

Оглавление

Об авторе	16
О научном редакторе.....	17
Благодарности	18
Введение	19
Почему именно PowerShell?.....	20
Для кого эта книга.....	20
О книге.....	21
От издательства.....	22

Часть I. Основы

Глава 1. Начало работы.....	24
Открытие консоли PowerShell.....	24
Использование команд DOS	25
Изучаем команды PowerShell	27
Подсказки.....	29
Отображение документов	29
Изучение общих аспектов	31
Обновление документов.....	33
Итоги.....	33
Глава 2. Основные понятия PowerShell.....	34
Переменные.....	34
Отображение и изменение переменной	35
Пользовательские переменные.....	35
Встроенные переменные.....	38
Типы данных.....	41
Логические значения.....	42
Целые числа и числа с плавающей точкой	42
Строки.....	44

Объекты.....	46
Проверка атрибутов.....	47
Использование командлета Get-Member.....	48
Вызов методов.....	48
Структуры данных.....	50
Массивы.....	50
Объекты ArrayList	54
Хеш-таблицы.....	56
Создание пользовательских объектов	58
Итоги.....	60
 Глава 3. Объединение команд.....	 61
Запуск службы Windows.....	61
Использование конвейера.....	62
Передача объектов между командами	62
Передача массивов между командами.....	63
Поговорим о привязке параметров.....	64
Написание сценариев.....	66
Настройка политики выполнения	67
Создание сценариев в PowerShell	69
Итоги.....	72
 Глава 4. Поток управления.....	 73
Немного о потоке управления.....	74
Использование условных операторов	75
Построение выражений с помощью операторов.....	75
Использование циклов	81
Цикл foreach.....	82
Цикл for.....	85
Цикл while	86
Циклы do/while и do/until.....	87
Итоги.....	88
 Глава 5. Обработка ошибок.....	 89
Работа с исключениями и ошибками.....	89
Обработка незавершающих ошибок	91
Обработка завершающих ошибок.....	93
Изучение автоматической переменной \$Error	95
Итоги.....	96

Глава 6. Пишем функции	97
Функции и командлеты	98
Определение функции.....	98
Добавление параметров в функции.....	99
Создание простого параметра	100
Атрибут параметра <i>Mandatory</i>	101
Значения параметров по умолчанию	102
Добавление атрибутов проверки параметров.....	103
Прием входных данных конвейера.....	105
Добавление еще одного параметра	105
Организация совместимости функции с конвейером.....	106
Добавление блока <i>process</i>	107
Итоги.....	108
Глава 7. Изучаем модули.....	109
Изучение встроенных модулей	110
Поиск модулей в сеансе	110
Поиск модулей на вашем компьютере.....	111
Импорт модулей.....	113
Компоненты PowerShell-модуля	114
Файл .psm1	114
Манифест модуля.....	114
Работа с пользовательскими модулями	116
Поиск модулей	116
Установка модулей	117
Деинсталляция модулей	118
Создание собственного модуля	119
Итоги.....	120
Глава 8. Удаленный запуск сценариев.....	121
Работа с блоками сценариев	122
Использование команды <i>Invoke-Command</i> для выполнения кода на удаленных системах.....	123
Запуск локальных сценариев на удаленных компьютерах	125
Удаленное использование локальных переменных	125
Работа с сессиями	127
Создание нового сеанса	128
Вызов команд в сеансе	129
Открытие интерактивных сеансов	130

Отключение от сеансов и повторное подключение к ним.....	131
Удаление сеансов с помощью команды Remove-PSSession	133
Общие сведения об авторизации при удаленном управлении PowerShell.....	133
Проблема двойного перехода.....	134
Двойной прыжок с использованием CredSSP.....	135
Итоги.....	138
Глава 9. Тестирование с помощью Pester	139
Знакомьтесь: Pester	139
Основы Pester	140
Файл Pester	140
Блок describe	141
Блок context.....	141
Блок it.....	142
Утверждения.....	142
Выполнение теста Pester.....	143
Итоги.....	144
Часть II. Автоматизация повседневных задач	
Глава 10. Парсинг структурированных данных.....	148
CSV-файлы.....	148
Чтение CSV-файлов.....	149
Создание CSV-файлов.....	153
Проект 1. Создание отчета об инвентаризации компьютеров	154
Таблицы Excel.....	158
Создание таблиц Excel.....	159
Чтение таблиц Excel.....	160
Добавление данных в таблицы Excel.....	161
Проект 2. Создание инструмента мониторинга служб Windows.....	162
Данные в формате JSON.....	164
Чтение данных в формате JSON.....	165
Создание строк JSON.....	166
Проект 3. Запрос и парсинг REST API.....	168
Итоги.....	170
Глава 11. Автоматизация Active Directory.....	171
Исходные требования.....	171
Установка модуля ActiveDirectory в PowerShell.....	172

Запросы и фильтрация объектов AD	173
Фильтрация объектов	173
Возврат отдельных объектов	175
Проект 4. Поиск учетных записей пользователей, пароль которых не менялся в течение 30 дней	176
Создание и изменение объектов AD	178
Пользователи и компьютеры	178
Группы	179
Проект 5. Создание сценария приема сотрудников	181
Синхронизация с другими источниками данных.....	184
Проект 6. Создание сценария синхронизации.....	185
Сопоставление атрибутов источника данных.....	186
Создание функций для возврата схожих свойств	187
Поиск совпадений в Active Directory.....	189
Изменение атрибутов Active Directory	191
Итоги.....	192
 Глава 12. Работа с Azure	193
Исходные требования.....	193
Авторизация в Azure	194
Создание субъекта-службы	194
Неинтерактивная авторизация с помощью команды Connect-AzAccount	196
Создание виртуальной машины Azure и всех зависимостей	197
Создание группы ресурсов.....	198
Создание сетевого стека.....	198
Создание учетной записи хранения.....	200
Создание образа операционной системы	201
Закругляемся.....	203
Автоматизация создания ВМ	204
Развертывание веб-приложения на Azure	204
Создание плана службы приложений и веб-приложения.....	205
Развертывание базы данных SQL Azure	206
Создание Azure SQL Server	206
Создание базы данных SQL Azure	207
Создание правила брандмауэра SQL Server.....	208
Тестирование вашей базы данных SQL	209
Итоги.....	210

Глава 13. Работа с Amazon Web Services.....	211
Исходные требования.....	211
Авторизация в AWS	212
Авторизация с пользователя root.....	212
Создание пользователя и роли IAM.....	213
Авторизация вашего пользователя IAM	216
Создание экземпляра AWS EC2.....	217
Виртуальное частное облако.....	217
Интернет-шлюз.....	218
Развертывание приложения Elastic Beanstalk	223
Создание приложения	224
Развертывание пакета.....	226
Создание базы данных SQL Server в AWS.....	228
Итоги.....	232
Глава 14. Создание сценария инвентаризации сервера	233
Исходные требования.....	233
Создание сценария проекта.....	234
Определение окончательного результата.....	234
Обнаружение и ввод сценария	234
Запрос всех серверов.....	236
Думаем наперед: объединение различных типов информации	237
Запрос файлов на удаленном расположении	240
Запрос инструментария управления Windows	242
Свободное место на диске.....	243
Информация об операционной системе.....	244
Память	245
Информация о сети	247
Службы Windows	251
Очистка и оптимизация скрипта.....	253
Итоги.....	256

Часть III. Создаем свой модуль

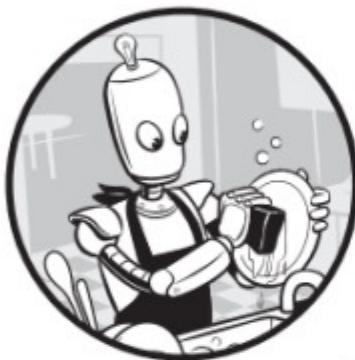
Глава 15. Создание виртуальной среды	261
Исходные требования для модуля PowerLab	262
Создание модуля.....	263
Создание пустого модуля	263
Создание манифеста модуля	264

Использование встроенных префиксов для имен функций	264
Импорт нового модуля.....	265
Автоматизация подготовки виртуальной среды	266
Виртуальные коммутаторы.....	266
Создание виртуальных машин.....	268
Виртуальные жесткие диски.....	270
Тестирование новых функций с помощью Pester	274
Итоги.....	275
Глава 16. Установка операционной системы.....	276
Исходные требования.....	276
Развертывание ОС.....	277
Создание VHDX	277
Подключение виртуальной машины.....	279
Автоматизация развертывания ОС	280
Хранение зашифрованных учетных данных на диске.....	282
PowerShell Direct	284
Тестирование с помощью Pester.....	285
Итоги.....	287
Глава 17. Развертывание Active Directory.....	288
Исходные требования.....	288
Создание леса Active Directory.....	289
Создаем лес	289
Сохранение защищенных строк на диск	290
Автоматизация создания леса.....	291
Заполнение домена.....	293
Сборка и запуск тестов Pester	298
Итоги.....	300
Глава 18. Создание и настройка SQL-сервера.....	301
Исходные требования.....	301
Создание виртуальной машины	302
Установка операционной системы	302
Добавление файла автоматического ответа Windows.....	303
Добавление SQL-сервера в домен.....	304
Установка SQL-сервера.....	306
Копирование файлов на SQL-сервер.....	306
Запуск установщика SQL-сервера.....	307

Автоматизация SQL-сервера.....	308
Запуск тестов Pester.....	312
Итоги.....	313
 Глава 19. Рефакторинг кода.....	 314
Еще раз о функции New-PowerLabSqlServer.....	314
Использование наборов параметров.....	319
Итоги.....	322
 Глава 20. Создание и настройка веб-сервера IIS	 323
Исходные требования.....	323
Установка и настройка	324
Создание веб-сервера с нуля.....	325
Модуль веб-администрирования.....	326
Сайты и пулы приложений.....	327
Сайты.....	327
Настройка SSL на веб-сайте.....	329
Итоги.....	333

17

Развертывание Active Directory



В этой главе мы воспользуемся информацией из нескольких последних глав части II и начнем разворачивать службы поверх виртуальных машин. Поскольку многим сервисам необходимо наличие Active Directory, сначала необходимо развернуть лес и домен Active Directory. Лес и домен AD будут выполнять задачи аутентификации и авторизации в оставшихся главах.

Я предполагаю, что вы прочитали предыдущие главы и настроили виртуальную машину LABDC, потому что именно ее мы и будем использовать для полной автоматизации леса Active Directory и заполнения тестовыми пользователями и группами.

Исходные требования

Нам понадобятся результаты нашей работы из главы 16, так что я надеюсь, что у вас уже настроена виртуальная машина LABDC, выполнена сборка с автоматизацией XML и загружен Windows Server 2016. Если это так, то все в порядке. Если нет, то в этой главе есть примеры автоматизации Active Directory. Но предупреждаю, что в таком случае вы не сможете повторить инструкции точь-в-точь.

Как всегда, запустите предварительный тест Pester, чтобы убедиться в выполнении всех исходных требований для этой главы.

Создание леса Active Directory

Хорошая новость заключается в том, что создать лес AD с помощью PowerShell довольно просто. Нам нужны всего две команды — `Install-WindowsFeature` и `Install-ADDSForest`. С их помощью вы можете построить лес и домен и подготовить сервер Windows на роль контроллера домена.

Поскольку мы будем использовать лес в лабораторной среде, нам понадобятся организационные единицы, пользователи и группы. Нахождение в лабораторной среде означает, что у вас нет производственных объектов для работы. Гораздо проще создать множество объектов, имитирующих производственные, чем пытаться синхронизировать реальные объекты с лабораторной средой.

Создаем лес

Первое, что вам нужно сделать при построении нового леса AD — это создать *контроллер домена*, краеугольный камень Active Directory. Чтобы создать рабочую среду AD, нужно завести хотя бы один контроллер домена.

Поскольку у нас лабораторная среда, нам нужен всего один контроллер домена. В реальной ситуации вам понадобятся как минимум два из них (про запас). В нашей ситуации в лабораторной среде нет данных — их нужно придумывать с нуля, — поэтому нам хватит одного контроллера. Перед началом необходимо установить Windows-компонент `AD-Domain-Services` на вашем сервере LABDC с помощью команды `Install-WindowsFeature`:

```
PS> $cred = Import-CliXml -Path C:\Files.xml
PS> Invoke-Command -VMName 'LABDC' -Credential $cred -ScriptBlock
{ Install-WindowsFeature -Name AD-Domain-Services }
PSComputerName : LABDC RunspaceId : 33d41d5e-50f3-475e-a624-4cc407858715
Success : True RestartNeeded : No FeatureResult : {Active Directory Domain Services, Remote Server Administration Tools, Active Directory module for Windows PowerShell, AD DS and AD LDS Tools...} ExitCode : Success
```

После предоставления учетных данных для подключения к серверу используем команду `Invoke-Command` для удаленного запуска команд `Install-WindowsFeature` на удаленном сервере.

После установки компонента с помощью команды `Install-ADDSForest` можно создать лес. Эта команда является частью PowerShell-модуля `ActiveDirectory`, который вы уже установили на LABDC.

Команда `Install-ADDSForest` — единственная, которая вам нужна для создания леса. У нее еще есть параметры, которые мы зададим с помощью кода, но

обычно их вводят с помощью графического интерфейса. Наш лес будет называться `powerlab.local`. Поскольку контроллером домена является Windows Server 2016, следует установить значение `WinThreshold` для режимов домена и леса. Все доступные значения параметров `DomainMode` и `ForestMode` можно посмотреть на странице документации Microsoft *Install-ADDSForest* (<http://bit.ly/2rrgUi6>).

Сохранение защищенных строк на диск

В главе 16, когда вам потребовались учетные данные, вы сохраняли их в объекты `PSCredential`, а затем повторно использовали их в своих командах. Но на этот раз нам не нужен объект `PSCredential`: нам хватит одной зашифрованной строки.

В этом разделе нам понадобится передать команде пароль администратора безопасного режима. Как и в случае с любой другой конфиденциальной информацией, необходимо использовать шифрование. Для сохранения и извлечения объектов PowerShell из файловой системы вы будете использовать команды `Export-CliXml` и `Import-CliXml`. Однако здесь вместо вызова `Get-Credential` мы создадим безопасную строку с помощью команды `ConvertTo-SecureString`, а затем сохраним этот объект в файл.

Чтобы сохранить зашифрованный пароль в файл, передайте его в виде обычного текста в `ConvertTo-SecureString`, а затем экспортируйте этот защищенный строковый объект в `Export-CliXml`, создав файл, на который можно ссылаться позже:

```
PS> 'P@$$w0rd12' | ConvertTo-SecureString -Force -AsPlainText  
| Export-Clixml -Path C:\PowerLab\SafeModeAdministratorPassword.xml
```

Итак, после сохранения пароля администратора безопасного режима на диск его можно считать с помощью `Import-CliXml` и передать все остальные параметры, которые необходимо запустить с `Install-ADDSForest`. Мы сделаем это с помощью следующего кода:

```
PS> $safeModePw = Import-CliXml -Path C:\PowerLab\  
SafeModeAdministratorPassword.xml  
PS> $cred = Import-CliXml -Path C:\PowerLab\VMCredential.xml  
PS> $forestParams = @{  
    >>> DomainName = 'powerlab.local' ①  
    >>> DomainMode = 'WinThreshold' ②  
    >>> ForestMode = 'WinThreshold'  
    >>> Confirm = $false ③
```

```
>>> $SafeModeAdministratorPassword = $safeModePw ④
>>> $WarningAction           = 'Ignore' ⑤
>>>
PS> Invoke-Command -VMName 'LABDC' -Credential $cred -ScriptBlock { $null =
Install-ADDSForest @using:forestParams }
```

Здесь мы создаем лес и домен под названием powerlab.local ①, работающий на функциональном уровне Windows Server 2016 (WinThreshold) ②. Затем мы обходим все подтверждения ③, передаем пароль администратора безопасного режима ④ и игнорируем возникающие нерелевантные сообщения с предупреждениями ⑤.

Автоматизация создания леса

Теперь, когда мы сделали все вручную, создадим в модуле PowerLab функцию, которая будет автоматически обрабатывать создание леса AD. Позже ее можно будет использовать и в других средах.

В модуле PowerLab, включенном в материалы этой главы, вы увидите функцию New-PowerLabActiveDirectoryForest, которая приведена в листинге 17.1.

Листинг 17.1. Функция New-PowerLabActiveDirectoryForest

```
function New-PowerLabActiveDirectoryForest {
    param(
        [Parameter(Mandatory)]
        [pscredential]$Credential,
        [Parameter(Mandatory)]
        [string]$SafeModePassword,
        [Parameter()]
        [string]$VMName = 'LABDC',
        [Parameter()]
        [string]$DomainName = 'powerlab.local',
        [Parameter()]
        [string]$DomainMode = 'WinThreshold',
        [Parameter()]
        [string]$ForestMode = 'WinThreshold'
    )

    Invoke-Command -VMName $VMName -Credential $Credential -ScriptBlock {
        Install-WindowsFeature -Name AD-Domain-Services

        $forestParams = @{
            DomainName           = $using:DomainName
```

```

        DomainMode          = $using:DomainMode
        ForestMode         = $using:ForestMode
        Confirm            = $false
        SafeModeAdministratorPassword = (ConvertTo-SecureString
                                         -AsPlainText -String $using:
                                         SafeModePassword -Force)
        WarningAction      = 'Ignore'
    }
    $null = Install-ADDSForest @forestParams
}
}

```

Как и в предыдущей главе, здесь мы определяем несколько параметров, которые будем использовать для передачи в команду `Install-ADDSForest` модуля `ActiveDirectory`. Обратите внимание, что эти два параметра учетных данных и пароля являются обязательными (`Mandatory`). Как следует из названия, пользователь должен задать эти параметры самостоятельно (прочие параметры имеют значения по умолчанию, поэтому пользователю не обязательно передавать их). С помощью этой функции вы можете прочитать сохраненный пароль администратора и учетные данные, а затем передать их в функцию:

```

PS> $safeModePw = Import-CliXml -Path C:\PowerLab\SafeModeAdministratorPassword.xml
PS> $cred = Import-CliXml -Path C:\PowerLab\VMCredential.xml
PS> New-PowerLabActiveDirectoryForest -Credential $cred
                  -SafeModePassword $safeModePw

```

После запуска этого кода у вас будет полностью рабочий лес AD! И все же, давайте найдем способ подтвердить, что все работает как надо. Как вариант, можно запросить все учетные записи пользователей в домене по умолчанию. Однако для этого необходимо создать другой объект `PSCredential` на диске. Поскольку LABDC теперь является контроллером домена, вместо учетной записи локального пользователя потребуется учетная запись пользователя домена. Мы создадим и сохраним данные с именем пользователя из `powerlab.local\administrator` и паролем из `P@$$w0rd12` в файле `C:\PowerLab\DomainCredential.xml`. Помните, что вам нужно сделать это только один раз. Затем вы сможете использовать новые учетные данные домена для подключения к LABDC:

```
PS> Get-Credential | Export-CliXml -Path C:\PowerLab\DomainCredential.xml
```

После создания учетных данных добавим в наш модуль PowerLab еще одну функцию под названием `Test-PowerLabActiveDirectoryForest`. Сейчас эта функция просто опрашивает всех пользователей в домене, но поскольку эта задача организована в функцию, вы можете настроить этот тест на свое усмотрение:

```
function Test-PowerLabActiveDirectoryForest {
    param(
        [Parameter(Mandatory)]
        [pscredential]$Credential,
        [Parameter()]
        [string]$VMName = 'LABDC'
    )
    Invoke-Command -Credential $Credential -ScriptBlock {Get-AdUser -Filter * }
}
```

Попробуйте выполнить функцию `Test-PowerLabActiveDirectoryForest`, используя учетные данные домена и имя виртуальной машины LABDC. Если в выводе будет несколько учетных записей пользователей, то поздравляю — все работает! Теперь вы успешно настроили контроллер домена и сохранили учетные данные для подключения к виртуальным машинам в рабочей группе (и к любым виртуальным машинам, подключенным к домену, на будущее).

Заполнение домена

В предыдущем разделе мы настроили контроллер домена в PowerLab. Давайте теперь создадим несколько тестовых объектов. Поскольку это тестовая среда, нам нужно самостоятельно создать различные объекты (подразделения, пользователи, группы и т. д.) для проверки базы данных. Можно запустить отдельную команду для создания каждого отдельного объекта, но это будет непрактично. Гораздо удобнее было бы определить все это в одном файле, прочитать каждый объект и создать их все за один раз.

Работа с таблицей объектов

В качестве исходного файла будем использовать электронную таблицу Excel для определения входных данных — она есть в прилагаемых к этой главе материалах. В таблице есть два листа: `Users` (рис. 17.1) и `Groups` (рис. 17.2).

Каждая строка в этих листах соответствует пользователю или группе, которую необходимо создать, а также содержит информацию для передачи в PowerShell. Как было сказано в главе 10, встроенная оболочка PowerShell не может обрабатывать электронные таблицы Excel без вашего участия. Однако с помощью готового модуля мы сделаем все проще. С модулем `ImportExcel` можно считывать электронные таблицы Excel столь же легко, как и CSV-файлы. Загрузите его из PowerShell Gallery с помощью команды `Install-Module -Name ImportExcel`. После нескольких запросов безопасности модуль готов к использованию.

	A	B	C	D	E
1	OUName	UserName	FirstName	LastName	MemberOf
2	PowerLab Users	jjones	Joe	Jones	Accounting
3	PowerLab Users	abertram	Adam	Bertram	Accounting
4	PowerLab Users	jhicks	Jeff	Hicks	Accounting
5	PowerLab Users	dtrump	Donald	Trump	Human Resources
6	PowerLab Users	alincoln	Abraham	Lincoln	Human Resources
7	PowerLab Users	bobama	Barack	Obama	Human Resources
8	PowerLab Users	tjefferson	Thomas	Jefferson	IT
9	PowerLab Users	bclinton	Bill	Clinton	IT
10	PowerLab Users	gbush	George	Bush	IT
11	PowerLab Users	rreagan	Ronald	Reagan	IT

Рис. 17.1. Таблица Users

	A	B	C
1	OUName	GroupName	Type
2	PowerLab Groups	Accounting	DomainLocal
3	PowerLab Groups	Human Resources	DomainLocal
4	PowerLab Groups	IT	DomainLocal

Рис. 17.2. Таблица Groups

Теперь воспользуемся командой Import-Excel для анализа таблиц:

```
PS> Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Users | Format-Table -AutoSize
OUName      UserName   FirstName LastName MemberOf
----      -----   -----   -----   -----
PowerLab Users jjones    Joe       Jones    Accounting
PowerLab Users abertram  Adam     Bertram  Accounting
PowerLab Users jhicks   Jeff     Hicks    Accounting
PowerLab Users dtrump   Donald   Trump    Human Resources
PowerLab Users alincoln Abraham Lincoln Human Resources
PowerLab Users bobama   Barack   Obama   Human Resources
PowerLab Users tjefferson Thomas  Jefferson IT
PowerLab Users bclinton  Bill    Clinton  IT
PowerLab Users gbush    George  Bush    IT
PowerLab Users rreagan   Ronald Reagan IT
```

```
PS> Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Groups | Format-Table -AutoSize
```

OUName	GroupName	Type
PowerLab Groups	Accounting	DomainLocal
PowerLab Groups	Human Resources	DomainLocal
PowerLab Groups	IT	DomainLocal

С помощью параметров `Path` и `WorksheetName` можно легко извлечь необходимые данные. Обратите внимание на команду `Format-Table`: она заставляет PowerShell отображать вывод в табличном формате. Параметр `AutoSize` сообщает PowerShell, что нужно попытаться «уложить» каждую строку в одну строку консоли.

Разработка плана

Теперь вы можете считывать данные из электронной таблицы Excel. Осталось понять, что с ними делать. В модуле PowerLab мы создадим функцию, которая считывает каждую строку и выполняет требуемое действие. Весь описываемый здесь код доступен с помощью `New-PowerLabActive DirectoryTestObject` в соответствующем модуле PowerLab.

Эта функция немного сложнее наших предыдущих сценариев, поэтому давайте разберем ее нестандартным образом, чтобы позже вы могли обращаться к этой информации. Этот шаг может показаться неважным, но в случае с более крупными функциями предварительное планирование значительно облегчит вам работу в долгосрочной перспективе. В этой функции вам необходимо сделать следующее:

1. Считать оба листа в электронной таблице Excel и получить все строки пользователей и групп.
2. Прочитать каждую строку на обоих листах и проверить, существует ли OU (подразделение), частью которого должен быть пользователь или группа.
3. Если OU не существует, его нужно создать.
4. Если пользователь/группа не существует, их нужно создать.
5. Добавить пользователя в указанную группу (этот шаг выполняется только для пользователя).

Теперь, когда у нас есть этот неформальный план, давайте приступим к написанию кода.

Создание AD-объектов

Для начала не будем все усложнять: сосредоточимся на обработке одного объекта. Нам незачем сейчас думать обо всем сразу. Ранее вы уже устанавливали Windows-компонент `AD-Domain-Services` на LABDC, поэтому у вас уже установлен модуль `ActiveDirectory`. Как вы убедились при изучении главы 11, в этом модуле есть большой набор полезных команд. Напомним, что многие команды следуют тому же соглашению об именах, что и `Get/Set/New-AD`.

Давайте откроем пустой сценарий .ps1 и приступим к работе. Начнем с написания всех необходимых команд (листинг 17.2) на основе предыдущего плана.

Листинг 17.2. Пишем код для проверки и создания новых пользователей и групп

```
Get-ADOrganizationalUnit -Filter "Name -eq 'OUName'" ❶
New-ADOrganizationalUnit -Name OUName ❷

Get-ADGroup -Filter "Name -eq 'GroupName'" ❸
New-ADGroup -Name GroupName -GroupScope GroupScope -Path "OU=OUName,DC=powerlab,
DC=local" ❹

Get-ADUser -Filter "Name -eq 'UserName'" ❺
New-ADUser -Name $user.UserName -Path "OU=$($user.OUName),DC=powerlab,DC=local" ❻

$UserName -in (Get-ADGroupMember -Identity GroupName).Name ❻
Add-ADGroupMember -Identity GroupName -Members UserName ❾
```

Из нашего плана следует, что сначала нужно проверить, существует ли OU ❶, и при отсутствии таковой создать ее ❷. Будем делать то же самое с каждой группой: проверять ее наличие ❸ и создавать ее, если она отсутствует ❹. Сделаем то же самое и для каждого пользователя: проверим наличие ❺ и создадим его ❻. У пользователей дополнительно проверяем, входят ли они в группу, имеющуюся в электронной таблице ❻, и по необходимости добавляем их в эту группу ❾.

Не хватает лишь условной структуры, которую мы добавим в листинге 17.3.

Листинг 17.3. Создание пользователей и групп в случае, если их еще нет

```
if (-not (Get-ADOrganizationalUnit -Filter "Name -eq 'OUName'")) {
    New-ADOrganizationalUnit -Name OUName
}

if (-not (Get-ADGroup -Filter "Name -eq 'GroupName'")) {
    New-ADGroup -Name GroupName -GroupScope GroupScope -Path "OU=OUName,
    DC=powerlab,DC=local"
}

if (-not (Get-ADUser -Filter "Name -eq 'UserName'")) {
    New-ADUser -Name $user.UserName -Path "OU=OUName,DC=powerlab,DC=local"
}

if ($UserName -notin (Get-ADGroupMember -Identity GroupName).Name) {
    Add-ADGroupMember -Identity GroupName -Members UserName
}
```

Теперь, когда ваш код делает все, что нужно для отдельного пользователя или группы, вам нужно выяснить, как сделать это для всех. Сначала необходимо

считать рабочие листы. Вы уже видели подходящие команды, осталось лишь сохранить все эти строки в переменных. Технически это не требуется, но код таким образом получается более внятным. Мы будем использовать циклы `foreach` для чтения всех пользователей и групп, как показано в листинге 17.4.

Листинг 17.4. Код, проходящий по каждой строке листа Excel

```
$users = Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Users
$groups = Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Groups

foreach ($group in $groups) {

}

foreach ($user in $users) {
```

Теперь, когда у вас есть структура для обхода каждой строки, давайте добавим в нее алгоритм для каждой из строк, как показано в листинге 17.5.

Листинг 17.5. Выполнение задач для всех пользователей и групп

```
$users = Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Users
$groups = Import-Excel -Path 'C:\Program Files\WindowsPowerShell\Modules\PowerLab\ActiveDirectoryObjects.xlsx' -WorksheetName Groups

foreach ($group in $groups) {
    if (-not (Get-ADOrganizationalUnit -Filter "Name -eq '$($group.OUName)'")) {
        New-ADOrganizationalUnit -Name $group.OUName
    }
    if (-not (Get-ADGroup -Filter "Name -eq '$($group.GroupName)'")) {
        New-ADGroup -Name $group.GroupName -GroupScope $group.Type
        -Path "OU=$($group.OUName),DC=powerlab,DC=local"
    }
}

foreach ($user in $users) {
    if (-not (Get-ADOrganizationalUnit -Filter "Name -eq '$($user.OUName)'")) {
        New-ADOrganizationalUnit -Name $user.OUName
    }
    if (-not (Get-ADUser -Filter "Name -eq '$($user.UserName)'")) {
        New-ADUser -Name $user.UserName -Path "OU=$($user.OUName),DC=powerlab,DC=local"
    }
    if ($user.UserName -notin (Get-ADGroupMember -Identity $user.MemberOf).Name) {
        Add-ADGroupMember -Identity $user.MemberOf -Members $user.UserName
    }
}
```

Почти все! Сценарий готов к работе, но теперь вам нужно запустить его на сервере LABDC. Поскольку мы не будем запускать этот код непосредственно на самой виртуальной машине LABDC, вам нужно запаковать все это в блок сценария и позволить команде `Invoke-Command` запустить его удаленно на LABDC. Поскольку мы хотим создать и заполнить лес за один раз, возьмем весь «рабочий» код и переместим его в функцию `New-PowerLabActiveDirectoryTestObject`. Вы можете загрузить копию этой готовой функции из прилагаемых к этой главе файлов.

Сборка и запуск тестов Pester

Сейчас у нас есть весь код, который нужен для создания нового леса AD и его заполнения. Теперь создадим несколько тестов Pester и убедимся, что все идет по плану. Вам предстоит многое проверить, поэтому здесь тесты Pester будут более сложными, чем прежде. Как и перед созданием сценария `New-PowerLab ActiveDirectoryTestObject.ps1`, сначала создадим сценарий тестирования Pester, а затем подумаем, какими должны быть тестовые примеры. Если вам нужно больше узнать о Pester, перечитайте главу 9. Я также включил все тесты Pester для этой главы в прилагаемые файлы.

Что конкретно нужно протестировать? В этой главе мы сделали следующее:

- Создали новый лес AD.
- Создали новый домен AD.
- Создали пользователей AD.
- Создали группы AD.
- Создали организационные подразделения AD.

После проверки наличия всех этих элементов вам необходимо убедиться, что у них правильные атрибуты (атрибуты, переданные в качестве параметров командам, которые их создали). Вот то, что вам нужно.

Таблица 17.1. Атрибуты AD

Объект	Атрибуты
Лес AD	<code>DomainName</code> , <code>DomainMode</code> , <code>ForestMode</code> , пароль администратора безопасного режима
Пользователь AD	Путь к OU, имя, член группы
Группа AD	Путь к OU, имя
Организационное подразделение AD	Имя