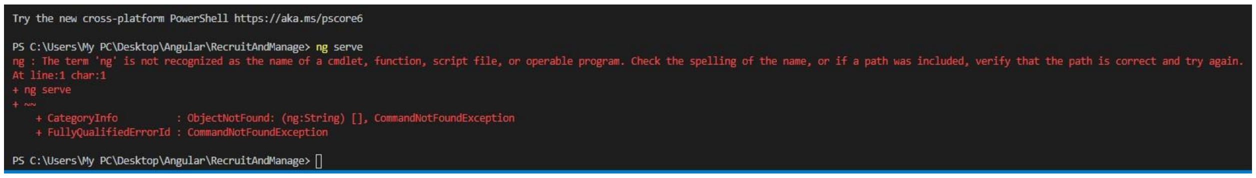
I got this error because I somehow accidentally imported the root `AppModule` into my lazy-loaded module. This caused the root `AppRoutingModule` to be called again when the lazy-loaded module was loaded, and thus `RouterModule.forRoot` was called twice.
If you are certain you aren't calling `RouterModule.forRoot` twice then this could be the cause of the issue. Check that you aren't importing any modules in your lazy-loaded module that direcly call `RouterModule.forRoot` or import a module that calls `RouterModule.forRoot`.

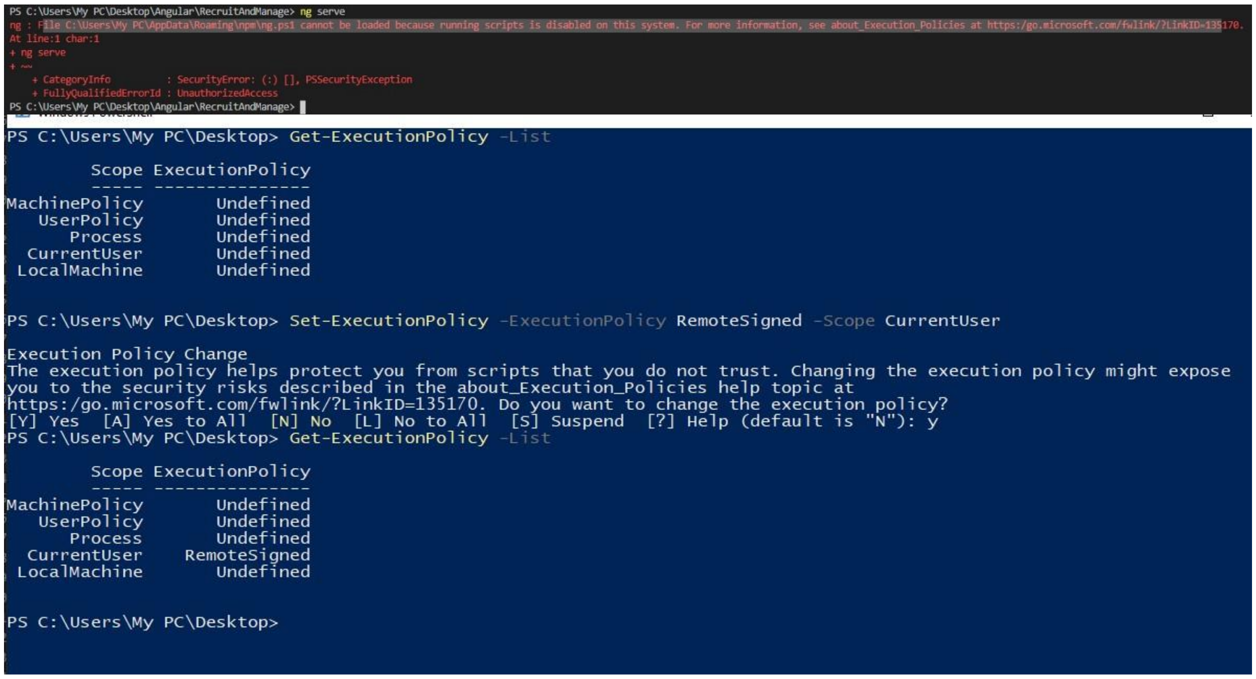


```
Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\My PC\Desktop\Angular\RecruitAndManage> ng serve
ng : The term 'ng' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ ng serve
+ ~~
    + CategoryInfo          : ObjectNotFound: (ng:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\My PC\Desktop\Angular\RecruitAndManage> 
```

Resolve : npm install -g @angular/cli



```
PS C:\Users\My PC\Desktop\Angular\RecruitAndManage> ng serve
ng : File C:\Users\My PC\AppData\Roaming\npm\ng.ps1 cannot be loaded because running scripts is disabled on this system. For more information, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170.
At line:1 char:1
+ ng serve
+ ~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
PS C:\Users\My PC\Desktop\Angular\RecruitAndManage> 
```

```
PS C:\Users\My PC\Desktop> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser       Undefined
 LocalMachine       Undefined


PS C:\Users\My PC\Desktop> Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser

Execution Policy Change
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy might expose
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): y
PS C:\Users\My PC\Desktop> Get-ExecutionPolicy -List

        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser    RemoteSigned
 LocalMachine       Undefined


PS C:\Users\My PC\Desktop>
```

https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.1

Short Description

Describes the PowerShell execution policies and explains how to manage them.

Long Description

PowerShell's execution policy is a safety feature that controls the conditions under which PowerShell loads configuration files and runs scripts. This feature helps prevent the execution of malicious scripts.

On a Windows computer you can set an execution policy for the local computer, for the current user, or for a particular session. You can also use a Group Policy setting to set execution policies for computers and users.

Execution policies for the local computer and current user are stored in the registry. You don't need to set execution policies in your PowerShell profile. The execution policy for a particular session is stored only in memory and is lost when the session is closed.

The execution policy isn't a security system that restricts user actions. For example, users can easily bypass a policy by typing the script contents at the command line when they cannot run a script. Instead, the execution policy helps users to set basic rules and prevents them from violating them unintentionally.

On non-Windows computers, the default execution policy is **Unrestricted** and cannot be changed. The `Set-ExecutionPolicy` cmdlet is available, but PowerShell displays a console message that it's not supported. While `Get-ExecutionPolicy` returns **Unrestricted** on non-Windows platforms, the behavior really matches **Bypass** because those platforms do not implement the Windows Security Zones.

PowerShell execution policies

Enforcement of these policies only occurs on Windows platforms. The PowerShell execution policies are as follows:

AllSigned

- Scripts can run.
- Requires that all scripts and configuration files be signed by a trusted publisher, including scripts that you write on the local computer.
- Prompts you before running scripts from publishers that you haven't yet classified as trusted or untrusted.
- Risks running signed, but malicious, scripts.

Bypass

- Nothing is blocked and there are no warnings or prompts.
- This execution policy is designed for configurations in which a PowerShell script is built in to a larger application or for configurations in which PowerShell is the foundation for a program that has its own security model.

Default

- Sets the default execution policy.
- **Restricted** for Windows clients.
- **RemoteSigned** for Windows servers.

RemoteSigned

- The default execution policy for Windows server computers.
- Scripts can run.
- Requires a digital signature from a trusted publisher on scripts and configuration files that are downloaded from the internet which includes email and instant messaging programs.
- Doesn't require digital signatures on scripts that are written on the local computer and not downloaded from the internet.
- Runs scripts that are downloaded from the internet and not signed, if the scripts are unblocked, such as by using the `Unblock-File` cmdlet.
- Risks running unsigned scripts from sources other than the internet and signed scripts that could be malicious.

Restricted

- The default execution policy for Windows client computers.
- Permits individual commands, but does not allow scripts.
- Prevents running of all script files, including formatting and configuration files (`.ps1xml`), module script files (`.psm1`), and PowerShell profiles (`.ps1`).

Undefined

- There is no execution policy set in the current scope.
- If the execution policy in all scopes is **Undefined**, the effective execution policy is **Restricted** for Windows clients and **RemoteSigned** for Windows Server.

Unrestricted

- The default execution policy for non-Windows computers and cannot be changed.
- Unsigned scripts can run. There is a risk of running malicious scripts.
- Warns the user before running scripts and configuration files that are not from the local intranet zone.

Note

On systems that do not distinguish Universal Naming Convention (UNC) paths from internet paths, scripts that are identified by a UNC path might not be permitted to run with the **RemoteSigned** execution policy.

Execution policy scope

You can set an execution policy that is effective only in a particular scope.

The valid values for **Scope** are **MachinePolicy**, **UserPolicy**, **Process**, **CurrentUser**, and **LocalMachine**. **LocalMachine** is the default when setting an execution policy.

The **Scope** values are listed in precedence order. The policy that takes precedence is effective in the current session, even if a more restrictive policy was set at a lower level of precedence.

For more information, see Set-ExecutionPolicy.

MachinePolicy

Set by a Group Policy for all users of the computer.

UserPolicy

Set by a Group Policy for the current user of the computer.

Process

The **Process** scope only affects the current PowerShell session. The execution policy is saved in the environment variable `$env:PSExecutionPolicyPreference`, rather than the registry. When the PowerShell session is closed, the variable and value are deleted.

CurrentUser

The execution policy affects only the current user. It's stored in the **HKEY_CURRENT_USER** registry subkey.

LocalMachine

The execution policy affects all users on the current computer. It's stored in the **HKEY_LOCAL_MACHINE** registry subkey.

Managing the execution policy with PowerShell

To get the effective execution policy for the current PowerShell session, use the `Get-ExecutionPolicy` cmdlet.

The following command gets the effective execution policy:

PowerShellCopy

```
Get-ExecutionPolicy
```

To get all of the execution policies that affect the current session and display them in precedence order:

PowerShellCopy

```
Get-ExecutionPolicy -List
```

The result looks similar to the following sample output:

OutputCopy

```
        Scope ExecutionPolicy
        ----- ---------------
MachinePolicy       Undefined
   UserPolicy       Undefined
      Process       Undefined
  CurrentUser    RemoteSigned
 LocalMachine       AllSigned
```

In this case, the effective execution policy is **RemoteSigned** because the execution policy for the current user takes precedence over the execution policy set for the local computer.

To get the execution policy set for a particular scope, use the **Scope** parameter of `Get-ExecutionPolicy`.

For example, the following command gets the execution policy for the **CurrentUser** scope:

PowerShellCopy

```
Get-ExecutionPolicy -Scope CurrentUser
```

Change the execution policy

To change the PowerShell execution policy on your Windows computer, use the `Set-ExecutionPolicy` cmdlet. The change is effective immediately. You don't need to restart PowerShell.

If you set the execution policy for the scopes **LocalMachine** or the **CurrentUser**, the change is saved in the registry and remains effective until you change it again.

If you set the execution policy for the **Process** scope, it's not saved in the registry. The execution policy is retained until the current process and any child processes are closed.

Note

In Windows Vista and later versions of Windows, to run commands that change the execution policy for the local computer, **LocalMachine** scope, start PowerShell with the **Run as administrator** option.

To change your execution policy:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy <PolicyName>
```

For example:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

To set the execution policy in a particular scope:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy <PolicyName> -Scope <scope>
```

For example:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

A command to change an execution policy can succeed but still not change the effective execution policy.

For example, a command that sets the execution policy for the local computer can succeed but be overridden by the execution policy for the current user.

Remove the execution policy

To remove the execution policy for a particular scope, set the execution policy to **Undefined**.

For example, to remove the execution policy for all the users of the local computer:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy Undefined -Scope LocalMachine
```

To remove the execution policy for a **Scope**:

PowerShellCopy

```
Set-ExecutionPolicy -ExecutionPolicy Undefined -Scope CurrentUser
```

If no execution policy is set in any scope, the effective execution policy is **Restricted**, which is the default for Windows clients.

Set a different policy for one session

You can use the **ExecutionPolicy** parameter of **pwsh.exe** to set an execution policy for a new PowerShell session. The policy affects only the current session and child sessions.

To set the execution policy for a new session, start PowerShell at the command line, such as **cmd.exe** or from PowerShell, and then use the **ExecutionPolicy** parameter of **pwsh.exe** to set the execution policy.

For example:

PowerShellCopy

```
pwsh.exe -ExecutionPolicy AllSigned
```

The execution policy that you set isn't stored in the registry. Instead, it's stored in the `$env:PSExecutionPolicyPreference` environment variable. The variable is deleted when you close the session in which the policy is set. You cannot change the policy by editing the variable value.

During the session, the execution policy that is set for the session takes precedence over an execution policy that is set in the registry for the local computer or current user. However, it doesn't take precedence over the execution policy set by using a Group Policy.

Use Group Policy to Manage Execution Policy

You can use the **Turn on Script Execution** Group Policy setting to manage the execution policy of computers in your enterprise. The Group Policy setting overrides the execution policies set in PowerShell in all scopes.

The **Turn on Script Execution** policy settings are as follows:

- If you disable **Turn on Script Execution**, scripts do not run. This is equivalent to the **Restricted** execution policy.
- If you enable **Turn on Script Execution**, you can select an execution policy. The Group Policy settings are equivalent to the following execution policy settings:

Group Policy
Allow all scripts
Allow local scripts and remote signed scripts

Group Policy
Allow only signed scripts

- If **Turn on Script Execution** is not configured, it has no effect. The execution policy set in PowerShell is effective.

The PowerShellExecutionPolicy.adm and PowerShellExecutionPolicy.admx files add the **Turn on Script Execution** policy to the Computer Configuration and User Configuration nodes in Group Policy Editor in the following paths.

For Windows XP and Windows Server 2003:

Administrative Templates\Windows Components\Windows PowerShell

For Windows Vista and later versions of Windows:

Administrative Templates\Classic Administrative Templates
Windows Components\Windows PowerShell

Policies set in the Computer Configuration node take precedence over policies set in the User Configuration node.

For more information, see about_Group_Policy_Settings.

Execution policy precedence

When determining the effective execution policy for a session, PowerShell evaluates the execution policies in the following precedence order:

- Group Policy: MachinePolicy
- Group Policy: UserPolicy
- Execution Policy: Process (or `pwsh.exe -ExecutionPolicy`)
- Execution Policy: CurrentUser
- Execution Policy: LocalMachine

Manage signed and unsigned scripts

In Windows, programs like Internet Explorer and Microsoft Edge add an alternate data stream to files that are downloaded. This marks the file as "coming from the Internet". If your PowerShell execution policy is **RemoteSigned**, PowerShell won't run unsigned scripts that are downloaded from the internet which includes email and instant messaging programs.

You can sign the script or elect to run an unsigned script without changing the execution policy.

Beginning in PowerShell 3.0, you can use the **Stream** parameter of the `Get-Item` cmdlet to detect files that are blocked because they were downloaded from the internet. Use the `Unblock-File` cmdlet to unblock the scripts so that you can run them in PowerShell.

For more information, see about_Signing, Get-Item, and Unblock-File.

Note

Other methods of downloading files may not mark the files as coming from the Internet Zone. Some examples include:

- `curl.exe`
- `Invoke-RestMethod`
- `Invoke-WebRequest`

Execution policy on Windows Server Core and Window Nano Server

When PowerShell 6 is run on Windows Server Core or Windows Nano Server under certain conditions, execution policies can fail with the following error:

OutputCopy

```
AuthorizationManager check failed.
At line:1 char:1
+ C:\scriptpath\scriptname.ps1
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : SecurityError: (:) [], PSSecurityException
    + FullyQualifiedErrorId : UnauthorizedAccess
```

PowerShell uses APIs in the Windows Desktop Shell (`explorer.exe`) to validate the Zone of a script file. The Windows Shell is not available on Windows Server Core and Windows Nano Server.

You could also get this error on any Windows system if the Windows Desktop Shell is unavailable or unresponsive. For example, during sign on, a PowerShell logon script could start execution before the Windows Desktop is ready, resulting in failure.

Using an execution policy of **ByPass** or **AllSigned** does not require a Zone check which avoids the problem.

See Also

about_Environment_Variables

about_Group_Policy_Settings

about_Signing

Get-ExecutionPolicy

Get-Item

Pwsh Console Help

Set-ExecutionPolicy

Unblock-File