# How to use PowerShell String Contains

lazyadmin.nl/powershell/string-contains

May 19, 2022

The Contains operator in PowerShell is a bit of a strange one. You probably want to check if a string contains a particular word or character. The `-contains` operator sounds the most logical one to use, but is it?

In PowerShell, the contains operator, doesn't do substring comparison. But instead, it's a collection operator. This means that it will test if a collection contains the matching value. It can't be used to find a character or substring inside a string. A better solution for this is to use the `-like` operator or the `.contains()` function

In this article, I will explain how you can check if a string contains a character and explain the differences between the different operators.

## PowerShell String Contains

When you want to test if a string contains a word you probably tried to use the `-contains` operator like this:

```
$string = "how to find a word"
if ($string -contains "find") {
# Do something
}
# Result
False
```

But this won't work as you may have noticed. This is because `-contains` will check if the $string object matches the test element. If you would test it on the complete string, contains will return true:

```
$string -contains "how to find a word" # True
```

So when you want to search for a substring or word inside a string with PowerShell it's better to use the `-like` operator. With -like you will need to use wildcards around your search string:

```
if ($string -like "*find*") {
Write-host "String contains find"
}
# Result
String contains find
```

There is however a way to use contains in PowerShell to find a word or character inside a string. Instead of the operator, we can use the built-in .NET function `.Contains()` for strings. This string function can actually check if the string contains a word:

```
if ($string.Contains("find")) {
Write-host "String contains find"
}
# Result
String contains find
```

Keep in mind that the Contains function is case sensitive, so to do a case insensitive comparison you will first need to convert the string to lower case:

```
if ($string.ToLower().Contains("find")) {
Write-host "String contains find"
}
```



## Using -Contains and -Notcontains on Objects and Arrays

The contains operator is meant to test if an object or array contains the given value. So for example, if we have an array with fruits, we can check if the array contains a pear:

```
# Array with fruits
$fruits = "apple","banana","pear"
# Check if the array contains pear
$fruits -contains "pear"
# Return
True
```

We can only search on the whole values, this means that we can't use wildcards as with the Like operator. If we try to search on only the first few letters of apple you will see that it returns false:

```
$fruits -contains "*app*"
# Result
False
```

If you want to check if a set doesn't contain an element you can use the `-notcontains` operator. So for example, if you want to make sure that the fruits array doesn't contain raspberry we can do the following:

```
$fruits -notcontains "raspberry"
# Returns
True
```

### -in and -notin operators

The `-in` and `-notin` operators in PowerShell are basically the same as the `-contains` and `-notcontains` operators. The difference is that they just work the other way around. With -contains the test object is on the right-hand side of the operator. With `-in`, it's on the left-hand side:

```
$fruits = "apple","banana","pear"
"apple" -in $fruits # True
"raspberry -notin $fruits # True
# Similar to
$fruits -contains "apple" # True
$fruits -notcontains "raspberry # True
```

Using the in operator instead of contains can improve the readability of your code. But besides that, there is no advantage in using it compared to contain.

## Wrapping Up

If you want to know in PowerShell if a string contains a particular string or word then you will need to use the `-like` operator or the `.contains()` function. The contains operator can only be used on objects or arrays just like its syntactic counterpart `-in` and `-notin`.

I hope this article helped with understanding the differences between the different operators and functions. Learn more about PowerShell scripting in this complete guide. If you have any questions or tips, just drop a comment below.