

avito.tech

SEZ Innopolis — 2020

GO MODULES В PRODUCTION

Илья Данилкин

Senior Engineer / Scrum Master



Илья Данилкин

Senior Engineer / Scrum Master

Пишу на Go, начиная с версии 1.6.
Работаю в Авито в команде Auto B2B.
Наша команда делает сервисы для профессиональных пользователей в категории Авто.

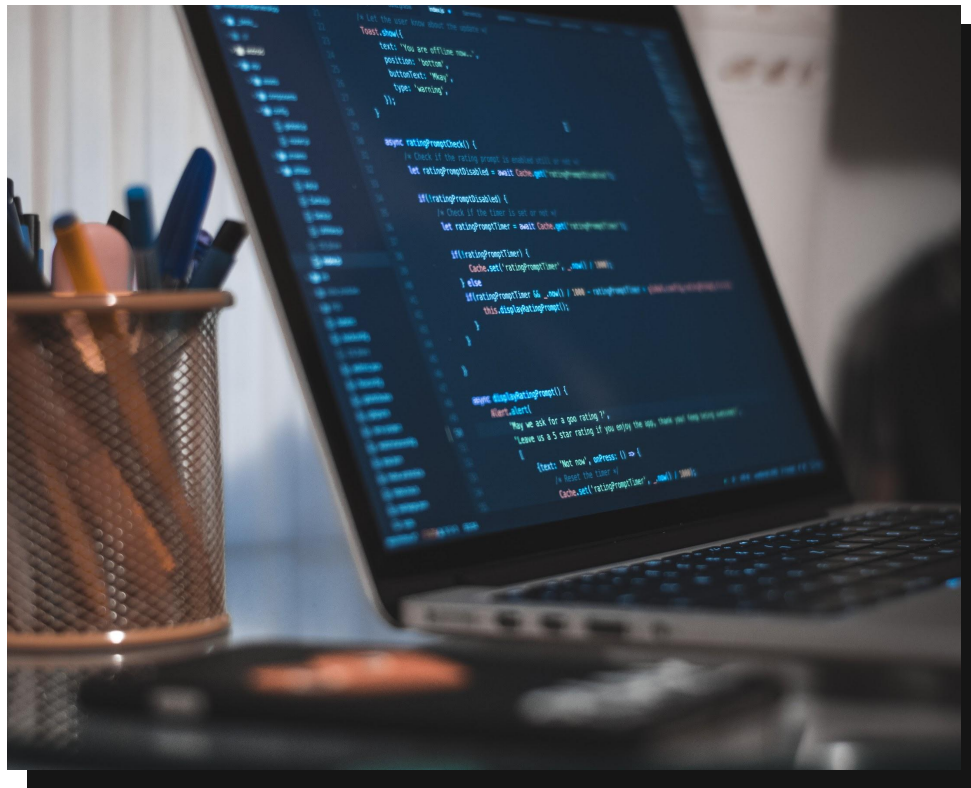
- ▶ [Спикер](#) GopherCon Russia 2018-2020
- ▶ [OpenGApps](#) contributor



AGENDA

- ▶ История создания модулей
- ▶ Как их готовить?
- ▶ Tips & tricks
- ▶ Как применять (live demo)
- ▶ Итоги

Ссылка на доклад:
местодляссылки.com





go modules в production

ИСТОРИЯ СОЗДАНИЯ МОДУЛЕЙ

THE ROAD SO FAR...



2018

[vgo proposal](#)
от Russ Cox

2018

Поддержка в 1.11
[Go Modules in 2019](#)

2019

Module-aware
“go, proxy & SumDB
[Using Go modules](#)

2020

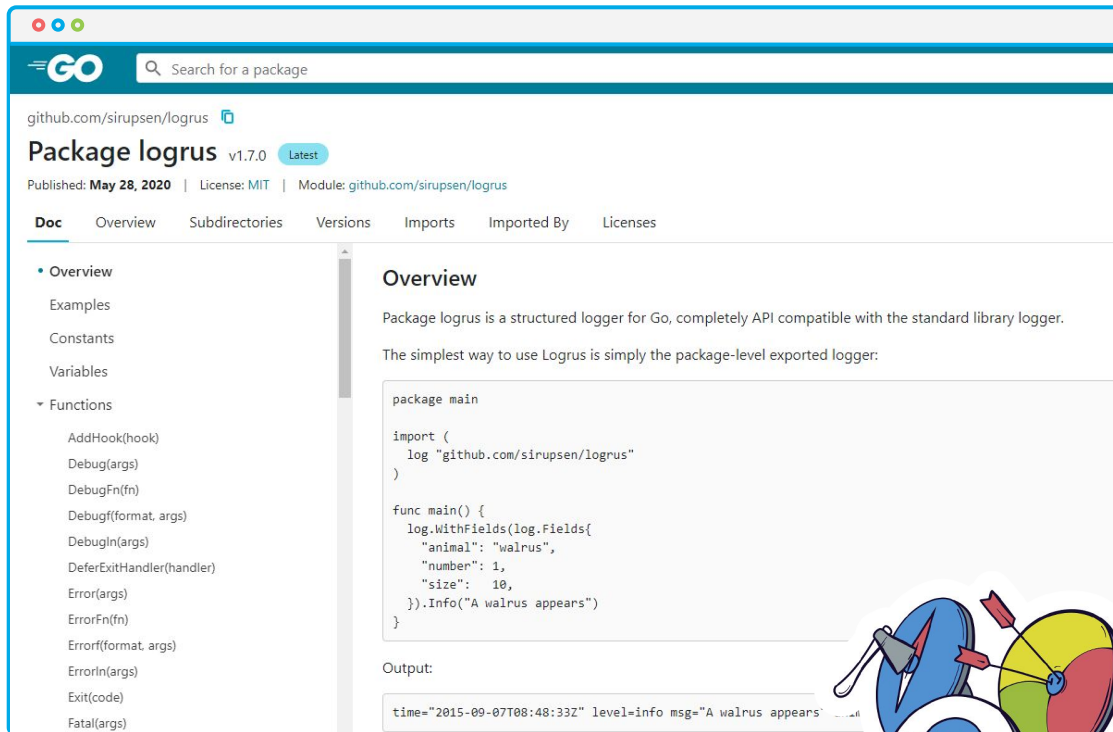
Community proxy-серверы
Поддержка /vendor
из коробки
[pkg.go.dev](#)
Ready for production!

PKG.GO.DEV

Современная замена
godoc.org:

- ▶ Поиск по базе proxy.golang.org
- ▶ Markdown
- ▶ Список версий, импортов и использования в opensource

<https://pkg.go.dev/>



The screenshot displays the pkg.go.dev website for the `github.com/sirupsen/logrus` package. The page includes a search bar at the top, the package name and version (v1.7.0), and a 'Latest' badge. It lists the publication date (May 28, 2020), license (MIT), and module path. The left sidebar contains navigation links for Overview, Examples, Constants, Variables, and Functions. The main content area shows the Overview section, which describes the package as a structured logger for Go, API compatible with the standard library logger. It provides a code example for using the package-level exported logger. The output of the code is shown as a log message: `time="2015-09-07T08:48:33Z" level=info msg="A walrus appears"`. In the bottom right corner, there is a cartoon illustration of three target-like characters with a large 'e' on one of them.



go modules в production

КАК ИХ ГОТОВИТЬ?

КЛЮЧЕВЫЕ КОМАНДЫ

01.

Инициализация:

`go mod
init`

02.

Загрузка:

`go mod
download`

03.

Чистка:

`go mod
tidy`

04.

Вендоринг:

`go mod
vendor`

+ module-aware go get

INIT

- ▶ [Поддержка](#) go/dep, Glide, Godeps etc.
- ▶ Директива **go 1.x** обозначает min API
- ▶ Больше нет необходимости в GOPATH!

```
> go mod init
```

```
module github.com/nezorflame/test
```

```
go 1.15
```

```
require github.com/sirupsen/logrus v1.7.0
```

DOWNLOAD

- ▶ Используем для скачивания зависимостей
- ▶ Проходит автоматически перед **go build**
- ▶ Встроен в **go mod tidy**

> go mod download

```
go: downloading github.com/sirupsen/logrus
v1.7.0
go: downloading golang.org/x/sys
v0.0.0-20191026070338-33540a1f6037
go: downloading github.com/stretchr/testify
v1.2.2
go: downloading github.com/pmezard/go-difflib
v1.0.0
go: downloading github.com/davecgh/go-spew
v1.1.1
```

GO GET

- ▶ Скачивает и добавляет пакет как зависимость (внутри пакета) / в кэш / ставит бинарник (снаружи)
- ▶ Можно указать версию / ветку / коммит через **@**
@latest - последняя
@none - убирает зависимость
- ▶ Управляется env
GOMODULE

```
> go get  
github.com/sirupsen/logrus@6699a89a232f3db797f2  
e280639854bbc4b89725
```

```
go: downloading github.com/sirupsen/logrus  
v1.7.0  
go: github.com/sirupsen/logrus  
6699a89a232f3db797f2e280639854bbc4b89725 ⇒  
v1.7.0  
go: downloading golang.org/x/sys  
v0.0.0-20191026070338-33540a1f6037
```

GO GET

- ▶ **go get x.y.z@version:**
обновит зависимость
до нужной версии
- ▶ Флаг **-u** обновит и
транзитивные
зависимости
-u=patch учтет только
patch-версии
- ▶ Флаг **-t** скачает
зависимости тестов
- ▶ Флаг **-d**
проигнорирует
установку

```
> go get -u
```

```
go: github.com/sirupsen/logrus upgrade =>  
v1.7.0
```

```
go: golang.org/x/sys upgrade =>  
v0.0.0-202010150000850-e3ed0017c211
```

TIDY

- ▶ Используем для скачивания нужных, упорядочивания и очистки лишних зависимостей
- ▶ Основная команда при работе (для **dep ensure**)

```
> go mod tidy
```

```
// before
require (
    github.com/golang/glog
    v0.0.0-20160126235308-23def4e6c14b // indirect
    github.com/sirupsen/logrus v1.7.0
    go.uber.org/zap v1.16.0 // indirect
)

// after
require (
    github.com/sirupsen/logrus v1.7.0
)
```

VENDOR

Используем для
скачивания модулей
внутри папки **vendor**

Но зачем?

- ▶ Поддержка CI без доступа к сети
- ▶ Обратная совместимость

```
> go mod vendor
```

```
> cat vendor/modules.txt
```

```
# github.com/sirupsen/logrus v1.7.0
## explicit
github.com/sirupsen/logrus
# golang.org/x/sys
v0.0.0-20191026070338-33540a1f6037
golang.org/x/sys/unix
golang.org/x/sys/windows
```




go modules в production

TIPS & TRICKS

IDE



Florin Păţan -
[Working with Go Modules](#)



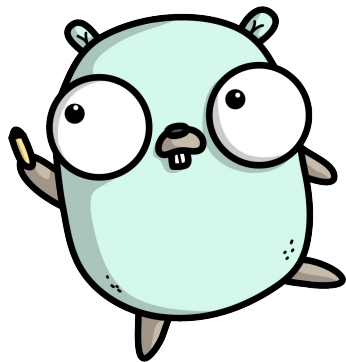
[vscode-go](#) + [gopls](#)
[nezorflame/](#)
[vscode-config](#)



[gopls: User guide](#) +
своя интеграция

ВЕРСИОНИРОВАНИЕ

01. Используйте [SemVer v2](#) с префиксом **v**:
v1.2.3, **v0.1.0-beta.1**, etc.
02. Подмодули работают с **go get**!
tags: **v1.2.3**, **mylib/v2.3.4** -
github.com/a/b@**v1.2.3**
github.com/a/b/**mylib@v2.3.4**
03. Для апгрейда модуля / зависимости используйте [marwan-at-work/mod](#)
04. Соблюдайте [Import Compatibility Rule](#) и думайте про релизы [v2](#)+



If an old package and a new package have the same import path, the new package must be backwards compatible with the old package.

Russ Cox & Go authors, 2018

GOPROXY



Используйте прокси-серверы
для скачивания модулей...

GOPROXY =

<https://proxy.golang.org/>

<https://goproxy.io/>

<https://gocenter.io/>,

direct ([1.15+](#))



...или используйте свой!

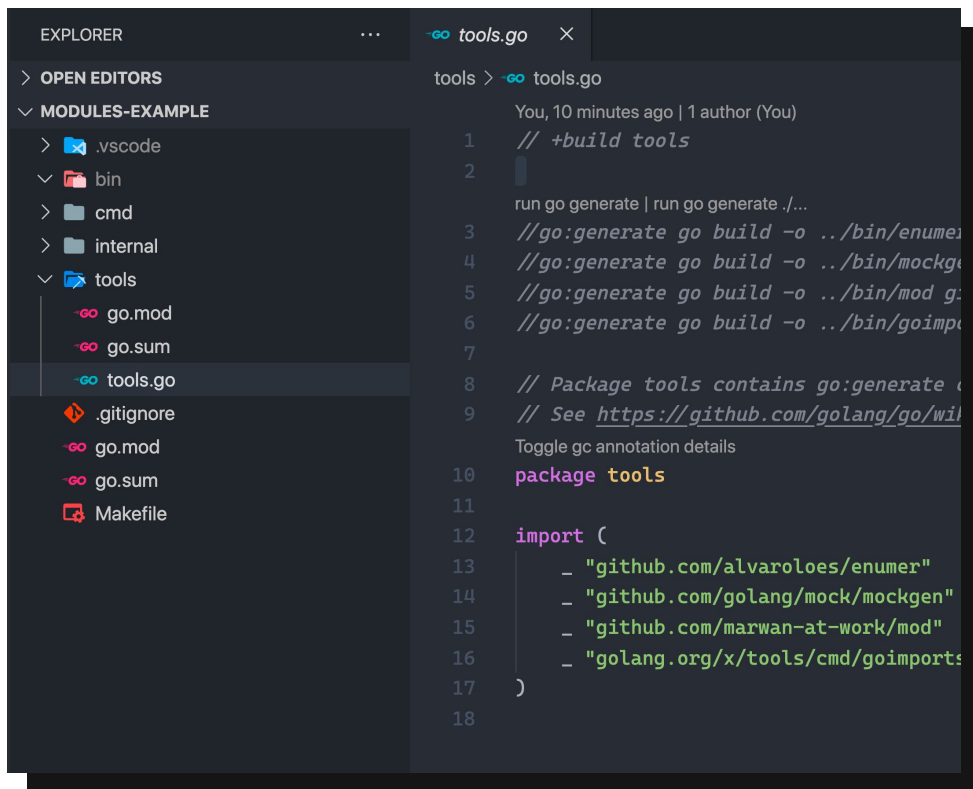
- ▶ [gomods/athens](https://github.com/gomods/athens)
- ▶ [elazarl/goproxy](https://github.com/elazarl/goproxy)
- ▶ [GitLab as a Go proxy](#)

TOOLS

- ▶ Создайте **tools/main.go**
- ▶ Добавьте туда модуль **your.project/tools**
- ▶ Поставьте билд-тэг **// +build tools**
- ▶ Добавьте пустые зависимости и генерацию бинарников
- ▶ **go generate -tags tools**

Modules Wiki:

[How can I track tool dependencies for a module?](#)

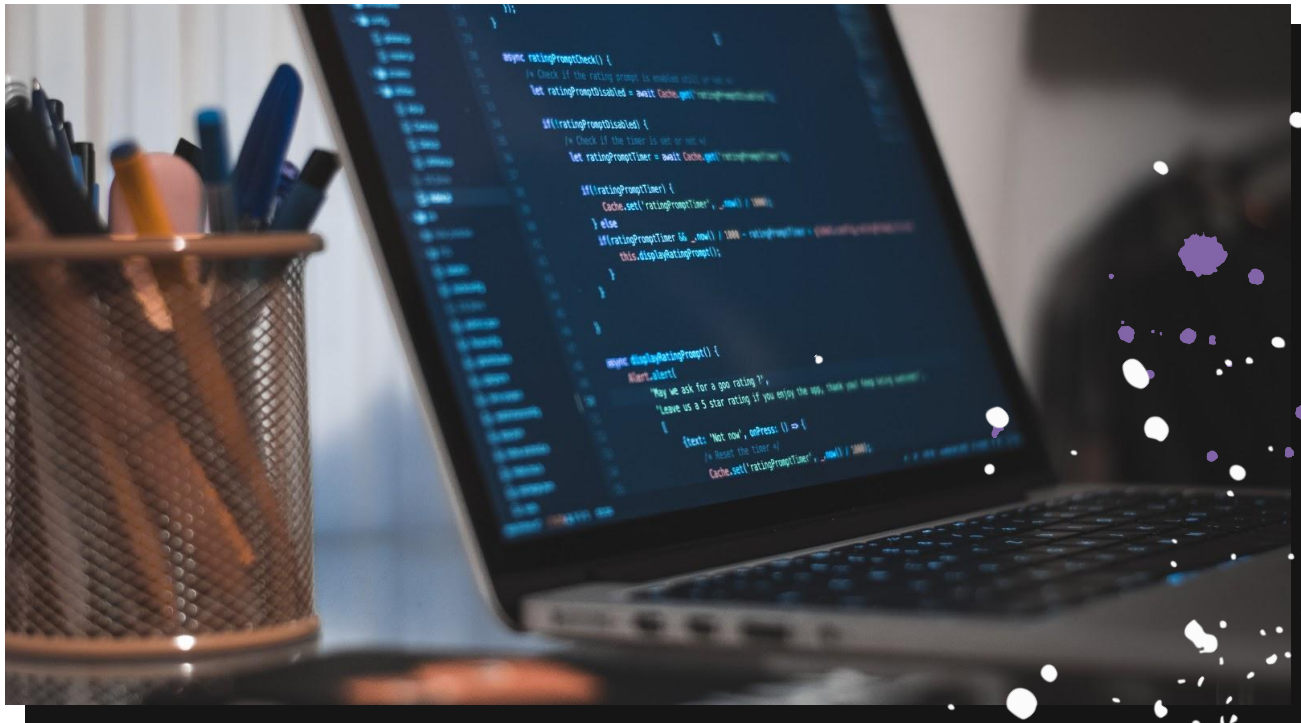




go modules в production

КАК ПРИМЕНЯТЬ?

LIVE DEMO





go modules в production

ИТОГИ

ИТОГИ

01.

Модули готовы к миграции и работе в production!

02.

go help modules +
[Modules wiki](#)

03.

Упрощайте себе жизнь с tips&tricks 🕶️

04.

Думайте о пользователях (особенно при релизах >**v1**)!

avito.tech

SEZ Innopolis — 2020

Илья Данилкин

Senior Engineer /
Scrum Master



[avito-tech](#)



[nezorflame](#)



[nezorflame](#)

Ссылки:

[Using Go modules](#)

[Go modules wiki](#)

Artwork:

[ashleymcnamara/gophers](#)