PowerShell Array Contains Explained

lazyadmin.nl/powershell/powershell-array-contains

October 16, 2024

Arrays allow you to store multiple values in one variable. This way you can easily loop through each value and process it in your script. But sometimes you need to check if the PowerShell array contains a specific value.

To check if a value exists in an array, we can use the `-Contains` operator in PowerShell. This will return return true if the value exists. But there are other options as well.

In this article, I will show how you can check if a **PowerShell array contains** a specific value and explain the different methods.

PowerShell Array Contains

The recommended way to check if an array contains a specific value is by using the `-contains` operator or the `.Contains()` method in PowerShell. Even though both look similar, there are some important differences between the two.

First, let's take a quick look at how to use the `-contains` operator. To simply check if a string is in an array you can do the following:

```
$array = @('apple', 'banana', 'cherry')
if ($array -contains 'banana') {
Write-Host "Array contains banana"
}else{
Write-Host "Banana not found in the array"
}
```

The method above will simply check if the string exists in the array. The comparison is case insensitive, and the operator will stop when the results are found. So when an item, a fruit in the example above, is listed multiple times in the array, then it will still just return True or False.

So there are a couple of other options when it comes to checking if a value is listed in an array. Besides the **.Contains()** method, we can also use:

- **-in operator** – checks if a value is in an array, just another way of writing it.
- **-ccontains operator** – case-sensitive comparison
- **-eq** – Can also check if an array contains a specific value
- **Where-Object** – Allows you to do more complex comparisons

Make sure that you read the PowerShell operators article as well if you want to know more about the different options.

Contains Operator vs Contains Method

So I want to first explain the difference between the contains operator and the contains method. Because they look really similar, but there is more to it.

The Contains method, written `.contains()`, is a method from the underlying .NET framework, whereas the operator is specific to PowerShell. Now the .NET method is faster, so when you need to look up a value in a large array, then there is a real benefit there.

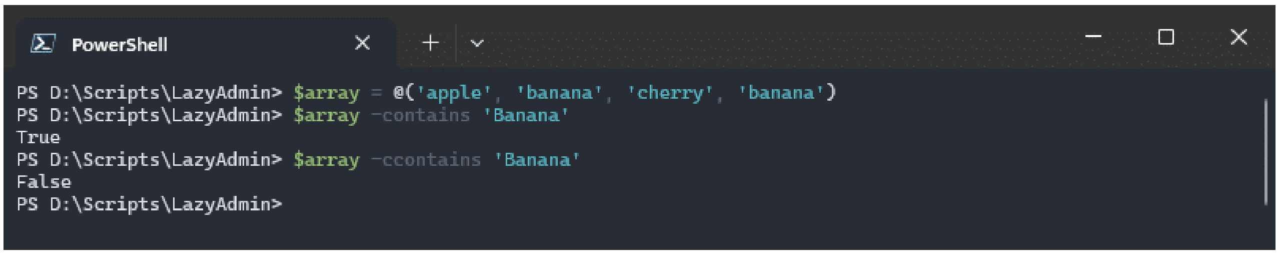
Another important difference between the two is that the contains method is case-sensitive by default, whereas the operator is not:

```
$array = @('apple', 'banana', 'cherry', 'banana')
$array -contains 'Banana' # Returns true
$array.Contains('Banana') # Returns false
```

Case-Sensitive Contains

As mentioned, the contains operator is case-in-sensitive, which is often perfectly fine. But when you can also do an exact case value match, by using the `-ccontains` operator (note the double c).

```
$array = @('apple', 'banana', 'cherry', 'banana')
# Case sensitive comparison
$array -ccontains 'Banana' # Returns false
```



```
PS D:\Scripts\LazyAdmin> $array = @('apple', 'banana', 'cherry', 'banana')
PS D:\Scripts\LazyAdmin> $array -contains 'Banana'
True
PS D:\Scripts\LazyAdmin> $array -ccontains 'Banana'
False
PS D:\Scripts\LazyAdmin>
```

Array Not Contains

Instead of checking if an array contains a specific value, we can also check if an array does not contains a specified value. To do this, we can use the `-notcontains` operator. The operator will return True if the value is not listed in the array.

For example, we have a list of users who have completed a security training, and we want to verify if a particular user has **not** completed it yet. In this case, `-notcontains` is a more logical option to use:

```
# List of users
$completedTraining = @('John', 'Alice', 'Robert', 'Sarah')
# Check if the user has not completed the training
if ($completedTraining -notcontains 'Jane'){
Write-Host "Jane didn't completed the training"
}
```

Using the -In Operator

The `-in` operator is the same as the contains operator, only reversed. It also returns true when the value is in the array, the only difference is that you place the array on the right of the operator, instead of the left.

You should use the -in operator when it's more intuitive and concise for the situation. For example, if we want to check if a user is a member of an Admin group, then it makes more sense to check if "Jane" is in the admin group, than to check if the admin group contains "Jane":

```
# List of admin users
$adminUsers = @('John', 'Alice', 'Robert')
# Using the -in operator
if ('Alice' -in $adminUsers) {
Write-Output "Alice is an admin."
}
```

The key here is to make your code easier to read and understand. You can also use the `-notin` operator, which checks if a value is not in the array.

Where can I send the Free PowerShell cheat sheet to?

Не удается связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.

Equal Operator

You might not expect it, but we can also use the -eq operator to check if an array contains a specific value in PowerShell. The big difference with the `-contains` operator, however, the equal operator does not return true or false, but it returns the value.

And, even more importantly, it does not stop searching for the value when the value is found. This means that we can also count or check if a value is listed multiple times in an array:

```
$array = @('apple', 'banana', 'cherry', 'banana')
$result = $array -eq "banana"
if ($result) {
```

```
Write-Host "Banana is listed $($result.Length) time(s) in the array"
}
# Result
Banana is listed 2 time(s) in the array
```

Using Where-Object

The last option that I want to show you is the `Where-Object` method to check if an array contains a specific value. This method is mainly used when you have an array of objects or need to do other complex comparisons.

For example, we have an array with user objects, and want to check if the array contains a specific user:

```
$users = @(
[PSCustomObject]@{ Name = 'John'; Age = 30 },
[PSCustomObject]@{ Name = 'Jane'; Age = 25 },
[PSCustomObject]@{ Name = 'Alice'; Age = 28 }
)
$result = $users | Where-Object { $_.Name -contains 'Jane'}
```

Wrapping Up

The most common way to check if an array contains a specific value is by using the `-contains` operator. Keep in mind though, that this is case-in-sensitive, whereas the `.contains()` method is case-sensitive.

Try to use the appropriate operator for the situation, this will make your code easier to read and understand.

Hope you liked this article, if you have any questions, just drop a comment below.

Did you **Liked** this **Article**?
Get the latest articles like this **in your mailbox**
or share this article

Не удается связаться с сервисом reCAPTCHA. Проверьте подключение к Интернету и перезагрузите страницу.

I hate spam to, so you can unsubscribe at any time.