Конфигурация Golang приложений

25 ноября 2016

Владимир Ананьев
АО "Программный Регион"

Что такое конфигурация?

Конфигурация программного обеспечения — совокупность настроек программы, задаваемая пользователем (Wiki (https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%84%D0%B8%D0%B3%D1%83%D1%80%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D)

)

- Переменные окружения (**GO_ROOT**, ...)
- Параметры командной строки (**./app start**)
- Текстовые файлы (**json**, **yaml**, **ini** для Windows)
- Специальные базы данных (**sqlite**, например)
- Системы конроля версий (**Git**, **Mercurial**)
- Специальные системы хранения конфигурации (**Consul** (http://consul.io), **etcd**)

Примеры файлов конфигурации

- json

```
{
    "domain": "www.example.com",
    "mongodb": {
        "host": "localhost",
        "port": 27017
    }
}
```

- yaml

```
consul:
  - { key: "ReporterConfig/APIPort", value: ":8081" }
  - { key: "ReporterConfig/GPDBName", value: "stats" }
  - { key: "ReporterConfig/GPHost", value: "" }
  - { key: "ReporterConfig/GPPort", value: "5432" }
  - { key: "ReporterConfig/GPUser", value: "gpadmin" }
```

Как это использовать в Golang?

- Готового шаблона конфигурации нет
- Есть какая-то структура **AppConfig**

```
type AppConfig struct {
    Host string
    Port int
}
```

- Дальше просто в начале приложения загружаем и парсим файл

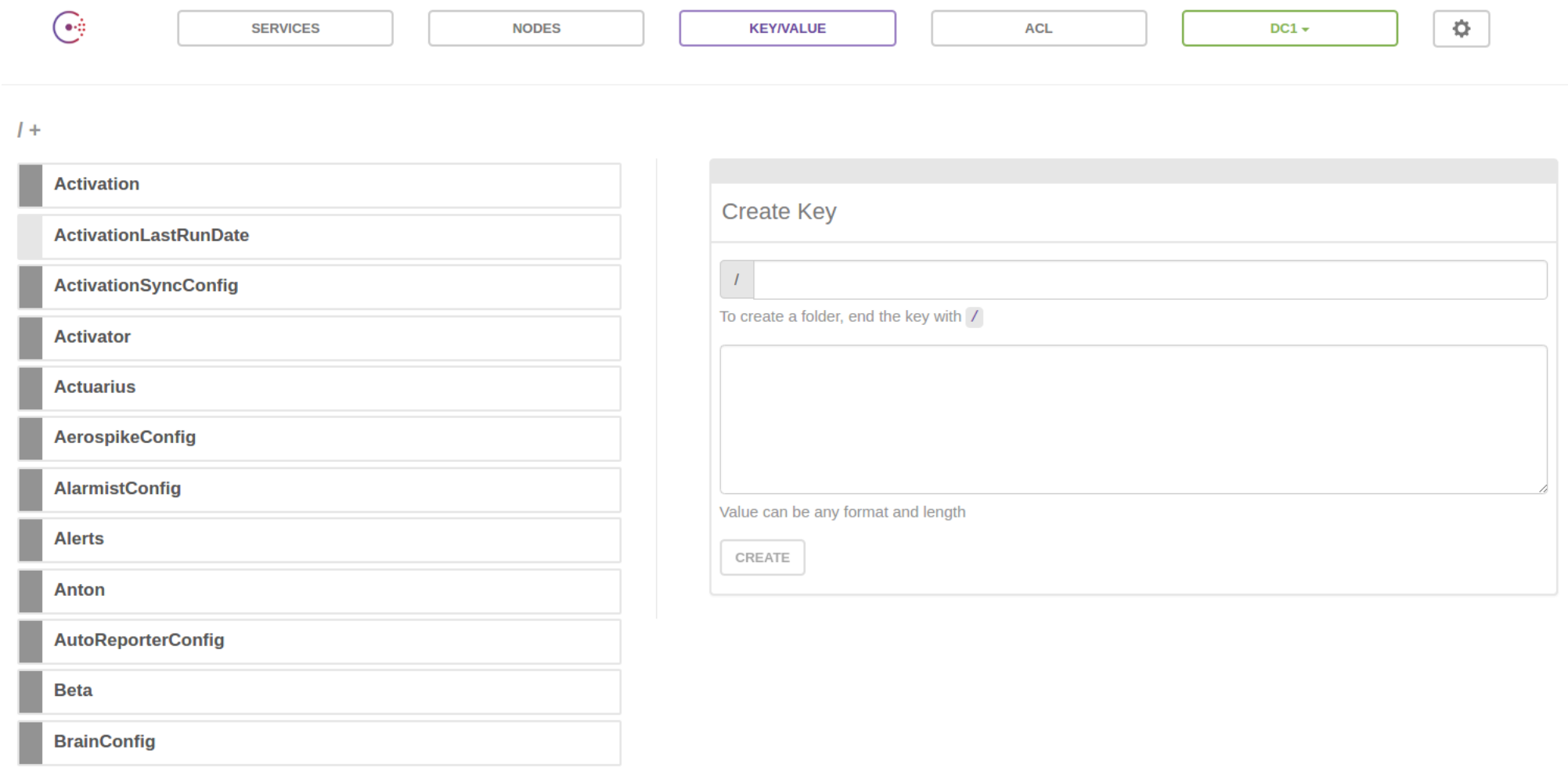
```
file, err := ioutil.ReadFile("config.json")
if err != nil {
    log.Fatal(err)
}
var cfg AppConfig
err = json.Unmarshal(file, &cfg)
...
```

Плюсы и минусы использования текстовых файлов

- Легко подгружаются в приложение
- Не требуется сетевого соединения, файлик всегда рядом
- Для микросервисной архитектуры происходит глобальное дублирование конфигурации
- Соответственно сложно вносить изменения, т.к. нужно обновлять целый ряд конфигурационных файлов
- Для монолитного приложения файл конфигурации - идеальный вариант

Готовые системы конфигурирования

- Consul



Преимущества использования Consul

- Удобное конфигурирование сервисов
- Нет дублирования общей конфигурации сервисов
- Можно создать кластер конфигурации для отказоустойчивости
- Отдельный мощный инструмент - **Service Discovery**

Как это использовать в Golang?

- Есть официальный клиент на Github ()
- Создается соединение с сервером Consul

```
import "github.com/hashicorp/consul/api"
...
// DefaultConfig -> будет коннект по адресу в
// переменной окружения CONSUL_HTTP_ADDR
cli, err := api.NewClient(api.DefaultConfig())
```

Примечание: создание соединения - это фундаментальный момент, который будет использоваться на протяжении всего курса

- Получаем объект для манипулирования key-value хранилищем

```
kv = cli.KV()
```

- Можем брать отдельные значения по ключу

```
kv.Get(key, nil)
```

Демо: как запустить Consul

- Consul binary
- Consul Web UI

```
./consul agent -server -bind 127.0.0.1 -ui-dir ~/consul/ui/ -data-dir ~/consul/data/
```

- Consul vs. Other Software (https://www.consul.io/intro/vs/index.html)

Что мы видим на странице администрирования

- **Services** - Сервисы, которые зарегистрированы в Consul
- **Nodes** - Отдельные элементы кластера, если есть
- **Key/Value** - Хранилище конфигурации
- **ACL** (Access Control List) - Специальные разграничения по правам доступа

Демо: Структура и пример **service discovery**

Демо: как считать конфигурацию из Consul

- ! CONSUL_HTTP_ADDR
- Все ключи и значения изначально строковые, связка с пакетом "**reflect**"

```
func Fill(cfg interface{}) {
    // Folder for searching inner keys
    folder := reflect.TypeOf(cfg).Elem().Name()
    elem := reflect.ValueOf(cfg).Elem()
    for i := 0; i < elem.NumField(); i++ {
        // Use field name as a key
        key := elem.Type().Field(i).Name
        val, err := Get(folder + "/" + key)
        if err != nil {
            return err
        }
        switch elem.Field(i).Interface().(type) {
        case int, int32:
            v, err := strconv.ParseInt(val, 0, 32)
            if err != nil {
                return fmt.Errorf("%v: %v", ErrInvalidInt, val)
            }
            elem.Field(i).SetInt(v)
        }
    }
```

Задание

- Скачать и запустить **Consul вместе с UI** (www.consul.io/downloads.html (https://www.consul.io/downloads.html))
- releases.hashicorp.com/consul/0.7.1/consul_0.7.1_linux_amd64.zip (https://releases.hashicorp.com/consul/0.7.1/consul_0.7.1_linux_amd64.zip)
- releases.hashicorp.com/consul/0.7.1/consul_0.7.1_web_ui.zip (https://releases.hashicorp.com/consul/0.7.1/consul_0.7.1_web_ui.zip)
- Взять приложение **track-server**
- Создать в нем пакет конфигурации со структурой конфигурации **TrackServerConfig**
- В эту структуру забить **порт**, на котором будет стартовать **track-server**
- Значение этого **порта** нужно получать из **Consul**
- Если не получится поднять свой Consul, то можно использовать открытый demo.consul.io (https://demo.consul.io)

Thank you

25 ноября 2016

Владимир Ананьев
АО "Программный Регион"
vladimir.ananyev@regium.com (mailto:vladimir.ananyev@regium.com)