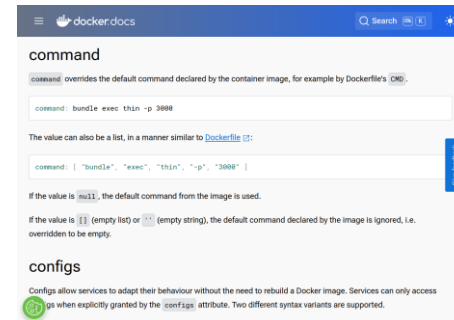


@ Raccoon Tech Connect #4 | 3分で語りつくせ！LTラッシュ！



1

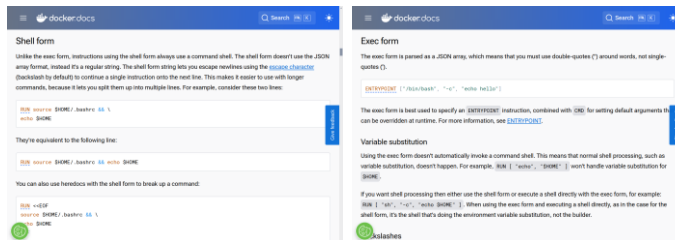
compose.yaml の command



<https://docs.docker.com/compose/compose-file/05-services/#command>

2

Shell and exec form



<https://docs.docker.com/reference/dockerfile/#shell-and-exec-form>

3

List style

compose.yaml	entrypoint.sh	output
<pre> services: bar: image: alpine entrypoint: /entrypoint.sh command: - --opt1 - foo - --opt2 - bar - --opt3 - baz qux volumes: - ./entrypoint.sh:/entrypoint.sh </pre>	<pre> #!/bin/sh echo "\$0" echo "\$1" echo "\$2" echo "\$3" echo "\$4" echo "\$5" echo "\$6" echo "\$7" echo "\$8" echo "\$9" </pre>	<pre> foo-bar-1 /entrypoint.sh foo-bar-1 --opt1 foo-bar-1 foo foo-bar-1 --opt2 foo-bar-1 bar foo-bar-1 --opt3 foo-bar-1 baz qux foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 exited with code 0 </pre>

4

String style

compose.yaml	entrypoint.sh	output
<pre>services: bar: image: alpine entrypoint: /entrypoint.sh command: --opt1 foo --opt2 bar --opt3 "baz qux" volumes: - ./entrypoint.sh:/entrypoint.sh</pre>	<pre>#!/bin/sh echo "\$0" echo "\$1" echo "\$2" echo "\$3" echo "\$4" echo "\$5" echo "\$6" echo "\$7" echo "\$8" echo "\$9"</pre>	<pre>foo-bar-1 /entrypoint.sh foo-bar-1 --opt1 foo-bar-1 foo foo-bar-1 --opt2 foo-bar-1 bar foo-bar-1 --opt3 foo-bar-1 baz qux foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 exited with code 0</pre>



5

String style (literal style)

compose.yaml	entrypoint.sh	output
<pre>services: bar: image: alpine entrypoint: /entrypoint.sh command: --opt1 foo --opt2 bar --opt3 "baz qux" volumes: - ./entrypoint.sh:/entrypoint.sh</pre>	<pre>#!/bin/sh echo "\$0" echo "\$1" echo "\$2" echo "\$3" echo "\$4" echo "\$5" echo "\$6" echo "\$7" echo "\$8" echo "\$9"</pre>	<pre>foo-bar-1 /entrypoint.sh foo-bar-1 --opt1 foo-bar-1 foo foo-bar-1 --opt2 foo-bar-1 bar foo-bar-1 --opt3 foo-bar-1 baz qux foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 foo-bar-1 exited with code 0</pre>



6

compose.yaml そのものの解析

- yaml そのものの仕様
<https://yaml.org/spec/1.2.2/>

- compose.yaml の解析

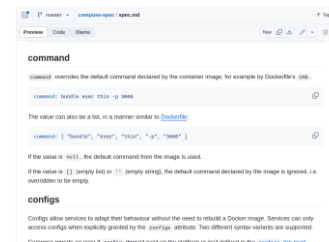
List style	String style	String style (literal style)
["--opt1", "foo", "--opt2", "bar", "--opt3", "baz qux"]	"--opt1 foo --opt2 bar --opt3 baz qux"	"--opt1 foo\n--opt2 bar\n--opt3 baz qux"

- String style (literal style) の場合は、文字列に改行が含まれている。その改行は、どのように扱われている？



7

改行文字はどのように扱われている？



- compose-spec/compose-spec (commit c35d193) に定義の仕様は、Docker Compose のドキュメントと同じ

<https://github.com/compose-spec/compose-spec/blob/c35d19323dfd7dbf9db30022498e29ef5eabeab2/spec.md#command>



8

改行文字はどのように扱われている？



```
17 package types
18
19 import "github.com/matttn/go-shellwords"
20
21 // ShellCommand is a string or list of string args.
22 //
23 // When normalized to YAML, all command strings will be treated as "strings".
24 // In YAML, a string can be a single line, or a multi-line string (e.g., "1", "2", "3").
25 // Empty string will be treated as an empty array ([""]).
26 //
27 // When normalized to JSON, the "strings" array must not be specified.
28 // If the command string is not, it will be specified as "null".
29 // Empty string will be treated as an empty array ([""]).
30 //
31 // The distinction between all and explicitly empty is important for Kubernetes.
32 // Because of some rules and a problem, the empty value should be
33 // preserved so that it can override any base value (e.g., container.restartPolicy).
34 //
35 // The difference between YAML and JSON is that in YAML, the
36 // "command" string is treated as the "command" string, which is not in JSON.
37 // In JSON, the "command" string is treated as the "command" string, which is not in JSON.
38 //
39 // In the future, it might make sense to use "command" of this type to
40 // "command" to avoid this situation, but that would require a
41 // breaking change.
42
43 type ShellCommand []string
```

- compose-spec/compose-go (v2.0.2) は、matttn/go-shellwords を利用

 <https://github.com/compose-spec/compose-go/blob/v2.0.2/types/command.go>

9

改行文字はどのように扱われている？



```
17 // Parse parses a string or list of string args.
18 //
19 // When normalized to YAML, all command strings will be treated as "strings".
20 // In YAML, a string can be a single line, or a multi-line string (e.g., "1", "2", "3").
21 // Empty string will be treated as an empty array ([""]).
22 //
23 // When normalized to JSON, the "strings" array must not be specified.
24 // If the command string is not, it will be specified as "null".
25 // Empty string will be treated as an empty array ([""]).
26 //
27 // The distinction between all and explicitly empty is important for Kubernetes.
28 // Because of some rules and a problem, the empty value should be
29 // preserved so that it can override any base value (e.g., container.restartPolicy).
30 //
31 // The difference between YAML and JSON is that in YAML, the
32 // "command" string is treated as the "command" string, which is not in JSON.
33 // In JSON, the "command" string is treated as the "command" string, which is not in JSON.
34 //
35 // In the future, it might make sense to use "command" of this type to
36 // "command" to avoid this situation, but that would require a
37 // breaking change.
38
39 type Shellwords []string
```

- matttn/go-shellwords (v1.0.12) は、改行文字を空白文字扱いする

 <https://github.com/matttn/go-shellwords/blob/v1.0.12/shellwords.go>

10

まとめ

- compose.yaml の command で literal style を使ったときの話
 - yaml に含まれる改行文字は、よしに空白文字扱いされるため、思った通りに動く
 - ただし、仕様ではなく（たぶん）、実装依存
この書き方はやめたほうがいいかも