

URL slug

Lowercase URL slug — letters, digits, hyphens.

```
^[a-z0-9]+(?:-[a-z0-9]+)*$
```

Token-by-token breakdown

Token	Meaning
^	start of string (or line in multiline mode)
[a-z0-9]+	one or more: any of: lowercase letters, digits
(?:	start of a non-capturing group
-	literal text “-”
[a-z0-9]+	one or more: any of: lowercase letters, digits
)*	end of group, zero or more
\$	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	<pre>const re = /^[a-z0-9]+(?:-[a-z0-9]+)*\$/; re.test(value);</pre>
Python	<pre>import re re.match(r"^[a-z0-9]+(?:-[a-z0-9]+)*\$", value)</pre>
Java	<pre>Pattern.compile("^[a-z0-9]+(?:-[a-z0-9]+)*\$").matcher(value).matches();</pre>
C# / .NET	<pre>Regex.IsMatch(value, @"^[a-z0-9]+(?:-[a-z0-9]+)*\$");</pre>
Go	<pre>regexp.MustCompile(`^[a-z0-9]+(?:-[a-z0-9]+)*\$`).MatchString(value)</pre>
Ruby	<pre>/^[a-z0-9]+(?:-[a-z0-9]+)*\$/ .match?(value)</pre>
PHP	<pre>preg_match('~^[a-z0-9]+(?:-[a-z0-9]+)*\$~', \$value);</pre>

Common pitfalls

- **Anchored to the whole string.** This pattern uses ^ and \$, so it requires the entire input to match. To find it inside a longer text, drop the anchors and use the global (g) flag.
- **Lowercase only.** Ranges like [a-z] won't match uppercase. Add A-Z (or the i flag) if uppercase input should be accepted.
- **Watch for backtracking.** This pattern nests quantifiers; on adversarial input that can cause exponential backtracking (ReDoS). Test with long non-matching strings, or use an RE2-based engine.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the url slug against a source of truth (database, API, or checksum) where it matters.