```
[System.Math] | Get-Member -Static -MemberType Methods  
[System.Math]::Max(2,7)  
[System.Math]::Min(2,7)
```

```
[System.Math]::Floor(3.9)
[System.Math]::Truncate(3.9)
```

## GeneratePassword

```
Add-Type -AssemblyName System.Web
[System.Web.Security.Membership]::GeneratePassword(10,2)
```

## SoundPlayer

```
$CriticalSound = New-Object System.Media.SoundPlayer
$CriticalSound.SoundLocation = "C:\WINDOWS\Media\Windows_Critical_Stop.wav"
$CriticalSound.Play()

$GoodSound = New-Object System.Media.SoundPlayer
$GoodSound.SoundLocation = "C:\WINDOWS\Media\tada.wav"
$GoodSound.Play()
```

## Static Class

```
[System.Environment] | Get-Member -Static
[System.Environment]::OSVersion
[System.Environment]::Version
[System.Environment]::MachineName
[System.Environment]::UserName

[System.Diagnostics.Process] | Get-Member -Static
[System.Diagnostics.Process]::Start('notepad.exe')
```

## Clicker

```
$cSource = @'
using System;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Windows.Forms;
public class Clicker
{
//https://msdn.microsoft.com/en-us/library/windows/desktop/ms646270\(v=vs.85\).aspx
[StructLayout(LayoutKind.Sequential)]
struct INPUT
{
    public int      type; // 0 = INPUT_MOUSE,
                     // 1 = INPUT_KEYBOARD
                     // 2 = INPUT_HARDWARE
    public MOUSEINPUT mi;
}
//https://msdn.microsoft.com/en-us/library/windows/desktop/ms646273\(v=vs.85\).aspx
[StructLayout(LayoutKind.Sequential)]
struct MOUSEINPUT
{
    public int      dx ;
    public int      dy ;
    public int      mouseData ;
    public int      dwFlags;
    public int      time;
    public IntPtr   dwExtraInfo;
}
//This covers most use cases although complex mice may have additional buttons
//There are additional constants you can use for those cases, see the msdn page
const int MOUSEEVENTF_MOVED      = 0x0001 ;
const int MOUSEEVENTF_LEFTDOWN   = 0x0002 ;
const int MOUSEEVENTF_LEFTUP     = 0x0004 ;
const int MOUSEEVENTF_RIGHTDOWN  = 0x0008 ;
const int MOUSEEVENTF_RIGHTUP    = 0x0010 ;
const int MOUSEEVENTF_MIDDLEDOWN = 0x0020 ;
const int MOUSEEVENTF_MIDDLEUP   = 0x0040 ;
const int MOUSEEVENTF_WHEEL      = 0x0080 ;
const int MOUSEEVENTF_XDOWN      = 0x0100 ;
const int MOUSEEVENTF_XUP        = 0x0200 ;
const int MOUSEEVENTF_ABSOLUTE   = 0x8000 ;
const int screen_length         = 0x10000 ;
//https://msdn.microsoft.com/en-us/library/windows/desktop/ms646310\(v=vs.85\).aspx
[System.Runtime.InteropServices.DllImport("user32.dll")]
extern static uint SendInput(uint nInputs, INPUT[] pInputs, int cbSize);
public static void LeftClickAtPoint(int x, int y)
{
    //Move the mouse
    INPUT[] input = new INPUT[3];
    input[0].mi.dx = x*(65535/System.Windows.Forms.Screen.PrimaryScreen.Bounds.Width);
    input[0].mi.dy = y*(65535/System.Windows.Forms.Screen.PrimaryScreen.Bounds.Height);
    input[0].mi.dwFlags = MOUSEEVENTF_MOVED | MOUSEEVENTF_ABSOLUTE;
    //Left mouse button down
    input[1].mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
    //Left mouse button up
    input[2].mi.dwFlags = MOUSEEVENTF_LEFTUP;
    SendInput(3, input, Marshal.SizeOf(input[0]));
}
}
'@
```

```
Add-Type -TypeDefintion $cSource -ReferencedAssemblies System.Windows.Forms, System.Drawing
[Clicker]::LeftClickAtPoint(1900,1070)
```

## Audio

```
Add-Type -Language CsharpVersion3 -TypeDefinition @"
using System.Runtime.InteropServices;
[Guid("5CDF2C82-841E-4546-9722-0CF74078229A"), InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IAudioEndpointVolume {
    // f(), g(), ... are unused COM method slots. Define these if you care
    int f(); int g(); int h(); int i();
    int SetMasterVolumeLevelScalar(float fLevel, System.Guid pguidEventContext);
    int j();
    int GetMasterVolumeLevelScalar(out float pfLevel);
    int k(); int l(); int m(); int n();
    int SetMute([MarshalAs(UnmanagedType.Bool)] bool bMute, System.Guid pguidEventContext);
    int GetMute(out bool pbMute);
}
[Guid("D666063F-1587-4E43-81F1-B948E807363F"), InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IMMDevice {
    int Activate(ref System.Guid id, int clsCtx, int activationParams, out IAudioEndpointVolume aev);
}
[Guid("A95664D2-9614-4F35-A746-DE8DB63617E6"), InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IMMDeviceEnumerator {
    int f(); // Unused
    int GetDefaultAudioEndpoint(int dataFlow, int role, out IMMDevice endpoint);
}
[ComImport, Guid("BCDE0395-E52F-467C-8E3D-C4579291692E")]
class MMDeviceEnumeratorComObject { }
public class Audio {
    static IAudioEndpointVolume Vol() {
        var enumerator = new MMDeviceEnumeratorComObject() as IMMDeviceEnumerator;
        IMMDevice dev = null;
        Marshal.ThrowExceptionForHR(enumerator.GetDefaultAudioEndpoint(/*eRender*/ 0, /*eMultimedia*/ 1, out dev));
        IAudioEndpointVolume epv = null;
        var epvid = typeof(IAudioEndpointVolume).GUID;
        Marshal.ThrowExceptionForHR(dev.Activate(ref epvid, /*CLSCTX_ALL*/ 23, 0, out epv));
        return epv;
    }
    public static float Volume {
        get {float v = -1; Marshal.ThrowExceptionForHR(Vol().GetMasterVolumeLevelScalar(out v)); return v;}
        set {Marshal.ThrowExceptionForHR(Vol().SetMasterVolumeLevelScalar(value, System.Guid.Empty));}
    }
    public static bool Mute {
        get { bool mute; Marshal.ThrowExceptionForHR(Vol().GetMute(out mute)); return mute; }
        set { Marshal.ThrowExceptionForHR(Vol().SetMute(value, System.Guid.Empty)); }
    }
}
@"
[Audio]::Volume = 0.50
[Audio]::Mute = $true
```

## NetSessionEnum

[Function](#)

[Source](#)

```

function Invoke-NetSessionEnum {
    param (
        [Parameter(Mandatory = $True)][string]$HostName
    )
    Add-Type -TypeDefinition @"
        using System;
        using System.Diagnostics;
        using System.Runtime.InteropServices;
        [StructLayout(LayoutKind.Sequential)]
        public struct SESSION_INFO_10
    {
        [MarshalAs(UnmanagedType.LPWStr)]public string OriginatingHost;
        [MarshalAs(UnmanagedType.LPWStr)]public string DomainUser;
        public uint SessionTime;
        public uint IdleTime;
    }
    public static class Netapi32
    {
        [DllImport("Netapi32.dll", SetLastError=true)]
        public static extern int NetSessionEnum(
            [In,MarshalAs(UnmanagedType.LPWStr)] string ServerName,
            [In,MarshalAs(UnmanagedType.LPWStr)] string UncClientName,
            [In,MarshalAs(UnmanagedType.LPWStr)] string UserName,
            Int32 Level,
            out IntPtr bufptr,
            int prefmaxlen,
            ref Int32 entriesread,
            ref Int32 totalentries,
            ref Int32 resume_handle);

        [DllImport("Netapi32.dll", SetLastError=true)]
        public static extern int NetApiBufferFree(
            IntPtr Buffer);
    }
    "@
    # Create SessionInfo10 Struct
    $SessionInfo10 = New-Object SESSION_INFO_10
    $SessionInfo10StructSize = [System.Runtime.InteropServices.Marshal]::SizeOf($SessionInfo10)` Grab size to loop bufptr
    $SessionInfo10 = $SessionInfo10.GetType()` Hacky, but we need this ;)
    # NetSessionEnum params
    $OutBuffPtr = [IntPtr]::Zero` Struct output buffer
    $EntriesRead = $TotalEntries = $ResumeHandle = 0` Counters & ResumeHandle
    $CallResult = [Netapi32]::NetSessionEnum($HostName, "", "", 10, [ref]$OutBuffPtr, -1, [ref]$EntriesRead, [ref]$TotalEntries, [ref]$ResumeHandle)
    if ($CallResult -ne 0){
        echo "Mmm something went wrong!\nError Code: $CallResult"
    }
    else {
        if ([System.IntPtr]::Size -eq 4) {
            echo "`nNetapi32::NetSessionEnum Buffer Offset --> 0x$("{0:X8}" -f $OutBuffPtr.ToInt32())"
        }
        else {
            echo "`nNetapi32::NetSessionEnum Buffer Offset --> 0x$("{0:X16}" -f $OutBuffPtr.ToInt64())"
        }
        echo "Result-set contains $EntriesRead session(s)!"
        # Change buffer offset to int
        $BufferOffset = $OutBuffPtr.ToInt64()
        # Loop buffer entries and cast pointers as SessionInfo10
        for ($Count = 0; ($Count -lt $EntriesRead); $Count++){
            $NewIntPtr = New-Object System.IntPtr -ArgumentList $BufferOffset
            $Info = [System.Runtime.InteropServices.Marshal]::PtrToStructure($NewIntPtr,[type]$SessionInfo10)
            $Info
            $BufferOffset = $BufferOffset + $SessionInfo10StructSize
        }
        echo "`nCalling NetApiBufferFree, no memleaks here!"
        [Netapi32]::NetApiBufferFree($OutBuffPtr) |Out-Null
    }
}

```

```
}
```

```
Invoke-NetSessionEnum localhost
```

## CopyFile

[Function](#)

[Source](#)

```
$MethodDefinition = @"
[DllImport("kernel32.dll", CharSet = CharSet.Unicode)]
public static extern bool CopyFile(string lpExistingFileName, string lpNewFileName, bool bFailIfExists);
"@
$Kernel32 = Add-Type -MemberDefinition $MethodDefinition -Name "Kernel32" -Namespace "Win32" -PassThru
$Kernel32::CopyFile("${Env:SystemRoot}\System32\calc.exe", "${Env:USERPROFILE}\Desktop\calc.exe", $False)
```

## ShowWindowAsync

[Function](#)

```
$Signature = @"
[DllImport("user32.dll")]public static extern bool ShowWindowAsync(IntPtr hWnd, int nCmdShow);
"@
$ShowWindowAsync = Add-Type -MemberDefinition $Signature -Name "Win32ShowWindowAsync" -Namespace Win32Functions -PassThru
$ShowWindowAsync | Get-Member -Static
$ShowWindowAsync::ShowWindowAsync((Get-Process -Id $pid).MainWindowHandle, 2)
$ShowWindowAsync::ShowWindowAsync((Get-Process -Id $Pid).MainWindowHandle, 3)
$ShowWindowAsync::ShowWindowAsync((Get-Process -Id $Pid).MainWindowHandle, 4)
```

## GetAsyncKeyState

[Function](#)

```
Add-Type -AssemblyName System.Windows.Forms
[int][System.Windows.Forms.Keys]::F1 определить номер [Int] клавиши по ее названию
65..90 | % {"{0} = {1}" -f $_, [System.Windows.Forms.Keys]$_. порядковый номер букв (A..Z)
```

```
function Get-ControlKey {
$key = 112
$Signature = '@'
[DllImport("user32.dll", CharSet=CharSet.Auto, ExactSpelling=true)]
public static extern short GetAsyncKeyState(int virtualKeyCode);
'@
Add-Type -MemberDefinition $Signature -Name Keyboard -Namespace PsOneApi
[bool]([PsOneApi.Keyboard]::GetAsyncKeyState($key) -eq -32767)
}

Write-Warning 'Press F1 to exit'
while ($true) {
Write-Host '.' -NoNewline
if (Get-ControlKey) {
break
}
Start-Sleep -Seconds 0.5
}
```

## Console API

[Source](#)

```

[Console] | Get-Member -Static
[Console]::BackgroundColor = "Blue"
[Console]::OutputEncoding используемая кодировка в текущей сессии
[Console]::OutputEncoding = [System.Text.Encoding]::GetEncoding("utf-8") изменить кодировку для отображения кириллицы
[Console]::OutputEncoding = [System.Text.Encoding]::GetEncoding("cp866") для ISE
[Console]::OutputEncoding = [System.Text.Encoding]::GetEncoding("windows-1251") для ps2exe
Get-Service | Out-File $home\Desktop\Service.txt -Encoding oem >
Get-Service | Out-File $home\Desktop\Service.txt -Append >>

do {
    if ([Console]::KeyAvailable) {
        $keyInfo = [Console]::ReadKey($true)
        break
    }
    Write-Host "." -NoNewline
    Start-Sleep 1
} while ($true)
Write-Host
$keyInfo

function Get-KeyPress {
    param (
        [Parameter(Mandatory)][ConsoleKey]$Key,
        [System.ConsoleModifiers]$ModifierKey = 0
    )
    if ([Console]::KeyAvailable) {
        $pressedKey = [Console]::ReadKey($true)
        $isPressedKey = $key -eq $pressedKey.Key
        if ($isPressedKey) {
            $pressedKey.Modifiers -eq $ModifierKey
        }
        else {
            [Console]::Beep(1800, 200)
            $false
        }
    }
}
}

Write-Warning 'Press Ctrl+Shift+Q to exit'
do {
    Write-Host "." -NoNewline
    $pressed = Get-KeyPress -Key Q -ModifierKey 'Control,Shift'
    if ($pressed) { break }
    Start-Sleep 1
} while ($true)

```

## Drawing

[API](#)

```

Add-Type -AssemblyName System.Drawing
$Width = 800
$Height = 400
$image = New-Object System.Drawing.Bitmap($Width,$Height)
$graphic = [System.Drawing.Graphics]::FromImage($image)
$background_color = [System.Drawing.Brushes]::Blue # задать цвет фона (синий)
$graphic.FillRectangle($background_color, 0, 0, $image.Width, $image.Height)
$text_color = [System.Drawing.Brushes]::White # задать цвет текста (белый)
$font = New-Object System.Drawing.Font("Arial", 20, [System.Drawing.FontStyle]::Bold) # задать шрифт
$text = "PowerShell" # указать текст
$text_position = New-Object System.Drawing.RectangleF(320, 180, 300, 100) # задать положение текста (x, y, width, height)
$graphic.DrawString($text, $font, $text_color, $text_position) # нанести текст на изображение
$image.Save("$home\desktop\powershell_image.bmp", [System.Drawing.Imaging.ImageFormat]::Bmp) # сохранить изображение
$image.Dispose() # освобождение ресурсов

$path = "$home\desktop\powershell_image.bmp"
Invoke-Item $path

$src_image = [System.Drawing.Image]::FromFile($path)
$Width = 400
$Height = 200
$dst_image = New-Object System.Drawing.Bitmap -ArgumentList $src_image, $Width, $Height # изменить размер изображения
$dst_image.Save("$home\desktop\powershell_image_resize.bmp", [System.Drawing.Imaging.ImageFormat]::Bmp)

$rotated_image = $src_image.Clone() # создать копию исходного изображения
$rotated_image.RotateFlip([System.Drawing.RotateFlipType]::Rotate180FlipNone) # перевернуть изображение на 180 градусов
$rotated_image.Save("$home\desktop\powershell_image_rotated.bmp", [System.Drawing.Imaging.ImageFormat]::Bmp)
$src_image.Dispose() # закрыть (отпустить) исходный файл

```

## ObjectEvent

```

$Timer = New-Object System.Timers.Timer
$Timer.Interval = 1000
Register-ObjectEvent -InputObject $Timer -EventName Elapsed -SourceIdentifier Timer.Output -Action {
$Random = Get-Random -Min 0 -Max 100
Write-Host $Random
}
$Timer.Enabled = $True

$Timer.Enabled = $False остановить
$Timer | Get-Member -MemberType Event отобразить список всех событий объекта
Get-EventSubscriber СПИСОК зарегистрированных подписок на события в текущей сессии
Unregister-Event -SourceIdentifier Timer.Output удаляет регистрацию подписки на событие по имени события (EventName) или все *
-Forward перенаправляет события из удаленного сеанса (New-PSSession) в локальный сеанс
-SupportEvent не выводит результат регистрации события на экран (и Get-EventSubscriber и Get-Job)

```

```

Register-EngineEvent -SourceIdentifier PowerShell.Exiting -Action {
$date = Get-Date -f hh:mm:ss
(New-Object -ComObject Wscript.Shell).Popup("PowerShell Exit: $date",0,"Action",64)
}

```

## Sockets

### UDP-Socket

[Source](#)

```

function Start-UDPServer {
    param(
        $Port = 5201
    )
    $RemoteComputer = New-Object System.Net.IPEndPoint([System.Net.IPAddress]::Any, 0)
    do {
        $UdpObject = New-Object System.Net.Sockets.UdpClient($Port)
        $ReceiveBytes = $UdpObject.Receive([ref]$RemoteComputer)
        $UdpObject.Close()
        $ASCIIEncoding = New-Object System.Text.ASCIIEncoding
        [string]$ReturnString = $ASCIIEncoding.GetString($ReceiveBytes)
        [PSCustomObject]@{
            LocalDateTime = $(Get-Date -UFormat "%Y-%m-%d %T")
            ClientIP     = $RemoteComputer.Address.ToString()
            ClientPort   = $RemoteComputer.Port.ToString()
            Message      = $ReturnString
        }
    } while (1)
}

```

Start-UDPServer -Port 5201

## Test-NetUDPConnection

```

function Test-NetUDPConnection {
    param(
        [string]$ComputerName = "127.0.0.1",
        [int32]$PortServer = 5201,
        [int32]$PortClient = 5211,
        $Message
    )
    begin {
        $UdpObject = New-Object System.Net.Sockets.UdpClient($PortClient)
        $UdpObject.Connect($ComputerName, $PortServer)
    }
    process {
        $ASCIIEncoding = New-Object System.Text.ASCIIEncoding
        if (!$Message) { $Message = Get-Date -UFormat "%Y-%m-%d %T" }
        $Bytes = $ASCIIEncoding.GetBytes($Message)
        [void]$UdpObject.Send($Bytes, $Bytes.length)
    }
    end {
        $UdpObject.Close()
    }
}

```

Test-NetUDPConnection -ComputerName 127.0.0.1 -PortServer 5201

Test-NetUDPConnection -ComputerName 127.0.0.1 -PortServer 514 -Message "<30>May 31 00:00:00 HostName multipathd[784]: Test message"

## TCP-Socket

```
function Start-TCPserver {
    param(
        $Port = 5201
    )
    do {
        $TcpObject = New-Object System.Net.Sockets.TcpListener($port)
        $ReceiveBytes = $TcpObject.Start()
        $ReceiveBytes = $TcpObject.AcceptTcpClient()
        $TcpObject.Stop()
        $ReceiveBytes.Client.RemoteEndPoint | select Address, Port
    } while (1)
}
```

```
Start-TCPserver -Port 5201
Test-NetConnection -ComputerName 127.0.0.1 -Port 5201
```

## WakeOnLan

Broadcast package consisting of 6 byte filled "0xFF" and then 96 byte where the mac address is repeated 16 times

```
function Send-WOL {
    param (
        [Parameter(Mandatory = $True)]$Mac,
        $IP,
        [int]$Port = 9
    )
    $Mac = $Mac.replace(":", "-")
    if (!$IP) { $IP = [System.Net.IPAddress]::Broadcast }
    $SynchronizationChain = [byte[]](, 0xFF * 6)
    $ByteMac = $Mac.Split("-") | % { [byte]("0x" + $_) }
    $Package = $SynchronizationChain + ($ByteMac * 16)
    $UdpClient = New-Object System.Net.Sockets.UdpClient
    $UdpClient.Connect($IP, $port)
    $UdpClient.Send($Package, $Package.Length)
    $UdpClient.Close()
}
```

```
Send-WOL -Mac "D8-BB-C1-70-A3-4E"
Send-WOL -Mac "D8-BB-C1-70-A3-4E" -IP 192.168.3.100
```

## HTTPListener

```
$httpListener = New-Object System.Net.HttpListener
$httpListener.Prefixes.Add("http://+:8888/")
$httpListener.Start()
while (!([console]::KeyAvailable)) {
    $info = Get-Service | select name, status | ConvertTo-HTML
    $context = $httpListener.GetContext()
    $context.Response.StatusCode = 200
    $context.Response.ContentType = 'text/HTML'
    $WebContent = $info
    $EncodingWebContent = [Text.Encoding]::UTF8.GetBytes($WebContent)
    $context.Response.OutputStream.Write($EncodingWebContent , 0, $EncodingWebContent.Length)
    $context.Response.Close()
    Get-NetTcpConnection -LocalPort 8888
    (Get-Date).datetime
}
$httpListener.Close()
```

## WebClient

```
[System.Net.WebClient] | Get-Member  
(New-Object Net.WebClient).DownloadString("https://raw.githubusercontent.com/Lifailon/PowerShell-Commands/rsa/README.md")
```

## HttpClient

```
$url = "https://github.com/PowerShell/PowerShell/releases/download/v7.4.2/PowerShell-7.4.2-win-x64.zip"  
$path = "$home\Downloads\$($Split-Path -Path $url -Leaf)"  
$httpClient = [System.Net.Http.HttpClient]::new()  
# Выполнение GET-запроса для загрузки файла (считывая заголовки ответа)  
$response = $httpClient.GetAsync($url, [System.Net.Http.HttpCompletionOption]::ResponseHeadersRead).Result  
# Получение потока содержимого из заголовка ответа  
$stream = $response.Content.ReadAsStreamAsync().Result  
# Открытие файла для записи  
$fileStream = [System.IO.File]::OpenWrite($path)  
try {  
    # Создание буфера размером 81920 байт (80 КБ) для чтения данных из потока  
    $buffer = New-Object byte[] 81920  
    # Чтение данных из потока и запись их в файл  
    while (($bytesRead = $stream.Read($buffer, 0, $buffer.Length)) -ne 0) {  
        $fileStream.Write($buffer, 0, $bytesRead)  
    }  
}  
finally {  
    # Освобождение ресурсов, связанных с потоками  
    $stream.Dispose()  
    $fileStream.Dispose()  
}
```

## Certificate

```
function Get-WebCertificate ($srv) {  
    $iwr = iwr $srv  
    $status_code = $iwr.StatusCode  
    $status = $iwr.BaseResponse.StatusCode  
    $info = $iwr.BaseResponse.Server  
    $spm = [System.Net.ServicePointManager]::FindServicePoint($srv)  
    $date_end = $spm.Certificate.GetExpirationDateString()  
    $cert_name = ($spm.Certificate.Subject) -replace "CN="  
    $cert_owner = (((($spm.Certificate.Issuer) -split ", ") | where { $_ -match "O=" }) -replace "O=")  
    $Collections = New-Object System.Collections.Generic.List[System.Object]  
    $Collections.Add([PSCustomObject]@{  
        Host      = $srv;  
        Server    = $info;  
        Status    = $status;  
        StatusCode = $status_code;  
        Certificate = $cert_name;  
        Issued    = $cert_owner;  
        End       = $date_end  
    })  
    $Collections  
}
```

```
Get-WebCertificate https://google.com
```

## Base64

### UTF8

```
$loginPassword = "login:password"
$Base64 = [Convert]::ToBase64String([System.Text.Encoding]::UTF8.GetBytes($loginPassword)) закодировать логин и пароль в строку Base64
[System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($Base64)) преобразовать в байты и обратно декодировать в исходную
строку с помощью UTF-8 кодировки
```

### Unicode

```
$text = "password"
$byte = [System.Text.Encoding]::Unicode.GetBytes($text) # преобразует строку $text в последовательность байтов, используя кодировку Unicode
$base64 = [System.Convert]::ToBase64String($byte) # байты конвертируются в строку Base64 с помощью метода ToBase64String
$decode_base64 = [System.Convert]::FromBase64String($base64) # декодировать строку Base64 обратно в последовательность байтов с помощью метода Fr
$decode_string = [System.Text.Encoding]::Unicode.GetString($decode_base64) # закодированные байты преобразуются обратно в строку с использованием
```

### Image

```
$path_image = "$home\Documents\1200x800.jpg"
$BBase64 = [System.Convert]::ToBase64String((Get-Content $path_image -Encoding Byte))
Add-Type -assembly System.Drawing
$image = [System.Drawing.Bitmap]::FromStream([IO.MemoryStream][Convert]::FromBase64String($BBase64))
$image.Save("$home\Desktop\1200x800.jpg")
```

# Excel

```
$path = "$home\Desktop\Services-to-Excel.xlsx"
$Excel = New-Object -ComObject Excel.Application
$Excel.Visible = $false` отключить открытие GUI
$ExcelWorkBook = $Excel.Workbooks.Add()` Создать книгу
$ExcelWorkSheet = $ExcelWorkBook.Worksheets.Item(1)` Создать лист
$ExcelWorkSheet.Name = "Services"` задать имя листа
$ExcelWorkSheet.Cells.Item(1,1) = "Name service"
# Задать имена столбцов:
$ExcelWorkSheet.Cells.Item(1,2) = "Description"
$ExcelWorkSheet.Cells.Item(1,3) = "Status"
$ExcelWorkSheet.Cells.Item(1,4) = "Startup type"
$ExcelWorkSheet.Rows.Item(1).Font.Bold = $true` выделить жирным шрифтом
$ExcelWorkSheet.Rows.Item(1).Font.size=14
# Задать ширину колонок:
$ExcelWorkSheet.Columns.Item(1).ColumnWidth=30
$ExcelWorkSheet.Columns.Item(2).ColumnWidth=80
$ExcelWorkSheet.Columns.Item(3).ColumnWidth=15
$ExcelWorkSheet.Columns.Item(4).ColumnWidth=25
$services = Get-Service
$count = 2` задать начальный номер строки для записи
foreach ($service in $services) {
    $status = $service.Status
    if ($status -eq 1) {
        $status_type = "Stopped"
    } elseif ($status -eq 4) {
        $status_type = "Running"
    }
    $Start = $service.StartType
    if ($Start -eq 1) {
        $start_type = "Delayed start"
    } elseif ($Start -eq 2) {
        $start_type = "Automatic"
    } elseif ($Start -eq 3) {
        $start_type = "Manually"
    } elseif ($Start -eq 4) {
        $start_type = "Disabled"
    }
    $ExcelWorkSheet.Columns.Item(1).Rows.Item($count) = $service.Name
    $ExcelWorkSheet.Columns.Item(2).Rows.Item($count) = $service.DisplayName
    $ExcelWorkSheet.Columns.Item(3).Rows.Item($count) = $status_type
    $ExcelWorkSheet.Columns.Item(4).Rows.Item($count) = $start_type
    if ($status_type -eq "Running") {
        $ExcelWorkSheet.Columns.Item(3).Rows.Item($count).Font.Bold = $true
    }
    # +1 увеличить для счетчика строки Rows
    $counter++
}
$ExcelWorkBook.SaveAs($path)
$ExcelWorkBook.Close($true)
$Excel.Quit()
```

## **Excel.Application.Open**

```
$path = "$home\Desktop\Services-to-Excel.xlsx"
$Excel = New-Object -ComObject Excel.Application
$Excel.Visible = $false
$ExcelWorkBook = $excel.Workbooks.Open($path)` открыть xlsx-файл
$ExcelWorkBook.Sheets | select Name,Index` отобразить листы
$ExcelWorkSheet = $ExcelWorkBook.Sheets.Item(1)` открыть лист по номеру Index
1..100 | %{$ExcelWorkSheet.Range("A$_").Text}` прочитать значение из столбца A строки с 1 по 100
$Excel.Quit()
```

## **ImportExcel**

```
Install-Module -Name ImportExcel
$data | Export-Excel .\Data.xlsx
$data = Import-Excel .\Data.xlsx

$data = ps
$Chart = New-ExcelChartDefinition -XRange CPU -YRange WS -Title "Process" -NoLegend
$data | Export-Excel .\ps.xlsx -AutoNameRange -ExcelChartDefinition $Chart -Show
```

## **CSV**

```
Get-Service | Select Name,DisplayName,Status,StartType | Export-Csv -path "$home\Desktop\Get-Service.csv" -Append -Encoding Default
экспортировать в csv (-Encoding UTF8)
Import-Csv "$home\Desktop\Get-Service.csv" -Delimiter "," импортировать массив
```

```
$data = ConvertFrom-Csv @"
Region,State,Units,Price
West,Texas,927,923.71
$null,Tennessee,466,770.67
@"
$systeminfo = systeminfo /FO csv | ConvertFrom-Csv ВЫВОД работы программы в CSV и конвертация в объект
$systeminfo."Полный объем физической памяти"
$systeminfo."Доступная физическая память"
```

## **ConvertFrom-String**

```
'  
log =  
{  
    level = 4;  
};  
' | ConvertFrom-String` создает PSCustomObject (разбивает по пробелам, удаляет все пробелы и пустые строки)
```

## **ConvertFrom-StringData**

```
"  
key1 = value1  
key2 = value2  
" | ConvertFrom-StringData # создает Hashtable
```

## XML

```
$xml = [xml](Get-Content $home\desktop\test.rdg)` прочитать содержимое XML-файла
$xml.load("$home\desktop\test.rdg")` открыть файл
$xml.RDCMan.file.group.properties.name` имена групп
$xml.RDCMan.file.group.server.properties` имена всех серверов
$xml.RDCMan.file.group[3].server.properties` список серверов в 4-й группе
($xml.RDCMan.file.group[3].server.properties | ? name -like ADIRK).Name = "New-Name"` изменить значение
$xml.RDCMan.file.group[3].server[0].properties.displayName = "New-displayName"
$xml.RDCMan.file.group[3].server[1].RemoveAll()` удалить объект (2-й сервер в списке)
$xml.Save($file)` сохранить содержимое объекта в файла
```

```
Get-Service | Export-Clixml -path $home\desktop\test.xml экспортировать объект PowerShell в XML
Import-Clixml -Path $home\desktop\test.xml импортировать объект XML в PowerShell
ConvertTo-Xml (Get-Service)
```

## Get-CredToXML

```
function Get-CredToXML {
    param (
        $CredFile = "$home\Documents\cred.xml"
    )
    if (Test-Path $CredFile) {
        Import-Clixml -path $CredFile
    }
    elseif (!(Test-Path $CredFile)) {
        $Cred = Get-Credential -Message "Enter credential"
        if ($Cred -ne $null) {
            $Cred | Export-Clixml -Path $CredFile
            $Cred
        }
    }
    else {
        return
    }
}
}

$Cred = Get-CredToXML
$Login = $Cred.UserName
$PasswordText = $Cred.GetNetworkCredential().password ПОЛУЧИТЬ пароль в текстовом виде
```

## XmIWriter

Extensible Markup Language

```

$xmlWriterSettings = New-Object System.Xml.XmlWriterSettings
$xmlWriterSettings.Indent = $true` включить отступы
$xmlWriterSettings.IndentChars = "    "` задать отступ

$xmlFilePath = "$home\Desktop\test.xml"
$xmlObjectWriter = [System.XML.XmlWriter]::Create($xmlFilePath, $xmlWriterSettings)` создать документ
$xmlObjectWriter.WriteStartDocument()` начать запись в документ

$xmlObjectWriter.WriteComment("Comment")
$xmlObjectWriter.WriteStartElement("Root")` создать стартовый элемент, который содержит дочерние объекты
    $xmlObjectWriter.WriteStartElement("Configuration")` создать первый дочерний элемент для BaseSettings
        $xmlObjectWriter.WriteLineString("Language", "RU")
        $xmlObjectWriter.WriteStartElement("Fonts")# <Fonts>
            $xmlObjectWriter.WriteLineString("Name", "Arial")
            $xmlObjectWriter.WriteLineString("Size", "12")
        $xmlObjectWriter.WriteEndElement()# </Fonts>
    $xmlObjectWriter.WriteEndElement()` конечный элемент </Configuration>
$xmlObjectWriter.WriteEndElement()` конечный элемент </Root>

$xmlObjectWriter.WriteEndDocument()` завершить запись в документ
$xmlObjectWriter.Flush()
$xmlObjectWriter.Close()

```

## CreateElement

```

$xml = [xml](gc $home\Desktop\test.xml)
$xml.Root.Configuration.Fonts
$newElement = $xml.CreateElement("Fonts")` выбрать элемент куда добавить
$newElement.set_InnerXML("<Name>Times New Roman</Name><Size>14</Size>")` Заполнить значениями дочерние элементы Fonts
$xml.Root.Configuration.AppendChild($newElement)` добавить элемент новой строкой в Configuration (родитель Fonts)
$xml.Save("$home\Desktop\test.xml")

```

## JSON

```

$log = '
{
    "log": {
        "level": 7
    }
}
' | ConvertFrom-Json

Get-Service | ConvertTo-Json

$00KLA = '
{
    "result" :
    {"date":1683534970,"id":14708271987,"connection_icon":"wireless","download":33418,"upload":35442,"latency":15,"distance":50,"country_code": "RU"
    [{"server_id":8191,"server_name": "Bryansk","sponsor_name": "SectorTelecom.ru"}, {"server_id":46278,"server_name": "Fokino","sponsor_name": "Fokks - P
    "path": "result\u002F14708271987","hasSecondary":true
    }
    ]
}
' | ConvertFrom-Json
$ookla.result

```

## YAML

```
Import-Module PSYaml` используется в Docker/Ansible
$netplan =
network: ` словарь по типу - ключ : значение с вложенными словарями
ethernets:
  ens160:
    dhcp4: yes
    dhcp6: no
    nameservers:
      addresses: `[8.8.8.8, 1.1.1.1]` список данных (строк)
        - 8.8.8.8
        - 1.1.1.1
    version: 2
"
$network = ConvertFrom-Yaml $netplan
$network.Values.ethernets.ens160.nameservers

$DataType =
int: !!int 10.1
flo: !!float 10.1
str: !!str string
bool: !!bool` boolean
"
```

## TOML

```
Install-Module -Name PSToml -Scope CurrentUser Устанавливаем модуль для чтения toml (PSToml)
$toml = Get-Content "$home\Documents\Git\lifailon.github.io\hugo.toml" Читаем содержимое конфигурации Hugo в формате toml
$json = ConvertFrom-Toml $toml | ConvertTo-Json -Depth 3 Конвертируем TOML в JSON
$json | Out-File "$home\Documents\Git\lifailon.github.io\hugo.json" Сохраняем конфигурационный файл в формате JSON
```

## Markdown

### Convert Markdown to HTML

```
function ConvertFrom-MarkdownToHtml {
    param (
        [Parameter(Mandatory = $true, ValueFromPipeline = $true)]$Markdown
    )
    $html = $(Get-Content index.md -Raw | ConvertFrom-Markdown).html
    @@
    <!DOCTYPE html>
    <html>
    <head>
    </head>
    <body>
    $html
    </body>
    </html>
    @@
}
```

```
Get-Content index.md -Raw | ConvertFrom-MarkdownToHtml | Out-File index.html
```

## PSMarkdown

```
Install-Module PSMarkdown
```

## ConvertFrom-MarkdownV2

```
Function ConvertFrom-MarkdownV2 {
    [CmdletBinding()]
    [OutputType([PSObject])]
    Param (
        [Parameter(
            Mandatory = $true,
            Position = 0,
            ValueFromPipeline = $true
        )]
        $InputObject
    )
    Begin {
        $items = @()
    }
    Process {
        $mddata = $InputObject
        $data = $mddata | Where-Object {$_. -notmatch "--" }
        $items += $data
    }
    End {
        $object = $items -replace ' +', '' | ConvertFrom-Csv -Delimiter '|'
        $object
    }
}
```

## ConvertTo-Markdown

```
Function ConvertTo-Markdown {
    [CmdletBinding()]
    [OutputType([string])]
    Param (
        [Parameter(
            Mandatory = $true,
            Position = 0,
            ValueFromPipeline = $true
        )]
        [PSObject[]]$InputObject
    )
    Begin {
        $items = @()
        $columns = @{}
    }
    Process {
        ForEach($item in $InputObject) {
            $items += $item
            $item.PSObject.Properties | ForEach-Object {
                if($null -ne $_.Value) {
                    if(-not $columns.ContainsKey($_.Name) -or $columns[$_.Name] -lt $_.Value.ToString().Length) {
                        $columns[$_.Name] = $_.Value.ToString().Length
                    }
                }
            }
        }
    }
    End {
        ForEach($key in $($columns.Keys)) {
            $columns[$key] = [Math]::Max($columns[$key], $key.Length)
        }
        $header = @()
        ForEach($key in $columns.Keys) {
            $header += ('{0,-' + $columns[$key] + '}') -f $key
        }
        $header -join ' '
        $separator = @()
        ForEach($key in $columns.Keys) {
            $separator += '-' * $columns[$key]
        }
        $separator -join ' '
        ForEach($item in $items) {
            $values = @()
            ForEach($key in $columns.Keys) {
                $values += ('{0,-' + $columns[$key] + '}') -f $item.($key)
            }
            $values -join ' '
        }
    }
}
```

# HTML

## ConvertFrom-Html

```
function ConvertFrom-Html {
    param (
        [Parameter(ValueFromPipeline)]$url
    )
    $irm = Invoke-RestMethod $url
    $HTMLFile = New-Object -ComObject HTMLFile
    $Bytes = [System.Text.Encoding]::Unicode.GetBytes($irm)
    $HTMLFile.write($Bytes)
    ($HTMLFile.all | where {$_TagName -eq "body"}).innerText
}

$apache_status = "http://192.168.3.102/server-status"
$apache_status | ConvertFrom-Html
```

## ConvertTo-Html

Get-Process | select Name, CPU | ConvertTo-Html -As Table > "\$home\desktop\proc-table.html" вывод в формате List (Format-List) или Table (Format-Table)

```

$servers = "ya.ru","ya.com","google.com"
$path = "$home\Desktop\Ping.html"
$header = @"
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
<head>
<title>Отчет о статусе серверов</title>
<style type="text/css">
<!--
body {
background-color: #E0E0E0;
font-family: sans-serif
}
table, th, td {
background-color: white;
border-collapse: collapse;
border: 1px solid black;
padding: 5px
}
-->
</style>
"@
$body = @"
<h1>Ping status</h1>
<p>$($get-date -Format "dd.MM.yyyy hh:mm").</p>
"@
$results = foreach ($server in $servers) {
    if (Test-Connection $server -Count 1 -ea 0 -Quiet) {
        $status = "Up"
    }
    else {
        $status = "Down"
    }
    [PSCustomObject]@{
        Name = $server
        Status = $status
    }
}
$results | ConvertTo-Html -head $header -body $body | foreach {
    $_ -replace "<td>Down</td>","<td style='background-color:#FF8080'>Down</td>" -replace "<td>Up</td>","<td style='background-color:#5BCCF3'>Up<
} | Out-File $path
Invoke-Item $path

```

## PSWriteHTML

```

Import-Module PSWriteHTML
(Get-Module PSWriteHTML).ExportedCommands
Get-Service | Out-GridHtml -FilePath ~\Desktop\Get-Service-GridHtml.html

```

## HtmlReport

```

Import-Module HtmlReport
$topVM = ps | Sort PrivateMemorySize -Descending | Select -First 10 | %{@(( $_.ProcessName + " " + $_.Id), $_.PrivateMemorySize)}
$topCPU = ps | Sort CPU -Descending | Select -First 10 | %{@(( $_.ProcessName + " " + $_.Id), $_.CPU)}
New-Report -Title "Piggy Processes" -Input {
    New-Chart Bar "Top VM Users" -input $topVm
    New-Chart Column "Top CPU Overall" -input $topCPU
    ps | Select ProcessName, Id, CPU, WorkingSet, *MemorySize | New-Table "All Processes"
} > ~\Desktop\Get-Process-HTMLReport.html

```

# HtmlAgilityPack

Source

```
# Загрузка библиотеки C#, которая позволяет парсить HTML-документы, выбирать элементы DOM и извлекать из них данные
Add-Type -Path "C:\Users\Lifailon\Downloads\HtmlAgilityPack\Net40\HtmlAgilityPack.dll"
$title = "новобранец"
$url = "http://fasts-torrent.net"
$ep = "engine/ajax/search_torrent.php?title=$title"
$html = Invoke-RestMethod "$url/$ep"
# Создание нового объекта HtmlDocument из HtmlAgilityPack, который будет использоваться для загрузки и обработки HTML-кода
$htmlDocument = New-Object HtmlAgilityPack.HtmlDocument
# Загрузка HTML в созданный объект HtmlDocument
$htmlDocument.LoadHtml($html)
$torrents = @()
# Использование XPath для выбора всех элементов <tr> (строк таблицы) в документе
$htmlDocument.DocumentNode.SelectNodes("//tr") | ForEach-Object {
    # Для каждой строки таблицы выбираем классы, соответствующие названию раздачи, размеру и ссылке для скачивания.
    $titleNode = $_.SelectSingleNode("./td[@class='torrent-title']")
    $sizeNode = $_.SelectSingleNode("./td[@class='torrent-sp']")
    $downloadLinkNode = $_.SelectSingleNode("./td[@class='torrent-d-btn']/a")
    # Проверяем, что все классы найдены
    if ($titleNode -ne $null -and $sizeNode -ne $null -and $downloadLinkNode -ne $null) {
        # Извлечение текста из классов
        $title = $titleNode.InnerText.Trim()
        $size = $sizeNode.InnerText.Trim()
        $downloadLink = $downloadLinkNode.Attributes["href"].Value
        $torrent = New-Object PSObject -Property @{
            Title = $title
            Size = $size
            DownloadLink = " $($url)$($downloadLink)"
        }
        $torrents += $torrent
    }
}
}

$torrents
```

Title	Size	DownloadLink
Новобранец (6 сезон: 1-3 серии из 13) (2024) WEBRip   RuDub	1,55 ГБ	http://fasts-torrent.net/download/449613/torrent/-6-1-3-13-2024
Новобранец (5 сезон: 1-22 серии из 22) (2023) WEBRip 1080p   RuDub	54,15 ГБ	http://fasts-torrent.net/download/433749/torrent/-5-1-22-22-202
Новобранец (5 сезон: 1-22 серии из 22) (2023) WEBRip 720p   RuDub	30,14 ГБ	http://fasts-torrent.net/download/433750/torrent/-5-1-22-22-202
Новобранец (5 сезон: 1-22 серии из 22) (2023) WEBRip   RuDub	11,55 ГБ	http://fasts-torrent.net/download/433751/torrent/-5-1-22-22-202
Новобранец (4 сезон: 1-22 серии из 22) (2021) WEB-DL 720p   RG.Paravozik	21.33 Gb	http://fasts-torrent.net/download/418618/torrent/-4-1-22-22-202
Полицейский с половиной: Новобранец (2017) WEB-DLRip 720p   Чистый звук	3.41 Gb	http://fasts-torrent.net/download/254846/torrent/-2017-web-dlri
Полицейский с половиной: Новобранец (2017) WEB-DLRip   Чистый звук	1.37 Gb	http://fasts-torrent.net/download/254845/torrent/-2017-web-dlri
Новобранец (2 сезон: 1-20 серии из 20) (2019) WEBRip   BaibaKo	11.28 Gb	http://fasts-torrent.net/download/364669/torrent/-2-1-20-20-201
Новобранец (2 сезон: 1-20 серии из 20) (2019) WEBRip 1080p   Octopus	45.97 Gb	http://fasts-torrent.net/download/364161/torrent/-2-1-20-20-201
Новобранец (2 сезон: 1-20 серии из 20) (2019) WEB-DLRip   LostFilm	11.95 Gb	http://fasts-torrent.net/download/364668/torrent/-2-1-20-20-201

# KeePass

```
# Определяем переменные до исполняемого файла, базы и пароль
$KeePassExecPath = "C:\Program Files\KeePass Password Safe 2\KeePass.exe"
$basePath = "$home\Documents\KeePass\base.kdbx"
$basePass = "12345"

# Загружаем сборку
[System.Reflection.Assembly]::LoadFrom($KeePassExecPath) | Out-Null
[KeePass.Program]::CommonInitialize()

# Открываем базу
$IOConnectionInfo = [KeePassLib.Serialization.IOConnectionInfo]::FromPath($basePath)
$CompositeKey = New-Object KeePassLib.Keys.CompositeKey
$KcpPassword = New-Object KeePassLib.Keys.KcpPassword @($basePass)
$CompositeKey.AddUserKey($KcpPassword)
$PwDatabase = New-Object KeePassLib.PwDatabase
$PwDatabase.Open($IOConnectionInfo, $CompositeKey, $null)

# Поиск в базе по частичному совпадению (фильтрация)
$SearchParameters = New-Object KeePassLib.SearchParameters
$SearchParameters.SearchString = 'test'
$PwObjectList = New-Object KeePassLib.Collections.PwObjectList[KeePassLib.PwEntry]
$PwDatabase.RootGroup.SearchEntries($SearchParameters, $PwObjectList)
$ucount = $PwObjectList.UCount - 1
if ($ucount -ne 0) {
    foreach ($index in $(0..$ucount)) {
        $PwObjectList.GetAt($index).Strings.ReadSafe([KeePassLib.PwDefs]::UserNameField)
    }
}

# Получаем список групп
$Groups = $PwDatabase.RootGroup.Groups
$groupCollections = New-Object System.Collections.Generic.List[System.Object]
# Проходим по группам
foreach ($Group in $Groups) {
    $entryCollections = New-Object System.Collections.Generic.List[System.Object]
    # Проходим по записям группы и добавляем во временную коллекцию
    foreach ($entry in $Group.Entries) {
        $entryCollections.Add([PSCustomObject]@{
            Title      = $entry.Strings.Get("Title").ReadString()
            Login      = $entry.Strings.Get("UserName").ReadString()
            Password   = $entry.Strings.Get("Password").ReadString()
            Url        = $entry.Strings.Get("URL").ReadString()
            Description = $entry.Strings.Get("Notes").ReadString()
        })
    }
    # Добавляем запись в основную коллекцию групп (название группы используется как ключ)
    $groupCollections += [PSCustomObject]@{
        # ($Group.Name) = $entryCollections
        Group      = $Group.Name
        Modification = $Group.LastAccessTime.ToString()
        Entries    = $entryCollections
    }
}

# Преобразуем структуру в JSON
$jsonData = $groupCollections | ConvertTo-Json -Depth 3
$jsonData

# Закрываем базу
$PwDatabase.Close()
```

## SQLite

```
$path = "$home\Documents\Get-Service.db"
$Module = Get-Module MySQLite
if ($Module -eq $null) {
    Install-Module MySQLite -Repository PSGallery -Scope CurrentUser
}
Import-Module MySQLite
New-MySQLiteDB -Path $path # создать БД
Invoke-MySQLiteQuery -Path $path -Query "CREATE TABLE Service (Name TEXT NOT NULL, DisplayName TEXT NOT NULL, Status TEXT NOT NULL);#" # создать т

$Service = Get-Service | select Name,DisplayName,Status
foreach ($S in $Service) {
    $Name = $S.Name
    $DName = $S.DisplayName
    $Status = $S.Status
    Invoke-MySQLiteQuery -Path $path -Query "INSERT INTO Service (Name, DisplayName, Status) VALUES ('$Name', '$DName', '$Status');"
}

(Get-MySQLiteDB $path).Tables список таблиц в базе
Invoke-MySQLiteQuery -Path $path -Query "SELECT name FROM sqlite_master WHERE type='table';" список таблиц в базе
Invoke-MySQLiteQuery -Path $path -Query "DROP TABLE Service;" удалить таблицу

$TableName = "Service"
Invoke-MySQLiteQuery -Path $path -Query "SELECT * FROM $TableName" # прочитать содержимое таблицы (в формате объекта)

Get-Service | select Name,DisplayName,Status | ConvertTo-MySQLiteDB -Path $path -TableName Service -force конвертировать объект в таблицу
```

## Database password

```
$Connection = New-SQLiteConnection -DataSource $path
$Connection.ChangePassword("password")
$Connection.Close()
Invoke-SqliteQuery -Query "SELECT * FROM Service" -DataSource "$path;Password=password"
```

## MySQL

```
apt -y install mysql-server mysql-client
mysql -V
systemctl status mysql
mysqladmin -u root password задать пароль root

nano /etc/mysql/mysql.conf.d/mysqld.cnf

[mysqld]
user          = mysql
# pid-file     = /var/run/mysqld/mysqld.pid
# socket       = /var/run/mysqld/mysqld.sock
# port         = 3306
# datadir      = /var/lib/mysql
# tmpdir        = /tmp
bind-address   = 0.0.0.0
mysqlx-bind-address = 0.0.0.0
log_error     = /var/log/mysql/error.log

systemctl restart mysql
ss -tulnp | grep 3306
ufw allow 3306/tcp
```

```
nc -zv 192.168.1.253 3306
tnc 192.168.1.253 -p 3306

mysql -u root -p
SELECT user(), now(), version();
quit;

mysql -u root -p -e 'SHOW TABLES FROM db_aduser;' отобразить список таблиц без подключения к консоли MySQL

CREATE создать БД, пользователя, таблицу
ALTER управление столбцами таблице
DROP удалить БД, пользователя, таблицу
USE выбрать БД
SHOW вывести список БД, прав доступа пользователя (GRANTS), названия столбцов и их свойства
GRANT дать доступ пользователю к БД
REVOKE удалить доступ пользователя к БД
UPDATE изменить права доступа, значения с таблице
FLUSH обновить права доступа
SELECT отобразить выбранную БД, вывести список пользователей, выборка данных в таблице
INSERT внести данные
DELETE удалить данные в (FROM) таблице
```

## DATA TYPE

VARCHAR(N) строка переменной длины, в формате ASCII, где один символ занимает 1 байт, числом N указывается максимальная возможная длина строки

NVARCHAR(N) строка переменной длины, в формате Unicode, где один символ занимает 2 байта

CHAR(N)/nchar(N) строка фиксированной длины, которая всегда дополняется справа пробелами до длины N и в базе данных она занимает ровно N символов

INT целое число, от -2147483648 до 2147483647, занимает 4 байта

FLOAT число, в котором может присутствовать десятичная точка (запятая)

BIT флаг, Да - 1 или Нет - 0

DATE формат даты, например 25.05.2023

TIME 23:30:55.1234567

DATETIME 25.05.2023 23:30:55.1234567

## DATABASE

```
SHOW databases; # вывести список БД
CREATE DATABASE db_aduser; # создать БД
CREATE DATABASE db_rep DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci; # создать БД с кодировкой UTF-8
DROP DATABASE db_rep; # удалить БД
USE db_aduser; # выбрать/переключиться на выбранную БД
SELECT database(); # отобразить выбранную БД
```

## USER

```
SELECT USER,HOST FROM mysql.user;          # вывести список УЗ
CREATE USER posh@localhost IDENTIFIED BY '1qaz!QAZ';    # создать УЗ, которая будет подключаться с локального сервера
CREATE USER posh@localhost IDENTIFIED BY '1qaz!QAZ';    # создать УЗ, которая будет подключаться с указанного сервера
CREATE USER posh@'192.168.3.99' IDENTIFIED BY '1qaz!QAZ'; # УЗ для доступа с конкретного сервера
CREATE USER 'admin'@'%' IDENTIFIED BY 'Admin12#';      # УЗ для доступа с любого сервера (% - wildcard)
DROP USER posh@localhost;                  # удалить пользователя
SHOW GRANTS FOR posh@'%';                # отобразить права доступа пользователя
GRANT ALL PRIVILEGES ON db_aduser.* TO posh@'192.168.3.99'; # полный доступ для posh к БД db_aduser
GRANT ALL PRIVILEGES ON *.* TO posh@'%';        # доступ к всем БД с любого клиентского хоста
GRANT SELECT,DELETE ON mysql.* TO posh@'%';     # права SELECT и DELETE на встроенную БД mysql
REVOKE DELETE ON mysql.* FROM posh@'%';         # удалить доступ DELETE
UPDATE mysql.user SET super_priv='Y' WHERE USER='posh' AND host='%;' # изменить привелегии для пользователя
SELECT USER,HOST,super_priv FROM mysql.user;      # список УЗ и таблица с правами SUPER privilege
FLUSH PRIVILEGES;                            # обновить права доступа
```

## TABLE

```
SHOW TABLES;          # отобразить список всех таблиц
SHOW TABLES LIKE '%user'; # поиск таблицы по wildcard-имени
CREATE TABLE table_aduser (id INT NOT NULL AUTO_INCREMENT, Name VARCHAR(100), email VARCHAR(100), PRIMARY KEY (ID)); # создать таблицу
DROP TABLE table_aduser; # удалить таблицу
```

## COLUMN

```
SHOW COLUMNS FROM table_aduser;          # отобразить название столбцов и их свойства
ALTER TABLE table_aduser DROP COLUMN id; # удалить столбец id
ALTER TABLE table_aduser ADD COLUMN info VARCHAR(10); # добавить столбец info
ALTER TABLE table_aduser CHANGE info new_info VARCHAR(100); # изменить имя столбца info на new_info и его тип данных
ALTER TABLE table_aduser ADD COLUMN (id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (ID)); # добавить столбец id
```

## INSERT

```
INSERT table_aduser (Name,email) VALUES ('Alex','no-email');
INSERT table_aduser (Name,email) VALUES ('Alex','no-email');
INSERT table_aduser (Name) VALUES ('Support');
INSERT table_aduser (Name) VALUES ('Jack');
```

## SELECT

```
SELECT * FROM table_aduser;          # содержимое всех столбцов в выбранной (FROM) таблице
SELECT Name,email FROM table_aduser; # содержимое указанных столбцов
SELECT DISTINCT Name,Email FROM table_aduser; # отобразить уникальные записи (без повторений)
SELECT * FROM table_aduser ORDER BY Name; # отсортировать по Name
SELECT * FROM table_aduser ORDER BY Name DESC; # обратная сортировка
SELECT COUNT(*) FROM table_aduser; # количество строк в таблице
SELECT COUNT(new_info) FROM table_aduser; # количество строк в столбце
```

## WHERE

```
NOT; AND; OR          # по приоритетам условий
SELECT * FROM table_aduser WHERE Name = 'Alex'; # поиск по содержимому
SELECT * FROM table_aduser WHERE NOT Name != 'Alex'; # условие NOT где Name не равен значению
SELECT * FROM table_aduser WHERE email != ''; # вывести строки, где содержимое email не рано null
SELECT * FROM table_aduser WHERE email != '' OR id > 1000; # или id выше 1000
SELECT * FROM table_aduser WHERE Name RLIKE "support"; # регистронезависимый (RLIKE) поиск
SELECT * FROM table_aduser WHERE Name RLIKE "^support"; # начинаются только с этого словосочетания
```

## **DELETE**

```
SELECT * FROM table_aduser WHERE Name RLIKE "alex";      # найти и проверить значения перед удалением
DELETE FROM table_aduser WHERE Name RLIKE "alex";        # Query OK, 2 rows affected # удалено две строки
DELETE FROM table_aduser;                                # удалить все значения
```

## **UPDATE**

```
SELECT * FROM table_aduser WHERE Name = 'Jack';          # найти и проверить значение перед изменением
UPDATE table_aduser SET Name = 'Alex' WHERE Name = 'Jack'; # изменить значение 'Jack' на 'Alex'
UPDATE db_aduser.table_aduser SET Name='BCA' WHERE id=1;   # изменить значение в строке с ID 1
```

## **CHECK**

```
CHECK TABLE db_aduser.table_aduser;      # проверить
ANALYZE TABLE db_aduser.table_aduser;    # анализировать
OPTIMIZE TABLE db_aduser.table_aduser;   # оптимизировать
REPAIR TABLE db_aduser.table_aduser;     # восстановить
TRUNCATE TABLE db_aduser.table_aduser;   # очистить
```

## **DUMP**

```
mysqldump -u root -p --databases db_aduser > /bak/db_aduser.sql
mysql -u root -p db_aduser < /bak/db_aduser.sql

crontab -e
00 22 * * * /usr/bin/mysqldump -uroot -p1qaz!QAZ db_zabbix | /bin/bzip2 > `date +/dump/zabbix/zabbix-\%d-\%m-\%Y-\%H:\%M.bz2`
00 23 * * * /usr/bin/mysqldump -uroot -p1qaz!QAZ db_zabbix > `date +/dump/smb/zabbix-\%d-\%m-\%Y-\%H:\%M.sql`
0 0 * * * find /dump/zabbix -mtime +7 -exec rm {} \;

mysqldump -u root --single-transaction db_zabbix > /dump/zabbix/db_zabbix.sql
mysql -u user_zabbix -p -e 'CREATE DATABASE db_zabbix;'
mysql -u user_zabbix -p db_zabbix < /root/db_zabbix.sql
```

## **innodb\_force\_recovery**

```
sed -i '/innodb_force_recovery/d' /etc/mysql/my.cnf # удалить
mode=6; sed -i "/^\\[mysqld\\]/{N;s/$\\innodb_force_recovery=$mode/}" /etc/mysql/my.cnf # добавить mode 6
systemctl restart mysql

[mysqld]
innodb_force_recovery=1 # сервер пытается начать работу независимо от того, есть ли поврежденные данные InnoDB или нет
innodb_force_recovery=2 # удается восстановить работу за счет остановки потока команд, которые были частично выполнены или не выполнены (не запускаются)
innodb_force_recovery=3 # отменяет откат после восстановления поврежденных файлов (не пытается откатить транзакции)
innodb_force_recovery=6 # запуск СУБД в режиме read only
```

# MySQL Connector .NET

## Add-ADUser

```
$ip = "192.168.1.253"
$user = "posh"
$pass = "1qaz!QAZ"
$db = "db_aduser"
Add-Type -Path "$home\Documents\MySQL-Connector-NET\8.0.31-4.8\MySql.Data.dll"
$Connection = [ MySql.Data.MySqlClient.MySqlConnection ]@{
    ConnectionString = "server=$ip;uid=$user;pwd=$pass;database=$db"
}
$Connection.Open()
$Command = New-Object MySql.Data.MySqlClient.MySqlCommand
$Command.Connection = $Connection
$UserList = Get-ADUser -filter * -properties name,EmailAddress
foreach ($user in $UserList) {
    $uname=$user.Name
    $uemail=$user.EmailAddress
    $Command.CommandText = "INSERT INTO table_aduser (Name,Email) VALUES ('$uname','$uemail')"
    $Command.ExecuteNonQuery()
}
$Connection.Close()
```

## Get-ADUser

```
$ip = "192.168.1.253"
$user = "posh"
$pass = "1qaz!QAZ"
$db = "db_aduser"
Add-Type -Path "$home\Documents\MySQL-Connector-NET\8.0.31-4.8\MySql.Data.dll"
$Connection = [ MySql.Data.MySqlClient.MySqlConnection ]@{
    ConnectionString = "server=$ip;uid=$user;pwd=$pass;database=$db"
}
$Connection.Open()
$Command = New-Object MySql.Data.MySqlClient.MySqlCommand
$Command.Connection = $Connection
$MySQLDataAdapter = New-Object MySql.Data.MySqlClient.MySqlDataAdapter
$MySQLDataSet = New-Object System.Data.DataSet
$Command.CommandText = "SELECT * FROM table_aduser"
$MySQLDataAdapter.SelectCommand = $Command
$NumberOfDataSets = $MySQLDataAdapter.Fill($MySQLDataSet, "data")
$Collections = New-Object System.Collections.Generic.List[ System.Object ]
foreach($DataSet in $MySQLDataSet.Tables[0]) {
    $Collections.Add([ PSCustomObject ]@{
        Name = $DataSet.name;
        Mail = $DataSet.email
    })
}
$Connection.Close()
$Collections
```

# MSSQL

```
wget -qO- https://packages.microsoft.com/keys/microsoft.asc | apt-key add - импортировать GPG-ключ для репозитория
https://packages.microsoft.com/config/ubuntu/ Выбрать репозиторий и скопировать URL
add-apt-repository "$(wget -qO- https://packages.microsoft.com/config/ubuntu/20.04/mssql-server-2019.list)"
apt-get update обновить список пакетов
apt-get install mssql-server
/opt/mssql/bin/mssql-conf setup скрипт начальной конфигурации (выбрать редакцию, 3 - express и русский язык 9 из 11)
systemctl status mssql-server
```

```

curl https://packages.microsoft.com/keys/microsoft.asc | apt-key add - установить клиент
curl https://packages.microsoft.com/config/ubuntu/20.04/prod.list | tee /etc/apt/sources.list.d/msprod.list
apt-get update
apt-get install mssql-tools
echo 'export PATH="$PATH:/opt/mssql-tools/bin"' >> ~/.bashrc добавить в домашний каталог файла bashrc, что бы не писать путь к исполняемому
файлу
export PATH="$PATH:/opt/mssql-tools/bin"
iptables -I INPUT 1 -p tcp --dport 1433 -j ACCEPT

sqlcmd -S localhost -U SA
CREATE DATABASE itinvent
go
SELECT name FROM master.dbo.sysdatabases
go

```

## System.Data.SqlClient

```

$user = "itinvent"
$pass = "itinvent"
$db   = "itinvent"
$srsv = "192.168.3.103"
$SqlConnection = New-Object System.Data.SqlClient.SqlConnection
$SqlConnection.ConnectionString = "server=$srsv;database=$db;user id=$user;password=$pass;Integrated Security=false"

$SqlCommand = New-Object System.Data.SqlClient.SqlCommand` класс формата команды
$SqlCommand.CommandText = "SELECT * FROM ITINVENT.dbo.USERS` отобразить содержимое таблицы
#$SqlCommand.CommandText = "SELECT LICENCE_DATE,DESCR,MODEL_NO,TYPE_NO FROM ITINVENT.dbo.ITEMS where LICENCE_DATE IS NOT NULL"
$SqlCommand.Connection = $SqlConnection` передать формат подключения
$SqlAdapter = New-Object System.Data.SqlClient.SqlDataAdapter` создать адаптер подключения для выполнения SELECT запросов к БД
$SqlAdapter.SelectCommand = $SqlCommand` передать команду

$DataSet = New-Object System.Data.DataSet` создать объект приема данных формата XML
$SqlAdapter.Fill($DataSet)` заполнить данными полученные от адаптера (возвращает кол-во объектов)
$SqlConnection.Close()
$data = $DataSet.Tables
$data[0] | ft

```

## SqlClient INSERT

```

$user = "itinvent"
$pass = "itinvent"
$db   = "db_test"
$srsv = "192.168.3.103"
$sql = "INSERT INTO table_test (column_user) VALUES ('lifailon')` добавить данные в таблицу table_test в колонку column_user
$SqlConnection = New-Object System.Data.SqlClient.SqlConnection
$SqlConnection.ConnectionString = "server=$srsv;database=$db;user id=$user;password=$pass;Integrated Security=false"
$SqlCommand = New-Object System.Data.SqlClient.SqlCommand
$SqlCommand.CommandText = $sql
$SqlCommand.Connection = $SqlConnection
$SqlConnection.Open()
$rowsAffected = $SqlCommand.ExecuteNonQuery();` для запросов INSERT/UPDATE/DELETE не используется SqlDataAdapter
$SqlConnection.Close()

```

## SSMS INSERT

```
USE [db_test]
GO
INSERT INTO [dbo].[table_test]
([column_user])
VALUES
('lifailon')
GO
SELECT TOP (1000) [column_user]
FROM [db_test].[dbo].[table_test]
```

## T-SQL

- DDL (Data Definition Language / Язык определения данных). К этому типу относятся команды, которые создают базу данных, таблицы, индексы, хранимые процедуры.

`CREATE` создает объекты базы данных (саму базу данных, таблицы, индексы и т.д.)

`ALTER` изменяет объекты базы данных

`DROP` удаляет объекты базы данных

`TRUNCATE` удаляет все данные из таблиц

- DML (Data Manipulation Language / Язык манипуляции данными). К этому типу относят команды по выбору, обновлению, добавлению и удалению данных.

`SELECT` извлекает данные из БД

`UPDATE` обновляет данные

`INSERT` добавляет новые данные

`DELETE` удаляет данные

- DCL (Data Control Language / Язык управления доступа к данным). К этому типу относят команды, которые управляют правами по доступу к данным.

`GRANT` предоставляет права для доступа к данным

`REVOKE` отзывает права на доступ к данным

```

-- Переменные
DECLARE @text NVARCHAR(20), @int INT;
SET @text='Test';
SET @int = 21;
select @text,@int

-- Имена сервера и экземпляра
Select @@SERVERNAME as [Server\Instance];

-- версия SQL Server
Select @@VERSION as SQLServerVersion;

-- Текущая БД (БД, в контексте которой выполняется запрос)
Select DB_NAME() AS CurrentDB_Name;

-- Время работы с момента запуска сервера
SELECT @@Servername AS ServerName ,
       create_date AS ServerStarted ,
       DATEDIFF(s, create_date, GETDATE()) / 86400.0 AS DaysRunning ,
       DATEDIFF(s, create_date, GETDATE()) AS SecondsRunning
  FROM sys.databases
 WHERE name = 'tempdb';

-- Количество активных соединений
SELECT @@Servername AS Server ,
       DB_NAME(database_id) AS DatabaseName ,
       COUNT(database_id) AS Connections ,
       Login_name AS LoginName ,
       MIN(Login_Time) AS Login_Time ,
       MIN(COALESCE(last_request_end_time, last_request_start_time))
                                         AS Last_Batch
  FROM sys.dm_exec_sessions
 WHERE database_id > 0
   AND DB_NAME(database_id) NOT IN ( 'master', 'msdb' )
 GROUP BY database_id ,
          login_name
 ORDER BY DatabaseName;

-- Статус Backup
SELECT @@Servername AS ServerName ,
       d.Name AS DBName ,
       MAX(b.backup_finish_date) AS LastBackupCompleted
  FROM sys.databases d
 LEFT OUTER JOIN msdb..backupset b
    ON b.database_name = d.name
   AND b.[type] = 'D'
 GROUP BY d.Name
 ORDER BY d.Name;

-- Путь к Backup
SELECT @@Servername AS ServerName ,
       d.Name AS DBName ,
       b.Backup_finish_date ,
       bmf.Physical_Device_name
  FROM sys.databases d
 INNER JOIN msdb..backupset b ON b.database_name = d.name
   AND b.[type] = 'D'
 INNER JOIN msdb.dbo.backupmediafamily bmf ON b.media_set_id = bmf.media_set_id
 ORDER BY d.NAME ,
        b.Backup_finish_date DESC;

-- Вывести список всех БД, модели восстановления и путь к mdf/ldf
EXEC sp_helpdb;
SELECT @@SERVERNAME AS Server ,
       d.name AS DBName ,

```

```

create_date ,
recovery_model_Desc AS RecoveryModel ,
m.physical_name AS FileName
FROM sys.databases d
JOIN sys.master_files m ON d.database_id = m.database_id
ORDER BY d.name;

-- Размер БД
with fs
as
(
    select database_id, type, size * 8.0 / 1024 size
    from sys.master_files
)
select
    name,
    (select sum(size) from fs where type = 0 and fs.database_id = db.database_id) DataFileSizeMB,
    (select sum(size) from fs where type = 1 and fs.database_id = db.database_id) LogFileSizeMB
from sys.databases

-- Поиск таблицы по маске имени (вывод: названия схемы где располагается объект, тип объекта, дата создания и последней модификации):
select [object_id], [schema_id],
    schema_name([schema_id]) as [schema_name],
    [name],
    [type],
    [type_desc],
    [create_date],
    [modify_date]
from sys.all_objects
-- where [name]='INVENT';
where [name] like '%INVENT%';

-- Кол-во строк в таблицах
SELECT @@ServerName AS Server ,
    DB_NAME() AS DBName ,
    OBJECT_SCHEMA_NAME(p.object_id) AS SchemaName ,
    OBJECT_NAME(p.object_id) AS TableName ,
    i.Type_Desc ,
    i.Name AS IndexUsedForCounts ,
    SUM(p.Rows) AS Rows
FROM sys.partitions p
JOIN sys.indexes i ON i.object_id = p.object_id
    AND i.index_id = p.index_id
WHERE i.type_desc IN ( 'CLUSTERED', 'HEAP' )
    -- This is key (1 index per table)
    AND OBJECT_SCHEMA_NAME(p.object_id) <> 'sys'
GROUP BY p.object_id ,
    i.type_desc ,
    i.Name
ORDER BY SchemaName ,
    TableName;

-- Найти строковое (nvarchar) значение 2023 по всем таблицам базы данных
-- Отображается в какой таблице и столбце хранится значение, а также количество найденных пары таблица-колонка
set nocount on
declare @name varchar(128), @substr nvarchar(4000), @column varchar(128)
set @substr = '%2023%'
declare @sql nvarchar(max);
create table`rslt
(table_name varchar(128), field_name varchar(128), [value] nvarchar(max))
declare s cursor for select table_name as table_name from information_schema.tables where table_type = 'BASE TABLE' order by table_name
open s
fetch next from s into @name
while @@fetch_status = 0
begin

```

```

declare c cursor for
select quotename(column_name) as column_name from information_schema.columns
where data_type in ('text', 'ntext', 'varchar', 'char', 'nvarchar', 'char', 'sysname', 'int', 'tinyint') and table_name = @name
set @name = quotename(@name)
open c
fetch next from c into @column
while @@fetch_status = 0
begin
--print 'Processing table - ' + @name + ', column - ' + @column
set @sql='insert into`rslt select ''' + @name + ''' as Table_name, ''' + @column + ''', cast(' + @column +
' as nvarchar(max)) from`' + @name + ' where cast(' + @column + ' as nvarchar(max)) like ''' + @substr + '''';
print @sql;
exec(@sql);
fetch next from c into @column;
end
close c
deallocate c
fetch next from s into @name
end
select table_name as [Table Name], field_name as [Field Name], count(*) as [Found Mathes] from`rslt
group by table_name, field_name
order by table_name, field_name
drop table`rslt
close s
deallocate s

-- Поиск в таблице [CI_HISTORY] и столбцу [HIST_ID]:
SELECT * FROM ITINVENT.dbo.CI_HISTORY where [HIST_ID] like '%2023%';

-- Узнать фрагментацию индексов
DECLARE @db_id SMALLINT;
SET @db_id = DB_ID('N\'itinvent');
IF @db_id IS NULL
BEGIN;
    PRINT N'Неверильное имя базы';
END;
ELSE
BEGIN;
    SELECT
        object_id AS [ID объекта],
        index_id AS [ID индекса],
        index_type_desc AS [Тип индекса],
        avg_fragmentation_in_percent AS [Фрагментация в %]

        FROM sys.dm_db_index_physical_stats(@db_id, NULL, NULL, NULL , 'LIMITED')

        ORDER BY [avg_fragmentation_in_percent] DESC;
END;
GO

-- TempDB
-- Initial size - начальный/минимальный размер БД (1024 МВ)
-- Autogrowth - прирост (512МВ)
-- По умолчанию tempdb настроена на авто-расширение (Autogrow) и при каждой перезагрузке SQL Server пересоздаёт файлы этой БД с минимальным разме
-- Увеличив размер инициализации файлов tempdb, можно свести к минимуму затраты системных ресурсов на операции авто-расширения.

-- Изменить путь к БД:
USE master;
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = tempdev, FILENAME = 'F:\tempdb.mdf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp2, FILENAME = 'F:\tempdb_mssql_2.ndf');
GO

```

```

ALTER DATABASE tempdb
MODIFY FILE (NAME = temp3, FILENAME = 'F:\tempdb_mssql_3.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp4, FILENAME = 'F:\tempdb_mssql_4.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp5, FILENAME = 'F:\tempdb_mssql_5.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp6, FILENAME = 'F:\tempdb_mssql_6.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp7, FILENAME = 'F:\tempdb_mssql_7.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = temp8, FILENAME = 'F:\tempdb_mssql_8.ndf');
GO
ALTER DATABASE tempdb
MODIFY FILE (NAME = templog, FILENAME = 'F:\templog.ldf');
GO

-- Указать размер файла:
MODIFY FILE (NAME = temp2, FILENAME = 'F:\tempdb_mssql_2.ndf' , SIZE = 1048576KB , FILEGROWTH = 524288KB);

```

## Тип резервной копии

- Full (Полная копия). Когда стартует полное резервирование, записывается Log Sequence Number (LSN - последовательный номер журнала), а так же LSN записывается и при завершении полного резервирования. Этот LSN является механизмом, используемым SQL Server, чтобы знать, в каком порядке выполнялись операторы INSERT, UPDATE или DELETE. При этом наличие записанных LSN начала и окончания, как части полного бэкапа, обеспечивает согласованное с точки зрения транзакций резервное копирование, поскольку при полном резервном копировании учитываются изменения, произошедшие во время резервного копирования. Это обеспечивает обработку таких транзакций в процессе восстановления бэкапа.
- Differential (дифференциальная/разностная копия). Хранит данных, изменившиеся с момента последней Полной резервной копии. При восстановлении нужно сначала восстановить Полную резервную копию в режиме NORECOVERY, потом можно применить любую из последующих Разностных копий, без предыдущей Полной резервной копии Разностная копия бесполезна. Каждая последующая Разностная копия будет хранить все данные, входящие в предыдущую Разностную резервную копию, сделанную после предыдущей Полной копии.
- Incremental (инкрементальная/копия журналов транзакций). Резервное копирования журнала транзакций копирует все транзакции, которые произошли с момента последнего резервного копирования, а затем урезает журнал транзакций для освобождения дискового пространства. Транзакции происходят в определенном порядке (LSN), бэкап журнала поддерживает этот порядок транзакций. Бэкапы журналов транзакций должны восстанавливаться по порядку. Для восстановления базы данных потребуется вся цепочка резервных копий: полная и все последующие инкрементальные журналы транзакций.

## Модели восстановления

- Simple (Простая). Хранится только необходимый для жизни остаток журнала транзакций. Журнал транзакций (лог) автоматически очищается. Создание резервных копий журнала транзакций невозможна, поэтому остается самое ограниченное число опций по восстановлению. Недоступен функционал: Always On, Point-In-Time восстановление, Резервные копии журнала транзакций.
- Full (Полная). Хранится журнал транзакций всех изменений в БД с момента последнего резервного копирования журнала транзакций. Журнал транзакций не будет очищаться до тех пор, пока не будет сделана резервная копия журнала транзакций.
- Bulk logged (С неполным протоколированием). Идентична Full, за исключение: SELECT INTO, BULK INSERT и BCP, INSERT INTO SELECT, операции с индексами (CREATE INDEX, ALTER INDEX REBUILD, DROP INDEX)

## Системные БД

- master. Хранятся все данные системного уровня (конфигурация системы, сведения об учетных записях входа, информация обо всех других базах данных) для экземпляра SQL Server.
- tempdb. Рабочее пространство для временных объектов, таких как глобальные или локальные временные таблицы, временные хранимые процедуры, табличные переменные и курсоры. Пересоздаётся при каждом запуске SQL Server.

- **model.** Используется в качестве шаблона для всех баз данных, создаваемых в экземпляре SQL Server, все содержимое базы данных **model**, включая параметры базы данных, копируется в создаваемую базу данных. Так как база данных **tempdb** создается каждый раз при запуске SQL Server, база данных **model** всегда должна существовать в системе SQL Server.
- **msdb.** Используется агентом SQL Server для создания расписания предупреждений (оператор) и выполнение заданий, а также другими компонентами. SQL Server хранит полный журнал резервного копирования и восстановления в базе данных **msdb**. Для отправки почты оператору используется: **USE [msdb]**.
- **resource.** Доступная только для чтения база данных, которая содержит все системные объекты, например **sys.objects**, физически хранятся в базе данных **resource**, но логически присутствуют в схеме **sys** каждой базы данных.

## Регламентные операции

- Проверка целостности базы данных

**DBCC CHECKDB**

- Индексы. Индексы используются для быстрого поиска данных без необходимости поиска/просмотра всех строк в таблице базы данных при каждом обращении к таблице базы данных. Индекс ускоряет процесс запроса, предоставляя быстрый доступ к строкам данных в таблице, аналогично тому, как указатель в книге помогает вам быстро найти необходимую информацию. Индексы предоставляют путь для быстрого поиска данных на основе значений в этих столбцах. Для каждого индекса обязательно хранится его статистика. MS SQL Server самостоятельно создает и изменяет индексы при работе с базой. С течением времени данные в индексе становятся фрагментированными, т.е. разбросанными по базе данных, что серьезно снижает производительность запросов. Если фрагментация составляет от 5 до 30% (стандартно в задании 15%), то рекомендуется ее устранить с помощью реорганизации, при фрагментации выше 30% (по умолчанию в задаче > 30% фрагментации и число страниц > 1000) необходимо полное перестроение индексов. После перестройки планово используется только реорганизация.
- Реорганизация (Reorganize) или дефрагментация индекса — это серия небольших локальных перемещений страниц так, чтобы индекс не был фрагментирован. После реорганизации статистика не обновляется. Во время выполнения почти все данные доступны, пользователи смогут работать.

**sp\_msforeachtable N'DBCC INDEXDEFRAG (<имя базы данных>, ''?''')**

- Перестройка (Rebuild) индексов (или задача в мастере планов обслуживания: Восстановить индекс) запускает процесс полного построения индексов. В версии MS SQL Server Standard происходит отключение всех клиентов от базы на время выполнения операции. После перестройки обязательно обновляется статистика.

**sp\_msforeachtable N'DBCC DBREINDEX (''?'')**

- Обновление статистики. Статистика — небольшая таблица (обычно до 200 строк), в которой хранится обобщенная информация о том, какие значения и как часто встречаются в таблице. На основании статистики сервер принимает решение, как лучше построить запрос. Когда происходят запросы к БД (например, SELECT) вы получаете данные, но не описываете то, как эти данные должны быть извлечены. В получении и обработке данных помогает статистика. Во время выполнения процедуры обновления статистики данные не блокируются.

**exec sp\_msforeachtable N'UPDATE STATISTICS ? WITH FULLSCAN'**

- Очистка процедурного кэша, выполняется после обновления статистики. Оптимизатор MS SQL Server кэширует планы запросов для их повторного выполнения. Это делается для того, чтобы экономить время, затрачиваемое на компиляцию запроса в том случае, если такой же запрос уже выполнялся и его план известен. После обновления статистики, не будет очищен процедурный кэш, то SQL Server может выбрать старый (неоптимальный) план запроса из кэша вместо того, чтобы построить новый (более оптимальный) план.

**DBCC FREEPROCCACHE**

## InfluxDB

[Download InfluxDB 1.x Open Source](#)

[InfluxDB-Studio](#)

## Install Windows

```
Invoke-RestMethod "https://dl.influxdata.com/influxdb/releases/influxdb-1.8.10_windows_amd64.zip" -OutFile "$home\Downloads\influxdb-1.8.10_windows_amd64.zip"
Expand-Archive "$home\Downloads\influxdb-1.8.10_windows_amd64.zip" -DestinationPath "$home\Documents\
Remove-Item "$home\Downloads\influxdb-1.8.10_windows_amd64.zip"
& "$home\Downloads\influxdb-1.8.10-1\influxd.exe"
```

## Install Ubuntu

```
wget https://dl.influxdata.com/influxdb/releases/influxdb_1.8.10_amd64.deb
sudo dpkg -i influxdb_1.8.10_amd64.deb
systemctl start influxdb
systemctl status influxdb

ps aux | grep influxdb | grep -Ev "grep"
netstat -natpl | grep 80[8-9][3-9]
```

## API

```
nano /etc/influxdb/influxdb.conf

[http]
enabled = true
bind-address = ":8086"
auth-enabled = false

systemctl restart influxdb
```

## Chronograf

```
wget https://dl.influxdata.com/chronograf/releases/chronograf-1.10.2_windows_amd64.zip -UseBasicParsing -OutFile chronograf-1.10.2_windows_amd64.zip
Expand-Archive .\chronograf-1.10.2_windows_amd64.zip -DestinationPath 'C:\Program Files\InfluxData\chronograf\'

wget https://dl.influxdata.com/chronograf/releases/chronograf_1.10.2_amd64.deb
sudo dpkg -i chronograf_1.10.2_amd64.deb
systemctl status influxdb
http://192.168.3.102:8888
```

## Grafana

### Download

```
invoke-RestMethod https://dl.grafana.com/enterprise/release/grafana-enterprise-10.3.1.windows-amd64.msi -OutFile "$home\Download\grafana.msi"

apt-get install -y adduser libfontconfig1 musl
wget https://dl.grafana.com/enterprise/release/grafana-enterprise_10.3.1_amd64.deb
dpkg -i grafana-enterprise_10.3.1_amd64.deb
systemctl start grafana-server
systemctl status grafana-server
```

## CLI Client

```
apt install influxdb-client
influx
influx --host 192.168.3.102 --username admin --password password
```

```
$influx_client_exec = "$home\Documents\influxdb-1.8.10-1\influx.exe"  
& $influx_client_exec -host 192.168.3.102 -port 8086  
help  
show databases  
use PowerShell  
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m
```

## USERS

```
SHOW USERS отобразить пользователей и их права доступа  
CREATE USER admin WITH PASSWORD 'password' WITH ALL PRIVILEGES создать пользователя  
GRANT ALL PRIVILEGES TO "admin" предоставить права доступа  
GRANT READ ON "database" TO "admin" доступ на чтение для БД или запись (WRITE)  
REVOKE ALL PRIVILEGES FROM "admin" отозвать права доступа  
SHOW GRANTS FOR "admin" БД и привилегии доступа для указанного пользователя  
SET PASSWORD FOR "admin" = 'new_password' изменить пароль  
DROP USER "admin" удалить пользователя
```

## DATABASE

```
CREATE DATABASE powershell создать БД  
SHOW DATABASES отобразить список БД  
DROP DATABASE powershell удалить БД  
USE powershell  
SHOW measurements отобразить все таблицы  
INSERT performance,host=console,counter=CPU value=0.88 записать данные в таблицу performance
```

## MEASUREMENT

```
SHOW TAG KEYS FROM "HardwareMonitor" отобразить все тэги в таблице  
SHOW TAG VALUES FROM "HardwareMonitor" WITH KEY = "HardwareName" отобразить все значения указанного тэга  
SHOW FIELD KEYS FROM "HardwareMonitor" отобразить все Field Tags и их тип данных  
SHOW SERIES FROM "HardwareMonitor" отобразить список всех уникальных серий в указанной таблице. Серия - это набор точек данных, которые имеют одинаковые значения для всех тегов, за исключением времени.  
DROP SERIES FROM "HardwareMonitor" очистить все данные в таблице  
DROP MEASUREMENT "HardwareMonitor" удалить таблицу
```

## SELECT/WHERE

```
SELECT * FROM performance отобразить все данные в таблице  
SELECT value FROM performance отфильтровать по столбцу value (только Field Keys)  
SELECT * FROM performance limit 10 отобразить 10 единиц данных  
SELECT * FROM performance WHERE time > now() -2d отобразить данные за последние 2 дня  
SELECT * FROM performance WHERE time > now() +3h -5m данные за последние 5 минут (+3 часа от текущего времени по UTC 0 -5 минут)  
SELECT * FROM performance WHERE counter = 'CPU' выборка по тэгу  
SELECT upload/1000 FROM speedtest WHERE upload/1000 <= 250 выборка по столбцу upload и разделить вывод на 1000, вывести upload меньше 250  
DELETE FROM performance WHERE time > now() -1h удалить данные за последние 1/4 часа  
DELETE FROM performance WHERE time < now() -24h удалить данные старше 24 часов
```

## REGEX

```
SELECT * FROM "win_pdisk" WHERE instance =~/.*C:/ and time > now() - 5m И  
SELECT * FROM "win_pdisk" WHERE instance =~/.*E:/ or instance =~ /.*F:/ ИЛИ  
SELECT * FROM "win_pdisk" WHERE instance !~ /.*Total/ Не равно (исключить)  
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and HardwareName =~ /Intel/ приблизительно равно  
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and HardwareName =~ /Intel.+i7/ эквивалент 12th_Gen_Intel_Core_i7-1260P  
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and HardwareName =~ /^Intel/ начинается на Intel  
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and HardwareName =~ /00$/ заканчивается на 00
```

## GROUP BY tag\_key

```
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and SensorName = 'Temperature' GROUP BY HardwareName создать уникальные группы по тэгу HardwareName
```

```
SELECT * FROM "HardwareMonitor" WHERE time > now() - 5m and SensorName = 'Temperature' GROUP BY Host,HardwareName больше групп по двум тэгам
```

## Functions(field\_key)

### Functions

```
SELECT instance, LAST(Avg._Disk_Read_Queue_Length) FROM "win_pdisk" GROUP BY instance отфильтровать вывод по последнему/текущему значению
```

```
SELECT instance, FIRST(Avg._Disk_Read_Queue_Length) FROM "win_pdisk" GROUP BY instance отфильтровать вывод по первому значению за весь или указанный отрезок времени
```

```
SELECT instance, MIN(Avg._Disk_Read_Queue_Length) FROM "win_pdisk" GROUP BY instance отфильтровать вывод с отображением минимального значения
```

```
SELECT instance, MAX(Avg._Disk_Read_Queue_Length) FROM "win_pdisk" GROUP BY instance отфильтровать вывод с отображением максимального значения
```

```
SELECT SUM(Bytes_Received_persec) FROM "win_net" GROUP BY instance суммакс всех значений
```

```
SELECT COUNT(Bytes_Received_persec) FROM "win_net" WHERE Bytes_Received_persec >= 0 GROUP BY instance кол-во данных, где значение выше или равно 0
```

```
SELECT MEAN(Bytes_Received_persec) FROM "win_net" WHERE Bytes_Received_persec < 1000 GROUP BY instance среднее значение данных с показателем от 0 до 1000 (509)
```

```
SELECT *, MAX(Value) FROM "HardwareMonitor" WHERE time > now() - 1h GROUP BY SensorName, Host создать группы для выявления максимального значения значения столбца Value каждого тэга SensorName и хоста за последний час
```

```
SELECT *, MAX(Value) FROM "HardwareMonitor" WHERE time > now() - 1h and SensorName = 'CPU_Package' GROUP BY Host максимальное значение CPU_Package за последний час для каждого хоста
```

```
SELECT MEAN(Value) FROM "HardwareMonitor" WHERE time > now() - 1h and SensorName = 'CPU_Package' GROUP BY Host среднее значение CPU_Package за последний час
```

## POLICY

```
CREATE DATABASE powershell WITH DURATION 48h REPLICATION 1 NAME "de12d" создать БД с политикой хранения 2 дня
```

```
CREATE RETENTION POLICY de12h ON powershell DURATION 2h REPLICATION 1 создать новую политику хранения для БД
```

```
CREATE RETENTION POLICY de16h ON powershell DURATION 6h REPLICATION 1 SHARD DURATION 2h указать период хранения 6 часов + 2 часа до очистки (по умолчанию 1ч или больше)
```

```
ALTER RETENTION POLICY de16h ON powershell DEFAULT изменить (ALTER) политику хранения для БД на de16h (DEFAULT)
```

```
DROP RETENTION POLICY de12d ON powershell удаление политики хранения приводит к безвозвратному удалению всех измерений (таблиц) и данных, хранящихся в политике хранения
```

```
SHOW RETENTION POLICIES ON PowerShell отобразить действующие политики базы данных PowerShell
```

## InfluxDB-api

```
$data = Invoke-RestMethod http://192.168.3.102:8086/query?q="SHOW RETENTION POLICIES ON PowerShell"
$data = $data.results.series.columns
$val = $data.results.series.values
$mass = @()
$mass += [string]$col
foreach ($v in $val) {
    $mass += [string]$v
}
$mass = $mass -replace '^',''
$mass = $mass -replace '$',''
$mass = $mass -replace '\s','',''
$mass | ConvertFrom-Csv
```

## API POST

Вместо таблиц в InfluxDB имеются измерения. Вместо столбцов в ней есть теги и поля.

Table	Tag (string/int)	Field (double/int)	TIMESTAMP
measurement, Tag_Keys1=Tag_Values1, Tag_Keys2=Tag_Values2	Field_Keys1="Values", Field_Keys2="Values"	00000000000000000000000000000000	
1	2	3	

```

$ip      = "192.168.3.104"
$port    = "8086"
$db      = "powershell"
$table   = "speedtest"
$ipp     = $ip+":"+$port
$url    = "http://$ipp/write?db=$db"
$user   = "admin"
$pass   = "password" | ConvertTo-SecureString -AsPlainText -Force
$cred   = [System.Management.Automation.PSCredential]::new($user,$pass)
$unixtime = (New-TimeSpan -Start (Get-Date "01/01/1970") -End (Get-Date)).TotalSeconds
$timestamp = ([string]$unixtime -replace "\..+") + "00000000"

```

Invoke-RestMethod -Method POST -Uri \$url -Body "\$table,host=\$(hostname) download=200000,upload=300000,ping=3 \$timestamp"

## API GET

curl http://192.168.3.104:8086/query --data-urlencode "q=SHOW DATABASES" pwsh7 (ConvertFrom-Json) and bash

```

$dbs = irm "http://192.168.3.104:8086/query?q=SHOW DATABASES"
$dbs = irm "http://192.168.3.104:8086/query?epoch=ms&u=admin&p=password&q=SHOW DATABASES"
$dbs.results.series.values

$ip      = "192.168.3.104"
$port    = "8086"
$db      = "powershell"
$table   = "speedtest"
$query  = "SELECT * FROM $table"
$ipp     = $ip+":"+$port
$url    = "http://$ipp/query?db=$db&q=$query"
$data   = Invoke-RestMethod -Method GET -Uri $url` -Credential $cred
$data.results.series.name ` имя таблицы
$data.results.series.columns` столбцы/ключи
$data.results.series.values ` данные построчно

```

## Endpoints

[API doc](#)

```

$stats = irm http://192.168.3.104:8086/debug/vars` статистика сервера
$stats."database:powershell".values` кол-во таблиц к БД
$stats.queryExecutor.values` количество query-запросов (обращений к endpoint /query)
$stats.write.values` количество write-запросов
$stats.system.uptime

```

<http://192.168.3.104:8086/debug/requests> КОЛ-ВО КЛИЕНТСКИХ HTTP-зАПРОСОВ К КОНЧЕЧНЫМ ТОЧКАМ /writeи /query

<http://192.168.3.104:8086/debug/pprof>

<http://192.168.3.104:8086/ping>

<http://192.168.3.104:8086/query>

<http://192.168.3.104:8086/write>

<http://192.168.3.99:8086/api/v2/setup>

<http://192.168.3.99:8086/api/v2/config>

<http://192.168.3.99:8086/api/v2/write>

## PingTo-InfluxDB

```
while ($true) {
    $tz = (Get-TimeZone).BaseUtcOffset.TotalMinutes
    $unixtime = (New-TimeSpan -Start (Get-Date "01/01/1970") -End ((Get-Date).AddMinutes(-$tz))).TotalSeconds` -3h UTC
    $timestamp = ([string]$unixtime -replace "\..+") + "00000000"
    $tnc = tnc 8.8.8.8
    $Status = $tnc.PingSucceeded
    $RTTime = $tnc.PingReplyDetails.RoundtripTime
    Invoke-RestMethod -Method POST -Uri "http://192.168.3.104:8086/write?db=powershell" -Body "ping,host=$(hostname) status=$status,rtime=$RT
    sleep 1
}

SELECT * FROM ping WHERE status = false
```

## PerformanceTo-InfluxDB

```
function ConvertTo-Encoding ([string]$From, [string]$To) {
    Begin {
        $encFrom = [System.Text.Encoding]::GetEncoding($from)
        $encTo = [System.Text.Encoding]::GetEncoding($to)
    }
    Process {
        $bytes = $encTo.GetBytes($_)
        $bytes = [System.Text.Encoding]::Convert($encFrom, $encTo, $bytes)
        $encTo.GetString($bytes)
    }
}

$localization = (Get-Culture).LCID` текущая локализация
if ($localization -eq 1049) {
    $performance = "\$(hostname)\Процессор(_Total)\% загруженности процессора" | ConvertTo-Encoding UTF-8 windows-1251` декодировать кирилли
} else {
    $performance = "\Processor(_Total)\% Processor Time"
}

$tz = (Get-TimeZone).BaseUtcOffset.TotalMinutes
while ($true) {
    $unixtime = (New-TimeSpan -Start (Get-Date "01/01/1970") -End ((Get-Date).AddMinutes(-$tz))).TotalSeconds` -3h UTC
    $timestamp = ([string]$unixtime -replace "\..+") + "00000000"
    [double]$value = (Get-Counter $performance).CounterSamples.CookedValue.ToString("0.00").replace(",",".")` округлить в тип данных Double
    Invoke-RestMethod -Method POST -Uri "http://192.168.3.104:8086/write?db=powershell" -Body "performance,host=$(hostname),counter=CPU value
    sleep 5
}
```

## Service

```
$powershell_Path = (Get-Command powershell).Source
$NSSM_Path = "C:\NSSM\NSSM-2.24.exe"
$Script_Path = "C:\NSSM\PerformanceTo-InfluxDB.ps1"
$Service_Name = "PerformanceTo-InfluxDB"
& $NSSM_Path install $Service_Name $powershell_Path -ExecutionPolicy Bypass -NoProfile -f $Script_Path
Get-Service $Service_Name | Start-Service
Get-Service $Service_Name | Set-Service -StartupType Automatic
```

## PSInfluxDB

```
Install-Module psinfluxdb -Repository NuGet
Get-InfluxUsers -server 192.168.3.102 список пользователей на сенвере InfluxDB и права доступа
Get-InfluxDatabases -server 192.168.3.102 список баз данных
Get-InfluxDatabases -server 192.168.3.102 -creat -database test создать базу данных
```

```

Get-InfluxDatabases -server 192.168.3.102 -delete -database test удалить базу данных
Get-InfluxPolicy -server 192.168.3.102 -database PowerShell список политик хранения указанной базы данных
Get-InfluxPolicy -server 192.168.3.102 -database PowerShell -create -policyName del2d -hours 48 создать политику хранения
Get-InfluxPolicy -server 192.168.3.102 -database PowerShell -policyName del2d -default применить политику хранения к базе данных по умолчанию
Get-InfluxTables -server 192.168.3.102 -database PowerShell список таблиц/измерений в базе данных
$Data = Get-InfluxData -server 192.168.3.102 -database PowerShell -table ping | Format-Table отобразить данные в таблице
$Data | Where-Object {$_.SensorName -match "CPU_Package" -and $_.Value -gt 90} | Format-Table отфильтровать данные по двум параметрам (сенсоры процессора с показателем выше 90)
$influx = Get-InfluxData -server 192.168.3.104 -database PowerShell -table speedtest
Get-InfluxChart -time ($influx.time) -data ($influx.download) -title "SpeedTest Download" -path "C:\Users\Lifailon\Desktop" создать график (WinForms Chart) измерений и сохранить в jpeg-файл

```

## Telegraf

### Plugins

```

iwr https://dl.influxdata.com/telegraf/releases/telegraf-1.27.1_windows_amd64.zip -UseBasicParsing -OutFile telegraf-1.27.1_windows_amd64.zip
Expand-Archive .\telegraf-1.27.1_windows_amd64.zip -DestinationPath "C:\Telegraf"
rm telegraf-1.27.1_windows_amd64.zip
cd C:\Telegraf
.\telegraf.exe -sample-config --input-filter cpu:mem:dns_query --output-filter influxdb > telegraf_nt.conf создать конфигурацию с выбарными плагинами для сбора метрик
Start-Process notepad++ C:\Telegraf\telegraf_nt.conf

[[outputs.influxdb]]
urls = ["http://192.168.3.104:8086"]
database = "telegraf_nt"
username = "user"
password = "pass"

[[inputs.cpu]]
percpu = false
totalcpu = true

[[inputs.dns_query]]
servers = ["8.8.8.8"]
network = "udp"
domains = [".."]
record_type = "A"
port = 53
timeout = "2s"

.\telegraf.exe --test -config C:\Telegraf\telegraf_nt.conf тест конфигурации (получения метрик с выводом в консоль)
C:\Telegraf\telegraf.exe -config C:\Telegraf\telegraf_nt.conf запустить telegraf (тест отправки данных)
.\telegraf.exe --config "C:\Telegraf\telegraf_nt.conf" --service install создать службу
Get-Service telegraf | Start-Service
.\telegraf.exe --service uninstall

USE telegraf
SELECT usage_idle,usage_system,usage_user FROM cpu

```

## Elasticsearch

```

Install-Module -Name Elastic.Console -AllowPrerelease github source
Get-Command -Module Elastic.Console
Get-ElasticsearchVersion
Set-ElasticsearchVersion 7.3.0
Invoke-Elasticsearch REST API запросы

```

# CData

PowerShell Gallery CData

Automate Elasticsearch Integration Tasks from PowerShell

```
Install-Module ElasticsearchCmdlets пакет драйвера в psgallery
Import-Module ElasticsearchCmdlets
Get-Command -Module ElasticsearchCmdlets

$elasticsearch = Connect-Elasticsearch -Server "$Server" -Port "$Port" -User "$User" -Password "$Password"
$shipcity = "New York"
$orders = Select-Elasticsearch -Connection $elasticsearch -Table "Orders" -Where "ShipCity = `'$ShipCity`'"` поиск и получение данных
$orders = Invoke-Elasticsearch -Connection $elasticsearch -Query 'SELECT * FROM Orders WHERE ShipCity = @ShipCity' -Params @{@"@ShipCity"='New Yor
```

## ADO.NET Assembly

```
Install-Package CData.Elasticsearch пакет драйвера в nuget
[Reflection.Assembly]::LoadFile("C:\Program Files\PackageManagement\NuGet\Packages\CData.Elasticsearch.23.0.8565\lib\net40\System.Data.CData.Elasti
$connect = New-Object System.Data.CData.Elasticsearch.ElasticsearchConnection("Server=127.0.0.1;Port=9200;User=admin;Password=123456;")
$connet.Open()
$sql = "SELECT OrderName, Freight from Orders"
$da = New-Object System.Data.CData.Elasticsearch.ElasticsearchDataAdapter($sql, $conn)
$dtt = New-Object System.Data.DataTable
$da.Fill($dt)
$dtt.Rows | foreach {
Write-Host $_.ordername $_.freight
}
```

## UPDATE

```
Update-Elasticsearch -Connection $Elasticsearch -Columns @('OrderName', 'Freight') -Values @('MyOrderName', 'MyFreight') -Table Orders -Id "MyId"
$cmd = New-Object System.Data.CData.Elasticsearch.ElasticsearchCommand("UPDATE Orders SET ShipCity='New York' WHERE Id = @myId", $conn)
$cmd.Parameters.Add(new System.Data.CData.Elasticsearch.ElasticsearchParameter("@myId","10456255-0015501366"))
$cmd.ExecuteNonQuery()
```

## INSERT

```
Add-Elasticsearch -Connection $Elasticsearch -Table Orders -Columns @("OrderName", "Freight") -Values @("MyOrderName", "MyFreight")
$cmd = New-Object System.Data.CData.Elasticsearch.ElasticsearchCommand("INSERT INTO Orders (ShipCity) VALUES (@myShipCity)", $conn)
$cmd.Parameters.Add(new System.Data.CData.Elasticsearch.ElasticsearchParameter("@myShipCity","New York"))
$cmd.ExecuteNonQuery()
```

## DELETE

```
Remove-Elasticsearch -Connection $Elasticsearch -Table "Orders" -Id "MyId"
$cmd = New-Object System.Data.CData.Elasticsearch.ElasticsearchCommand("DELETE FROM Orders WHERE Id=@myId", $conn)
$cmd.Parameters.Add(new System.Data.CData.Elasticsearch.ElasticsearchParameter("@myId","001d000000YBRseAAH"))
$cmd.ExecuteNonQuery()
```

# ODBC

```
Get-Command -Module Wdac
Get-OdbcDriver | ft список установленных драйверов
```

## Elasticsearch ODBC драйвер для доступа к данным Elasticsearch из Microsoft PowerShell

```
$connectstring = "DSN=Local Elasticsearch;"  
$sql = "SELECT * FROM library"  
$conn = New-Object System.Data.Odbc.OdbcConnection($connectstring)  
$conn.open()  
$cmd = New-Object system.Data.Odbc.OdbcCommand($sql,$conn)  
$da = New-Object system.Data.Odbc.OdbcDataAdapter($cmd)  
$dt = New-Object system.Data.datatable  
$null = $da.fill($dt)  
$conn.close()  
$dt
```

## PostgreSQL

### ODBC Driver

[Скачать и установить драйвер](#)

```
$dbServer = "192.168.3.101"  
$port = "5432"  
$dbName = "test"  
$dbUser = "admin"  
$dbPass = "admin"  
$szConnect = "Driver={PostgreSQL Unicode(x64)};Server=$dbServer;Port=$port;Database=$dbName;Uid=$dbUser;Pwd=$dbPass;"  
  
$cnDB = New-Object System.Data.Odbc.OdbcConnection($szConnect)  
$dsDB = New-Object System.Data.DataSet  
try {  
    $cnDB.Open()  
    $adDB = New-Object System.Data.Odbc.OdbcDataAdapter  
    $adDB.SelectCommand = New-Object System.Data.Odbc.OdbcCommand("SELECT id, name, age, login FROM public.users" , $cnDB)  
    $adDB.Fill($dsDB)  
    $cnDB.Close()  
}  
catch [System.Data.Odbc.OdbcException] {  
    $_.Exception  
    $_.Exception.Message  
    $_.Exception.ItemName  
}  
foreach ($row in $dsDB[0].Tables[0].Rows) {  
    $row.login  
    $row.age  
}
```

## npgsql

[Source](#)

[Package](#)

```

# Подключаем сборку Npgsql
Add-Type -Path "$home\Documents\Npgsql.PostgreSQL\8.0.4\net8.0\Npgsql.EntityFrameworkCore.PostgreSQL.dll"
# Определяем строку подключения к базе данных PostgreSQL
$connString = "Host=myserver;Username=mylogin;Password=mypass;Database=mydatabase"
# Создаем объект подключения
$conn = New-Object Npgsql.NpgsqlConnection($connString)
# Открываем соединение с базой данных
$conn.Open()
try {
    # Создаем SQL команду
    $cmd = $conn.CreateCommand()
    $cmd.CommandText = "INSERT INTO data (some_field) VALUES (@p)"
    # Добавляем параметр и его значение
    $param = $cmd.Parameters.Add("p", [NpgsqlTypes.NpgsqlDbType]::Text)
    $param.Value = "Hello world"
    # Выполняем команду
    $cmd.ExecuteNonQuery() | Out-Null
} finally {
    $cmd.Dispose()
}
# Извлекаем все строки из таблицы
try {
    # Создаем команду SQL для извлечения данных
    $cmd = $conn.CreateCommand()
    $cmd.CommandText = "SELECT some_field FROM data"
    # Выполняем команду и получаем объект чтения данных
    $reader = $cmd.ExecuteReader()
    # Читаем и выводим данные построчно
    while ($reader.Read()) {
        Write-Host $reader.GetString(0)
    }
} finally {
    # Освобождаем ресурсы чтения данных и команды
    $reader.Dispose()
    $cmd.Dispose()
}
# Закрываем соединение с базой данных
$conn.Close()
$conn.Dispose()

```

## WMI

### WMI/CIM

Windows Management Instrumentation/Common Information Model

Get-WmiObject -ComputerName localhost -Namespace root -class "\_\_NAMESPACE" | select name,\_\_namespace отобразить дочерние Namespace (логические иерархические группы)

Get-WmiObject -List отобразить все классы пространства имен "root\cimv2" (по умолчанию), свойства (описывают конфигурацию и текущее состояние управляемого ресурса) и их методы (какие действия позволяет выполнить над этим ресурсом)

Get-WmiObject -List | Where-Object {\$\_.name -match "video"} поиск класса по имени, его свойств и методов

Get-WmiObject -ComputerName localhost -Class Win32\_VideoController отобразить содержимое свойств класса

gwmi -List | where name -match "service" | ft -auto если в таблице присутствуют Methods, то можно взаимодействовать {StartService, StopService}

gwmi -Class win32\_service | select \* отобразить список всех служб и всех их свойств

Get-CimInstance Win32\_service обращается на прямую к "root\cimv2"

gwmi win32\_service -Filter "name='Zabbix Agent'" отфильтровать вывод по имени

(gwmi win32\_service -Filter "name='Zabbix Agent'").State отобразить конкретное свойство

gwmi win32\_service -Filter "State = 'Running'" отфильтровать запущенные службы

gwmi win32\_service -Filter "StartMode = 'Auto'" отфильтровать службы по методу запуска

```

gwmi -Query 'select * from win32_service where startmode="Auto"' WQL-запрос (WMI Query Language)
gwmi win32_service | Get-Member -MemberType Method отобразить все методы взаимодействия с описание применения (Delete, StartService)
(gwmi win32_service -Filter 'name="Zabbix Agent").Delete() удалить службу
(gwmi win32_service -Filter 'name="MSSQL$MSSQUEL").StartService() запустить службу

Get-CimInstance -ComputerName $srv Win32_OperatingSystem | select LastBootUpTime время последнего включения
gwmi -ComputerName $srv -Class Win32_OperatingSystem | select LocalDateTime,LastBootUpTime текущее время и время последнего включения
gwmi Win32_OperatingSystem | Get-Member -MemberType Method методы reboot и shutdown
(gwmi Win32_OperatingSystem -EnableAllPrivileges).Reboot() используется с ключом повышения привилегий
(gwmi Win32_OperatingSystem -EnableAllPrivileges).Win32Shutdown(0) завершение сеанса пользователя

$system = Get-WmiObject -Class Win32_OperatingSystem
$InstallDate = [Management.ManagementDateTimeconverter]::ToDateTime($system.installdate)` Получаем дату установки ОС
$AfterInstallDays = ((Get-Date) - $InstallDate).Days` Вычисляем время, прошедшее с момента установки
$ShortInstallDate = "{0:yyyy-MM-dd HH:MM}" -f ($InstallDate)
"Система установлена: $ShortInstallDate (Прошло $AfterInstalldays дней)"

(Get-WmiObject win32_battery).estimatedChargeRemaining заряд батареи в процентах
gwmi Win32_UserAccount Доменные пользователи
(gwmi Win32_SystemUsers).PartComponent
Get-CimInstance -ClassName Win32_LogonSession
Get-CimInstance -ClassName Win32_BIOS

gwmi -list -Namespace root\cimv2\TerminalServices
(gwmi -Class Win32_TerminalServiceSetting -Namespace root\cimv2\TerminalServices).AllowTSConnections
(gwmi -Class Win32_TerminalServiceSetting -Namespace root\cimv2\TerminalServices).SetAllowTSConnections(1) включить RDP

$srvc = "localhost"
gwmi Win32_logicalDisk -ComputerName $srvc | where {$_._Size -ne $null} | select @{
    Label="Value"; Expression={$_._DeviceID}, @{{Label="AllSize"; Expression={
        [string]([int]($_._Size/1Gb))+" GB"}},@{{Label="FreeSize"; Expression={
            [string]([int]($_._FreeSpace/1Gb))+" GB"}}, @{{Label="Free%"; Expression={
                [string]([int]($_._FreeSpace/$_.Size*100))+" %}}}

```

## NLA

Network Level Authentication

```

(gwmi -class "Win32_TSGeneralSetting" -Namespace root\cimv2\TerminalServices -Filter "TerminalName='RDP-tcp'").UserAuthenticationRequired
(gwmi -class "Win32_TSGeneralSetting" -Namespace root\cimv2\TerminalServices -Filter "TerminalName='RDP-tcp'").SetUserAuthenticationRequired(1)
включить NLA

Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name SecurityLayer отобразить значение (2)
Get-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name UserAuthentication отобразить значение (1)
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name SecurityLayer -Value 0 изменить значение
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Control\Terminal Server\WinStations\RDP-Tcp" -Name UserAuthentication -Value 0
REG ADD HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System\CredSSP\Parameters /v AllowEncryptionOracle /t REG_DWORD /d 2 отключить на клиентском компьютере проверку версии CredSSP, если на целевом компьютере-сервере не установлены обновления KB4512509 от мая 2018 года

```

## Regedit

```

Get-PSDrive список всех доступных дисков/разделов, их размер и веток реестра
cd HKLM:\ HKEY_LOCAL_MACHINE
cd HKCU:\ HKEY_CURRENT_USER
Get-Item получить информацию о ветке реестра
New-Item создать новый раздел реестра
Remove-Item удалить ветку реестра

```

```

Get-ItemProperty получить значение ключей/параметров реестра (это свойства ветки реестра, аналогично свойствам файла)
Set-ItemProperty изменить название или значение параметра реестра
New-ItemProperty создать параметр реестра
Remove-ItemProperty удалить параметр

Get-ItemProperty HKLM:\Software\Microsoft\Windows\CurrentVersion\Uninstall\* | Select DisplayName СПИСОК установленных программ
Get-Item HKCU:\SOFTWARE\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002 посмотреть содержимое Items
(Get-ItemProperty HKCU:\SOFTWARE\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002). "New Signature"
отобразить значение (Value) свойства (Property) Items
$reg_path = "HKCU:\SOFTWARE\Microsoft\Office\16.0\Outlook\Profiles\Outlook\9375CFF0413111d3B88A00104B2A6676\00000002"
$sig_name = "auto"
Set-ItemProperty -Path $reg_path -Name "New Signature" -Value $sig_name изменить или добавить в корне ветки (Path) свойство (Name) со
значением (Value)
Set-ItemProperty -Path $reg_path -Name "Reply-Forward Signature" -Value $sig_name

Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\taskmgr.exe]
"Debugger"="\"C:\\Windows\\System32\\Taskmgr.exe\""

```

## Performance

```

lodctr /R пересоздать счетчиков производительности из системного хранилища архивов (так же исправляет счетчики для CIM, например, для
cpu Win32_PerfFormattedData_PerfOS_Processor и iops Win32_PerfFormattedData_PerfDisk_PhysicalDisk)
( Get-Counter -ListSet * ).CounterSetName вывести список всех доступных счетчиков производительности в системе
( Get-Counter -ListSet *memory* ).Counter ПОИСК по wildcard-имени во всех счетчиках (включая дочерний)
Get-Counter "\Memory\Available MBytes" объем свободной оперативной памяти
Get-Counter -cn $srv "\LogicalDisk(*)%\ Free Space" % свободного места на всех разделах дисков
( Get-Counter "\Process(*)\ID Process" ).CounterSamples
Get-Counter "\Processor(_Total)\% Processor Time" -ComputerName $srv -MaxSamples 5 -SampleInterval 2 5 проверок каждые 2 секунды
Get-Counter "\Процессор(_Total)\% загруженности процессора" -Continuous непрерывно
( Get-Counter "\Процессор(*)\% загруженности процессора" ).CounterSamples

( Get-Counter -ListSet *интерфейс* ).Counter Найти все счетчики
Get-Counter "\Сетевой интерфейс(*)\Всего байт/с" отобразить все адаптеры (выбрать действующий по трафику)

$WARNING = 25
$CRITICAL = 50
$TransferRate = ((Get-Counter "\\huawei-mb-x-pro\сетевой интерфейс(intel[r] wi-fi 6e ax211 160mhz)\всего байт/с"
).Countersamples | select -ExpandProperty CookedValue)*8
$NetworkUtilisation = [math]::round($TransferRate/1000000000*100,2)
if ($NetworkUtilisation -gt $CRITICAL){
Write-Output "CRITICAL: $($NetworkUtilisation) % Network utilisation, $($TransferRate.ToString('N0')) b/s"
#exit 2
}
if ($NetworkUtilisation -gt $WARNING){
Write-Output "WARNING: $($NetworkUtilisation) % Network utilisation, $($TransferRate.ToString('N0')) b/s"
#exit 1
}
Write-Output "OK: $($NetworkUtilisation) % Network utilisation, $($TransferRate.ToString('N0')) b/s"
#exit 0

```

## SNMP

### Setup SNMP Service

```

Install-WindowsFeature SNMP-Service,SNMP-WMI-Provider -IncludeManagementTools установить роль SNMP и WMI провайдер через Server Manager
Get-WindowsFeature SNMP*

```

```
Add-WindowsCapability -Online -Name SNMP.Client~~~0.0.1.0 установить компонент Feature On Demand для Windows 10/11 \ Get-Service  
SNMP* \ Get-NetFirewallrule -DisplayName snmp | ft \ Get-NetFirewallrule -DisplayName snmp | Enable-NetFirewallRule'
```

## Setting SNMP Service via Regedit

Agent:

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\services\SNMP\Parameters\RFC1156Agent" -Name "sysContact" -Value "lifailon-user" создать  
(New) или изменить (Set)  
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\services\SNMP\Parameters\RFC1156Agent" -Name "sysLocation" -Value "plex-server"
```

Security:

```
New-Item -Path "HKLM:\SYSTEM\CurrentControlSet\services\SNMP\Parameters\TrapConfiguration\public" создать новый community string  
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\ValidCommunities" -Name "public" -Value 16 назначить права на  
public
```

1 – NONE

2 – NOTIFY позволяет получать SNMP ловушки

4 – READ ONLY позволяет получать данные с устройства

8 – READ WRITE позволяет получать данные и изменять конфигурацию устройства

16 – READ CREATE позволяет читать данные, изменять и создавать объекты

```
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\SNMP\Parameters\PermittedManagers" -Name "1" -Value "192.168.3.99" от кого  
разрешено принимать запросы
```

Get-Service SNMP | Restart-Service

## snmpwalk

```
snmpwalk -v 2c -c public 192.168.3.100
```

```
snmpwalk -v 2c -c public -O e 192.168.3.100
```

## SNMP Modules

Install-Module -Name SNMP

```
Get-SnmpData -IP 192.168.3.100 -OID 1.3.6.1.2.1.1.4.0 -UDPPort 161 -Community public
```

```
(Get-SnmpData -IP 192.168.3.100 -OID 1.3.6.1.2.1.1.4.0).Data
```

Invoke-SnmpWalk -IP 192.168.3.100 -OID 1.3.6.1.2.1.1 пройтись по дереву OID

Invoke-SnmpWalk -IP 192.168.3.100 -OID 1.3.6.1.2.1.25.6.3.1.2 список установленного ПО

Invoke-SnmpWalk -IP 192.168.3.100 -OID 1.3.6.1.2.1.25.2.3.1 список разделов и памяти (C: D: Virtual Memory и Physical Memory)

Set-SnmpData изменение данных на удаленном устройстве

Install-Module -Name SNMPv3

Invoke-SNMPv3Get получение данных по одному OID

Invoke-SNMPv3Set изменение данных

Invoke-SNMPv3Walk обход по дереву OID

```
Invoke-SNMPv3Walk -UserName lifailon -Target 192.168.3.100 -AuthSecret password -PrivSecret password -OID 1.3.6.1.2.1.1 -AuthType MD5 -PrivType AES
```

## Lextm.SharpSnmpLib

Синтаксис

Download lib

```
Add-Type -LiteralPath "$home\Desktop\lextm.sharpsnmplib-12.5.2\net471\SharpSnmpLib.dll"
```

```

$port = 161
$OID = "1.3.6.1.2.1.1.4.0"
$variableList = New-Object Collections.Generic.List[Lextm.SharpSnmpLib.Variable]
$variableList.Add([Lextm.SharpSnmpLib.Variable]::new([Lextm.SharpSnmpLib.ObjectIdentifier]::new($OID)))
$timeout = 3000
[Net.IPEndPoint]$ip = "192.168.3.100"
$endpoint = New-Object Net.IpEndPoint $ip, $port
$Community = "public"
[Lextm.SharpSnmpLib.VersionCode]$Version = "V2"

$message = [Lextm.SharpSnmpLib.Messaging.Messenger]::Get(
$Version,
$endpoint,
$Community,
$variableList,
$TimeOut
)
$message.Data.ToString()

```

## Walk

```

[Lextm.SharpSnmpLib.ObjectIdentifier]$OID = "1.3.6.1.2.1.1" # дерево или конечный OID
$WalkMode = [Lextm.SharpSnmpLib.Messaging.WalkMode]::WithinSubtree # режим обхода по дереву
$results = New-Object Collections.Generic.List[Lextm.SharpSnmpLib.Variable]
$message = [Lextm.SharpSnmpLib.Messaging.Messenger]::Walk(
$Version,
$endpoint,
$Community,
$OID,
$results,
$TimeOut,
$WalkMode
)
$results

$results2 = @()
foreach ($d in $results) {
$results2 +=[PSCustomObject]@{'ID'=$d.id.ToString(); 'Data'=$d.Data.ToString()} # перекодировать вывод построчно в строку
}
$results2

```

## Zabbix

### Zabbix Agent

**Zabbix Agent Deploy:**

```

$url = "https://cdn.zabbix.com/zabbix/binaries/stable/6.4/6.4.5/zabbix_agent2-6.4.5-windows-amd64-static.zip"
$path = "$home\Downloads\zabbix-agent2-6.4.5.zip"
$WebClient = New-Object System.Net.WebClient
$WebClient.DownloadFile($url, $path) # скачать файл
Expand-Archive $path -DestinationPath "C:\zabbix-agent2-6.4.5\" # разархивировать
Remove-Item $path # удалить архив
New-NetFirewallRule -DisplayName "Zabbix-Agent" -Profile Any -Direction Inbound -Action Allow -Protocol TCP -LocalPort 10050,10051 # открыть порт

$Zabbix_Server = "192.168.3.102"
$conf = "C:\zabbix-agent2-6.4.5\conf\zabbix_agent2.conf"
$cat = cat $conf
$rep = $cat -replace "Server=.+", "Server=$Zabbix_Server"
$rep | Select-String Server=
$rep > $conf

$exe = "C:\zabbix-agent2-6.4.5\bin\zabbix_agent2.exe"
.$exe --config $conf --install # установить службу
Get-Service *Zabbix*Agent* | Start-Service # запустить службу
#.exe --config $conf --uninstall # удалить службу

```

### **zabbix\_agent2.conf**

```

# Агент может работать в пассивном (сервер забирает сам информацию) и активном режиме (агент сам отправляет):
Server=192.168.3.102
ServerActive=192.168.3.102
# Требуется указать hostname для ServerActive:
Hostname=huawei-book-01
# Если не указано, используется для генерации имени хоста (игнорируется, если имя хоста определено):
# HostnameItem=system.hostname
# Как часто обновляется список активных проверок, в секундах (Range: 60-3600):
RefreshActiveChecks=120
# IP-адрес источника для исходящих соединений:
# SourceIP=
# Агент будет слушать на этом порту соединения с сервером (Range: 1024-32767):
# ListenPort=10050
# Список IP-адресов, которые агент должен прослушивать через запятую
# ListenIP=0.0.0.0
# Агент будет прослушивать этот порт для запросов статуса HTTP (Range: 1024-32767):
# StatusPort=
ControlSocket=\\.\\pipe\\agent.sock
# Куда вести журнал (file/syslog/console):
LogType=file
LogFile=C:\zabbix-agent2-6.4.5\zabbix_agent2.log
# Размер лога от 0-1024 MB (0 - отключить автоматическую ротацию логов)
LogFileSize=100
# Уровень логирования. 4 - для отладки (выдает много информации)
DebugLevel=4

```

## **Zabbix Sender**

Используется для отправки данных на сервер

Создать host - задать произвольное имя (powershell-host) и добавить в группу на сервере

Создать Items вручную:

```

Name : Service Count
Type : Zabbix trapper
Key : service.count
Type of Information : Numeric

```

```
$path = "C:\zabbix-agent2-6.4.5\bin"
$count = (Get-Service).Count
.$path\zabbix_sender.exe -z 192.168.3.102 -s "powershell-host" -k service.count -o $count # отправить данные на сервер
```

## Zabbix Get

Используется для получения данных с агента (как их запрашивает сервер)

```
apt install zabbix-get
nano /etc/zabbix/zabbix_agentd.conf
Server=127.0.0.1,192.168.3.102,192.168.3.99 Добавить сервера для получения данных через zabbix_get с агента (как их запрашивает сервер)

.$path\zabbix_get -s 192.168.3.101 -p 10050 -k agent.version проверить версию агента
.$path\zabbix_get -s 192.168.3.101 -p 10050 -k agent.ping 1 - ok
.$path\zabbix_get -s 192.168.3.101 -p 10050 -k net.if.discovery список сетевых интерфейсов
.$path\zabbix_get -s 192.168.3.101 -p 10050 -k net.if.in["ens33"]
.$path\zabbix_get -s 192.168.3.101 -p 10050 -k net.if.out["ens33"]
```

## UserParameter

Пользовательские параметры

```
UserParameter=process.count,powershell -Command "(Get-Process).Count"
UserParameter=process.vm[*],powershell -Command "(Get-Process $1).ws"
```

Получение данных:

```
C:\zabbix-agent2-6.4.5\bin\zabbix_get.exe -s 127.0.0.1 -p 10050 -k process.count
C:\zabbix-agent2-6.4.5\bin\zabbix_get.exe -s 127.0.0.1 -p 10050 -k process.vm[zabbix_agent2]
C:\zabbix-agent2-6.4.5\bin\zabbix_get.exe -s 127.0.0.1 -p 10050 -k process.vm[powershell]
```

Создать новые Items на сервере:

```
key: process.count
key: process.vm[zabbix_agent2]
```

## Include Plugins

- Добавить параметр `Include` для включения конфигурационных файлов подключаемых плагинов

```
'Include=.\\zabbix_agent2.d\\plugins.d\\*.conf' >> C:\\zabbix-agent2-6.4.5\\conf\\zabbix_agent2.conf
```

- Создать конфигурационный файл с пользовательскими параметрами в каталоге, путь к которому указан в `zabbix_agentd.conf`

```
'UserParameter=Get-Query-Param[*],powershell.exe -noprofile -executionpolicy bypass -File C:\\zabbix-agent2-6.4.5\\conf\\zabbix_agent2.d\\scripts\\User-
```

- Поместить скрипт `Get-Query-Param.ps1` в каталог, путь к которому указан в `User-Sessions.conf`. Скрипт содержит пользовательские параметры, которые он принимает от Zabbix сервера для передачи их в функции скрипта.

```
Param([string]$select)
if ($select -eq "ACTIVEUSER") {
    (Get-Query | where status -match "Active").User
}
if ($select -eq "INACTIVEUSER") {
    (Get-Query | where status -match "Disconnect").User
}
if ($select -eq "ACTIVECOUNT") {
    (Get-Query | where status -match "Active").Status.Count
}
if ($select -eq "INACTIVECOUNT") {
    (Get-Query | where status -match "Disconnect").Status.Count
}
```

- Проверить работу скрипта:

```
$path = "C:\zabbix-agent2-6.4.5\conf\zabbix_agent2.d\scripts\User-Sessions"
$path\Get-Query-Param.ps1 ACTIVEUSER
$path\Get-Query-Param.ps1 INACTIVEUSER
$path\Get-Query-Param.ps1 ACTIVECOUNT
$path\Get-Query-Param.ps1 INACTIVECOUNT
```

- Создать Items с ключами:

```
Get-Query-Param[ACTIVEUSER] Type: Text
Get-Query-Param[INACTIVEUSER] Type: Text
Get-Query-Param[ACTIVECOUNT] Type: Int
Get-Query-Param[INACTIVECOUNT] Type: Int
```

- Макросы:

```
{$ACTIVEMAX} = 16
{$ACTIVEMIN} = 0
```

- Триггеры:

```
last(/Windows-User-Sessions/Get-Query-Param[ACTIVECOUNT])>{$ACTIVEMAX}
min(/Windows-User-Sessions/Get-Query-Param[ACTIVECOUNT],24h)={$ACTIVEMIN}
```

## Zabbix API

### Documentation

```
$ip = "192.168.3.102"
$url = "http://$ip/zabbix/api_jsonrpc.php"
```

Получение токена доступа:

```
$data = @{
    "jsonrpc"="2.0";
    "method"="user.login";
    "params"=@{
        "username"="Admin"; # в версии до 6.4 параметр "user"
        "password"="zabbix";
    };
    "id"=1;
}
$token = (Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result
```

\$token = "2eefd25fdf1590ebcd7978b5bcea1fff755c65b255da8cbd723181b639bb789" сгенерировать токен в UI (<http://192.168.3.102/zabbix/zabbix.php?action=token.list>)

- user.get method

```
$data = @{
    "jsonrpc"="2.0";
    "method"="user.get";
    "params"=@{
    };
    "auth"=$token;
    "id"=1;
}
$users = (Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result
```

- problem.get method

```

$data = @{
    "jsonrpc"="2.0";
    "method"="problem.get";
    "params"=@{
    };
    "auth"=$token;
    "id"=1;
}
(Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result

```

- host.get method

Получить список всех хостов (имя и id)

#### [Endpoint host documentation](#)

**host.create** — создание новых хостов  
**host.delete** — удаление хостов  
**host.get** — получить список хостов  
**host.massadd** - добавление (привязка) объектов на хосты  
**host.massremove** - удаление объектов  
**host.massupdate** - замена или обновление объектов  
**host.update** - обновление хостов

```

$data = @{
    "jsonrpc"="2.0";
    "method"="host.get";
    "params"=@{
        "output"=@( # отфильтровать вывод
            "hostid";
            "host";
        );
    };
    "id"=2;
    "auth"=$token;
}
$hosts = (Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result
$host_id = $hosts[3].hostid # забрать id хоста по индексу

```

- item.get

Получить id элементов данных по наименованию ключа для конкретного хоста

```

$data = @{
    "jsonrpc"="2.0";
    "method"="item.get";
    "params"=@{
        "hostids"=@($host_id); # отфильтровать по хосту
    };
    "auth"=$token;
    "id"=1;
}
$item = (Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result
$item_id = ($item | where key_ -match system.uptime).itemid # забрать id элемента данных

```

- history.get

Получить всю историю элемента данных по его id

```

$data = @{
    "jsonrpc"="2.0";
    "method"="history.get";
    "params"=@{
        "hostids"=@($host_id); # фильтрация по хосту
        "itemids"=@($items_id); # фильтрация по элементу данных
    };
    "auth"=$token;
    "id"=1;
}
$items_data_uptime = (Invoke-RestMethod -Method POST -Uri $url -Body ($data | ConvertTo-Json) -ContentType "application/json").Result # получить

```

Конвертация секунд в TimeSpan :

```

$sec = $items_data_uptime.value

function ConvertSecondsTo-TimeSpan {
    param (
        $insec
    )
    $TimeSpan = [TimeSpan]::fromseconds($insec)
    "{0:dd' day 'hh\:mm\:ss}" -f $TimeSpan
}

```

```
$UpTime = ConvertSecondsTo-TimeSpan $sec[-1]
```

Конвертация из времени Unix :

```

$time = $items_data_uptime.clock

function ConvertFrom-UnixTime {
    param (
        $intime
    )
    $EpochTime = [DateTime]"1/1/1970"
    $TimeZone = Get-TimeZone
    $UTCTime = $EpochTime.AddSeconds($intime)
    $UTCTime.AddMinutes($TimeZone.BaseUtcOffset.TotalMinutes)
}

$GetDateTime = ConvertFrom-UnixTime $time[-1]

($hosts | where hostid -eq $host_id).host получить имя хоста
$UpTime последнее полученное значение времени работы хоста
$GetDateTime время последнего полученного значения

```

## pki

```
New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\My -DnsName "$env:computername" -FriendlyName "Test Certificate" -NotAfter (Get-Dat
создать самоподписанный сертификат (в LocalMachine\My - Сертификаты компьютера\Личное) с сроком действия 5 лет
```

```
Get-ChildItem -Path Cert:\CurrentUser\Root\ список всех установленных сертификатов в хранилище Доверенные корневые ЦС Текущего
пользователя
```

```
Get-ChildItem -Path Cert:\CurrentUser\My\ список самозаверяющих сертификатов в Личное хранилище Текущего пользователя
```

```
Get-ChildItem -Path Cert:\LocalMachine\My\ список самозаверяющих сертификатов в Личное хранилище Локального компьютера
```

```
Get-ChildItem -Path Cert:\LocalMachine\My\ | select NotBefore,NotAfter,Thumbprint,Subject срок действия сертификата
```

```
Get-ChildItem -Path Cert:\LocalMachine\My\ | where Thumbprint -eq D9356FB774EE0E6206B7D5B59B99102CA5B17BDA поиск сертификат по отпечатку
```

```
Get-ChildItem -Path $env:APPDATA\Microsoft\SystemCertificates\My\Certificates\ сертификаты в файловой системе, каждый файл соответствует
сертификату, установленному в личном хранилище текущего пользователя
```

```
Get-ChildItem -Path $env:APPDATA\Microsoft\SystemCertificates\My\Keys\ ссылки на объекты закрытых ключей, созданных поставщиком хранилища
```

ключей (KSP)

```
Get-ChildItem -Path HKCU:\Software\Microsoft\SystemCertificates\CA\Certificates | ft -AutoSize список сертификатов в реестре вошедшего в систему пользователя
```

```
$cert = (Get-ChildItem -Path Cert:\CurrentUser\My\)[1] выбрать сертификат
$cert | Remove-Item удалить сертификат

Export-Certificate -FilePath $home\Desktop\certificate.cer -Cert $cert экспортировать сертификат
$cert.HasPrivateKey проверить наличие закрытого ключа
$pass = "password" | ConvertTo-SecureString -AsPlainText -Force создать пароль для шифрования закрытого ключа
Export-PfxCertificate -FilePath $home\Desktop\certificate.pfx -Password $pass -Cert $certificate экспортировать сертификат с закрытым ключем

Import-Certificate -FilePath $home\Desktop\certificate.cer -CertStoreLocation Cert:\CurrentUser\My импортировать сертификат
Import-PfxCertificate -Exportable -Password $pass -CertStoreLocation Cert:\CurrentUser\My -FilePath $home\Desktop\certificate.pfx
```

## OpenSSL

```
Invoke-WebRequest -Uri https://slproweb.com/download/Win64OpenSSL_Light-3_1_1.msi -OutFile $home\Downloads\OpenSSL-Light-3.1.1.msi
Start-Process $home\Downloads\OpenSSL-Light-3.1.1.msi -ArgumentList '/quiet' -Wait` установить msi пакет в тихом режиме (запуск от имени Администратора)
cd "C:\Program Files\OpenSSL-Win64\bin"
```

- Изменить пароль для PFX  
openssl pkcs12 -in "C:\Cert\domain.ru.pfx" -out "C:\Cert\domain.ru.pem" -nodes экспортируем имеющийся сертификат и закрытый ключ в . pem-файл без пароля с указанием текущего пароля  
openssl pkcs12 -export -in "C:\Cert\domain.ru.pem" -out "C:\Cert\domain.ru\_password.pfx" -nodes конвертируем . pem обратно в . pfx с указанием нового пароля
- Конвертация из закрытого и открытого ключа PEM в PFX  
openssl pkcs12 -export -in "C:\tmp\vpn\vpn.itproblog.ru-crt.pem" -inkey "C:\tmp\vpn\vpn.itproblog.ru-key.pem" -out "C:\tmp\vpn\vpn.iiproblog.ru" in – путь до файла с открытым ключом  
inkey – путь до файла с закрытым ключом  
out – путь до файла, в который будет конвертирован сертификат (pfx)
- Конвертация PFX в CRT  
openssl pkcs12 -in "C:\OpenSSL-Win64\bin\\_.domain.ru.pfx" -clcerts -out "C:\OpenSSL-Win64\bin\\_.domain.ru.crt" указывается текущий и 2 раза новый пароль PEM pass phrase (файл содержит EGIN CERTIFICATE и BEGIN ENCRYPTED PRIVATE KEY)  
openssl pkcs12 -in "C:\OpenSSL-Win64\bin\\_.domain.ru.pfx" -clcerts -nokeys -out "C:\OpenSSL-Win64\bin\\_.domain.ru.crt" без ключа, получить открытую часть (файл содержит только EGIN CERTIFICATE)
- Конвертация PFX в KEY  
openssl pkcs12 -in "C:\OpenSSL-Win64\bin\\_.domain.ru.pfx" -nocerts -out "C:\OpenSSL-Win64\bin\\_.domain.ru.key" файл содержит только BEGIN ENCRYPTED PRIVATE KEY
- Снять пароль с закрытого ключа .key  
openssl rsa -in "C:\OpenSSL-Win64\bin\\_.domain.ru.key" -out "C:\OpenSSL-Win64\bin\\_.domain.ru-decrypted.key"
- CRT и KEY в PFX:  
openssl pkcs12 -inkey certificate.key -in certificate.crt -export -out certificate.pfx

## OpenVPN

```
Invoke-WebRequest -Uri https://swupdate.openvpn.org/community/releases/OpenVPN-2.6.5-I001-amd64.msi -OutFile $home\Downloads\OpenVPN-2.6.5.msi
Start-Process $home\Downloads\OpenVPN-2.6.5.msi -ArgumentList '/quiet /SELECT_OPENSSL_UTILITIES=1' -Wait
msiexec /i $home\Downloads\OpenVPN-2.6.5.msi ADDLOCAL=EasyRSA /passive /quiet # установить отдельный компонент EasyRSA Certificate Management Script
# msiexec /i $home\Downloads\OpenVPN-2.6.5.msi ADDLOCAL=OpenVPN.Service,Drivers,Drivers.Wintun,OpenVPN,OpenVPN.GUI,OpenVPN.GUI.OnLogon,EasyRSA /pass
выборочная установка
# Invoke-WebRequest -Uri https://github.com/OpenVPN/easy-rsa/releases/download/v3.1.5/EasyRSA-3.1.5-win64.zip -OutFile $home\Downloads\EasyRSA-3.1.
скачать отдельный пакет EasyRSA
rm $home\Downloads\OpenVPN-2.6.5.msi

cd "C:\Program Files\OpenVPN\easy-rsa"
Copy-Item vars.example vars файл конфигурации для EasyRSA
```

```

set_var EASYRSA_TEMP_DIR "$EASYRSA_PKI"
set_var EASYRSA_REQ_COUNTRY "RU"
set_var EASYRSA_REQ_PROVINCE "MSK"
set_var EASYRSA_REQ_CITY "MSK"
set_var EASYRSA_REQ_ORG "FAILON.NET"
set_var EASYRSA_REQ_EMAIL "lifailon@domain.ru"
set_var EASYRSA_REQ_OU "IT"
#set_var EASYRSA_KEY_SIZE 2048
#set_var EASYRSA_CA_EXPIRE 3650
#set_var EASYRSA_CERT_EXPIRE 825

.\EasyRSA-Start.bat среда EasyRSA Shell
easyrsa init-pki инициализация PKI, создает директорию: C:\Program Files\OpenVPN\easy-rsa\pk и читает переменные файла \easy-rsa\vars
easyrsa build-ca генерация корневого CA с указанием пароля и произвольное имя сервера (\pk\ca.crt и \pk\private\ca.key)
easyrsa gen-req server nopass генерация запроса сертификата и ключ для сервера OpenVPN - yes (\pk\reqs\server.req и \pk\private\server.key)
easyrsa sign-req server server подписать запрос на выпуск сертификата сервера с помощью CA - yes (\pk\issued\server.crt)
easyrsa gen-dh создать ключ Диффи-Хеллмана (\pk\dh.pem)
easyrsa gen-req client1 nopass генерация запроса сертификата и ключ для клиента OpenVPN (\pk\reqs\client1.req и \pk\private\client1.key)
easyrsa sign-req client client1 подписать запрос на выпуск сертификата клиента с помощью CA - yes (\pk\issued\client1.crt)
easyrsa revoke client1 отозвать сертификат пользователя
openssl rsa -in "C:\Program Files\OpenVPN\easy-rsa\pk\private\client1.key" -out "C:\Program Files\OpenVPN\easy-rsa\pk\private\client1_nopass.key"
снять защиту паролем для ключа (BEGIN ENCRYPTED PRIVATE KEY -> BEGIN PRIVATE KEY)
exit
cd "C:\Program Files\OpenVPN\bin"
.\openvpn --genkey secret ta.key генерация ключа tls-auth (\bin\ta.key)
Move-Item "C:\Program Files\OpenVPN\bin\ta.key" "C:\Program Files\OpenVPN\easy-rsa\pk\"

```

## server.ovpn

```

# Copy-Item "C:\Program Files\OpenVPN\sample-config\server.ovpn" "C:\Program Files\OpenVPN\config-auto\server.ovpn"
New-Item -ItemType File -Path "C:\Program Files\OpenVPN\config-auto\server.ovpn"

```

```

port 1194
proto udp
# Что именно инкапсулировать в туннеле (ethernet фреймы - tap или ip пакеты - tun)
dev tun
ca "C:\\Program Files\\OpenVPN\\easy-rsa\\pki\\ca.crt"
cert "C:\\Program Files\\OpenVPN\\easy-rsa\\pki\\issued\\server.crt"
key "C:\\Program Files\\OpenVPN\\easy-rsa\\pki\\private\\server.key"
dh "C:\\Program Files\\OpenVPN\\easy-rsa\\pki\\dh.pem"
server 192.168.4.0 255.255.255.0
# Хранит список сопоставления ip для клиентов, что бы назначить тот же адрес при перезапуске сервера
# ifconfig-pool-persist "C:\\Program Files\\OpenVPN\\dhcp-client-list.txt"
# Разрешить клиентам подключаться под одним ключом
# duplicate-cn
# max-clients 30
# Разрешить обмен трафиком между клиентами
client-to-client
# compress
tls-auth "C:\\Program Files\\OpenVPN\\easy-rsa\\pki\\ta.key" 0
cipher AES-256-GCM
keepalive 20 60
# Не перечитывать файлы ключей при перезапуске туннеля
persist-key
# Оставляет без изменения устройства tun/tap при перезапуске OpenVPN
persist-tun
status "C:\\Program Files\\OpenVPN\\log\\status.log"
log "C:\\Program Files\\OpenVPN\\log\\openvpn.log"
verb 3
mute 20
windows-driver wintun
# Открыть доступ к подсети за сервером
push "route 192.168.3.0 255.255.255.0"
push "route 192.168.4.0 255.255.255.0"
# Завернуть все запросы клиента (в том числе Интернет трафик) на OpenVPN сервер
# push "redirect-gateway def1"
# push "dhcp-option DNS 192.168.3.101"
# push "dhcp-option DOMAIN failon.net"

```

```

New-NetFirewallRule -DisplayName "AllowOpenVPN-In" -Direction Inbound -Protocol UDP -LocalPort 1194 -Action Allow На сервере
New-NetFirewallRule -DisplayName "AllowOpenVPN-Out" -Direction Outbound -Protocol UDP -LocalPort 1194 -Action Allow на клиенте
Get-Service *openvpn* | Restart-Service

```

## client.ovpn

```

# Copy-Item "C:\\Program Files\\OpenVPN\\sample-config\\client.ovpn" "C:\\Program Files\\OpenVPN\\config-auto\\client.ovpn"
New-Item -ItemType File -Path "C:\\Program Files\\OpenVPN\\config-auto\\client.ovpn"

```

```

client
dev tun
proto udp
remote 26.115.154.67 1194
resolv-retry infinite
nobind
persist-key
persist-tun
ca ca.crt
cert client1.crt
key client1.key
remote-cert-tls server
tls-auth ta.key 1
cipher AES-256-GCM
connect-retry-max 25
# Использовать драйвер wintun и полный путь до сертификатов при использовании openvpn gui
windows-driver wintun
verb 3

```

## Client

iwr -Uri https://openvpn.net/downloads/openvpn-connect-v3-windows.msi -OutFile "\$home\downloads\OpenVPN-Connect-3.msi"

Передать конфигурацию и ключи:

```

client.ovpn
ca.crt
dh.pem
ta.key
client1.crt
client1.key

```

## Route

```

Get-Service RemoteAccess | Stop-Service
Set-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters" -Name "IPEnableRouter" -Value 1 включает IP маршрутизацию
(Get-ItemProperty "HKLM:\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters").IPEnableRouter
Get-NetIPInterface | select ifIndex,InterfaceAlias,AddressFamily,ConnectionState,Forwarding | ft отобразить сетевые интерфейсы
Set-NetIPInterface -ifIndex 13 -Forwarding Enabled ВКЛЮЧИТЬ переадресацию на интерфейсе

```

```

sysctl net.ipv4.ip_forward=1
echo "sysctl net.ipv4.ip_forward = 1" >> /etc/sysctl.conf

```

```

Get-NetRoute
New-NetRoute -DestinationPrefix "192.168.3.0/24" -NextHop "192.168.4.1" -InterfaceIndex 8
route -p add 192.168.3.0 mask 255.255.255.0 192.168.4.1 metric 1
route -p change 192.168.3.0 mask 255.255.255.0 192.168.4.1 metric 2
route -p add 192.168.3.0 mask 255.255.255.0 192.168.4.1 metric 1 if 7 указать номер сетевого интерфейса на который необходимо посыпать
пакет (Wintun Userspace Tunnel)
route print -4
route delete 192.168.3.0

```

```
tracert 192.168.3.101 c 192.168.4.6
```

```

1    17 ms      *      22 ms  192.168.4.1
2    12 ms     13 ms   14 ms  192.168.3.101

```

```

route add -net 192.168.4.0 netmask 255.255.255.0 gw 192.168.3.100
route -e

```

```
traceroute 192.168.4.6 c 192.168.3.101
```

```
1 192.168.3.100 (192.168.3.100) 0.148 ms 0.110 ms 0.106 ms
2 192.168.4.6 (192.168.4.6) 14.573 ms * *
```

```
ping 192.168.3.101 -t c 192.168.4.6
```

```
tcpdump -n -i ens33 icmp на 192.168.3.101
```

```
14:36:34.533771 IP 192.168.4.6 > 192.168.3.101: ICMP echo request, id 1, seq 2962, length 40 # отправил запрос
14:36:34.533806 IP 192.168.3.101 > 192.168.4.6: ICMP echo reply, id 1, seq 2962, length 40 # отправил ответ
```

## NAT

```
Get-Command -Module NetNat
New-NetNat -Name LocalNat -InternalIPInterfaceAddressPrefix "192.168.3.0/24"
Add-NetNatStaticMapping -NatName LocalNat -Protocol TCP -ExternalIPAddress 0.0.0.0 -ExternalPort 80 -InternalIPAddress 192.168.3.102 -InternalPort
Remove-NetNatStaticMapping -StaticMappingID 0
Remove-NetNat -Name LocalNat
```

## WireGuard

```
Invoke-WebRequest "https://download.wireguard.com/windows-client/wireguard-amd64-0.5.3.msi" -OutFile "$home\Downloads\WireGuard-Client-0.5.3.msi"
msiexec.exe /i "$home\Downloads\WireGuard-Client-0.5.3.msi" DO_NOT_LAUNCH=1 /qn
Invoke-WebRequest "http://www.wiresock.net/downloads/wiresock-vpn-gateway-x64-1.1.4.1.msi" -OutFile "$home\Downloads\WireSock-VPN-Gateway-1.1.4.1.m
msiexec.exe /i "http://www.wiresock.net/downloads/wiresock-vpn-gateway-x64-1.1.4.1.msi" /qn
$env:Path = [System.Environment]::GetEnvironmentVariable("Path","Machine") + ";" + [System.Environment]::GetEnvironmentVariable("Path","User")
wg-quick-config -add -start
26.115.154.67:8181
192.168.21.4/24
Successfully saved client configuration: C:\ProgramData\NT KERNEL\WireSock VPN Gateway\wsclient_1.conf
Successfully saved server configuration: C:\ProgramData\NT KERNEL\WireSock VPN Gateway\wiresock.conf
get-service *wire*
wg show
wg-quick-config -add -restart add client
```

wiresock.conf

```
[Interface]
PrivateKey = gCHC0g2JPwr6sXPiaOL4/KTkMyjN9TculrJUA/GORV8=
Address = 192.168.21.5/24
ListenPort = 8181

[Peer]
PublicKey = NoSxjew2RCHiUzI6mlahjd4I+0EcLsoYom/H01z91yU=
AllowedIPs = 192.168.21.6/32
```

wsclient\_1.conf (добавить маршруты для клиента в AllowedIPs)

```
[Interface]
PrivateKey = yIpQRMaGrrk9Y+49E8JhEpFmKzSeecvUAdeNgf1hUM=
Address = 192.168.21.6/24
DNS = 8.8.8.8, 1.1.1.1
MTU = 1420

[Peer]
PublicKey = Fp7674VSYeGj8CYt6RCKR7Qz1y/IKUXCw8ImOFhX3hk=
AllowedIPs = 192.168.21.0/24, 192.168.3.0/24
Endpoint = 26.115.154.67:8181
PersistentKeepalive = 25
```

## VpnClient

```
Get-Command -Module VpnClient

Add-VpnConnection -Name "vpn-failon" -ServerAddress "26.115.154.67" -TunnelType L2TP -L2tpPsk "123098" -EncryptionLevel "Required" -AuthenticationMethod PPTP/L2TP/SSTP/IKEv2/Automatic
-L2tpPsk использовать общий ключ для аутентификации (без параметра, для L2TP аутентификации используется сертификат)
-AuthenticationMethod Pap/Chap/MSChapv2/Eap/MachineCertificate
-EncryptionLevel NoEncryption/Optional/Required/Maximum/Custom
-SplitTunneling заворачивать весь трафик через VPN-туннель (включение Use default gateway on remote network в настройках параметра VPN адаптера)
-UseWinlogonCredential использовать учетные данные текущего пользователя для аутентификации на VPN сервере
-RememberCredential разрешить сохранять учетные данные для VPN подключения (учетная запись и пароль сохраняются в диспетчер учетных данных Windows после первого успешного подключения)
-DnsSuffix domain.local
-AllUserConnection разрешить использовать VPN подключение для всех пользователей компьютера (сохраняется в конфигурационный файл: C:\ProgramData\Microsoft\Network\Connections\Pbk\rasphone.pbk)

Install-Module -Name VPNCredentialsHelper модуль для сохранения логина и пароля в Windows Credential Manager для VPN подключения
Set-VpnConnectionUsernamePassword -connectionname vpn-failon -username user1 -password password

rasdial "vpn-failon" ПОДКЛЮЧИТЬСЯ
Get-VpnConnection -AllUserConnection | select * список VPN подключения, доступных для всех пользователей, настройки и текущий статус подключения (ConnectionStatus)
Add-VpnConnectionRoute -ConnectionName vpn-failon -DestinationPrefix 192.168.3.0/24 -PassThru динамически добавить в таблицу маршрутизации маршрут, который будет активен при подключении к VPN
Remove-VpnConnection -Name vpn-failon -AllUserConnection -Force удалить

Set-VpnConnection -Name "vpn-failon" -SplitTunneling $True включить раздельное туннелирование
Add-VpnConnectionRoute -ConnectionName "vpn-failon" -DestinationPrefix 172.22.22.0/24 настроить маршрутизацию к указанной подсети через VPN-соединение
(Get-VpnConnection -ConnectionName "vpn-failon").routes отобразить таблицу маршрутизации для указанного соединения
Remove-VpnConnectionRoute -ConnectionName "vpn-failon" -DestinationPrefix "172.22.23.0/24"
```

## ProxyClient

```
$user = "lifailon"
$pass = "Proxy"
$SecureString = ConvertTo-SecureString $pass -AsPlainText -Force
$Credential = New-Object System.Management.Automation.PSCredential($user, $SecureString)
[System.Net.Http.HttpClient]::DefaultProxy = New-Object System.Net.WebProxy("http://192.168.3.100:9090")
[System.Net.Http.HttpClient]::DefaultProxy.Credentials = [System.Net.CredentialCache]::DefaultCredentials
[System.Net.Http.HttpClient]::DefaultProxy.Credentials = $Credential
Invoke-RestMethod http://ifconfig.me/ip узнать внешний ip-адрес (по умолчанию в текущей сессии подключения будут происходить через заданный прокси сервер)
Invoke-RestMethod https://kinozal.tv/rss.xml
```

## netsh

### Reverse Proxy

```
netsh interface portproxy add v4tov4 listenport=8080 listenaddress=0.0.0.0 connectport=80 connectaddress=192.168.3.108 настраивает входящее подключение на 8080 порту и переадресует трафик на 80 порт указанного хоста
netsh interface portproxy show all отобразить список всех настроек
netsh interface portproxy delete v4tov4 listenport=8080 listenaddress=0.0.0.0 удалить переадресацию
```

## **Wlan**

```
netsh wlan show profile список сохраненны профилей Wi-Fi и паролей  
netsh wlan show interfaces хар-ки текущей сети (MAC, speed)  
netsh wlan show profile SSID-Name-Network key=clear очистить пароль  
netsh wlan show networks СПИСОК ВИДЕМЫХ СЕТЕЙ  
netsh wlan disconnect ОТКЛЮЧИТЬСЯ от Wi-Fi  
netsh wlan connect name="SSID-Name-Network" подключиться  
netsh wlan show drivers драйвер Wi-Fi  
netsh wlan set hostednetwork mode=allow ssid="WiFi-Test" key="password" создание точки доступа Wi-Fi (SoftAP)
```

## **Firewall**

```
netsh advfirewall set allprofiles state off ОТКЛЮЧИТЬ fw  
netsh advfirewall reset сбросить настройки  
netsh advfirewall firewall add rule name="Open Remote Desktop" protocol=TCP dir=in localport=3389 action=allow открыть порт 3389  
netsh advfirewall firewall add rule name="All ICMP V4" dir=in action=allow protocol=icmpv4 открыть icmp
```

## **OpenSSH**

```
Get-WindowsCapability -Online | ? Name -like 'OpenSSH.Client*'  
Add-WindowsCapability -Online -Name OpenSSH.Client*  
dism /Online /Add-Capability /CapabilityName:OpenSSH.Client~~~0.0.1.0  
iwr https://github.com/PowerShell/Win32-OpenSSH/releases/download/v9.2.2.0p1-Beta/OpenSSH-Win64-v9.2.2.0.msi -OutFile $home\Downloads\OpenSSH-Win64  
скачать  
msiexec /i $home\Downloads\OpenSSH-Win64-v9.2.2.0.msi установить msi пакет  
Set-Service sshd -StartupType Automatic  
Get-NetTCPConnection | where LocalPort -eq 22  
New-NetFirewallRule -Name sshd -DisplayName 'OpenSSH Server (sshd)' -Enabled True -Direction Inbound -Protocol TCP -Action Allow -LocalPort 22  
Get-NetFirewallRule -Name *ssh*  
Start-Process notepad++ C:\Programdata\ssh\sshd_config конфигурационный файл  
GSSAPIAuthentication yes включить Kerberos аутентификацию (через AD)  
SyslogFacility LOCAL0 включить локальное ведение журнала в файл (C:\ProgramData\ssh\logs\sshd.log)  
LogLevel INFO  
Restart-Service sshd  
ssh -K $srv выполнить Kerberos аутентификацию  
ssh lifailon@192.168.3.99 -p 22  
pwsh -command Get-Service  
ssh -L 3101:192.168.3.101:22 -R 3101:192.168.3.101:22 lifailon@192.168.3.101 -p 22 SSH Tunnel lifailon@localhost:3101 -> 192.168.3.101:3101
```

## **PSRemoting over SSH**

```
Add-WindowsCapability -Online -Name OpenSSH.Server~~~0.0.1.0 установка OpenSSH Server  
Get-WindowsCapability -Online | ? Name -like 'OpenSSH.Ser*'  
iex "& { $(irm https://aka.ms/install-powershell.ps1) } -UseMSI" установка PowerShell Core последней версии (требуется на клиентской стороне)  
Set-Service -Name sshd -StartupType "Automatic"  
Start-Service sshd  
Get-NetTCPConnection -State Listen|where {$_.localport -eq '22'}  
Enable-NetFirewallRule -Name *OpenSSH-Server*  
New-ItemProperty -Path "HKLM:\SOFTWARE\OpenSSH" -Name DefaultShell -Value "C:\Program Files\PowerShell\7\pwsh.exe" -PropertyType String -Force  
изменить интерпритатор по умолчанию на pwsh  
notepad $Env:ProgramData\ssh\sshd_config  
  
PasswordAuthentication yes  
Subsystem powershell c:/progra~1/powershell/7/pwsh.exe -sshs -NoLogo # запуск интерпретатора pwsh для удаленных SSH подключений  
  
Restart-Service sshd
```

```
$session = New-PSSession -HostName 192.168.3.100 -Port 2121 -UserName lifailon -SSHTransport  
Invoke-Command -Session $session -ScriptBlock {Get-Service}
```

## WinRM

```
Enter-PSSession -ComputerName $srv подключиться к PowerShell сессии через PSRemoting. Подключение возможно только по FQDN-имени  
Invoke-Command $srv -ScriptBlock {Get-ComputerInfo} выполнение команды через PSRemoting  
$session = New-PSSession $srv открыть сессию  
Get-PSSession отобразить активные сессии  
icm -Session $session {$srsv = $using:srv} передать переменную текущей сессии ($using) удаленную 'Disconnect - PSSession'  
session закрыть сессию \ Remove-PSSession session 'удалить сессию 'Import - Module - Name ActiveDirectory - PSSessionsrv'  
импортировать модуль с удаленного компьютера в локальную сессию
```

## WinRM Configuration

```
winrm quickconfig -quiet изменит запуск службы WinRM (Windows Remote Management) на автоматический, задаст стандартные настройки  
WinRM и добавить исключения для портов в fw  
Enable-PSRemoting -Force включить PowerShell Remoting, работает только для доменного и частного сетевых профилей Windows  
Enable-PSRemoting -SkipNetworkProfileCheck -Force для настройки компьютера в общей (public) сети (работает с версии powershell 6)  
  
$NetProfiles = Get-NetConnectionProfile отобразить профили сетевых подключений  
Set-NetConnectionProfile -InterfaceIndex $NetProfiles[1].InterfaceIndex -NetworkCategory Private изменить тип сети для профиля  
(DomainAuthenticated/Public)  
(Get-CimInstance -ClassName Win32_ComputerSystem).PartOfDomain проверить, что компьютер добавлен в домен AD  
Get-Service WinRM | Set-Service -StartupType AutomaticDelayedStart отложенный запуск  
Get-Service -Name winrm -RequiredServices статус зависимых служб  
New-NetFirewallRule -Profile Any -DisplayName "WinRM HTTP" -Direction Inbound -Protocol TCP -LocalPort 5985,5986  
Test-NetConnection $srv -port 5895 проверить порт  
Test-WSMan $srv -ErrorAction Ignore проверить работу WinRM на удаленном компьютере (игнорировать вывод ошибок для скрипта) или локально  
(localhost)  
  
$Cert = New-SelfSignedCertificate -CertStoreLocation Cert:\LocalMachine\My -DnsName "$env:computername" -FriendlyName "WinRM HTTPS Certificate" -No  
создать самоподписанный сертификат  
$Thumbprint = $Cert.Thumbprint забрать отпечаток  
New-Item -Path WSMAN:\localhost\Listener -Transport HTTPS -Address * -CertificateThumbprint $Thumbprint -Name WinRM_HTTPS_Listener -Force создать  
прослушиватель  
New-NetFirewallRule -DisplayName 'WinRM HTTPS' -Profile Domain,Private -Direction Inbound -Action Allow -Protocol TCP -LocalPort 5986 открыть  
порт в fw  
  
$selector_set = @{  
    Address = "*"  
    Transport = "HTTPS"  
}  
$value_set = @{  
    CertificateThumbprint = "66ABFDA044D8C85135048186E2FDC0DBE6125163"  
}  
New-WSManInstance -ResourceURI "winrm/config/Listener" -SelectorSet $selector_set -ValueSet $value_set  
  
winrm get winrm/config отобразить всю конфигурацию (Client/Service)  
winrm get winrm/config/service/auth конфигурация авторизации на сервере  
winrm enumerate winrm/config/listener текущая конфигурация прослушивателей WinRM (отображает отпечаток сертификата для HTTPS 5986)  
Get-ChildItem -Path Cert:\LocalMachine\My\ | where Thumbprint -eq D9356FB774EE0E6206B7D5B59B99102CA5B17BDA | select * информация о  
сертификате  
  
ls WSMAN:\localhost\Client конфигурацию клиента  
ls WSMAN:\localhost\Service конфигурация сервера  
ls WSMAN:\localhost\Service\auth список всех конфигураций аутентификации WinRM сервера  
Set-Item -path WSMAN:\localhost\Service\auth\basic -value $true разрешить локальную аутентификацию к текущему серверу  
ls HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WSMAN Настройки в реестре (например, для включения аутентификации в \Service\auth_basic =
```

1)

```

Set-Item WSMAN:\localhost\Client\TrustedHosts -Value 192.168.* -Force добавить доверенные хосты в конфигурацию на клиенте, чтобы работала
Negotiate аутентификация через NTLM
Set-Item WSMAN:\localhost\Client\TrustedHosts -Value 192.168.3.100 -Concatenate -Force добавить второй компьютер
ls WSMAN:\localhost\Client\TrustedHosts
Set-Item WSMAN:\localhost\Client\AllowUnencrypted $true Включить передача незашифрованных данных конфигурации клиента
Set-Item WSMAN:\localhost\Service\AllowUnencrypted $true Включить передача незашифрованных данных конфигурации сервера (необходимо
быть в private сети)

Get-PSSessionConfiguration проверить, включен ли PSremoting и вывести список пользователей и групп, которым разрешено подключаться через
WinRM
Set-PSSessionConfiguration -Name Microsoft.PowerShell -ShowSecurityDescriptorUI назначить права доступа через дескриптор безопасности
текущей сессии (до перезагрузки)
(Get-PSSessionConfiguration -Name "Microsoft.PowerShell").SecurityDescriptorSDDL получить настройки дескриптора в формате SDDL
Set-PSSessionConfiguration -Name Microsoft.PowerShell -SecurityDescriptorSDDL $SDDL применить настройки дескриптора на другом компьютере без
использования GUI \

```

```

New-LocalUser "WinRM-Writer" -Password (ConvertTo-SecureString -AsPlainText "123098") создать пользователя
Add-LocalGroupMember -Group "Remote Management Users" -Member "WinRM-Writer" добавить пользователя WinRM-Writer в локальную группу доступа
"Пользователи удаленного управления"
cmdkey /add:192.168.3.99 /user:WinRM-Writer /pass:123098 сохранить пароль в CredentialManager
cmdkey /list
Import-Module CredentialManager
Add-Type -AssemblyName System.Web
New-StoredCredential -Target 192.168.3.99 -UserName WinRM-Writer -Password 123098 -Comment WinRM сохранить пароль в CredentialManager (из PS5)
Get-StoredCredential -AsCredentialObject
$cred = Get-StoredCredential -Target 192.168.3.99
Enter-PSSession -ComputerName 192.168.3.99 -Credential $cred -Authentication Negotiate
Enter-PSSession -ComputerName 192.168.3.99 -Credential $cred -Authentication Basic -Port 5985 работает при отключении allowunencrypted на
стороне сервера и клиента
winrs -r:http://192.168.3.100:5985/wsman -u:WinRM-Writer -p:123098 ipconfig передать команду через winrs (-?)
winrs -r:https://192.168.3.100:5985/wsman -u:WinRM-Writer -p:123098 -ssl ipconfig через https
pwsh -Command "Install-Module -Name PSWSMan" установить модуль для использования в Linux системе

```

## Kerberos

```

.\CheckMaxTokenSize.ps1 -Principals login -OSEmulation $true -Details $true узнать размер токена пользователя в домене
Get-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\Kerberos\Parameters | select maxtokensize максимальный размер токена на
сервере
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\HTTP\Parameters изменить размера, если заголовок пакета аутентификации превышает 16 Кб
(из за большого кол-ва групп)
MaxFieldLength увеличить до 0000ffff (65535)
MaxRequestBytes увеличить до 0000ffff (65535)

```

## PackageManagement

```

Import-Module PackageManagement импортировать модуль
Get-Module PackageManagement информация о модуле
Get-Command -Module PackageManagement отобразить все командлеты модуля
Get-Package отобразить все установленные пакеты PowerShellGallery
Get-Package -ProviderName msi,Programs список установленных программ
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12 Включить использование протокол TLS 1.2 (если не отключены
протоколы TLS 1.0 и 1.1)
Get-PackageSource источники установки пакетов
Get-PackageProvider отобразить список провайдеров менеджеров пакетов
Get-PackageProvider | Where-Object Name -Match nuget
Find-PackageProvider отображение всех доступных менеджеров пакетов
Find-PackageProvider nuget

```

```
Install-PackageProvider NuGet -Force установить менеджер пакетов nuget
Install-PackageProvider PSGallery -Force установить источник
Install-PackageProvider Chocolatey -Force
Set-PackageSource nuget -Trusted разрешить установку пакетов из указанного источника
Register-PSRepository -Name "NuGet" -SourceLocation "https://www.nuget.org/api/v2" -InstallationPolicy Trusted зарегистрировать менеджер пакетов используя url (работает для Power Shell Core)
Set-PackageSource -Name NuGet -Trusted изменить источник по умолчанию
Find-Package PSEverything поиск пакетов по имени во всех менеджерах
Find-Package PSEverything -Provider NuGet поиск пакета у выбранного провайдера
Install-Module PSEverything -Scope CurrentUser установить модуль для текущего пользователя
Install-Package -Name VeeamLogParser -ProviderName PSGallery -scope CurrentUser
Get-Command *Veeam*
Import-Module -Name VeeamLogParser загрузить модуль
Get-Module VeeamLogParser | select -ExpandProperty ExportedCommands отобразить список функций
```

## winget

[Source](#)  
[Web](#)

```
winget list список установленных пакетов
winget search VLC найти пакет
winget show VideoLAN.VLC информация о пакете
winget show VideoLAN.VLC --versions список доступных версий в репозитории
winget install VideoLAN.VLC установить пакет
winget uninstall VideoLAN.VLC удалить пакет
winget download jqlang.jq загрузить пакет (https://github.com/jqlang/jq/releases/download/jq-1.7/jq-windows-amd64.exe)
winget install jqlang.jq добавляет в переменную среду и псевдоним командной строки jq
winget uninstall jqlang.jq
```

## jqlang

```
[uri]$url = $($($irm https://api.github.com/repos/jqlang/jq/releases/latest).assets.browser_download_url -match "windows-amd64").ToString() # получить url
irm $url -OutFile "C:\Windows\System32\jq.exe" # загрузить jq.exe
```

## Scoop

```
Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
irm get.scoop.sh | iex установка
scoop help
scoop search jq
scoop info jq
(scoop info jq).version
scoop cat jq
scoop download jq C:\Users\lifailon\scoop\cache
scoop install jq C:\Users\lifailon\scoop\apps\jq\1.7
scoop list
(scoop list).version
scoop uninstall jq
```

## Chocolatey

```
Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocols
```

```
choco -v
choco -help
```

```
choco list  
choco install adobereader
```

## NuGet

```
Invoke-RestMethod https://dist.nuget.org/win-x86-commandline/latest/nuget.exe -OutFile "$home\Documents\NuGet.exe"  
Invoke-Expression "$home\Documents\NuGet.exe search Selenium.WebDriver"  
Invoke-Expression "$home\Documents\NuGet.exe install Selenium.WebDriver"  
Get-Item $home\Documents\*Selenium*  
  
& "$home\Documents\NuGet.exe" list console-translate  
$nuget_api_key = "<API-KEY>"  
$source = "https://api.nuget.org/v3/index.json"  
$Name = "Console-Translate"  
$path = "$home\Documents\$Name"  
New-Item -Type Directory $path  
Copy-Item "$home\Documents\Git\$Name\$Name\0.2\*" "$path\"  
Copy-Item "$home\Documents\Git\$Name\LICENSE" "$path\"  
Copy-Item "$home\Documents\Git\$Name\README.md" "$path\"  
  
'<?xml version="1.0"?>  
<package>  
  <metadata>  
    <id>Console-Translate</id>  
    <version>0.2.2</version>  
    <authors>Lifailon</authors>  
    <owners>Lifailon</owners>  
    <description>Cross-platform client for translating text in the console, uses API Google (edded public free token), MyMemory and DeepLX (no to</description>  
    <tags>PowerShell, Module, Translate, api</tags>  
    <repository type="git" url="https://github.com/Lifailon/Console-Translate" />  
    <projectUrl>https://github.com/Lifailon/Console-Translate</projectUrl>  
    <licenseUrl>https://github.com/Lifailon/Console-Translate/blob/rsa/LICENSE</licenseUrl>  
    <contentFiles>  
      <files include="Console-Translate.ps1" buildAction="Content" />  
      <files include="Console-Translate.psd1" buildAction="Content" />  
      <files include="lang-iso-639-1.csv" buildAction="Content" />  
      <files include="README.md" buildAction="Content" />  
      <files include="LICENSE" buildAction="Content" />  
    </contentFiles>  
  </metadata>  
</package>' > "$path\$Name.nuspec"  
  
Set-Location $path  
& "$home\Documents\NuGet.exe" pack "$path\$Name.nuspec"  
& "$home\Documents\NuGet.exe" push "$path\$Name.0.2.2.nupkg" -ApiKey $nuget_api_key -Source $source  
& "$home\Documents\NuGet.exe" push "$path\$Name.0.2.2.nupkg" -ApiKey $nuget_api_key -Source $source -SkipDuplicate  
  
Install-Package Console-Translate -Source nuget.org  
Get-Package Console-Translate | select *  
  
Register-PSRepository -Name "NuGet" -SourceLocation "https://www.nuget.org/api/v2" -InstallationPolicy Trusted  
Get-PSRepository  
Find-Module -Name Console-Translate  
Install-Module Console-Translate -Repository NuGet  
  
& "$home\Documents\NuGet.exe" delete Console-Translate 0.2.0 -Source https://api.nuget.org/v3/index.json -ApiKey $nuget_api_key -NoPrompt
```

# Modules

## Get-Query

```
Install-Module Get-Query -Repository NuGet установить модуль
Get-Help Get-Query
Get-Query localhost отобразить всех авторизованных пользователей, их статус и время работы (по умолчанию localhost)
Get-Query 192.168.1.1.1 -proc список всех пользовательских процессов (по умолчанию -user *)
Get-Query 192.168.1.1.1 -proc -user username список процессов указанного пользователя
```

## Console-Translate

```
Install-Module Console-Translate -Repository NuGet
Get-Translate "Module for text translation"
Get-Translate "Модуль для перевода текста"
Get-Translate "Привет world" -LanguageSelected т.к. больше латинских символов (на 1), то перевод будет произведен на английский язык
Get-Translate "Hello друг" -LanguageSelected перевод на русский язык
Get-Translate -Text "Модуль для перевода текста" -LanguageSource ru -LanguageTarget tr
Get-Translate -Provider MyMemory -Text "Hello World" -Alternatives выбрать провайдер перевода и добавить альтернативные варианты вывода
Get-DeepLX "Get select" ru
Start-DeepLX -Job запустить сервер в режиме процесса
Start-DeepLX -Status
Get-DeepLX -Server 192.168.3.99 -Text "Module for text translation" ru
Stop-DeepLX
Get-LanguageCode получение кодов языков по стандарту ISO-639-1
```

## Console-Download

```
Install-Module Console-Download -Repository NuGet установить модуль из менеджера пакетов NuGet
Invoke-Expression $(Invoke-RestMethod "https://raw.githubusercontent.com/Lifailon/Console-Download/rsa/module/Console-Download/Console-Download.ps1"
или импортировать модуль из GitHub репозитория в текущую сессию PowerShell
Invoke-Download -Url "https://github.com/PowerShell/PowerShell/releases/download/v7.4.2/PowerShell-7.4.2-win-x64.zip" загрузить файл в один поток с отображением скорости загрузки в реальном времени (путь загрузки файла по умолчанию: $home\downloads)
Invoke-Download -Url "https://github.com/PowerShell/PowerShell/releases/download/v7.4.2/PowerShell-7.4.2-win-x64.zip" -Thread 3 загрузить один и тоже файл в 3 потока (создается 3 файла, доступно от 1 до 20 потоков)
```

```
$urls = @(
    "https://github.com/PowerShell/PowerShell/releases/download/v7.4.2/PowerShell-7.4.2-win-x64.zip",
    "https://github.com/Lifailon/helperd/releases/download/0.0.1/Helper-Desktop-Setup-0.0.1.exe"
)
Invoke-Download $urls # загрузить параллельно 2 файла

$urls = Get-Content "$home\Desktop\links.txt"
Invoke-Download $urls передайте список URL-адресов из файла для загрузки файлов, эквивалентному числу url

$urls = Get-LookingGlassList отобразить актуальный список конечных точек Looking Glass через LookingHouse (выдает 3 ссылки на загрузку файлов по 10, 100 и 1000 мбайт для каждого региона)
$usaNy = $urls | Where-Object region -like *USA*New*York* отфильтровать список по региону и городу
$url = $usaNy[0].url100mb забрать первую ссылку с файлом на 100 Мбайт
Invoke-Download $url начать загрузку файла
Invoke-Download $url -Thread 3 начать загрузку 3-х одинаковых файлов
```

## PSEverything

```
Install-Module PSEverything -Repository NuGet
Find-Everything pingui найти все файлы в системе (на всех локальных дисках) с именем pingui через dll csharp версии Everything (по умолчанию)
Find-Everything pingui-0.1.py | Format-List
Find-Everything pingui -es использовать cli версию Everything для поиска (при первом использовании версии, необходимо дождаться
```

автоматической установки файлов зависимостей)

Find-Everything pingui-0.1 -ComputerName localhost поиск на удаленном компьютере через REST API, если запущен HTTP-сервер Everything

## HardwareMonitor

Install-Module HardwareMonitor -Repository NuGet -Scope AllUsers

Install-LibreHardwareMonitor установить и запустить LibreHardwareMonitor в систему

(<https://github.com/LibreHardwareMonitor/LibreHardwareMonitor>)

Install-OpenHardwareMonitor установить OpenHardwareMonitor (<https://github.com/openhardwaremonitor/openhardwaremonitor>)

Get-Sensor | Where-Object {(\$\_.SensorName -match "Temperature") -or (\$\_.SensorType -match "Temperature") } | Format-Table Использовать

LibreHardwareMonitor и WMI/CIM (по умолчанию) и отфильтровать вывод по наименованию датчикам или типу сенсора для вывода датчиков температуры

Get-Sensor -Open использовать OpenHardwareMonitor

Get-Sensor -ComputerName 192.168.3.99 -Port 8086 | Format-Table вывести датчики системы через REST API

Get-Sensor -ComputerName 192.168.3.99 -Port 8085 -User hardware -Password monitor | Where-Object Value -notmatch "^0,0" | Format-Table использовать авторизацию и отфильтровать вывод не пустых датчиков

Get-Sensor -Library | Where-Object Value -ne 0 | Format-Table использовать динамическую библиотеку (dll) через .NET

## CrystalDisk-Cli

Install-Module CrystalDisk-Cli -Repository NuGet

Get-DiskInfoSettings отобразить настройки программы Crystal-DiskInfo (<https://github.com/hiyohiyo/CrystalDiskInfo>)

Get-DiskInfoSettings -AutoRefresh 5 изменить время сканирования на 5 минут

Get-DiskInfo -List отобразить список дисков

Get-DiskInfo получить результаты последнего сканирования (статус, температура и т.п.)

Get-DiskInfo -Report | Select-Object Name, Date, HealthStatus, Temperature получить актуальный отчет (запустить сканирование и дождаться результатов)

## PS-Pi-Hole

```
$path_psm = ($env:PSModulePath.Split(";"))[0]+"\Invoke-Pi-Hole\Invoke-Pi-Hole.psm1"
```

```
if (!(Test-Path $path_psm)) {
```

```
    New-Item $path_psm -ItemType File -Force
```

```
}
```

```
irm https://raw.githubusercontent.com/Lifailon/PS-Pi-Hole/rsa/Invoke-Pi-Hole/Invoke-Pi-Hole.psm1 | Out-File $path_psm -Force
```

sudo cat /etc/pihole/setupVars.conf | grep WEBPASSWORD получить токен доступа

\$Server = "192.168.1.253"

\$Token = "5af9bd44aebce0af6206fc8ad4c3750b6bf2dd38fa59bba84ea9570e16a05d0f"

Invoke-Pi-Hole -Versions -Server \$Server -Token \$Token получить текущую версию ядра (backend) и веб (frontend) на сервере а также последнюю доступную версию для обновления

Invoke-Pi-Hole -Releases -Server \$Server -Token \$Token узнать последнюю доступную версию в репозитории на GitHub

Invoke-Pi-Hole -QueryLog -Server \$Server -Token \$Token отобразить полный журнал запросов (клиент, домен назначения, тип записи, статус времени запроса и адрес пересылки forward dns - куда ушел запрос)

Invoke-Pi-Hole -AdList -Server \$Server -Token \$Token получить списки блокировок используемых на сервере (дата обновления, количество доменов и url-источника)

Invoke-Pi-Hole -Status -Server \$Server -Token \$Token статус работы режима блокировки

Invoke-Pi-Hole -Enable -Server \$Server -Token \$Token включить блокировку

Invoke-Pi-Hole -Disable -Server \$Server -Token \$Token отключить блокировку

Invoke-Pi-Hole -Stats -Server \$Server -Token \$Token подключиться к серверу Pi-Hole для получения статистики (метрики: количество доменов для блокировки, количество запросов и блокировок за сегодня и т.д.)

Invoke-Pi-Hole -QueryTypes -Server \$Server -Token \$Token статистика запросов по типу записей относительно 100%

Invoke-Pi-Hole -TopClient -Server \$Server -Token \$Token список самых активных клиентов (ip/name и количество запросов проходящих через сервер)

Invoke-Pi-Hole -TopPermittedDomains -Count 100 -Server \$Server -Token \$Token список самых посещаемых доменов и количество запросов

Invoke-Pi-Hole -LastBlockedDomain -Server \$Server -Token \$Token адрес последнего заблокированного домена

Invoke-Pi-Hole -ForwardServer -Server \$Server -Token \$Token список серверов для пересылки, которым обычно выступает DNS-сервер стоящий за

Pi-Hole в локальной сети, например AD

Invoke-Pi-Hole -Data -Server \$Server -Token \$Token количество запросов за каждые 10 минут в течение последних 24 часов

## Check-Host

```
$path = $($env:PSModulePath -split ";")[0] + "\Get-CheckHost"
if (Test-Path $path) {
    Remove-Item $path -Force -Recurse
    New-Item -ItemType Directory -Path $path
} else {
    New-Item -ItemType Directory -Path $path
}
Invoke-RestMethod "https://raw.githubusercontent.com/Lifailon/Check-Host/rsa/Get-CheckHost/Get-CheckHost.ps1" -OutFile "$path\Get-CheckHost.ps1"
```

Install-Module CheckHost установить модуль работы с Check-Host (<https://check-host.net>) через ари

Get-CheckHost -List список хостов (node) разных регионов (40)

Get-CheckHost -Server google.com -Type ping -Count 5 использовать 5 любых хостов для 4 пингов с каждого до указанного url (google)

Get-CheckHost -Server google.com -Type dns -Count 5 проверить DNS (возвращает А-запись или диапазон с ip-адресом)

Get-CheckHost -Server google.com:443 -Type http -Count 5 проверить доступность порта

Get-CheckHost -Server google.com:443 -Type tcp -Count 5 проверить доступность TCP или UDP порта

## PSDomainTest

Install-Module PSDomainTest -Repository NuGet -Scope CurrentUser

Get-DomainTest -Domain github.com -Warning протестировать домен и DNS записи на ошибки (вывести только ошибки) через ZoneMaster (<https://github.com/zonemaster/zonemaster>)

Get-DomainTest -Domain github.com -Warning -json вывод в формате json

Get-DomainTest -Domain github.com -html | Out-File .\result.html получить отчет в формате HTML-таблицы с фильтрацией по столбцам

## WinAPI

Install-Module ps.win.api -Repository NuGet -AllowClobber

Import-Module ps.win.api

Get-Command -Module ps.win.api

Start-WinAPI запустить сервер

Test-WinAPI статус сервера

Stop-WinAPI остановить сервер

Read-WinAPI отобразить лог в реальном времени

Get-Hardware вывести сводную информацию о системе с использованием потоков (фоновых заданий)

Get-DiskPhysical отобразить список физических дисков, их размер, интерфейс и статус

Get-DiskPhysical -ComputerName 192.168.3.100 -Port 8443 -User rest -Pass api получить информацию с удаленного сервера, на котором запущен сервер WinAPI (доступно для всех функций модуля)

Get-DiskLogical список логических дисков (включая виртуальные диски), их файловая система, общий и используемый размер в гб и процентах

Get-DiskPartition список разделов физических дисков (отобразит скрытые разделы, загрузочный и порядок назначения байт)

Get-Smart статус работы всех дисков и текущая температура

Get-IOps количество операций ввода/вывода дисковой подсистемы

Get-Files подробная информация о файлах (добавляет дату создания, изменения, доступа, количество дочерних файлов и директорий)

Find-Process поиск пути к исполняемому файлу по имени остановленного (не запущенного) процесса в директориях

Get-ProcessPerformance информация о процессах (Get-Process) в человеко-читаемом формате

Get-CPU список всех ядер и нагрузка на каждое ядро и на все сразу (суммарное процессорное время, привилегированное и пользовательское время процессора)

Get-MemorySize размер оперативной, виртуальной памяти, суммарное потребление памяти процессов и путь к файлу подкачки

Get-MemorySlots список всех слотов оперативной памяти

Get-NetInterfaceStat статистика активного сетевого интерфейса за время с момента загрузки системы (количество пакетов и ГБ)

Get-NetInterfaceStat -Current текущая статистика активного сетевого интерфейса (количество пакетов и МБ/с)

Get-NetIpConfig конфигурация всех сетевых интерфейсов (ip и мас-адрес, статус DHCP сервера, дата аренды)

Get-NetStat развернутая статистика сетевых интерфейсов (из Get-NetTCPConnection), добавляет имя процесса, имя удаленного хоста (через nslookup), время работы процесса (не процессорное время) и путь к исполняемому файлу

Get-Performance информация из счетчиков (Get-Counter) человеко-читаемом формате  
Get-VideoCard информация о всех видео-картах (наименование, частота и объем видео-памяти)  
Get-Software список установленного программного обеспечения  
Get-WinUpdate список обновлений Windows (дата установки и источник)  
Get-Driver список установленных драйверов (имя, провайдер, версия и дата установки)

## pSyslog

```
Install-Module pSyslog -Repository NuGet
Start-pSyslog -Port 514 запустить сервер на порту 514 (по умолчанию)
Start-pSyslog -RotationSize 500 указать размер файла локального журнала для его ротации (обрезания) в КБ
Get-pSyslog -Status | Format-List отобразить статус работы
Get-pSyslog вывести журнал сообщений в реальном времени
Stop-pSyslog остановить сервер
Send-pSyslog -Content "Test" -Server 192.168.3.99 отправить сообщение на Syslog-сервер
Send-pSyslog -Content "Test" -Server 192.168.3.99 -Type Informational -PortServer 514 -PortClient 55514
(Get-Service -Name WinRM).Status | Send-pSyslog -Server 192.168.3.102 -Tag Service[WinRM]
Send-pSyslog -Content "test" -Server 192.168.3.99 -PortServer 514 -Base64 использовать шифрование при отправки сообщения (расшифровка
работает только для сервера pSyslog)
Start-UDPRelay -inPort 515 -outIP 192.168.3.102 -outPort 514 запустить сервер в режиме UDP-Relay, который слушает на порту 515 и
переадресует сообщения на Syslog сервер 192.168.3.102 с портом 514
Send-pSyslog -Server 192.168.3.99 -PortServer 515 -Content $(Get-Date) отправить сообщение на сервере UDP-Relay
Show-pSyslog -Type Warning -Count отобразить метрики (количество ошибок)
Show-pSyslog -Type Alert -Count
Show-pSyslog -Type Critical -Count
Show-pSyslog -Type Error -Count
Show-pSyslog -Type Emergency -Count
Show-pSyslog -Type Informational -Count
Show-pSyslog -LogFile 05-06 | Out-GridView прочитать локальный журнал логирования и вывести в GridView для фильтрации сообщений
Show-pSyslog -Count отобразить количество сообщений локального журнала
Show-pSyslog -Count -LogFile 10-06 выбрать журнал по дате
```

## Syslog source message

```
Add-Type -TypeDefintion @"
public enum Syslog_Facility {
    kern,      // 0 kernel (core) messages
    user,      // 1 user level messages
    mail,      // 2 mail system
    daemon,    // 3 system daemons
    auth,      // 4 security/authorization messages (login/su)
    syslog,    // 5 syslog daemon
    lpr,       // 6 line printer subsystem (creating jobs and send to spool for print by using lpd)
    news,      // 7 network news subsystem (USENET)
    uucp,      // 8 Unix-to-Unix Copy subsystem
    cron,      // 9 scheduling daemon
    authpriv,  // 10 security/authorization private messages
    ftp,       // 11 FTP daemon
    ntp,       // 12 NTP daemon
    security,  // 13 security log audit
    console,   // 14 console log alert
    clock,     // 15 clock subsystem
    local0,    // 16 local use
    local1,    // 17
    local2,    // 18
    local3,    // 19
    local4,    // 20
    local5,    // 21
    local6,    // 22
    local7    // 23
}
"@"
```

## Syslog type message

```
Add-Type -TypeDefintion @"
public enum Syslog_Severity {
    Emergency, // 0 emerg
    Alert,      // 1 alert
    Critical,   // 2 crit
    Error,      // 3 err
    Warning,    // 4 warning
    Notice,     // 5 notice
    Informational, // 6 info
    Debug       // 7 debug
}
"@"
```

## Atlassian

### Bitbucket

```
$url = "https://github.com/AtlassianPS/BitbucketPS/archive/refs/heads/master.zip"
Invoke-RestMethod $url -OutFile $home\Downloads\BitbucketPS.zip
Expand-Archive -Path "$home\Downloads\BitbucketPS.zip" -OutputPath "$home\Downloads"
Copy-Item -Path "$home\Downloads\BitbucketPS-master\*" -Destination $($env:PSModulePath.Split(";")[0])\PSBitBucket" -Recurse
Remove-Item "$home\Downloads\Bitbucket*" -Recurse -Force

Import-Module PSBitBucket
Get-Command -Module PSBitBucket
Set-BitBucketConfigServer -Url $url -User username -Password password установить конфигурацию сервера BitBucket
Get-BitBucketConfigServer получить текущую конфигурацию сервера BitBucket
```

```
Get-Repositories получить список всех репозиториев для текущей конфигурации сервера BitBucket  
Get-ProjectKey получить ключ проекта BitBucket  
Get-BranchList -Repository pSyslog список всех веток в репозитории  
Get-Branch -Repository pSyslog -Branch main получить информацию о конкретной ветке репозитория  
Get-CommitMessage -Repository pSyslog -CommitHash $hash получить сообщение коммита по его хэшу  
Get-Commits -Repository pSyslog -Limit 10 список последних 10 коммитов в репозитории  
Get-CommitsForBranch -Repository pSyslog -Branch main список коммитов для конкретной ветки в репозитории
```

## Jira

```
Install-Module JiraPS -Scope CurrentUser -Repository PSGallery -AllowClobber -Force  
Get-Command -Module JiraPS  
Get-JiraServerInfo информация о сервере  
Add-JiraFilterPermission добавить разрешения для фильтра  
Add-JiraGroupMember добавить участника в группу  
Add-JiraIssueAttachment добавить вложения к задаче  
Add-JiraIssueComment добавить комментария к задаче  
Add-JiraIssueLink добавить ссылки на задачу  
Add-JiraIssueWatcher добавить наблюдателя к задаче  
Add-JiraIssueWorklog добавить рабочего журнала к задаче  
Find-JiraFilter поиск фильтра  
Format-Jira форматирование данных Jira  
Get-JiraComponent получение компонента проекта  
Get-JiraConfigServer получение конфигурации сервера Jira  
Get-JiraField получение поля Jira  
Get-JiraFilter получение фильтра  
Get-JiraFilterPermission получение разрешения фильтра  
Get-JiraGroup получение группы  
Get-JiraGroupMember получение участников группы  
Get-JiraIssue получение задачи  
Get-JiraIssueAttachment получение вложения задачи  
Get-JiraIssueAttachmentFile получение файла вложения задачи  
Get-JiraIssueComment получение комментария задачи  
Get-JiraIssueCreateMetadata получение метаданных создания задачи  
Get-JiraIssueEditMetadata получение метаданных редактирования задачи  
Get-JiraIssueLink получение ссылки задачи  
Get-JiraIssueLinkType получение типа ссылки задачи  
Get-JiraIssueType получение типа задачи  
Get-JiraIssueWatcher получение наблюдателя задачи  
Get-JiraIssueWorklog получение рабочего журнала задачи  
Get-JiraPriority получение приоритета задачи  
Get-JiraProject получение проекта  
Get-JiraRemoteLink получение удаленной ссылки  
Get-JiraServerInformation получение информации о сервере Jira  
Get-JiraSession получение сессии  
Get-JiraUser получение пользователя  
Get-JiraVersion получение версии проекта  
Invoke-JiraIssueTransition выполнение перехода задачи  
Invoke-JiraMethod выполнение метода Jira  
Move-JiraVersion перемещение версии проекта  
New-JiraFilter создание нового фильтра  
New-JiraGroup создание новой группы  
New-JiraIssue создание новой задачи  
New-JiraSession создание новой сессии  
New-JiraUser создание нового пользователя  
New-JiraVersion создание новой версии проекта  
Remove-JiraFilter удаление фильтра
```

```
Remove-JiraFilterPermission удаление разрешения фильтра
Remove-JiraGroup удаление группы
Remove-JiraGroupMember удаление участника группы
Remove-JiraIssue удаление задачи
Remove-JiraIssueAttachment удаление вложения задачи
Remove-JiraIssueLink удаление ссылки задачи
Remove-JiraIssueWatcher удаление наблюдателя задачи
Remove-JiraRemoteLink удаление удаленной ссылки
Remove-JiraSession удаление сессии
Remove-JiraUser удаление пользователя
Remove-JiraVersion удаление версии проекта
Set-JiraConfigServer установка конфигурации сервера Jira
Set-JiraFilter установка фильтра
Set-JiraIssue установка задачи
Set-JiraIssueLabel установка метки задачи
Set-JiraUser установка пользователя
Set-JiraVersion установка версии проекта
```

## Confluence

```
Install-Module ConfluencePS -Scope CurrentUser -Repository PSGallery -AllowClobber -Force
Get-Command -Module ConfluencePS
Add-ConfluenceAttachment добавить вложения к странице
Add-ConfluenceLabel добавить метки к странице
ConvertTo-ConfluenceStorageFormat конвертация содержимого в формат хранения Confluence
ConvertTo-ConfluenceTable конвертация данных в таблицу Confluence
Get-ConfluenceAttachment получение вложения страницы
Get-ConfluenceAttachmentFile получение файла вложения страницы
Get-ConfluenceChildPage получение дочерних страниц
Get-ConfluenceLabel получение меток страницы
Get-ConfluencePage получение информации о странице
Get-ConfluenceSpace получение информации о пространстве
Invoke-ConfluenceMethod выполнение метода Confluence
New-ConfluencePage создание новой страницы
New-ConfluenceSpace создание нового пространства
Remove-ConfluenceAttachment удаление вложения страницы
Remove-ConfluenceLabel удаление метки со страницы
Remove-ConfluencePage удаление страницы
Remove-ConfluenceSpace удаление пространства
Set-ConfluenceAttachment установка вложения страницы
Set-ConfluenceInfo установка информации о странице
Set-ConfluenceLabel установка метки страницы
Set-ConfluencePage установка страницы
```

## Pester

### Source

```
Install-Module -Name Pester -Repository PSGallery -Force -AllowClobber
Import-Module Pester
$(Get-Module Pester -ListAvailable).Version
.Tests.ps1
```

```

function Add-Numbers {
    param (
        [int]$a,
        [int]$b
    )
    $a + $b
}

Describe "Add-Numbers" {
    Context "При сложении двух чисел" {
        It "Должна вернуться правильная сумма" {
            $result = Add-Numbers -a 3 -b 4
            $result | Should -Be 7
        }
    }
    Context "При сложении двух чисел" {
        It "Должна вернуться ошибка (5+0 -не 4)" {
            $result = Add-Numbers -a 5 -b 0
            $result | Should -Be 4
        }
    }
}

function Get-RunningProcess {
    return Get-Process | Select-Object -ExpandProperty Name
}

Describe "Get-RunningProcess" {
    Context "При наличии запущенных процессов" {
        It "Должен возвращать список имен процессов" {
            $result = Get-RunningProcess
            $result | Should -Contain "svchost"
            $result | Should -Contain "explorer"
        }
    }
    Context "Когда нет запущенных процессов" {
        It "Должен возвращать пустой список" {
            # Замокать функцию Get-Process, чтобы она всегда возвращала пустой список процессов
            Mock Get-Process { return @() }
            $result = Get-RunningProcess
            $result | Should -BeEmpty
        }
    }
}

```

## DSC

```

Import-Module PSDesiredStateConfiguration
Get-Command -Module PSDesiredStateConfiguration
(Get-Module PSDesiredStateConfiguration).ExportedCommands
Get-DscLocalConfigurationManager

Get-DscResource
Get-DscResource -Name File -Syntax СИНТАКСИС

Ensure = Present настройка должна быть включена (каталог должен присутствовать, процесс должен быть запущен, если нет – создать, запустить)

Ensure = Absent настройка должна быть выключена (каталога быть не должно, процесс не должен быть запущен, если нет – удалить, остановить)

```

```

Configuration TestConfiguraion
{
    Ctrl+Space
}

Configuration DSConfigurationProxy
{
    Node vproxy-01
    {
        File CreateDir
        {
            Ensure = "Present"
            Type = "Directory"
            DestinationPath = "C:\Temp"
        }
        Service StopW32time
        {
            Name = "w32time"
            State = "Stopped" ` Running
        }
        WindowsProcess RunCalc
        {
            Ensure = "Present"
            Path = "C:\WINDOWS\system32\calc.exe"
            Arguments = ""
        }
        Registry RegSettings
        {
            Ensure = "Present"
            Key = "HKEY_LOCAL_MACHINE\SOFTWARE\MySoft"
            ValueName = "TestName"
            ValueData = "TestValue"
            ValueType = "String"
        }
        # WindowsFeature IIS
        #
        # {
        #     Ensure = "Present"
        #     Name = "Web-Server"
        # }
    }
}

$Path = (DSConfigurationProxy).DirectoryName
Test-DscConfiguration -Path $Path | select * ResourcesInDesiredState - уже настроено, ResourcesNotInDesiredState - не настроено (не
соответствует)
Start-DscConfiguration -Path $Path
Get-Job
$rv = "vproxy-01"
Get-Service -ComputerName $rv | ? name -match w32time # Start-Service
icm $rv {Get-Process | ? ProcessName -match calc} | ft # Stop-Process -Force
icm $rv {ls C:\ | ? name -match Temp} | ft rm

```

```

Configuration InstallPowerShellCore {
    Import-DscResource -ModuleName PSDesiredStateConfiguration
    Node localhost {
        Script InstallPowerShellCore {
            GetScript = {
                return @{
                    GetScript = $GetScript
                }
            }
            SetScript = {
                [string]$url = $(Invoke-RestMethod https://api.github.com/repos/PowerShell/PowerShell/releases/latest).assets.bro
                $downloadPath = "$home\Downloads\PowerShell.zip"
                $installPath = "$env:ProgramFiles\PowerShell\7"
                Invoke-WebRequest -Uri $url -OutFile $downloadPath
                Expand-Archive -Path $downloadPath -DestinationPath $installPath -Force
            }
            TestScript = {
                return Test-Path "$env:ProgramFiles\PowerShell\7\pwsh.exe"
            }
        }
    }
}

$Path = (InstallPowerShellCore).DirectoryName
Test-DscConfiguration -Path $Path
Start-DscConfiguration -Path $path -Wait -Verbose
Get-Job

```

## PSAppDeployToolkit

### Install-DeployToolkit

```

$githubRepository = "psappdeploytoolkit/psappdeploytoolkit"
$filenamePatternMatch = "PSAppDeployToolkit*.zip"
$psadtReleaseUri = "https://api.github.com/repos/$githubRepository/releases/latest"
$psadtDownloadUri = ((Invoke-RestMethod -Method GET -Uri $psadtReleaseUri).assets | Where-Object name -like $filenamePatternMatch ).browser_download_url
$zipExtractionPath = Join-Path $env:USERPROFILE "Downloads" "PSAppDeployToolkit"
$zipTempDownloadPath = Join-Path -Path $($([System.IO.Path]::GetTempPath()) -ChildPath $($Split-Path -Path $psadtDownloadUri -Leaf))
## Download to a temporary folder
Invoke-WebRequest -Uri $psadtDownloadUri -Out $zipTempDownloadPath
## Remove any Zone.Identifier alternate data streams to unblock the file (if required)
Unblock-File -Path $zipTempDownloadPath
New-Item -Type Directory $zipExtractionPath
Expand-Archive -Path $zipTempDownloadPath -OutputPath $zipExtractionPath -Force
Write-Host ("File: {0} extracted to Path: {1}" -f $psadtDownloadUri, $zipExtractionPath) -ForegroundColor Yellow
Remove-Item $zipTempDownloadPath

```

### Deploy-Notepad-Plus-Plus

```

$url_notepad = "https://github.com/notepad-plus-plus/notepad-plus-plus/releases/download/v8.6.6/npp.8.6.6.Installer.x64.exe"
Invoke-RestMethod $url_notepad -OutFile "$home\Downloads\PSAppDeployToolkit\Toolkit\Files\npp.8.6.6.Installer.x64.exe"

```

```

'# Подключаем модуль PSAppDeployToolkit
Import-Module "$PSScriptRoot\AppDeployToolkit\AppDeployToolkitMain.ps1"
# Название приложения
$AppName = "Notepad++"
# Версия приложения
$AppVersion = "8.6.6"
# Путь к установщику Notepad++
$InstallerPath = "$PSScriptRoot\Files\npp.$AppVersion.Installer.x64.exe"
# Проверка существования установщика
If (-not (Test-Path $InstallerPath)) {
    Write-Host "Установщик Notepad++ не найден: $InstallerPath"
    Exit-Script -ExitCode 1
}
# Настройки установки Notepad++
$InstallerArguments = "/S /D=$ProgramFiles\Notepad++"
Function Install-Application {
    # Выводим сообщение о начале установки
    Show-InstallationWelcome -CloseApps "iexplore" -CheckDiskSpace -PersistPrompt
    # Запускаем установку
    Execute-Process -Path $InstallerPath -Parameters $InstallerArguments -WindowStyle Hidden -IgnoreExitCodes "3010"
    # Выводим сообщение об успешной установке
    Show-InstallationPrompt -Message "Установка $AppName завершена." -ButtonRightText "Закрыть" -Icon Information -NoWait
    # Завершаем процесс установки
    Exit-Script -ExitCode $AppDependentExitCode
}
Install-Application' | Out-File "$home\Downloads\PSAppDeployToolkit\Toolkit\Deploy-Application.ps1" -Encoding unicode

```

powershell -File "\$home\Downloads\PSAppDeployToolkit\Toolkit\Deploy-Application.ps1"

## Uninstall-Notepad-Plus-Plus

```

'Import-Module "$PSScriptRoot\AppDeployToolkit\AppDeployToolkitMain.ps1"
$AppName = "Notepad++"
$UninstallerPath = "C:\Program Files\Notepad++\uninstall.exe"
If (-not (Test-Path $UninstallerPath)) {
    Write-Host "Деинсталлятор Notepad++ не найден: $UninstallerPath"
    Exit-Script -ExitCode 1
}
Function Uninstall-Application {
    Show-InstallationWelcome -CloseApps "iexplore" -CheckDiskSpace -PersistPrompt
    Execute-Process -Path $UninstallerPath -Parameters "/S" -WindowStyle Hidden -IgnoreExitCodes "3010"
    Show-InstallationPrompt -Message "Программа $AppName удалена." -ButtonRightText "Закрыть" -Icon Information -NoWait
    Exit-Script -ExitCode $AppDependentExitCode
}
Uninstall-Application' | Out-File "$home\Downloads\PSAppDeployToolkit\Toolkit\Deploy-Application.ps1" -Encoding unicode

```

powershell -File "\$home\Downloads\PSAppDeployToolkit\Toolkit\Deploy-Application.ps1"

## Deploy-WinSCP

```

$PSAppDeployToolkit = "$home\Downloads\PSAppDeployToolkit\
$version = "6.3.3"
$url_winscp = "https://cdn.winscp.net/files/WinSCP-$version.msi?secure=P2HLWGKaMDigpDQw-H9BgA==,1716466173"
$WinSCP_Template = Get-Content "$PSAppDeployToolkit\Examples\WinSCP\Deploy-Application.ps1" # читаем пример конфигурации для WinSCP
$WinSCP_Template_Latest = $WinSCP_Template -replace "6.3.2","$version" # обновляем версию на актуальную
$WinSCP_Template_Latest > "$PSAppDeployToolkit\Toolkit\Deploy-Application.ps1" # заменяем скрипт развертывания
Invoke-RestMethod $url_winscp -OutFile "$PSAppDeployToolkit\Toolkit\Files\WinSCP-$version.msi" # загружаем msi-пакет
powershell -File "$PSAppDeployToolkit\Toolkit\Deploy-Application.ps1" # запускаем установку

```

# Load Testing

## Apache Benchmark

```
$path = "$HOME\Downloads\apache"
New-Item $path -Type Directory
cd $path
curl -L -o apache.zip "https://www.apachelounge.com/download/VS17/binaries/httpd-2.4.63-250207-win64-VS17.zip"
Expand-Archive -Path apache.zip
Copy-Item .\Apache24\bin\ $HOME\Documents\apache\ -Recurse
cd .. && Remove-Item "$HOME\Downloads\apache" -Recurse

$ab = "$HOME\Documents\apache\ab.exe"
. $ab -n 10000 -c 100 http://192.168.3.100:8444/api/provider/list
```

```
Количество одновременных запросов: 100
Время проведения тестов: 52,402 секунды
Выполненные запросы: 10000
Неудачные запросы: 0
Передано всего: 6830000 байтов
Передано HTML: 3290000 байт
RPS (Requests Per Second): 190,83 [#/sec] (среднее)
Время одного запроса: 524.017 [MS] (среднее)
Время одного запроса: 5.240 [MS] (среднее, во всех одновременных запросах)
Скорость передачи: 127,28 [Kbytes/Sec]
```

## Locust

[Locust](#) - это инструмент нагрузочного тестирования для HTTP и других протоколов на Python.

```
pip3 install locust
```

```

echo '
import os
from locust import HttpUser, task, between
class TorApiUser(HttpUser):
    # Каждый виртуальный пользователь будет ждать от 2 до 5 секунд перед выполнением следующего @task
    wait_time = between(2, 5)
    # Определяем заголовки запросов
    headers = {
        "User-Agent": "Locust"
    }
    # Получаем параметры из переменных окружения или использовать значение по умолчанию
    QUERY = os.getenv("QUERY", "test")
    # GET запросы (вес приоритета задачи для частоты ее выполнения, чем выше, тем чаще выполнение)
    @task(1)
    def test_status(self):
        self.client.get("/api/provider/list", headers=self.headers)
    @task(2)
    def test_search(self):
        # Словарь параметров, который автоматически конвертируется в строку запроса (?key=value&key2=value2)
        searchParams = {
            "query": {self.QUERY},
            "category": 0,
            "page": 0
        }
        self.client.get("/api/search/title/rutracker", headers=self.headers, params=searchParams)
    # POST запрос с телом запроса
    # @task(3)
    # def test_post_auth(self):
    #     self.client.post("/api/auth", json={"username": "admin", "password": "password"})
' > locustfile.py

```

```

locust -f locustfile.py --host http://192.168.3.100:8444
$env:QUERY = "The+Rookie" определяем переменную окружения для параметра запросов
locust -f locustfile.py --host http://192.168.3.100:8444 -u 10 -r 2 -t 30s количество виртуальных пользователей (VU), частота появления
новых пользователей в секунду (10 пользователей будут созданы за 5 секунд) и длительность 30 секунд
locust -f locustfile.py --host http://192.168.3.100:8444 -u 10 -r 2 -t 30s --headless --csv locustresult запуск без веб-интерфейса с выгрузкой
результатов в csv файлы

```

Запуск Web-интерфейса в контейнере Docker:

```
mkdir locust && cd locust
```

```

FROM alpine:latest
RUN apk add --no-cache python3 py3-pip gcc musl-dev linux-headers python3-dev
RUN python3 -m venv /venv
RUN /venv/bin/pip install --no-cache-dir locust
ENV PATH="/venv/bin:$PATH"
COPY locustfile.py .
EXPOSE 8089
CMD ["locust", "-f", "/locustfile.py"]

```

```
sudo docker build -t locust-alpine-web . && sudo docker run -d --name locust -p 8089:8089 --restart=unless-stopped locust-alpine-web
```

## LLM

## OpenAI

Пример запроса для перевода текста

```

$text = "The OpenAI API uses API keys for authentication. You can create API keys at a user or service account level." # https://platform.openai.com/account/api-keys
$toLang = "Russian"
$presetPrompt = "Translate the following text into $toLang :
$text
Respond ONLY with the translated text. Do not include any other explanations, context, or comments.
"
apiKey = "sk-proj-XXXXXXXXXX"
apiUrl = "https://api.openai.com/v1/chat/completions"

$body = @{
    model = "gpt-4o-mini"
    messages = @(@{
        role = "user";
        content = $presetPrompt
    })
    temperature = 0.7
} | ConvertTo-Json -Depth 10 -Compress

$response = Invoke-RestMethod -Uri $apiUrl -Method Post -Headers @{
    "Content-Type" = "application/json"
    "Authorization" = "Bearer $apiKey"
} -Body $body

$response.choices.message.content

```

## Mock

Создаем серверную заглушку для API OpenAI через [JSON Server](#)

```
npm install -g json-server@0.17.4
```

Конфигурация ответов в файле `openai.json`

```
{
  "completions": {
    "model": "gpt-4o-mini-2024-07-18",
    "choices": [
      {
        "message": {
          "role": "assistant",
          "content": "Response from JSON Server",
        },
        "finish_reason": "stop"
      }
    ]
  }
}
```

Настройка маршрутизации в файле `routes.json`

```
{
  "/v1/chat/completions": "/completions"
}
```

Конфигурация сервера в файле `json-server.json`

```
{
  "port": 3001
}
```

Запускаем сервер:

```
json-server --watch openai.json --routes routes.json
```

Делаем запрос:

```
$(Invoke-RestMethod -Uri "http://localhost:3001/v1/chat/completions").choices.message.content
```

## OpenRouter

Регистрируем аккаунт на [OpenRouter](#) через Google, выпускаем [api ключ](#) и выбираем [бесплатную модель](#).

```
$OPENROUTER_API_KEY = "sk-or-v1-KEY"
$OPENROUTER_MODEL = "deepseek/deepseek-r1:free"
$headers = @{
    "Content-Type" = "application/json"
    "Authorization" = "Bearer $OPENROUTER_API_KEY"
}
$body = @{
    "model" = $OPENROUTER_MODEL
    "messages" = @(
        @{
            "role" = "system"
            "content" = "Your role is a translator. You only translate the text into Russian and do not analyze the answer."
        },
        @{
            "role" = "user"
            "content" = "Hello! I translate the text into English!"
        }
    )
} | ConvertTo-Json -Depth 10 -Compress
$response = Invoke-RestMethod -Uri "https://openrouter.ai/api/v1/chat/completions" -Method Post -Headers $headers -Body $body
$response.choices.message.content
```

## LM Studio

API в [LM Studio](#) совместим с OpenAI

Получить список моделей:

```
$(Invoke-RestMethod -Uri "http://127.0.0.1:1234/v1/models/").data.id
deepseek-r1-distill-llama-8b
llama-3.2-3b-instruct
text-embedding-nomic-embed-text-v1.5
```

Режим чата (когда `stream` установлен в `True`, ответ приходит по частям):

```
curl http://127.0.0.1:1234/v1/chat/completions \
-H "Content-Type: application/json" \
-d '{
    "model": "deepseek-r1-distill-llama-8b",
    "messages": [
        { "role": "system", "content": "Твоя роль переводчик. Ты только переводишь текст на русский язык и не анализируешь ответ." },
        { "role": "user", "content": "Привет! Я перевожу текст на английский язык!" }
    ],
    "temperature": 0.7,
    "max_tokens": -1,
    "stream": false
}'
```

# Ollama

```
cd $home\Downloads
$versionLatest = $(Invoke-RestMethod "https://api.github.com/repos/ollama/ollama/releases/latest").tag_name
irm https://github.com/ollama/ollama/releases/download/${versionLatest}/ollama-windows-amd64.zip -OutFile ollama.zip
Expand-Archive -Path ollama.zip -OutputPath ".\ollama" # -DestinationPath for Windows PowerShell 5.1
Remove-Item ollama.zip; cd ollama

.\ollama serve запускаем сервер
.\ollama pull mistral:7b-instruct загружаем модель (https://ollama.com/library/mistral)
.\ollama run mistral запустить консоль для общения с LLM в режиме чата

# Отправляем API запрос
$data = curl -sS -X POST http://localhost:11434/api/generate -d '{
    "model": "mistral",
    "prompt": "Return only the word test in the answer"
}'

# Собираем ответ из частей response
[string]$($data | ConvertFrom-Json).response.trim()
```

# GigaChat

## Windows

### Developers chat

- Установка сертификатов:

```
Invoke-WebRequest "https://gu-st.ru/content/lending/russian_trusted_root_ca.pem.crt" -OutFile "$home\Downloads\russian_trusted_root_ca.cer"
скачать сертификат минцифры
Invoke-WebRequest "https://gu-st.ru/content/lending/russian_trusted_sub_ca.pem.crt" -OutFile "$home\Downloads\russian_trusted_sub_ca.cer"
Import-Certificate -FilePath "$home\Downloads\russian_trusted_root_ca.cer" -CertStoreLocation "Cert:\CurrentUser\Root" установить сертификат
минцифры
Import-Certificate -FilePath "$home\Downloads\russian_trusted_sub_ca.cer" -CertStoreLocation "Cert:\CurrentUser\CA"
```

- ii. Авторизация по Sber ID и генерация новых авторизационных данных для получения токена: [Developers](#) (время жизни 30 минут)
- iii. Формирование авторизационных данных в формате Base64 из Client ID и Client Secret:

```
$Client_ID      = "7e6d2f9f-825e-49b7-98f4-62fb7506427" # [System.Guid]::Parse("7e6d2f9f-825e-49b7-98f4-62fb7506427")
$client_Secret = "c35113ee-6757-47ba-9853-ea1d0d9db1ef" # [System.Guid]::Parse("c35113ee-6757-47ba-9853-ea1d0d9db1ef")
$client_Join   = $Client_ID+":"+$client_Secret # объединяем два UUID в одну строку, разделяя их символом ':'
$Bytes         = [System.Text.Encoding]::UTF8.GetBytes($Client_Join) # преобразуем строку в массив байт
$cred_Base64  = [Convert]::ToBase64String($Bytes) # кодируем байты в строку Base64
```

- iv. Получение токена:

```
$cred_Base64  = "N2U2ZDJmOWYtODI1ZS00WI3LTk4ZjQtNjJmYmI3NTA2NDI3OmIyYzgwZmZmLTEzOGUtNDg1Mi05MjgwLWE2MGI4NTc0YTM2MQ=="
$UUID = [System.Guid]::NewGuid() генерируем UUID для журналирования входящих вызовов и разбора инцидентов
```

```
$url = "https://ngw.devices.sberbank.ru:9443/api/v2/oauth"
$headers = @{
    "Authorization" = "Basic $cred_Base64"
    "RqUID" = "$UUID"
    "Content-Type" = "application/x-www-form-urlencoded"
}
$body = @{
    scope = "GIGACHAT_API_PERS"
}
$GIGA_TOKEN = $(Invoke-RestMethod -Uri $url -Method POST -Headers $headers -Body $body).access_token
```

- v. Параметры:

```
[string]$content = "Посчитай сумму чисел: 22+33"
[string]$role = "user" # роль автора сообщения (user/assistant/system)
[float]$temperature = 0.7 # температура выборки в диапазоне от 0 до 2. Чем выше значение, тем более случайным будет ответ модели.
[float]$top_p = 0.1 # используется как альтернатива temperature и изменяется в диапазоне от 0 до 1. Задает вероятностную массу токенов, которые д
[int64]$n = 1 # количество вариантов ответов (1..4), которые нужно сгенерировать для каждого входного сообщения
[int64]$max_tokens = 512 # максимальное количество токенов, которые будут использованы для создания ответов
[boolean]$stream = $false # передавать сообщения по частям в потоке
```

- vi. Составление запросов:

```
$url = "https://gigachat.devices.sberbank.ru/api/v1/chat/completions"
$headers = @{
    "Authorization" = "Bearer $GIGA_TOKEN"
    "Content-Type" = "application/json"
}

$(Invoke-RestMethod -Uri "https://gigachat.devices.sberbank.ru/api/v1/models" -Headers $headers).data # список доступных моделей

$body = @{
    model = "GigaChat:latest"
    messages = @(
        @{
            role = $role
            content = $content
        }
    )
    temperature = $temperature
    n = $n
    max_tokens = $max_tokens
    stream = $stream
} | ConvertTo-Json
$request = Invoke-RestMethod -Method POST -Uri $url -Headers $headers -Body $body
$request.choices.message.content
```

## Linux

- Установка сертификатов в Ubuntu:

```
wget https://gu-st.ru/content/lending/russian_trusted_root_ca.pem.crt
wget https://gu-st.ru/content/lending/russian_trusted_sub_ca.pem.crt
mkdir /usr/local/share/ca-certificates/russian_trusted
cp russian_trusted_root_ca.pem.crt russian_trusted_sub_ca.pem.crt /usr/local/share/ca-certificates/russian_trusted
update-ca-certificates -v
wget -qS --spider --max-redirect=0 https://www.sberbank.ru
```

- Получение токена:

```
Cred_Base64="N2U2ZDJmOWYtODI1ZS000WI3LTk4ZjQtNjJmYmI3NTA2NDI3OmIyYzgwZmZmLTEzOGUtNDg1Mi05MjgwLWE2MGI4NTc0YTM2MQ=="
UUID=$(uuidgen)
GIGA_TOKEN=$(curl -s --location --request POST "https://ngw.devices.sberbank.ru:9443/api/v2/oauth" \
--header "Authorization: Basic $Cred_Base64" \
--header "RqUID: $UUID" \
--header "Content-Type: application/x-www-form-urlencoded" \
--data-urlencode 'scope=GIGACHAT_API_PERS' | jq -r .access_token)
```

```
curl -s --location "https://gigachat.devices.sberbank.ru/api/v1/models" --header "Authorization: Bearer $GIGA_TOKEN" | jq . для проверки
```

- Составление запроса:

```

request=$(curl -s https://gigachat.devices.sberbank.ru/api/v1/chat/completions \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $GIGA_TOKEN" \
-d '{
  "model": "GigaChat:latest",
  "messages": [
    {
      "role": "user",
      "content": "Когда уже ИИ захватит этот мир?"
    }
  ],
  "temperature": 0.7
}')
echo $request | jq -r .choices[].message.content

```

## YandexGPT

- Получить OAuth-Token:

[Create AIM Token](#) время жизни IAM-токена не больше 12 часов

yandexPassportOauthToken="y0\_AgAAAAAGaLFLAATuwQAAAAD3xtRLQE4hvIazQ5euK043XXXXXXXXXX" для bash

\$yandexPassportOauthToken = "y0\_AgAAAAAGaLFLAATuwQAAAAD3xtRLQE4hvIazQ5euK043XXXXXXXXXX" для PowerShell

- Обменять OAuth-Token на IAM-Token:

```
IAM_TOKEN=$(curl -s -d "{\"yandexPassportOauthToken\": \"$yandexPassportOauthToken\"}" "https://iam.api.cloud.yandex.net/iam/v1/tokens" | jq -r .iam_token)
$IAM_TOKEN = $(Invoke-RestMethod -Method POST -Uri "https://iam.api.cloud.yandex.net/iam/v1/tokens" -Body $($yandexPassportOauthToken = "$yandexPassportOauthToken"))
```

- Получить FOLDER\_ID:

```
CLOUD_ID=$(curl -s -H "Authorization: Bearer $IAM_TOKEN" https://resource-manager.api.cloud.yandex.net/resource-manager/v1/clouds | jq -r .clouds[])
curl -s --request GET -H "Authorization: Bearer $IAM_TOKEN" https://resource-manager.api.cloud.yandex.net/resource-manager/v1/folders -d "{\"cloud_id\": \"$CLOUD_ID\"}")
curl -s --request POST -H "Authorization: Bearer $IAM_TOKEN" https://resource-manager.api.cloud.yandex.net/resource-manager/v1/folders -d "{\"cloud_id\": \"$CLOUD_ID\", \"name\": \"my_folder\"}")
FOLDER_ID=$(curl -s --request GET -H "Authorization: Bearer $IAM_TOKEN" https://resource-manager.api.cloud.yandex.net/resource-manager/v1/folders | jq -r .folders[] | jq -r .id)
```

```
$CLOUD_ID = $(Invoke-RestMethod -Method Get -Uri "https://resource-manager.api.cloud.yandex.net/resource-manager/v1/clouds" -Headers @{"Authorization" = "Bearer $yandexPassportOauthToken"})
$FOLDER_ID = $(Invoke-RestMethod -Method Get -Uri "https://resource-manager.api.cloud.yandex.net/resource-manager/v1/folders" -Headers @{"Authorization" = "Bearer $yandexPassportOauthToken"})
```

- Составление запроса:

```

model="gpt://$FOLDER_ID/yandexgpt/latest" # https://cloud.yandex.ru/ru/docs/yandexgpt/concepts/models
body=$(cat <<EOF
{
  "modelUri": "$model",
  "completionOptions": {
    "stream": false,
    "temperature": 0.6,
    "maxTokens": 2000
  },
  "messages": [
    {
      "role": "user",
      "text": "Посчитай сумму 22+33"
    }
  ]
}
EOF)
curl --request POST \
-H "Content-Type: application/json" \
-H "Authorization: Bearer $IAM_TOKEN" \
-H "x-folder-id: $FOLDER_ID" \
-d "$body" \
"https://llm.api.cloud.yandex.net/foundationModels/v1/completion"

$model = "gpt://$FOLDER_ID/yandexgpt/latest"
$body = @"
{
  "modelUri": "$model",
  "completionOptions": {
    "stream": false,
    "temperature": 0.6,
    "maxTokens": 2000
  },
  "messages": [
    {
      "role": "user",
      "text": "Посчитай сумму 22+33"
    }
  ]
}
@"
Invoke-RestMethod -Method POST -Uri "https://llm.api.cloud.yandex.net/foundationModels/v1/completion" -Headers @{
  "Content-Type" = "application/json"
}

```

## SuperAGI

[Source](#)

[Playground generate](#)

[API Doc \(examples\)](#)

```

$SUPERAGI_API_KEY="31f72164129XXXXX"
prompt="посчитай сумму 22+33, дай только ответ без лишнего текста"
request=$(curl -s -X POST 'https://api.superagi.com/v1/generate/65437cbf227a4018516ad1ce' \
-H 'Content-Type: application/json' \
-H "Authorization: Bearer $SUPERAGI_API_KEY" \
-d '{
    "prompt": ["$prompt"],
    "max_tokens": 500,
    "temperature": 0.9,
    "top_p": 0.15,
    "repetition_penalty": 0,
    "best_of": 1.05,
    "top_k": 50,
    "stream": false
}')
echo $request | sed "s/data: //\" | jq -r .choices[].text

$SUPERAGI_API_KEY = "31f72164129XXXXX"
$prompt = "посчитай сумму 22+33, дай только ответ без лишнего текста"
$request = Invoke-RestMethod -Method Post -Uri 'https://api.superagi.com/v1/generate/65437cbf227a4018516ad1ce' -Headers @{
    'Content-Type' = 'application/json'
    'Authorization' = "Bearer $SUPERAGI_API_KEY"
} -Body (@{
    prompt = $($prompt)
    max_tokens = 500
    temperature = 0.9
    top_p = 0.15
    repetition_penalty = 0
    best_of = 1.05
    top_k = 50
    stream = $false
} | ConvertTo-Json)
$request -replace "^data: " | ConvertFrom-Json).choices.text

```

## Replicate

### API curl examples

```

REPLICATE_API_TOKEN="r8_STyeUNXiGonkLfxE1FSKaql1261XXXXXXXXXX"
prompt="Жираф в полоску зебры"
request=$(curl -s -X POST \
-H "Authorization: Token $REPLICATE_API_TOKEN" \
-H "Content-Type: application/json" \
-d '${
    "version": "ac732df83cea7fff18b8472768c88ad041fa750ff7682a21affe81863cbe77e4",
    "input": {
        "prompt": "$prompt"
    }
}' \
https://api.replicate.com/v1/predictions)
request_url=$(echo $request | jq -r .urls.get)
response_status=$(curl -s -H "Authorization: Token $REPLICATE_API_TOKEN" $request_url | jq -r .status)
while [[ $response_status != succeeded ]]; do
    response_status=$(curl -s -H "Authorization: Token $REPLICATE_API_TOKEN" $request_url | jq -r .status)
done
curl -s -H "Authorization: Token $REPLICATE_API_TOKEN" $request_url | jq -r .output[]

```

```

$REPLICATE_API_TOKEN = "r8_STyeUNXiGonkLfxE1FSKaql1261XXXXXXXXXX"
$prompt = "Жираф в полоску зебры"
$body = @{
    version = "ac732df83cea7fff18b8472768c88ad041fa750ff7682a21affe81863cbe77e4"
    input = @{
        prompt = $prompt
    }
} | ConvertTo-Json
$headers = @{
    "Authorization" = "Token $REPLICATE_API_TOKEN"
    "Content-Type" = "application/json"
}
$request = Invoke-RestMethod -Uri "https://api.replicate.com/v1/predictions" -Method POST -Body $body -Headers $headers
$response = Invoke-RestMethod $($request.urls.get) -Headers @{"Authorization" = "Token $REPLICATE_API_TOKEN"}
while ($response.status -ne "succeeded") {
    $response = Invoke-RestMethod $($request.urls.get) -Headers @{"Authorization" = "Token $REPLICATE_API_TOKEN"}
}
$response.output

```

## Google API

### Google Translate

```

$key = "<TOKEN_API>" # получить токен: https://console.cloud.google.com/apis/credentials
$text = "You can see in the right corner how long each translation request takes (this does not depend on the amount of text being transferred)."
$languageTarget = "RU"
$languageSource = "EN"
$url = "https://translation.googleapis.com/language/translate/v2?key=$key"
$header = @{
    "Content-Type" = "application/json"
}
$body = @{
    "q" = "$text"
    "target" = "$languageTarget"
    "source" = "$languageSource"
} | ConvertTo-Json
$webClient = New-Object System.Net.WebClient
foreach ($key in $header.Keys) {
    $webClient.Headers.Add($key, $header[$key])
}
$response = $webClient.UploadString($url, "POST", $body) | ConvertFrom-Json
$response.data.translations.translatedText

```

### Google Search

```

$key = "<TOKEN_API>" # получить токен: https://developers.google.com/custom-search/v1/overview?hl=ru (пользовательский поиск JSON API предоставляет)
$cx = "35c78340f49eb474a" # создать поисковую систему https://programmablesearchengine.google.com/controlpanel/all
$query = "как создать бота discord"
$lang = "ru"
$num = 10
$start = 0
$response = Invoke-RestMethod "https://www.googleapis.com/customsearch/v1?q=$query&key=$key&cx=$cx&lr=$lang_$lang&num=$num&start=$start"
$response.items | Select-Object title,snippet,displayLink,link | Format-List

```

### Google Search via RapidAPI

[Google-Search72](#)

```

$key = "<TOKEN_API>"
$headers=@{}
$headers.Add("X-RapidAPI-Key", "$key")
$headers.Add("X-RapidAPI-Host", "google-search72.p.rapidapi.com")
$query = "как создать бота discord"
$response = Invoke-RestMethod "https://google-search72.p.rapidapi.com/search?q=$query%20gitgub&gl=us&lr=lang_ru&num=20&start=0" -Method GET -Head
$response.items | Select-Object title,snippet,displayLink,link | Format-List

```

## Google Filter

<https://www.google.com/search?q=the+rookie+2018+imdb> формат url-запроса поиска с пробелами  
<https://www.google.com/search?q=the+rookie+2018+site:imdb.com> поиск по сайту  
[https://www.google.com/search?q=the+rookie+intitle:index.of+"last modified"+\(mkv|avi\)](https://www.google.com/search?q=the+rookie+intitle:index.of+"last modified"+(mkv|avi)) искать страницы, на которых указано "last modified" (последние изменения), заголовок страницы через расширенный оператор поиска (все перечисленные слова должны встречаться в заголовке) содержит слово "index.of" (указывает на директорию на веб-сервере, которая содержит список файлов) и искать файлы с расширениями .mkv или () .avi  
<https://www.google.com/search?q=the+rookie+2018+filetype:torrent>  
инструкция gopro hero 11 filetype:pdf искать сразу документ (на странице .pdf или загрузка)  
"действия/глаголы, утвержденные для использования в командах" искать по фразе целиком, без разбиения на отдельные слова  
"ягуар скорость -животное -xe -xj" узнаем скорость Ягуара, исключаем животное и модели автомобиля  
"intitle:лучшие фильмы 2023" запрос ищет страницы, заголовки (title HTML документа) которых содержат слова "лучшие", "фильмы" и "2023" (все слова должны быть в заголовке)  
"allintitle:лучшие фильмы 2023" запрос ищет страницы, заголовки (title HTML документа) которых содержат слова "лучшие", "фильмы" или "2023" (одно из)  
"intext:telegram бот powershell" поиск страниц, содержащих указанное ключевое слово в тексте страницы (а не только в заголовке)  
"inurl:lifailon" поиск страниц, в URL которых содержится указанное ключевое слово  
intitle:index.of "game of thrones" mkv daterange:2010..2015 фильтрация по дате изменения, оператор позволяет задать диапазон дат в формате YYYYMMDD..YYYYMMDD  
intitle:index.of "game of thrones" mkv after:2015 ограничить результаты поиска файлов, измененных до (before) или после (after) указанной даты  
intitle:index.of "game of thrones" mkv from:2010 to:2015 фильтрация по диапазону дат  
<https://www.google.com/search?q=the-rookie-2018+site:imdb.com&btnI> редирект на первый url

## Media API

### IMDb

#### IMDb8

```

$key = "<TOKEN_API>" # 500 запросов в месяц
$query="Break"
$headers=@{}
$headers.Add("X-RapidAPI-Key", "$key")
$headers.Add("X-RapidAPI-Host", "imdb8.p.rapidapi.com")
$response = Invoke-RestMethod "https://imdb8.p.rapidapi.com/title/find?q=$query" -Method GET -Headers $headers
$response.results | select title,titleType,year,runningTimeInMinutes,id | Format-Table
"https://www.imdb.com \$\(\$response.results.id\[0\]\)"
$response.results.principals # актеры
$response.results.image

```

## MoviesDatabase

#### MoviesDatabase

```

$key = "<TOKEN_API>"
$imdb_id = "tt0455275"
$headers=@{}
$headers.Add("X-RapidAPI-Key", "$key")
$headers.Add("X-RapidAPI-Host", "moviesdatabase.p.rapidapi.com")
$response = Invoke-RestMethod "https://moviesdatabase.p.rapidapi.com/titles/$imdb_id" -Method GET -Headers $headers
$response.results

```

## TMDB

### Developer TMDB

```

$TOKEN = "548e444e7812575caa0a7eXXXXXXXXXX"
$Endpoint = "search/tv" # поиск сериала (tv) и фильма (movie) по названию
$Query = "зимородок"
$url = $($("https://api.themoviedb.org/3/$Endpoint"+"?api_key=$TOKEN&query=$Query"))
$(Invoke-RestMethod -Uri $url -Method Get).results
$id = $(Invoke-RestMethod -Uri $url -Method Get).results.id # забрать id сериала (210865) https://www.themoviedb.org/tv/210865

$Endpoint = "tv/$id" # получение информации о сериале по его ID
$url = $($("https://api.themoviedb.org/3/$Endpoint"+"?api_key=$TOKEN"))
$(Invoke-RestMethod -Uri $url -Method Get) # список сезонов (.seasons), количество эпизодов (.seasons.episode_count)

(Invoke-RestMethod -Uri "https://api.themoviedb.org/3/tv/$id/season/2?api_key=$Token" -Method Get).episodes # вывести 2 сезон
Invoke-RestMethod -Uri "https://api.themoviedb.org/3/tv/$id/season/2/episode/8?api_key=$Token" -Method Get # вывести 8 эпизод

```

## OMDb

### Получение API ключа по email

```

$API_KEY = "XXXXXXX"
$IMDb_ID = "tt7587890"
curl -s "https://omdbapi.com/?apikey=$($API_KEY)&i=$($IMDb_ID)" | jq .
curl -s "https://omdbapi.com/?apikey=$($API_KEY)&i=$($IMDb_ID)" | ConvertFrom-Json
Invoke-RestMethod "https://omdbapi.com/?apikey=$($API_KEY)&s=The Rookie"
Invoke-RestMethod "https://omdbapi.com/?apikey=$($API_KEY)&t=The Rookie" ПОИСК по Title
Invoke-RestMethod "https://omdbapi.com/?apikey=$($API_KEY)&t=The Rookie&y=1990" поиск по Title и году выхода
Invoke-RestMethod "https://omdbapi.com/?apikey=$($API_KEY)&t=The Rookie&type=movie" поиск только фильма (movie) или сериала (series)
$(Invoke-RestMethod "https://omdbapi.com/?apikey=$($API_KEY)&s=The Rookie").Search поиск всех совпадений (фильмы и сериалы)

```

## ivi

### ivi api doc

```

Invoke-RestMethod https://api.ivi.ru/mobileapi/categories список категорий и жанров (genres/meta_genres)
Invoke-RestMethod https://api.ivi.ru/mobileapi/collections подборки

(Invoke-RestMethod "https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok").result.seasons.number КОЛ-ВО сезонов
(Invoke-RestMethod "https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok").result.seasons[1].episode_count КОЛ-ВО серий во втором сезоне
(Invoke-RestMethod "https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok").result.seasons[1].ivi_release_info.date_interval_min дата выхода следующей серии
(Invoke-RestMethod "https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok").result.kp_rating рейтинг в Кинопоиск (8.04)

$id = (Invoke-RestMethod "https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok").result.kp_id получить id в Кинопоиск (5106881)
$id=$(curl -s https://api.ivi.ru/mobileapi/search/v7/?query=zimorodok | jq .result[].kp_id) получить id в Кинопоиск

```

# Kinopoisk

```
id=5106881
get=$(curl -s https://www.kinopoisk.ru/film/$id/episodes/)
printf "%s\n" "${get[@]}" | grep -A 1 "Сезон 2" | grep "эпизодов" | sed -r "s/^.*\: //" # количество эпизодов во втором сезоне
```

## kinopoisk.dev

[Получить токен](#)

[Документация по API в формате OpenAPI](#)

GET /v1.4/movie/{id} поиск по id

```
$id = 5106881
$API_KEY = "ZYMNJJJA-0J8MNPN-PB4N7R7-XXXXXX"

$header = @{
    "accept" = "application/json"
    "X-API-KEY" = "$API_KEY"
}
$irm = Invoke-RestMethod "https://api.kinopoisk.dev/v1.4/movie/$id" -Method GET -Headers $header
$irm.rating.kp # рейтинг в Кинопоиск (8,079)
$irm.seasonsInfo # количество сезонов и эпизодов в них
```

```
id=5106881
API_KEY="ZYMNJJJA-0J8MNPN-PB4N7R7-XXXXXX"
get=$(curl -s -X GET \
"https://api.kinopoisk.dev/v1.4/movie/$id" \
-H "accept: application/json" \
-H "X-API-KEY: $API_KEY")
echo $get | jq .rating.kp # рейтинг в Кинопоиск (8,079)
echo $get | jq .seasonsInfo[1].episodesCount # количество эпизодов во втором [1] сезоне (6)
```

GET /v1.4/movie/search

```
query="zimorodok"
page=1 # кол-во страниц для выборки
limit=1 # кол-во элементов на странице
curl -s -X GET \
"https://api.kinopoisk.dev/v1.4/movie/search?page=$page&limit=$limit&query=$query" \
-H "accept: application/json" \
-H "X-API-KEY: $API_KEY" | jq .

limit=5
request=$(curl -s -X GET \
"https://api.kinopoisk.dev/v1.4/movie/search?page=$page&limit=$limit&query=%D0%B7%D0%B8%D0%BC%D0%BE%D1%80%D0%BE%D0%B4%D0%BE%D0%BA" \
-H "accept: application/json" \
-H "X-API-KEY: $API_KEY" | jq .)
echo $request | jq '.docs[] | select(.year == 2022)' # отфильтровать вывод по году выхода
```

```
$API_KEY = "ZYMNJJJA-0J8MNPN-PB4N7R7-XXXXXX"
$page = 1
$limit = 5
$query = "%D0%B7%D0%B8%D0%BC%D0%BE%D1%80%D0%BE%D0%B4%D0%BE%D0%BA"
$request = Invoke-RestMethod -Uri "https://api.kinopoisk.dev/v1.4/movie/search?page=$page&limit=$limit&query=$query" -Headers @{"accept"="appli
$request.docs | Where-Object year -eq 2022
```

## UrlCode

```
function Get-PercentEncode ($str) {
    $bytes = [System.Text.Encoding]::UTF8.GetBytes($str)
    ($bytes | ForEach-Object { "{0:X2}" -f $_ }) -join '%' -replace '^', "%"
}
Get-PercentEncode "зимородок"

function Get-UrlEncode($str) {
    [System.Web.HttpUtility]::UrlEncode($str)
}
UrlEncode "зимородок"

percent-encode() {
    str=$1
    echo -n "$1" | iconv -t utf8 | od -An -tx1 | tr ' ' % | tr -d '\n'
}
percent-encode "зимородок"

function Get-UrlDecode($encoded) {
    [System.Uri]::UnescapeDataString($encoded)
}
Get-UrlDecode "%D0%B7%D0%B8%D0%BC%D0%BE%D1%80%D0%BE%D0%B4%D0%BE%D0%BA"

percent-decode() {
    encoded=$1
    local url_encoded="${1//+/ }"
    printf '%b' "${url_encoded//%/\\x}"
}
percent-decode "%D0%B7%D0%B8%D0%BC%D0%BE%D1%80%D0%BE%D0%B4%D0%BE%D0%BA"
```

## KinopoiskApiUnofficial

Бесплатно 500 запросов в сутки. [Swagger documentation](#)

```
API_KEY="828ec96a-f45d-4e3d-84b1-XXXXXXXXXXXX"
$headers = @{
    "accept" = "application/json"
    "X-API-KEY" = "$API_KEY"
}
Invoke-RestMethod -Uri 'https://kinopoiskapiunofficial.tech/api/v2.2/films/1142153' -Headers $headers

curl -s "https://kinopoiskapiunofficial.tech/api/v2.2/films/1142153" -H "accept: application/json" -H "X-API-KEY: $API_KEY" | jq .
```

## Kinobox

```
$url = "https://www.kinopoisk.ru/film/694051"
$kp_id = $url -replace ".+/"
https://kinomix.web.app/#694051
curl -s -X GET "https://kinobox.tv/api/players/main?kinopoisk=$kp_id" -H "accept: application/json" ПОИСК по id Кинопоиск
curl -s -X GET "https://kinobox.tv/api/players/main?imdb=tt2293640" -H "accept: application/json" ПОИСК по id IMDb
curl -s -X GET "https://kinobox.tv/api/players/main?title=minions" -H "accept: application/json" ПОИСК основных плееров по названию
curl -s -X GET "https://kinobox.tv/api/players/all?title=minions" -H "accept: application/json" ПОИСК всех плееров
curl -s -X GET "https://kinobox.tv/api/popular/films" -H "accept: application/json" ПОПУЛЯРНЫЕ фильмы
curl -s -X GET "https://kinobox.tv/api/popular/series" -H "accept: application/json" ПОПУЛЯРНЫЕ сериалы
```

# VideoCDN

[API](#)

[Source](#)

[API JSON](#)

```
$kp_id = 5106881
$token = "YfTWH2p3Mai7ziqDoGjS3yXXXXXXXXXX"
$sep = "tv-series"
$(Invoke-RestMethod $("https://videocdn.tv/api/$ep"+"?api_token=$token&field=kinopoisk_id&query=$kp_id")).data.episodes | Where-Object season_num
```

```
kp_id=5106881
token="YfTWH2p3Mai7ziqDoGjS3yXXXXXXXXXX"
ep="tv-series"
curl -s "https://videocdn.tv/api/$ep?api_token=$token&field=kinopoisk_id&query=$kp_id" | jq ".data[].episodes | length" # количество серий
curl -s "https://videocdn.tv/api/$ep?api_token=$token&field=kinopoisk_id&query=$kp_id" | jq ".data[].episodes[] | select(.season_num == 2) | {epi
```

# Torrent

## Jackett

[Source](#)

```
mkdir /jackett
docker-compose.yml
```

```
---
services:
  jackett:
    image: lsqr.io/linuxserver/jackett:latest
    container_name: jackett
    environment:
      - PUID=1000
      - PGID=1000
      - TZ=Etc/UTC
    volumes:
      - /jackett/data:/config
      - /jackett/blackhole:/downloads
    ports:
      - 9117:9117
    restart: unless-stopped
```

```
docker-compose up -d jackett
docker exec -it jackett /bin/bash доступ к оболочке во время работы контейнера
docker logs -f jackett Мониторинг журналов контейнера
```

```
/jackett/data/Jackett/ServerConfig.json место хранения конфигурации сервера
/jackett/data/Jackett/Indexers/*.json место хранения конфигурации индексаторов
```

```
$API_KEY = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/rutor/results/torznab/api?apikey=$API_KEY" Прочитать RSS ленту RuTor
$query = "the+rookie"
Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/rutor/results/torznab/api?apikey=$API_KEY&t=search&cat=&q=$query" Поиск в RuTor
Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/kinozal/results/torznab/api?apikey=$API_KEY&t=search&q=$query" ПОИСК В КИНОЗАЛ
Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/kinozal/results/torznab/api?apikey=$API_KEY&t=search&q=$query&cat=5000"
отфильтровать вывод по сериалам (Capabilities: 5000)
Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/all/results/torznab/api?apikey=$API_KEY&t=search&q=riverdale" ПОИСК ВО ВСЕХ
индексаторах
```

```
$(Invoke-RestMethod "http://127.0.0.1:9117/api/v2.0/indexers/all/results/torznab/api?apikey=$API_KEY&t=indexers&configured=true").indexers.indexer
список всех настроенных индексаторов (трекеров)
```

## Torrent-API-py

[Source](#)

[Documentation](#)

```
git clone https://github.com/Ryuk-me/Torrent-Api-py
cd Torrent-Api-py
pip install virtualenv
py -3 -m venv api-py
# Активировать виртуальную среду для Windows
.\api-py\Scripts\activate
# Активировать виртуальную среду для Linux
# $ source api-py/bin/activate
# Установить зависимости и запустить
pip install -r requirements.txt
python main.py
# Proxy: https://github.com/dperson/torproxy
# export HTTP_PROXY="http://proxy-host:proxy-port"

$srv = "http://localhost:8009" local
$srv = "https://torrent-api-py-nx0x.onrender.com" public
Invoke-RestMethod $srv/api/v1/sites список доступных трекеров
Invoke-RestMethod "$srv/api/v1/search?site=torlock&query=the+rookie&limit=0&page=1" ПОИСК В выбранном трекере
Invoke-RestMethod "$srv/api/v1/all/search?query=the+rookie&limit=0" ПОИСК ПО НАЗВАНИЮ ВО ВСЕХ ТРЕКЕРАХ
```

## Plex

```
$API_TOKEN = "XXXXXXXXXXXXXXXXXXXXXX"
```

```
$headers = @{
    "X-Plex-Token" = $API_TOKEN
    "accept" = "application/json"
}

$(Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/servers).MediaContainer.Server версия сервера
Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/diagnostics/logs -OutFile log.zip выгрузить лог с сервера
$(Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/library/sections).MediaContainer.Directory список секций добавленных на сервер
$section_key = $(Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/library/sections).MediaContainer.Directory.key[0]
Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/library/sections/$section_key/refresh Синхронизация указанной секции в Plex по ключу
$(Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/library/sections/2/folder).MediaContainer.Metadata получить список директорий и файлов в корне выбранной секции
$(Invoke-RestMethod -Headers $headers -Uri http://localhost:32400/library/sections/2/folder?parent=204).MediaContainer.Metadata получить список всех файлов в указанной директории через ключ (MediaContainer.Metadata.key) конечной точки
```

## Jellyfin

[Source](#)

[API Docs](#)

```
$API_TOKEN "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} http://localhost:8096/Users список пользователей и их id
$Users = Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} http://localhost:8096/Users
$UserId = $($Users | Where-Object Name -match "Lifailon").Id забрать id пользователя
Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} http://localhost:8096/System/Info информация о системе
$(Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} http://localhost:8096/Items).Items список добавленных объектов директорий
```

```

$ItemId = $($Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} http://localhost:8096/Items).Items[-1].Id забрать id директории
$data = $($Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} "http://localhost:8096/Users/$UserId/Items?ParentId=$ItemId").Items получить
содержимое корневой директории по Id из Items
$TvId = $($data | Where-Object Name -match "Rookie").Id найти сериал или фильм по имени и забрать его Id
$(Invoke-RestMethod -Headers @{"X-Emby-Token" = $API_TOKEN} "http://localhost:8096/Users/$UserId/Items?ParentId=$TvId").Items получить
содержимое дочерней директории по Id ее родительской директории

```

## Telegram

@BotFather (<https://t.me/BotFather>) /newbot

Format: <https://api.telegram.org/bot<token>/<endpoint>>

[getupdates](#)

```

function Get-FromTelegram {
    param (
        $Token = "687...:AAF...",
        [switch]$Date,
        [switch]$Last,
        [switch]$ChatID
    )
    $endpoint = "getUpdates"
    $url      = "https://api.telegram.org/bot$Token/$endpoint"
    $result   = Invoke-RestMethod -Uri $url
    if ($Date) {
        $Collections = New-Object System.Collections.Generic.List[System.Object]
        foreach ($r in $($result.result)) {
            $EpochTime = [DateTime]"1/1/1970"
            $TimeZone = Get-TimeZone
            $UTCTime = $EpochTime.AddSeconds($r.message.date)
            $d = $UTCTime.AddMinutes($TimeZone.BaseUtcOffset.TotalMinutes)
            $Collections.Add([PSCustomObject]@{
                Message = $r.message.text;
                Date    = $d
            })
        }
        $Collections
    }
    else {
        if ($Last) {
            $result.result.message.text[-1]
        }
        elseif ($ChatID) {
            $Collections = New-Object System.Collections.Generic.List[System.Object]
            foreach ($r in $($result.result)) {
                $Collections.Add([PSCustomObject]@{
                    Message = $r.message.text;
                    UserName = $r.message.chat.username;
                    ChatID = $r.message.chat.id;
                    ChatType = $r.message.chat.type
                })
            }
            $Collections
        }
        else {
            $result.result.message.text
        }
    }
}

Get-FromTelegram
Get-FromTelegram -Last

```

```
Get-FromTelegram -Date  
Get-FromTelegram -ChatID
```

## sendmessage

```
function Send-ToTelegram {  
    param (  
        [Parameter(Mandatory = $True)]$Text,  
        $Token      = "687...:AAF...",  
        $Chat       = "125468108",  
        $Keyboard  
)  
    $endpoint = "sendMessage"  
    $url      = "https://api.telegram.org/bot$Token/$endpoint"  
    $Body     = @{  
        chat_id = $Chat  
        text    = $Text  
    }  
    if ($keyboard -ne $null) {  
        $Body += @{reply_markup = $keyboard}  
    }  
    Invoke-RestMethod -Uri $url -Body $Body  
}
```

```
Send-ToTelegram -Text "Send test from powershell"
```

```
$LastDate = (Get-FromTelegram -date)[-1].Date  
while ($true) {  
    $LastMessage = (Get-FromTelegram -date)[-1]  
    Start-Sleep 1  
    $LastDateTest = $LastMessage.Date  
    if (($LastMessage.Message -match "/Service") -and ($LastDate -ne $LastDateTest)) {  
        $ServiceName = $($LastMessage.Message -split " ")[-1]  
        $Result = $(Get-Service $ServiceName -ErrorAction Ignore).Status  
        if ($Result) {  
            Send-ToTelegram -Text $Result  
        } else {  
            Send-ToTelegram -Text "Service not found"  
        }  
        $LastDate = $LastDateTest  
    }  
}
```

```
/Service vpnagent  
/Service WinRM  
/Service test
```

## Button

```
$keyboard = '{  
    "inline_keyboard": [[  
        {"text": "Uptime", "callback_data": "/Uptime"},  
        {"text": "Test", "callback_data": "/Test"}  
    ]]  
}'  
Send-ToTelegram -Text "Test buttons" -Keyboard $keyboard  
$request = (Invoke-RestMethod -Uri "https://api.telegram.org/bot$Token/getUpdates").result.callback_query  
$request.data # прочитать callback_data нажатой кнопки  
$request.message.date
```

## Send-ToTelegramFile

<https://core.telegram.org/bots/api#senddocument>

```
function Send-ToTelegramFile {
    param (
        [Parameter(Mandatory = $true)][string]$Path,
        [Parameter(Mandatory = $true)][string]$Token,
        [Parameter(Mandatory = $true)][string]$Chat,
        $Keyboard
    )
    $endpoint = "senddocument"
    $url      = "https://api.telegram.org/bot$Token/$endpoint"
    $multipartContent = [System.Net.Http.MultipartFormDataContent]::new()
    $fileStream = [System.IO.FileStream]::new($Path, [System.IO.FileMode]::Open)
    $fileContent = [System.Net.Http.StreamContent]::new($fileStream)
    $fileHeader = [System.Net.Http.Headers.ContentDispositionHeaderValue]::new("form-data")
    $fileHeader.Name = "document"
    $fileHeader.FileName = [System.IO.Path]::GetFileName($Path)
    $fileContent.Headers.ContentDisposition = $fileHeader
    $multipartContent.Add($fileContent, "document")
    if ($Keyboard) {
        $keyboardContent = [System.Net.Http.StringContent]::new($Keyboard)
        $keyboardContent.Headers.ContentType.MediaType = "application/json"
        $multipartContent.Add($keyboardContent, "reply_markup")
    }
    $chatContent = [System.Net.Http.StringContent]::new($Chat)
    $multipartContent.Add($chatContent, "chat_id")
    $response = Invoke-RestMethod -Uri $url -Method Post -Body $multipartContent -ContentType "multipart/form-data"
    $fileStream.Dispose()
    return $response
}
```

```
Send-ToTelegramFile -Path "C:\Users\Lifailon\Documents\lake.jpg" -Token "7777777777:AAF..." -Chat "777777777"
```

## Discord

### Developers

Создаем Applications (General Information). В Bot привязываем к Application и копируем токен авторизации. В OAuth2 - URL Generator выбираем bot и права Administrator и копируем созданный URL для добавления на канал. Переходим по url и добавляем бота на сервер. Получаем ID канала на сервере (текстовые каналы, правой кнопкой мыши копируем ссылку и забираем последний id в url).

## Send to Discord

```
DISCORD_TOKEN="MTE5NzE1NjM0NTM3NjQxMTcyQQ.XXXXXXX.EzBF6RA9Kx_MSuhLW5e1H1U-XXXXXXXXXXXXXX"
DISCORD_CHANNEL_ID="119403124XXXXXXXXX"
TEXT="test from bash"
URL="https://discordapp.com/api/channels/$DISCORD_CHANNEL_ID/messages"
curl -s -X POST $URL \
-H "Authorization: Bot $DISCORD_TOKEN" \
-H "Content-Type: application/json" \
-d "{\"content\": \"$TEXT\"}"
```

```
$DISCORD_TOKEN = "MTE5NzE1NjM0NTM3NjQxMTcyQQ.XXXXXXX.EzBF6RA9Kx_MSuhLw5e1H1U-XXXXXXXXXXXXXX"
$DISCORD_CHANNEL_ID = "119403124XXXXXXXXXX"
$TEXT = "test from PowerShell"
$URL = "https://discordapp.com/api/channels/$DISCORD_CHANNEL_ID/messages"
$Body = @{
    content = $TEXT
} | ConvertTo-Json
curl -s $URL -X POST -H "Authorization: Bot $DISCORD_TOKEN" -H "Content-Type: application/json" -d $Body
```

## Read from Discord

```
curl -s -X GET $URL \
-H "Authorization: Bot $DISCORD_TOKEN" \
-H "Content-Type: application/json" | jq -r .[0].content

$messages = (curl -s -X GET $URL -H "Authorization: Bot $DISCORD_TOKEN" -H "Content-Type: application/json" | ConvertFrom-Json)
$messages | Select-Object content,timestamp,{$_author.username}
```

## HttpClient

```
$DISCORD_TOKEN = "MTE5NzE1NjM0NTM3NjQxMTcyQQ.XXXXXXX.EzBF6RA9Kx_MSuhLw5e1H1U-XXXXXXXXXXXXXX"
$DISCORD_CHANNEL_ID = "119403124XXXXXXXXXX"
$URL = "https://discordapp.com/api/channels/$DISCORD_CHANNEL_ID/messages"
$HttpClient = New-Object System.Net.Http.HttpClient
$HttpClient.DefaultRequestHeaders.Authorization = "Bot $DISCORD_TOKEN"
$response = $HttpClient.GetAsync($URL).Result
$messages = $response.Content.ReadAsStringAsync().Result
($messages | ConvertFrom-Json).content
```

## Button

```
curl -X POST $URL \
-H "Content-Type: application/json" \
-H "Authorization: Bot $DISCORD_TOKEN" \
-d '{
    "content": "Test text for button",
    "components": [
        {
            "type": 1,
            "components": [
                {
                    "type": 2,
                    "label": "Button",
                    "style": 1,
                    "custom_id": "button_click"
                }
            ]
        }
    ]
}'
```

## Discord.Net.Webhook

```
Add-Type -Path $(ls "$home\Documents\Discord.NET\*.dll").FullName
# https://discordapp.com/api/webhooks/<webhook_id>/<webhook_token> (Настроить канал - Интеграция)
$webhookId = 119757728000000000
$webhookToken = "rs8AA-XXXXXXXXXX_Vk5RUI4A6HuSGhpCCTepq25duwCwLXasfv6u23a7XXXXXXXXX"
$messageContent = "Test dotNET"
$client = New-Object Discord.Webhook.DiscordWebhookClient($webhookId, $webhookToken)
$client.SendMessageAsync($messageContent).Wait()
```

## Discord.Net.WebSocket

```
$DiscordAssemblies = $(ls "$home\Documents\Discord.NET\*.dll").FullName
foreach ($assembly in $DiscordAssemblies) {
    Add-Type -Path $assembly
}
$DISCORD_TOKEN = "MTE5NzE1NjM0NTM3NjQxMTcyOQ.XXXXXX.EzBF6RA9Kx_MSuhLW5e1H1U-XXXXXXXXXXXX"
$Client = New-Object Discord.WebSocket.DiscordSocketClient
$Client.Add_MessageReceived({
    param($message)
    if ($message.Author.Id -ne $Client.CurrentUser.Id) {
        Write-Host ("Received message from " + $message.Author.Username + ": " + $message.Content)
        if ($message.Content.Contains("ping")) {
            $message.Channel.SendMessageAsync("pong").GetAwaiter().GetResult()
        }
    }
})
$Client.LoginAsync([Discord.TokenType]::Bot, $DISCORD_TOKEN).GetAwaiter().GetResult()
#$Client.StartAsync().Wait()
$Client.StartAsync().GetAwaiter().GetResult()
$Client.ConnectionState

[console]::ReadKey($true)
$Client.LogoutAsync().GetAwaiter().GetResult()
$Client.Dispose()
```

## oh-my-posh

### Install

```
winget install JanDeDobbeleer.OhMyPosh -s winget
choco install oh-my-posh -y
scoop install https://github.com/JanDeDobbeleer/oh-my-posh/releases/latest/download/oh-my-posh.json
Set-ExecutionPolicy Bypass -Scope Process -Force; Invoke-Expression ((New-Object System.Net.WebClient).DownloadString('https://ohmyposh.dev/install'))
```

### Themes

```
Get-PoshThemes отобразить список всех тем
oh-my-posh init pwsh --config "$env:POSH_THEMES_PATH/di4am0nd.omp.json" | Invoke-Expression применить (использовать) тему в текущей сессии
oh-my-posh init pwsh --config "https://raw.githubusercontent.com/JanDeDobbeleer/oh-my-posh/main/themes/cert.omp.json" | Invoke-Expression считать тему из репозитория
New-Item -Path $PROFILE -Type File -Force создает файл профилья PowerShell
'oh-my-posh init pwsh --config "$env:POSH_THEMES_PATH/di4am0nd.omp.json" | Invoke-Expression' > $PROFILE сохранить тему профиля (загружать тему при запуске PowerShell)
```

## themes-performance

```
Install-Module themes-performance -Repository NuGet установить модуль с темами
Set-PoshTheme -Theme System-Sensors использовать тему с датчиками из LibreHardwareMonitor
Set-PoshTheme -Theme System-Sensors -Save загрузить тему из репозитория на локальный компьютер и сохранить тему в профиле
```

```
Set-PoshTheme -Theme System-Performance использовать тему с датчиками системы, получаемыми из системы WMI/CIM (заряд батареи ноутбука |  
загрузка CPU в % | использование оперативной памяти | скорость активного сетевого интерфейса)  
Set-PoshTheme -Theme System-Performance -Save  
Set-PoshTheme -Theme Pwsh-Process-Performance время работы текущего процесса pwsh (процессорное время), количество работающих/общее  
(статус успех/ошибка) фоновых заданий, Working Set текущего процесса и всех процессов PowerShell в системе  
Set-PoshTheme -Theme Pwsh-Process-Performance -Save
```

## Windows-Terminal

### Terminal-Icons

```
Install-Module -Name Terminal-Icons -Repository PSGallery  
scoop bucket add extras  
scoop install terminal-icons  
  
notepad $PROFILE  
Import-Module -Name Terminal-Icons
```

Использует шрифты, которые необходимо установить и настроить в параметрах профиля PowerShell: [Nerd Fonts](#)

#### Список шрифтов

Скачать и установить шрифт похожий на Cascadia Code - [CaskaydiaCove](#)

Установить шрифт в конфигурацию Windows Terminal для PowerShell Core:

```

"profiles":
{
    "defaults":
    {
        "colorScheme": "One Half Dark",
        "experimental.retroTerminalEffect": true,
        "font":
        {
            "size": 10.0
        },
        "useAtlasEngine": true
    },
    "list":
    [
        // PowerShell Core
        {
            "font":
            {
                "face": "CaskaydiaMono Nerd Font" // устанавливаем шрифт для работы Terminal-Icons
            },
            "guid": "{574e775e-4f2a-5b96-ac1e-a2962a402336}",
            "hidden": false,
            "name": "PowerShell Core",
            "source": "Windows.Terminal.PowershellCore"
        },
        // WSL (Ubuntu)
        {
            "guid": "{2c4de342-38b7-51cf-b940-2309a097f518}",
            "hidden": false,
            "name": "WSL",
            "source": "Windows.Terminal.Wsl"
        },
        // ssh
        {
            "commandline": "ssh lifailon@192.168.1.100 -p 22",
            "guid": "{a3ec86f6-2bc1-59dd-814d-2a0d935af5f8}",
            "icon": "\ud83d\udcbb",
            "name": "devops-01"
        }
    ]
}

```

## Custom Actions

Custom actions: <https://learn.microsoft.com/ru-ru/windows/terminal/customize-settings/actions>

Escape-последовательности: <https://learn.microsoft.com/ru-ru/cpp/c-language/escape-sequences?view=msvc-170>

```

"actions":
[
    {
        "command": {
            "action": "copy",
            "singleLine": false
        },
        "keys": "ctrl+c"
    },
    {
        "command": "paste",
        // Сохраняем классическую вставку интерпритатора, не заставляя выполнять код построчно
        "keys": "ctrl+shift+v" // default: ctrl+v
    },
    {
        "command": "find",
        "keys": "ctrl+f" // default: ctrl+shift+f
    },
    {
        "command": {
            "action": "splitPane",
            "split": "left", // default: auto
            "splitMode": "duplicate"
        },
        "keys": "ctrl+shift+d" // default: alt+shift+d
    },
    // Очистить строку
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~"
        },
        "keys": "ctrl+k"
    },
    // Очистить терминал
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~clear\r"
        },
        "keys": "ctrl+l"
    },
    // Вставить шаблон модуля для перевода текста через Google Translate
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~Get-Translate -Alternatives -Provider Google ''\u001b[D"
        },
        "keys": "ctrl+g"
    },
    // Вставить шаблон модуля для перевода текста через MyMemory
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~Get-Translate -Alternatives -Provider MyMemory ''\u001b[D"
        },
        "keys": "ctrl+q"
    },
    // Быстрый перевод текста из буфера обмена
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~Get-Translate -Alternatives -Provider MyMemory -Text $(Get-Clipboard)\u001b[D\r"
        }
    }
]

```

```

        },
        "keys": "ctrl+shift+q"
    },
    // Быстрый пинг dns google
    {
        "command": {
            "action": "sendInput",
            "input": "\u0001\u001b[3~ping 8.8.8.8 -t\r"
        },
        "keys": "ctrl+p"
    }
]

```

## Pandoc

```

$release_latest = Invoke-RestMethod "https://api.github.com/repos/jgm/pandoc/releases/latest"
$url = $($release_latest.assets | Where-Object name -match "windows-x86_64.zip").browser_download_url
Invoke-RestMethod $url -OutFile $home\Downloads\pandoc.zip
Expand-Archive -Path "$home\Downloads\pandoc.zip" -DestinationPath "$home\Downloads\" 
$path = $(Get-ChildItem "$home\Downloads\pandoc-*\*.exe").FullName
Copy-Item -Path $path -Destination "C:\Windows\System32\pandoc.exe"
Remove-Item "$home\Downloads\pandoc*" -Force -Recurse

pandoc -s README.md -o index.html конвертация из Markdown в HTML
pandoc README.md -o index.html --css=styles.css применить стили из CSS
pandoc -s index.html -o README.md конвертация из HTML в Markdown
pandoc -s README.md -o README.docx конвертация в Word
pandoc -s README.md -o README.epub конвертация в открытый формат электронных версий книг
pandoc -s README.md -o README.pdf конвертация в PDF (требуется rsvg-convert)
pandoc input.md -f markdown+hard_line_breaks -o output.md конвертация из markdown документа, который не содержит обратный слэш в конце каждой строки для переноса (), который их добавит

```

## Convert Excel to Markdown

```

Import-Module ImportExcel
Import-Excel -Path srv.xlsx | Export-Csv -Path $csvFilePath -NoTypeInformation -Encoding UTF8 # конвертация Excel в csv
pandoc -s -f csv -t markdown input.csv -o output.md # конвертация таблицу csv в markdown

```

## FFmpeg

```

$release_latest = Invoke-RestMethod "https://api.github.com/repos/BtbN/FFmpeg-Builds/releases/latest"
$url = $($release_latest.assets | Where-Object name -match "ffmpeg-master-latest-win64-gpl.zip").browser_download_url
Invoke-RestMethod $url -OutFile $home\Downloads\ffmpeg-master-latest-win64-gpl.zip
Expand-Archive -Path "$home\Downloads\ffmpeg-master-latest-win64-gpl.zip" -DestinationPath "$home\Downloads\" 
Copy-Item -Path "$home\Downloads\ffmpeg-master-latest-win64-gpl\bin\ffmpeg.exe" -Destination "C:\Windows\System32\ffmpeg.exe"
Remove-Item "$home\Downloads\ffmpeg*" -Force -Recurse

ffmpeg -i input.mp4 output.gif конвертировать mp4 в gif
ffmpeg -i input.mp4 -filter_complex "scale=1440:-1:flags=lanczos" output.gif изменить разрешение на выходе
ffmpeg -i input.mp4 -filter_complex "scale=1440:-1:flags=lanczos" -r 10 output.gif изменить количество кадров в секунду на выходе
ffmpeg -i input.mp4 -filter_complex "fps=5,scale=960:-1:flags=lanczos,split[s0][s1];[s0]paletteneg=max_colors=32[p];[s1][p]paletteuse=dither=bayer" сжатие за счет цветовой политики
ffmpeg -i input.mp4 -ss 00:00:10 -frames:v 1 -q:v 1 output.jpg вытащить скриншот из видео на 10 секунде
ffmpeg -i input.mp4 -ss 00:00:05 -to 00:00:10 -c copy output.mp4 вытащить кусок видео
ffmpeg -i "%d.jpeg" -framerate 2 -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4 создать видео из фото (1.jpeg, 2.jpeg и т.д.) с framerate (частотой кадров) в создаваемом видео 2 кадра в секунду
ffmpeg -i "rtsp://admin:password@192.168.3.201:554" -rtsp_transport tcp -c:v copy -c:a aac -strict experimental output.mp4 запись без

```

перекодирования (copy) RTSP-потока с камеры видеонаблюдения (+ аудио в кодеке AAC) в файл

```
ffmpeg -i "rtsp://admin:password@192.168.3.201:554" -rtsp_transport tcp -c:v copy -c:a aac -strict experimental -movflags +faststart+frag_keyframe+переместить метаданные в начало файла, что позволяет начать воспроизведение файла в видеоплеере до его полной загрузки
ffmpeg -i "rtsp://admin:password@192.168.3.201:554" -rtsp_transport tcp -frames:v 1 -c:v mjpeg output.jpg сделать скриншот
ffmpeg -i input.mp4 -vf "pad=width=iw:height=ih+100:x=0:y=100:color=black" -c:a copy output.mp4 width=iw: (ширина видео остается как у исходного файла), height=ih+100 (высота видео увеличивается на 100 пикселей), x=0 (горизонтальное смещение установлено в 0), y=100 (вертикальное смещение установлено в 100 пикселей вниз, чтобы добавить черное пространство сверху), color=black (цвет добавленного пространства — черный)
```

## HandBrake

```
$url = "https://github.com/HandBrake/HandBrake/releases/download/1.8.0/HandBrakeCLI-1.8.0-win-x86_64.zip"
Invoke-RestMethod $url -OutFile $home\Downloads\HandBrakeCLI.zip
Expand-Archive -Path $home\Downloads\HandBrakeCLI.zip -OutputPath "$home\Downloads\""
Copy-Item -Path "$home\Downloads\HandBrakeCLI.exe" -Destination "C:\Windows\System32\HandBrakeCLI.exe"
Remove-Item "$home\Downloads\doc" -Force -Recurse
Remove-Item "$home\Downloads\HandBrakeCLI*"
```

HandBrakeCLI -i input.mp4 -o output.mkv конвертирует видео в формат mp4 в формат mkv с использованием стандартных настроек HandBrake  
 HandBrakeCLI -i input.mp4 -o output.mkv -q 20 установить качество видео 20, значения варьируются от 0 (максимальное качество) до 51  
 (минимальное качество), где 20 считается хорошим качеством для большинства видео  
 HandBrakeCLI -i input.mp4 -o output.mkv -r 30 установить частоту кадров на 30 fps  
 HandBrakeCLI -i input.mp4 -o output.mkv --maxWidth 1280 --maxHeight 720 изменить размер на 1280x720  
 HandBrakeCLI -i input.mp4 -o output.mkv -b 1500 установить битрейт видео 1500 кбит/с  
 HandBrakeCLI -i input.mp4 -o output.mkv -e x264 преобразовать видео с использованием кодека x264  
 HandBrakeCLI -i input.mp4 -o output.mp4 --crop 0:200:0:0 обрезать видео снизу на 200px (верх:низ:лево:право)  
 HandBrakeCLI -i input.mp4 -o output.mp4 --start-at duration:5 --stop-at duration:15 обрезать видео (на выходе будет 15-секундное видео с 5 по 20 секунды)

## ImageMagick

Source: [ImageMagick](#)

```
magick identify -verbose PowerShell-Commands.png извлечь метаданные изображения
magick PowerShell-Commands.png output.jpg конвертация формата изображения
magick PowerShell-Commands.png -resize 800x600 output.jpg изменить размер (увеличить или уменьшить)
magick PowerShell-Commands.png -crop 400x300+100+50 output.jpg обрезать
magick PowerShell-Commands.png -rotate 90 output.jpg повернуть изображение
magick PowerShell-Commands.png -fill white -pointsize 24 -gravity center -annotate +0+0 "PowerShell" output.jpg наложить текст на изображение
magick PowerShell-Commands.png -brightness-contrast +20x+10 output.jpg изменить яркость и контрастность
magick convert -delay 100 1.png 2.png 3.png output.gif создать gif из изображений
magick convert image1.jpg image2.jpg -append output.jpg вертикально объединить изображения
```

## YouTube

```
$release_latest = Invoke-RestMethod "https://api.github.com/repos/yt-dlp/yt-dlp/releases/latest"
$url = $($release_latest.assets | Where-Object name -match "yt-dlp.exe").browser_download_url
Invoke-RestMethod $url -OutFile "C:\Windows\System32\yt-dlp.exe"
```

```
yt-dlp -F https://www.youtube.com/watch?v=gxplizjhqiw отобразить список всех доступных форматов
yt-dlp -J https://www.youtube.com/watch?v=gxplizjhqiw вывести данные в формате JSON
yt-dlp -J https://www.youtube.com/watch?v=gxplizjhqiw | jq -r .formats[].format id -resolution (format_note)
yt-dlp -f 137 https://www.youtube.com/watch?v=gxplizjhqiw загрузить только видео в указанном формате по id
yt-dlp -f bestaudio https://www.youtube.com/watch?v=gxplizjhqiw загрузить только аудио
yt-dlp -f best https://www.youtube.com/watch?v=gxplizjhqiw загрузить видео с аудио в лучшем качестве
```

```
yt-dlp -f 'bestvideo[height<=1080]+bestaudio/best[height<=1080]' https://www.youtube.com/watch?v=gxplizjhqiw загрузить в указанном качестве  
yt-dlp -r 2m https://www.youtube.com/watch?v=gxplizjhqiw ограничить скорость загрузки до 2 Мбит/с
```

```
function Get-YouTube {  
    param (  
        $url  
    )  
    $result = yt-dlp -J $url  
    $($result | ConvertFrom-Json).formats |  
    Where-Object filesize -ne $null |  
    Select-Object format_id,  
    @{Name="FileSize";  
        Expression={[string]([int]($_.filesize / 1024kb)).ToString("0.0")+" Mb"}  
    },  
    resolution,format_note,quality,fps,ext,language  
}  
  
$formats = Get-YouTube "https://www.youtube.com/watch?v=gxplizjhqiw"  
$video = $($formats | Where-Object format_note -match 1080 | Where-Object ext -match mp4)[-1].format_id  
$audio = $($formats | Where-Object resolution -match "audio" | Where-Object ext -match m4a)[-1].format_id  
cd "$home\Downloads"  
yt-dlp -f $video+$audio https://www.youtube.com/watch?v=gxplizjhqiw -o '%(title)s.%ext)s'
```