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

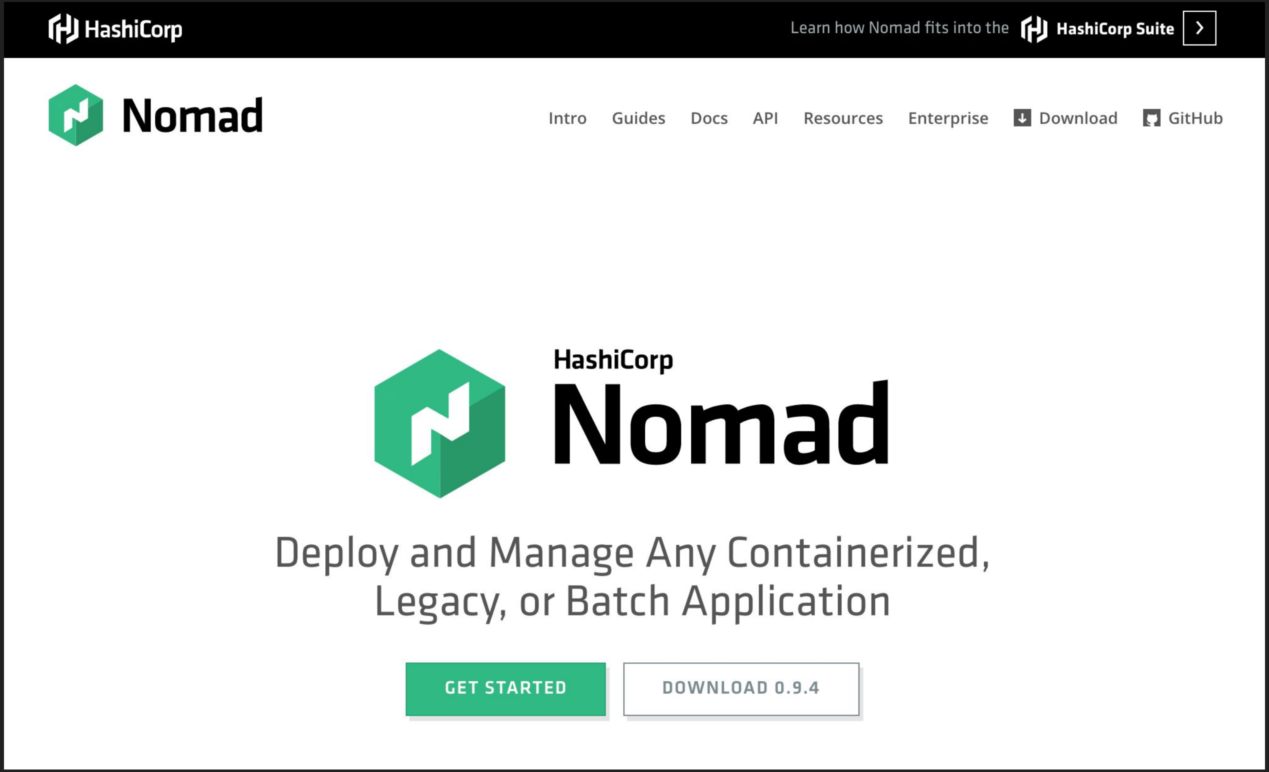
Terrible amounts of string manipulation.

Example One

Fun Project at Work

Deploy to...

What is Nomad?



Learn how Nomad fits into the HashiCorp Suite

Intro Guides Docs API Resources Enterprise Download GitHub

HashiCorp Nomad

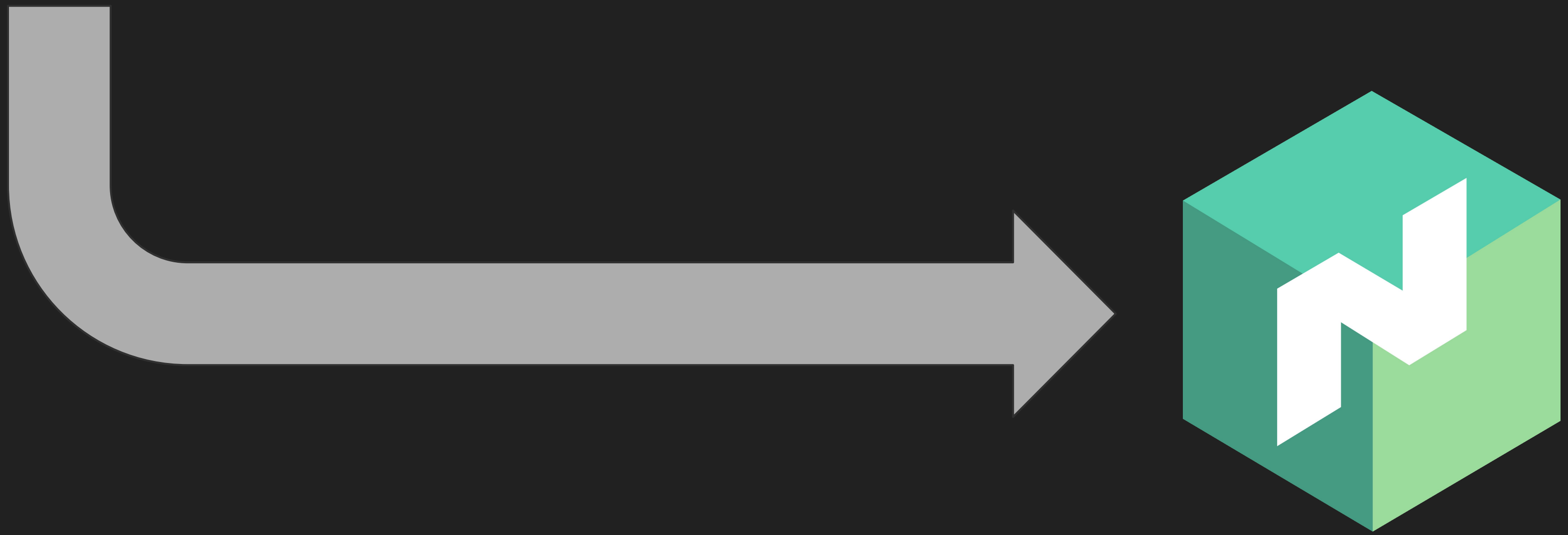
Deploy and Manage Any Containerized, Legacy, or Batch Application

First Rule of Orchestration

The deployment story is never great.

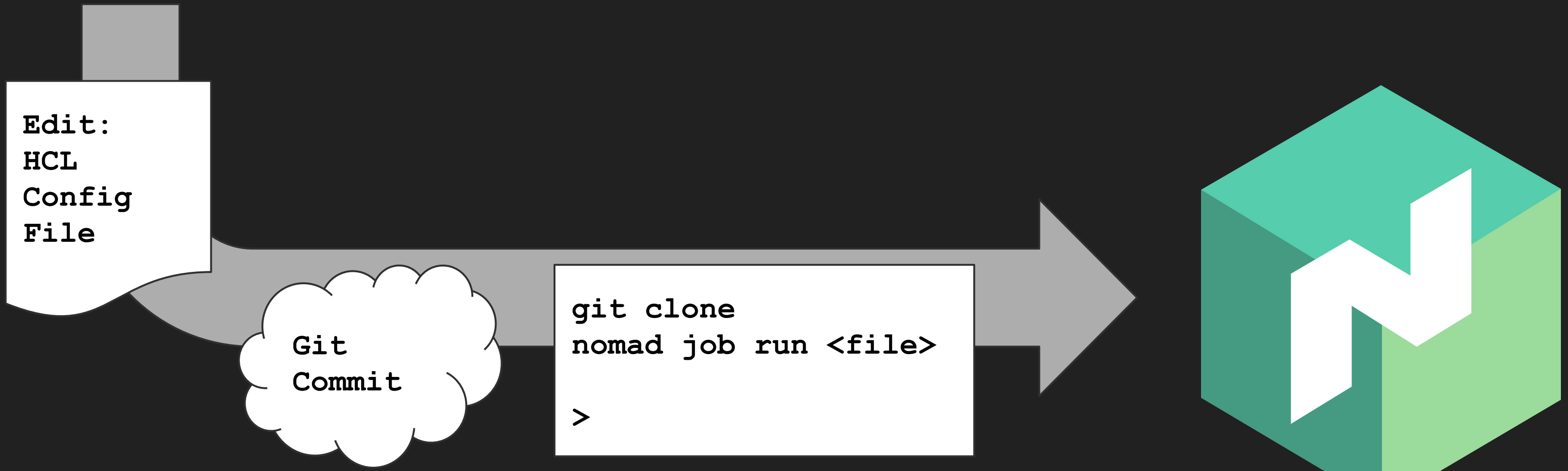
Fun Project at Work

Deploy to...



Fun Project at Work

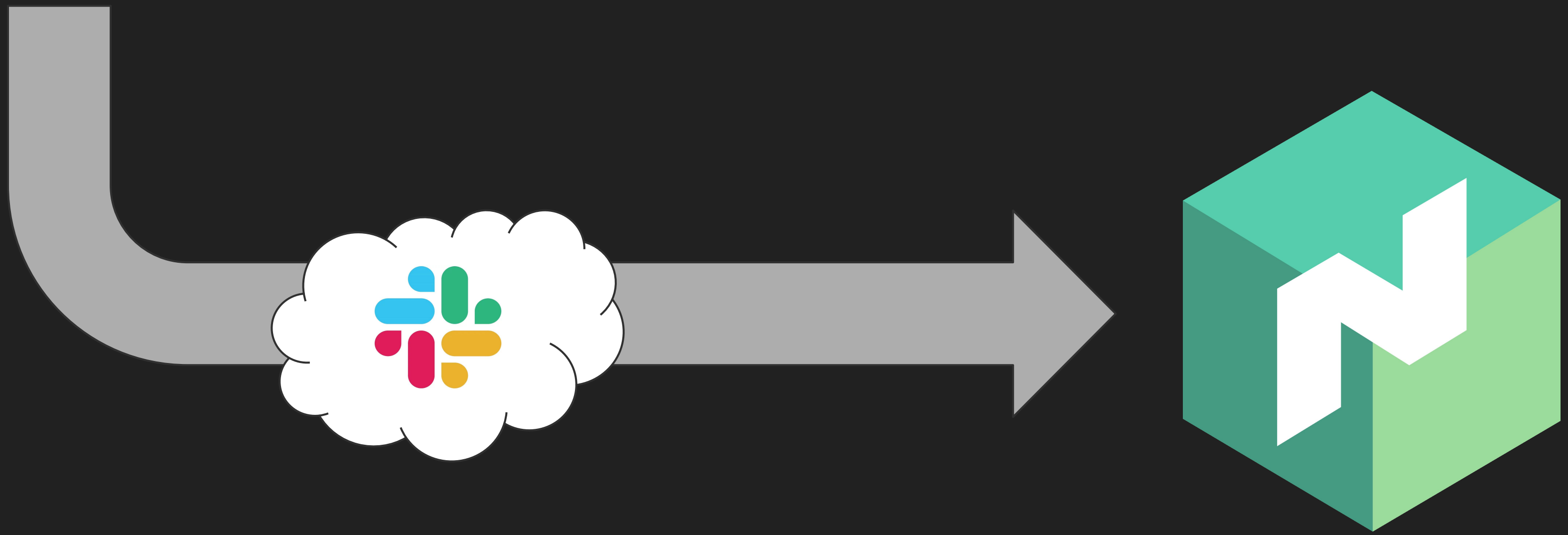
Deploy to...



```
git clone
nomad job run <file>

>
```

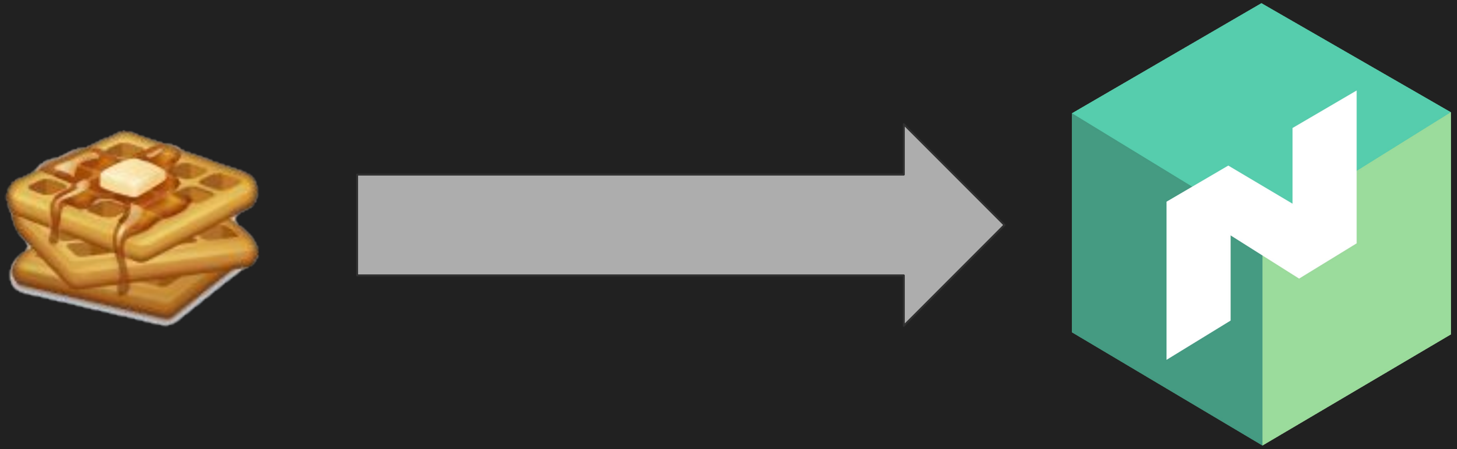
Fun Project at Work

Deploy to...



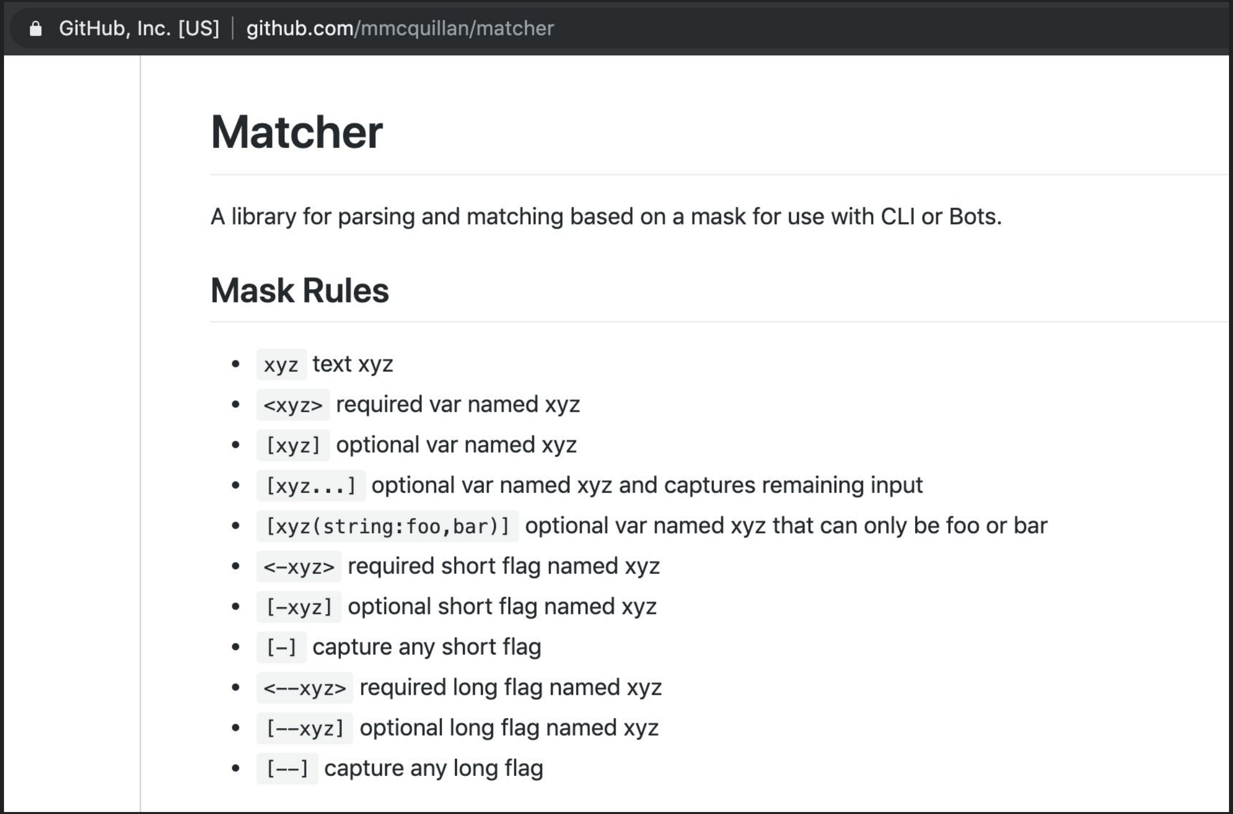
Nomad Deployment Bot

Waffles



Waffles demo

Matcher



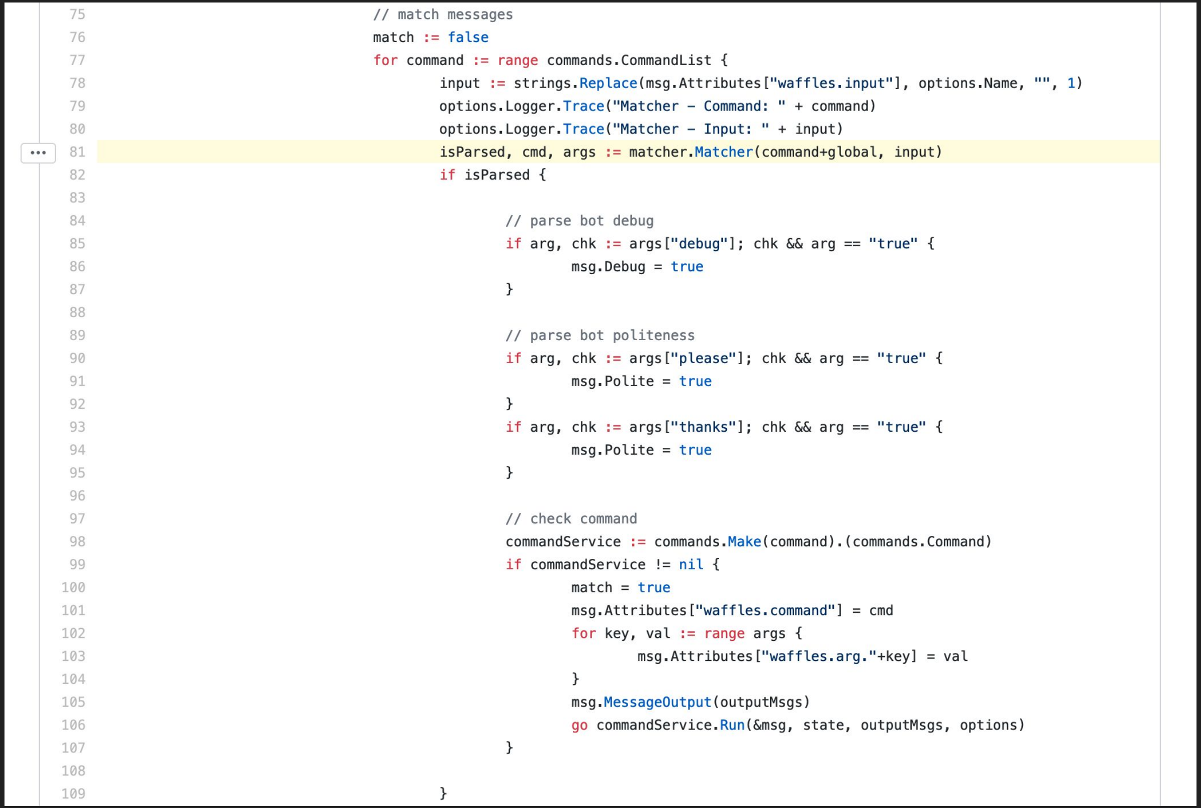
Matcher

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

Mask Rules

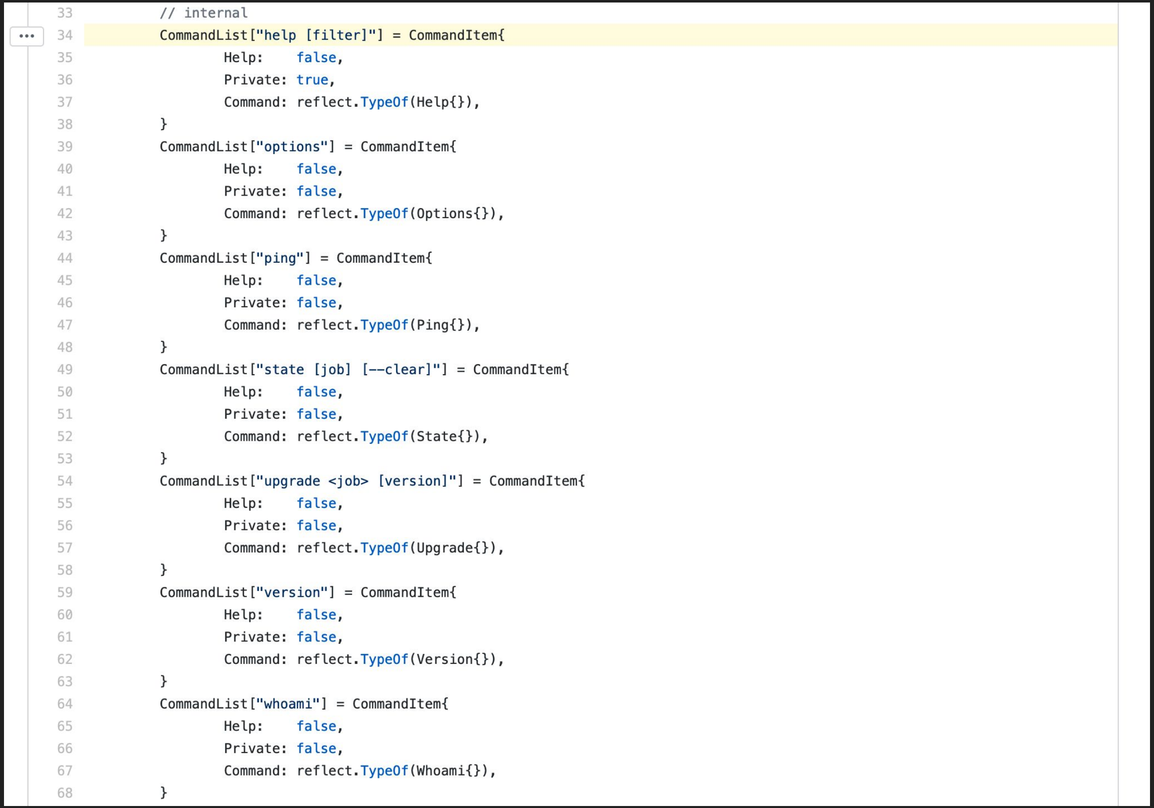
- `xyz` text xyz
- `<xyz>` required var named xyz
- `[xyz]` optional var named xyz
- `[xyz...]` optional var named xyz and captures remaining input
- `[xyz(string:foo,bar)]` optional var named xyz that can only be foo or bar
- `<-xyz>` required short flag named xyz
- `[-xyz]` optional short flag named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



```go
// match messages
match := false
for command := range commands.CommandList {
	input := strings.Replace(msg.Attributes["waffles.input"], options.Name, "", 1)
	options.Logger.Trace("Matcher - Command: " + command)
	options.Logger.Trace("Matcher - Input: " + input)
	isParsed, cmd, args := matcher.Matcher(command+global, input)
	if isParsed {

		// parse bot debug
		if arg, chk := args["debug"]; chk && arg == "true" {
			msg.Debug = true
		}

		// parse bot politeness
		if arg, chk := args["please"]; chk && arg == "true" {
			msg.Polite = true
		}
		if arg, chk := args["thanks"]; chk && arg == "true" {
			msg.Polite = true
		}

		// check command
		commandService := commands.Make(command).(commands.Command)
		if commandService != nil {
			match = true
			msg.Attributes["waffles.command"] = cmd
			for key, val := range args {
				msg.Attributes["waffles.arg."+key] = val
			}
			msg.MessageOutput(outputMsgs)
			go commandService.Run(&msg, state, outputMsgs, options)
		}

	}
```

Example Command



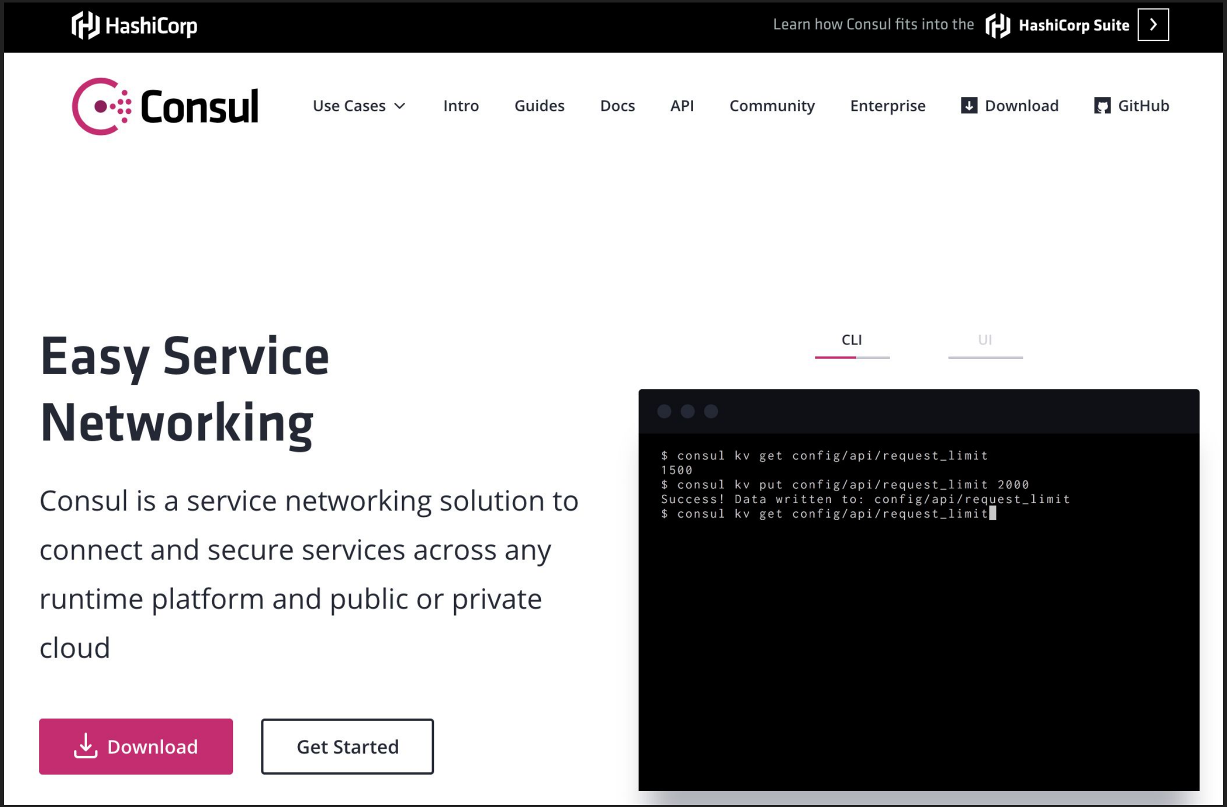
```go
	// internal
	CommandList["help [filter]"] = CommandItem{
		Help:    false,
		Private: true,
		Command: reflect.TypeOf(Help{}),
	}
	CommandList["options"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(Options{}),
	}
	CommandList["ping"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(Ping{}),
	}
	CommandList["state [job] [--clear]"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(State{}),
	}
	CommandList["upgrade <job> [version]"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(Upgrade{}),
	}
	CommandList["version"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(Version{}),
	}
	CommandList["whoami"] = CommandItem{
		Help:    false,
		Private: false,
		Command: reflect.TypeOf(Whoami{}),
	}
```

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

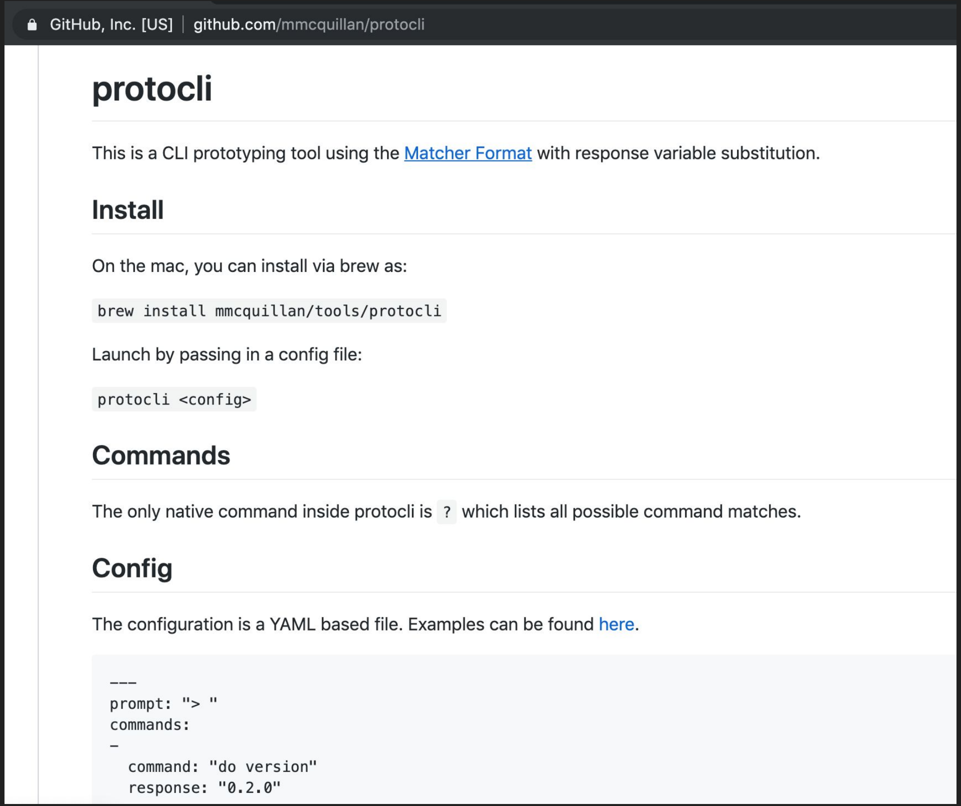


Easy Service Networking

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

```
$ 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