PowerShell Groups Tutorial

This tutorial covers how to manage both local groups and Active Directory groups using PowerShell. You'll learn to create, modify, query, and delete groups across different environments.

Prerequisites

- PowerShell 5.1 or later
- Administrative privileges for local group management
- Active Directory module for AD operations (`Install-Module ActiveDirectory`)
- Domain access for Active Directory operations

Part 1: Local Groups Management

Viewing Local Groups

```powershell
# List all local groups
Get-LocalGroup

# Get specific group information
Get-LocalGroup -Name "Administrators"

# List groups with wildcard matching
Get-LocalGroup -Name "*Admin*"
```

Creating Local Groups

```powershell
# Create a basic local group
New-LocalGroup -Name "ProjectTeam" -Description "Members of the special project team"

# Create group with additional parameters
New-LocalGroup -Name "DevOps" -Description "Development Operations Team" -Verbose
```

Managing Local Group Membership

powershell

```
# Add user to local group
Add-LocalGroupMember -Group "ProjectTeam" -Member "DOMAIN\username"
Add-LocalGroupMember -Group "ProjectTeam" -Member "localuser"

# Add multiple users at once
Add-LocalGroupMember -Group "ProjectTeam" -Member @("user1", "user2", "DOMAIN\user3")

# View group members
Get-LocalGroupMember -Group "ProjectTeam"

# Remove user from group
Remove-LocalGroupMember -Group "ProjectTeam" -Member "username"
```

Modifying Local Groups

powershell

```
# Change group description
Set-LocalGroup -Name "ProjectTeam" -Description "Updated project team description"

# Rename a group (requires removing and recreating)
$members = Get-LocalGroupMember -Group "OldGroupName"
New-LocalGroup -Name "NewGroupName" -Description "Renamed group"
$members | ForEach-Object { Add-LocalGroupMember -Group "NewGroupName" -Member $_.Name }
Remove-LocalGroup -Name "OldGroupName"
```

Removing Local Groups

powershell

```
# Remove a local group
Remove-LocalGroup -Name "ProjectTeam"

# Remove with confirmation prompt
Remove-LocalGroup -Name "ProjectTeam" -Confirm
```

Part 2: Active Directory Groups Management

Prerequisites for AD Operations

powershell

```powershell
# Import Active Directory module
Import-Module ActiveDirectory

# Verify connection to domain
Get-ADDomain
```

Viewing AD Groups

powershell

```powershell
# List all AD groups
Get-ADGroup -Filter *

# Get specific group
Get-ADGroup -Identity "Domain Admins"

# Search groups by name pattern
Get-ADGroup -Filter "Name -like '*IT*'"

# Get groups with detailed properties
Get-ADGroup -Filter * -Properties Description, ManagedBy, Members

# Find groups in specific OU
Get-ADGroup -Filter * -SearchBase "OU=Groups,DC=domain,DC=com"
```

Creating AD Groups

powershell

```powershell
# Create basic security group
New-ADGroup -Name "IT-Support" -GroupScope Global -GroupCategory Security -Path "OU=Groups,DC=domain,DC=

# Create distribution group with description
New-ADGroup -Name "Marketing-List" -GroupScope Universal -GroupCategory Distribution -Description "Marketing te

# Create group with additional properties
New-ADGroup -Name "Project-Alpha" `
    -GroupScope DomainLocal `
    -GroupCategory Security `
    -Description "Project Alpha team members" `
    -ManagedBy "DOMAIN\projectmanager" `
    -Path "OU=ProjectGroups,DC=domain,DC=com"
```

Managing AD Group Membership

powershell

```powershell
# Add user to AD group
Add-ADGroupMember -Identity "IT-Support" -Members "username"

# Add multiple users
Add-ADGroupMember -Identity "IT-Support" -Members @("user1", "user2", "user3")

# Add computer to group
Add-ADGroupMember -Identity "Workstations" -Members "COMPUTER01$"

# View group members
Get-ADGroupMember -Identity "IT-Support"

# Get detailed member information
Get-ADGroupMember -Identity "IT-Support" | Get-ADUser -Properties DisplayName, EmailAddress

# Remove user from group
Remove-ADGroupMember -Identity "IT-Support" -Members "username"

# Remove multiple members
Remove-ADGroupMember -Identity "IT-Support" -Members @("user1", "user2")
```

Modifying AD Groups

powershell

```powershell
# Change group properties
Set-ADGroup -Identity "IT-Support" -Description "Updated IT Support team description"
Set-ADGroup -Identity "IT-Support" -ManagedBy "DOMAIN\newmanager"

# Change group scope (with restrictions)
Set-ADGroup -Identity "IT-Support" -GroupScope Universal

# Add custom attributes
Set-ADGroup -Identity "IT-Support" -Add @{info="Custom group information"}
```

Advanced AD Group Operations

powershell

```powershell
# Find groups a user belongs to
Get-ADUser -Identity "username" -Properties MemberOf | Select-Object -ExpandProperty MemberOf

# Find empty groups
Get-ADGroup -Filter * | Where-Object { -not (Get-ADGroupMember -Identity $_.Name) }

# Get nested group membership
function Get-NestedGroupMembership {
    param([string]$GroupName)

    $members = Get-ADGroupMember -Identity $GroupName
    foreach ($member in $members) {
        if ($member.objectClass -eq "group") {
            Write-Host "Nested Group: $($member.Name)"
            Get-NestedGroupMembership -GroupName $member.Name
        } else {
            Write-Host "User: $($member.Name)"
        }
    }
}

# Copy group membership from one user to another
$sourceUser = Get-ADUser -Identity "sourceuser" -Properties MemberOf
$targetUser = "targetuser"

foreach ($group in $sourceUser.MemberOf) {
    Add-ADGroupMember -Identity $group -Members $targetUser
}
```

Removing AD Groups

powershell

```powershell
# Remove AD group
Remove-ADGroup -Identity "IT-Support"

# Remove with confirmation
Remove-ADGroup -Identity "IT-Support" -Confirm:$true

# Remove multiple groups
$groupsToDelete = @("TempGroup1", "TempGroup2", "TempGroup3")
$groupsToDelete | ForEach-Object { Remove-ADGroup -Identity $_ -Confirm:$false }
```

Part 3: Practical Examples and Scripts

Bulk Group Creation

powershell

```
# Create multiple groups from CSV
$groupData = @"
Name,Description,Scope,Category,OU
Finance-Team,Finance Department,Global,Security,OU=Departments,DC=domain,DC=com
HR-Team,Human Resources,Global,Security,OU=Departments,DC=domain,DC=com
IT-Helpdesk,IT Support Team,DomainLocal,Security,OU=IT,DC=domain,DC=com
"@ | ConvertFrom-Csv

foreach ($group in $groupData) {
    New-ADGroup -Name $group.Name -Description $group.Description -GroupScope $group.Scope -GroupCategory $
    Write-Host "Created group: $($group.Name)"
}
```

Group Audit Report

powershell

```
# Generate group membership report
$report = @()

$groups = Get-ADGroup -Filter * -Properties Description, ManagedBy
foreach ($group in $groups) {
    $members = Get-ADGroupMember -Identity $group.Name -ErrorAction SilentlyContinue

    foreach ($member in $members) {
        $report += [PSCustomObject]@{
            GroupName = $group.Name
            GroupDescription = $group.Description
            MemberName = $member.Name
            MemberType = $member.objectClass
            GroupManager = $group.ManagedBy
        }
    }
}

# Export to CSV
$report | Export-Csv -Path "GroupMembershipReport.csv" -NoTypeInformation
```

Cleanup Script for Orphaned Groups

powershell

```powershell
# Find and optionally remove empty groups
$emptyGroups = Get-ADGroup -Filter * | Where-Object {
    -not (Get-ADGroupMember -Identity $_.Name -ErrorAction SilentlyContinue)
}

Write-Host "Found $($emptyGroups.Count) empty groups:"
$emptyGroups | Select-Object Name, Description | Format-Table

# Uncomment to remove (use with caution!)
# $emptyGroups | ForEach-Object { Remove-ADGroup -Identity $_.Name -Confirm:$false }
```

Best Practices

Naming Conventions

- Use consistent prefixes: `DEPT-TeamName`, `ROLE-FunctionName`
- Include purpose: `APP-DatabaseAdmins`, `RES-PrinterAccess`
- Avoid spaces in group names for better scripting compatibility

Security Considerations

- Use principle of least privilege
- Regularly audit group memberships
- Document group purposes and owners
- Implement approval processes for sensitive groups

PowerShell Tips

- Always test scripts in development environment first
- Use `-WhatIf` parameter when available
- Implement error handling with `try-catch` blocks
- Use `-Confirm:$false` carefully in automated scripts

Error Handling Example

```powershell
function Add-UserToGroup {
    param(
        [string]$UserName,
        [string]$GroupName
    )

    try {
        Add-ADGroupMember -Identity $GroupName -Members $UserName -ErrorAction Stop
        Write-Host "Successfully added $UserName to $GroupName" -ForegroundColor Green
    }
    catch {
        Write-Error "Failed to add $UserName to $GroupName`: $($_.Exception.Message)"
    }
}
```

Common Troubleshooting

Permission Issues

```powershell
# Check if running as administrator
if (-NOT ([Security.Principal.WindowsPrincipal] [Security.Principal.WindowsIdentity]::GetCurrent()).IsInRole([Security.Princ
    Write-Warning "This script requires Administrator privileges!"
}
```

Module Not Found

```powershell
# Install Active Directory module
Install-WindowsFeature RSAT-AD-PowerShell
# Or on Windows 10/11
Add-WindowsCapability -Online -Name "Rsat.ActiveDirectory.DS-LDS.Tools~~~~0.0.1.0"
```

This tutorial provides a foundation for managing groups in PowerShell. Practice these commands in a test environment before implementing in production, and always maintain proper backups and documentation.