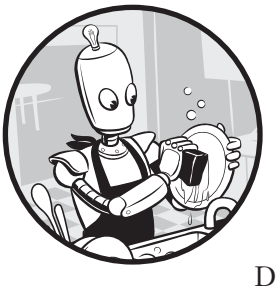
17

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 работающую 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' ❶
>>> DomainMode                    = 'WinThreshold' ❷
>>> ForestMode                    = 'WinThreshold'
>>> Confirm                       = $false ❸
```

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

Здесь мы создаем лес и домен под названием powerlab.local ❶, работающий на функциональном уровне Windows Server 2016 (`WinThreshold`) ❷. Затем мы обходим все подтверждения ❸, передаем пароль администратора безопасного режима ❹ и игнорируем возникающие нерелевантные сообщения с предупреждениями ❺.

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 домена 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 ❼
Add-ADGroupMember -Identity GroupName -Members UserName ❽
```

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 наличие❸ 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 ❼, и по необходимости добавляем их в эту группу ❽.

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	`DomainName`, `DomainMode`, `ForestMode`, пароль администратора безопасного режима
Пользователь AD	Путь к OU, имя, член группы
Группа AD	Путь к OU, имя
Организационное подразделение AD	Имя