

String Matching

Matt McQuillan

Hello 🖐️

Matt McQuillan

- Work at HashiCorp*
- Dev > DB > DevOps > SRE > Mgr
- At this for 20+ years
- Not a great developer
- I ❤️ Golang & CLI's
- Twitter: @automatedmatt

* see me for jobs or stickers

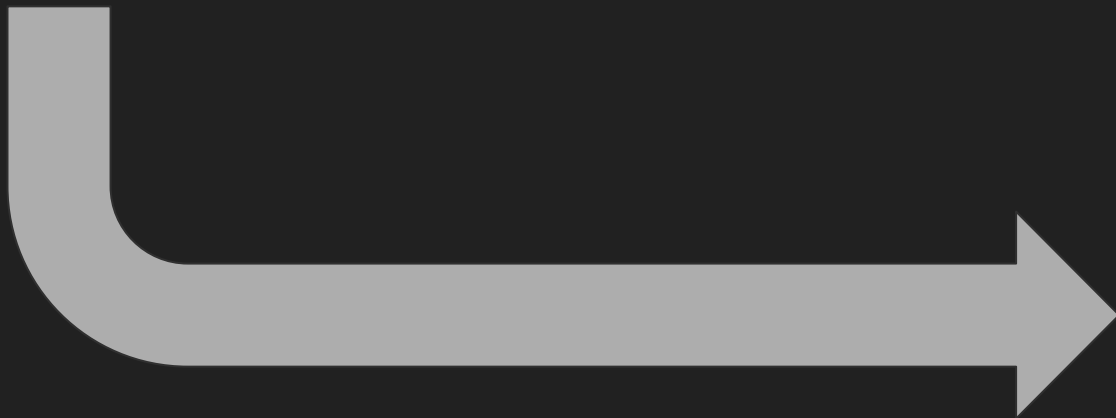
Summary of Technical Experience

Terrible amounts of string manipulation.

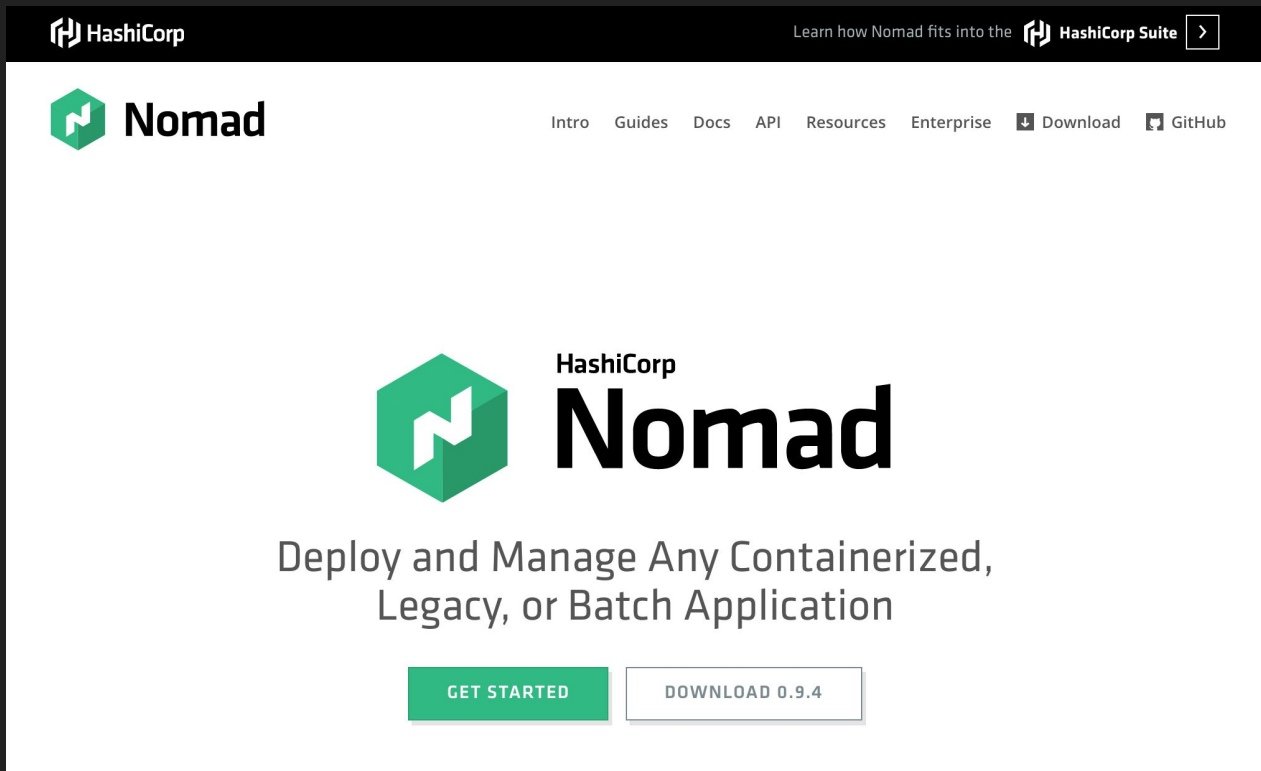
Example One

Fun Project at Work

Deploy to...



What is Nomad?



The screenshot shows the HashiCorp Nomad website. At the top, there is a dark navigation bar with the HashiCorp logo on the left and the text "Learn how Nomad fits into the HashiCorp Suite" with a right-pointing arrow on the right. Below this, the main header features the Nomad logo (a green hexagon with a white 'N') and the word "Nomad" in a large, bold, black font. To the right of the logo, a horizontal menu contains links for "Intro", "Guides", "Docs", "API", "Resources", "Enterprise", "Download" (with a download icon), and "GitHub" (with a GitHub icon). The main content area has a large, centered graphic of the Nomad logo and the word "Nomad" in a large, bold, black font. Below this, the tagline "Deploy and Manage Any Containerized, Legacy, or Batch Application" is displayed in a smaller, gray font. At the bottom, there are two buttons: a green "GET STARTED" button and a white "DOWNLOAD 0.9.4" button with a gray border.

HashiCorp

Learn how Nomad fits into the HashiCorp Suite >

Nomad

Intro Guides Docs API Resources Enterprise Download GitHub

HashiCorp
Nomad

Deploy and Manage Any Containerized,
Legacy, or Batch Application

GET STARTED

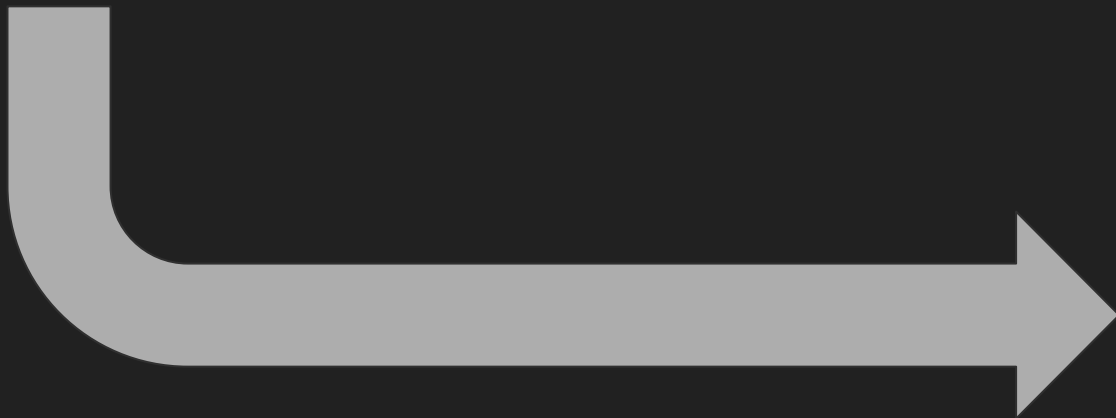
DOWNLOAD 0.9.4

First Rule of Orchestration

The deployment story is never great.

Fun Project at Work

Deploy to...



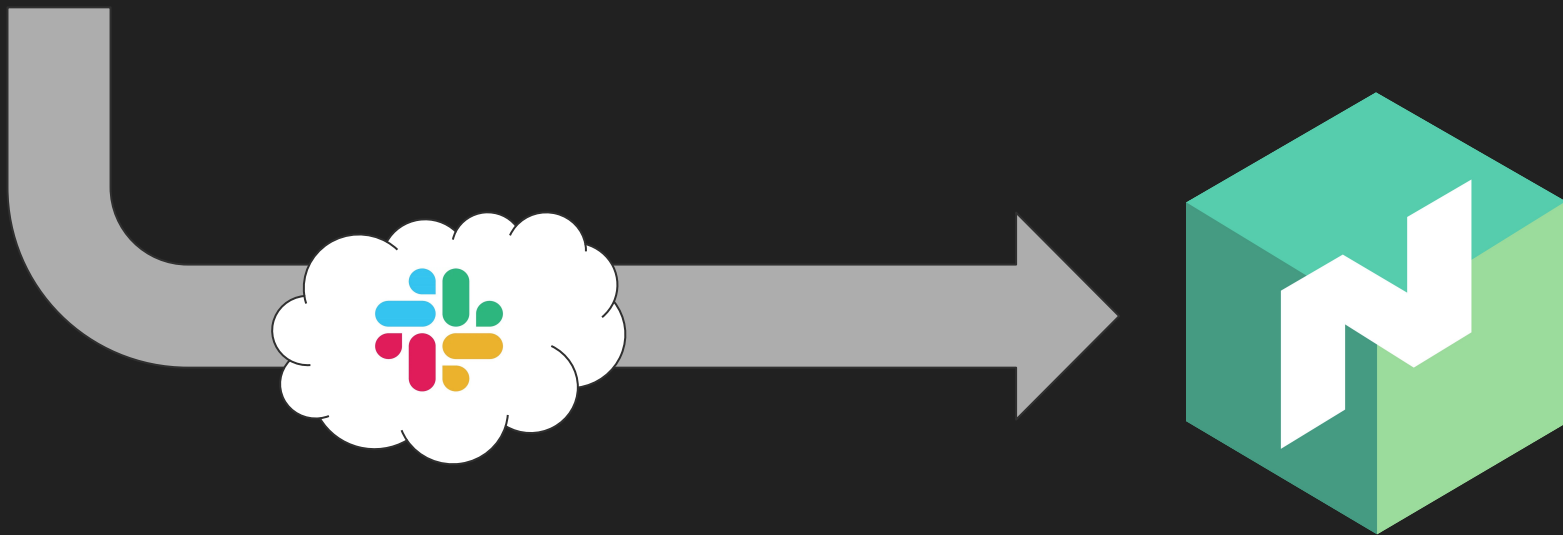
Fun Project at Work

Deploy to...



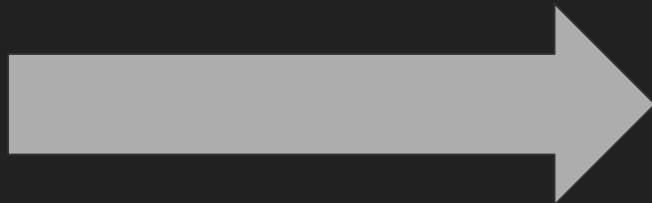
Fun Project at Work

Deploy to...



Nomad Deployment Bot

Waffles



Waffles demo



Matcher

GitHub, Inc. [US] | github.com/mmcquillan/matcher

Matcher

A library for parsing and matching based on a mask for use with CLI or Bots.

Mask Rules

- `xyz` text xyz
- `<xyz>` required var named xyz
- `[xyz]` optional var named xyz
- `[xyz...]` optional var named xyz and captures remaining input
- `[xyz(string:foo,bar)]` optional var named xyz that can only be foo or bar
- `<-xyz>` required short flag named xyz
- `[-xyz]` optional short flag named xyz
- `[-]` capture any short flag
- `<--xyz>` required long flag named xyz
- `[--xyz]` optional long flag named xyz
- `[--]` capture any long flag

Avoiding a Switch/If Control Flow

```
75 // match messages
76 match := false
77 for command := range commands.CommandList {
78     input := strings.Replace(msg.Attributes["waffles.input"], options.Name, "", 1)
79     options.Logger.Trace("Matcher - Command: " + command)
80     options.Logger.Trace("Matcher - Input: " + input)
81     isParsed, cmd, args := matcher.Matcher(command+global, input)
82     if isParsed {
83
84         // parse bot debug
85         if arg, chk := args["debug"]; chk && arg == "true" {
86             msg.Debug = true
87         }
88
89         // parse bot politeness
90         if arg, chk := args["please"]; chk && arg == "true" {
91             msg.Polite = true
92         }
93         if arg, chk := args["thanks"]; chk && arg == "true" {
94             msg.Polite = true
95         }
96
97         // check command
98         commandService := commands.Make(command).(commands.Command)
99         if commandService != nil {
100             match = true
101             msg.Attributes["waffles.command"] = cmd
102             for key, val := range args {
103                 msg.Attributes["waffles.arg."+key] = val
104             }
105             msg.MessageOutput(outputMsgs)
106             go commandService.Run(&msg, state, outputMsgs, options)
107         }
108     }
109 }
```

Example Command

```
33 // internal
34 CommandList["help [filter]"] = CommandItem{
35     Help:    false,
36     Private: true,
37     Command: reflect.TypeOf(Help{}),
38 }
39 CommandList["options"] = CommandItem{
40     Help:    false,
41     Private: false,
42     Command: reflect.TypeOf(Options{}),
43 }
44 CommandList["ping"] = CommandItem{
45     Help:    false,
46     Private: false,
47     Command: reflect.TypeOf(Ping{}),
48 }
49 CommandList["state [job] [--clear]"] = CommandItem{
50     Help:    false,
51     Private: false,
52     Command: reflect.TypeOf(State{}),
53 }
54 CommandList["upgrade <job> [version]"] = CommandItem{
55     Help:    false,
56     Private: false,
57     Command: reflect.TypeOf(Upgrade{}),
58 }
59 CommandList["version"] = CommandItem{
60     Help:    false,
61     Private: false,
62     Command: reflect.TypeOf(Version{}),
63 }
64 CommandList["whoami"] = CommandItem{
65     Help:    false,
66     Private: false,
67     Command: reflect.TypeOf(Whoami{}),
68 }
```

Matcher

Mask: run <speed> [distance] [--jump]

Input: run fast far --jump=high

Tokenize: run fast far --jump=high

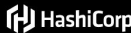
Parser: run fast far --jump=high
 arg[0] arg[1] arg[2] var[jump]


Masker: run <speed> [distance] [--jump]
 run fast far --jump=high


Matcher: true / cmd=run / var[speed]=fast
 var[distance]=far
 var[jump]=high

Example Two

Prototyping CLI's

 HashiCorp

Learn how Consul fits into the  HashiCorp Suite >

 Consul

[Use Cases](#) [Intro](#) [Guides](#) [Docs](#) [API](#) [Community](#) [Enterprise](#) [Download](#) [GitHub](#)

Easy Service Networking

Consul is a service networking solution to connect and secure services across any runtime platform and public or private cloud

[Download](#) [Get Started](#)

CLI

UI

```
$ consul kv get config/api/request_limit
1500
$ consul kv put config/api/request_limit 2000
Success! Data written to: config/api/request_limit
$ consul kv get config/api/request_limit
```

protocli

GitHub, Inc. [US] | github.com/mmcquillan/protocli

protocli

This is a CLI prototyping tool using the [Matcher Format](#) with response variable substitution.

Install

On the mac, you can install via brew as:

```
brew install mmcquillan/tools/protocli
```

Launch by passing in a config file:

```
protocli <config>
```

Commands

The only native command inside protocli is `?` which lists all possible command matches.

Config

The configuration is a YAML based file. Examples can be found [here](#).

```
---
prompt: "> "
commands:
-
  command: "do version"
  response: "0.2.0"
```

protocli demo



Thank You!

Matt McQuillan
matt@hashicorp.com

<https://github.com/mmcquillan/matcher>

<https://github.com/mmcquillan/protocli>