Semantic version

Semver version like 1.2.3 with pre-release and build metadata.

```
^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$
```

Token-by-token breakdown

Token	Meaning
`^`	start of string (or line in multiline mode)
`(`	start of a capturing group
`0`	literal text “0”
`|`	OR — try the alternative
`[1-9]`	any of: 1–9
`\d*`	zero or more: digit (0–9)
`)`	end of group
`\.`	literal “.”
`(`	start of a capturing group
`0`	literal text “0”
`|`	OR — try the alternative
`[1-9]`	any of: 1–9
`\d*`	zero or more: digit (0–9)
`)`	end of group
`\.`	literal “.”
`(`	start of a capturing group
`0`	literal text “0”
`|`	OR — try the alternative
`[1-9]`	any of: 1–9
`\d*`	zero or more: digit (0–9)
`)`	end of group
`(?:`	start of a non-capturing group
`-`	literal text “-”
`(`	start of a capturing group
`(?:`	start of a non-capturing group
`0`	literal text “0”
`|`	OR — try the alternative

Token	Meaning
`[1-9]`	any of: 1–9
`\d*`	zero or more: digit (0–9)
`|`	OR — try the alternative
`\d*`	zero or more: digit (0–9)
`[a-zA-Z-]`	any of: lowercase letters, uppercase letters, “-”
`[0-9a-zA-Z-]*`	zero or more: any of: digits, lowercase letters, uppercase letters, “-”
`)`	end of group
`(?:`	start of a non-capturing group
`\.`	literal “.”
`(?:`	start of a non-capturing group
`0`	literal text “0”
`|`	OR — try the alternative
`[1-9]`	any of: 1–9
`\d*`	zero or more: digit (0–9)
`|`	OR — try the alternative
`\d*`	zero or more: digit (0–9)
`[a-zA-Z-]`	any of: lowercase letters, uppercase letters, “-”
`[0-9a-zA-Z-]*`	zero or more: any of: digits, lowercase letters, uppercase letters, “-”
`)`	end of group
`)*`	end of group, zero or more
`)`	end of group
`)?`	end of group, optional (zero or one)
`(?:`	start of a non-capturing group
`\+`	literal “+”
`(`	start of a capturing group
`[0-9a-zA-Z-]+`	one or more: any of: digits, lowercase letters, uppercase letters, “-”
`(?:`	start of a non-capturing group
`\.`	literal “.”
`[0-9a-zA-Z-]+`	one or more: any of: digits, lowercase letters, uppercase letters, “-”
`)*`	end of group, zero or more
`)`	end of group
`)?`	end of group, optional (zero or one)
`$`	end of string (or line in multiline mode)

Quick usage in 7 languages

JavaScript	`const re = /^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$/;` `re.test(value);`
Python	`import re` `re.match(r"^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$", value)`
Java	`Pattern.compile("^(0|[1-9]\\d*)\\.(0|[1-9]\\d*)\\.(0|[1-9]\\d*)(?:-((?:0|[1-9]\\d*|\\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\\.(?:0|[1-9]\\d*|\\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\\+([0-9a-zA-Z-]+(?:\\.[0-9a-zA-Z-]+)*))?$").matcher(value).matches();`
C# / .NET	`Regex.IsMatch(value, @"^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$");`
Go	`regexp.MustCompile(`^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$`).MatchString(value)`
Ruby	`/^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$/.match?(value)`
PHP	`preg_match('~^(0|[1-9]\d*)\.(0|[1-9]\d*)\.(0|[1-9]\d*)(?:-((?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*)(?:\.(?:0|[1-9]\d*|\d*[a-zA-Z-][0-9a-zA-Z-]*))*))?(?:\+([0-9a-zA-Z-]+(?:\.[0-9a-zA-Z-]+)*))?$~', $value);`

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.
- **ASCII letters only.** [a-zA-Z] excludes accented and non-Latin letters (é, ü, ß, ñ, and non-Latin scripts). For international input use Unicode properties like \p{L} with the u flag.
- **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.
- **Escape it correctly per language.** In Java and JavaScript strings each backslash must be doubled (\\d); in Python, Go, and C# use raw/verbatim strings so the backslashes survive.
- **Validate beyond format.** Matching the format doesn't guarantee the value is real. Confirm the semantic version against a source of truth (database, API, or checksum) where it matters.