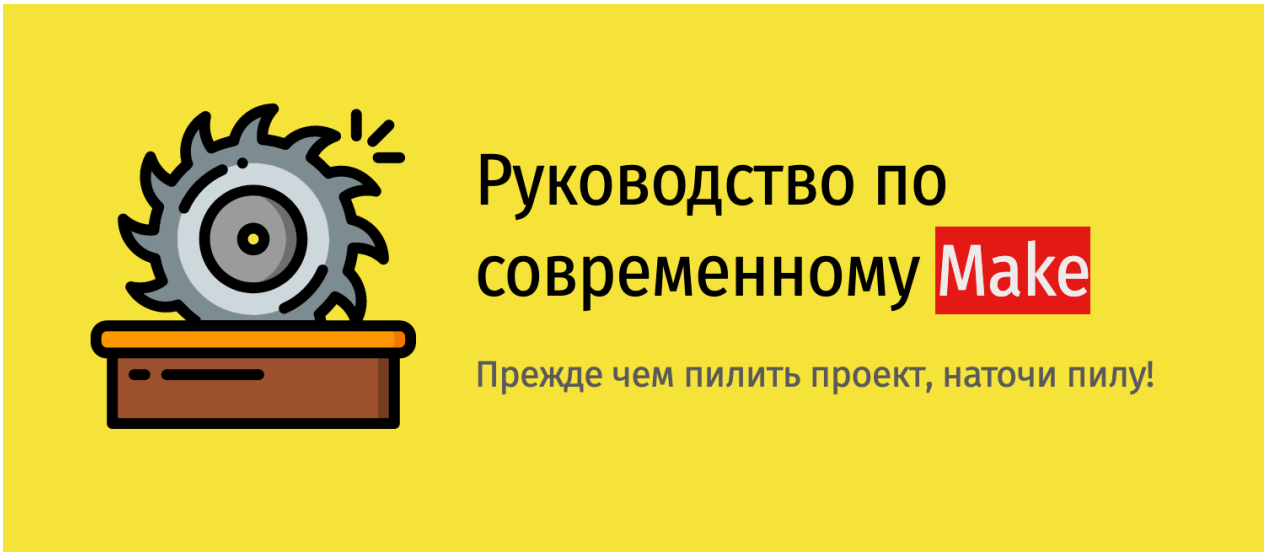
Modern Make Handbook (v. 0.5-ru)



1. Making your library of shortcuts

– Что говорит кошка, когда хочет кушать?

– Мяу!

– Что говорит собака, когда чует опасность?

– Гав-гав!

– А что говорит Вова когда хочет задеплоить проект?

– `ansible-playbook -i inventory/production --tags "deploy" app-server.yml -vvv --become-user=app --extra-vars=extra.txt --vault-password-file="~/.ansible/vault.txt"`

Вот с таким и будем бороться.

Метод борьбы прост и приятен, и напоминает процесс уборки в шкафу.

Открываем `Makefile`, и складываем туда свою мегакоманду:

```
deploy:
  ansible-playbook -i inventory/production --tags "deploy" app-server.yml -vvv --
become-user=app --extra-vars=extra.txt --vault-password-file="~/.ansible/vault.txt"
```

В следующий раз Вова просто скажет `make deploy`.

Классно ведь!? Это значит, что Вова:

- не потратит лишнюю минуту вспоминая синтаксис
- не запутается в ключах команды деплоя
- не будет страдать когда Ansible решит поменять названия половины своих ключей

Дальше действуем по аналогии.

Почему-то то что изначально задумывалось простым, например `rails server`, часто обрастает дополнительными изнуряющими подробностями: `bundle exec bin/rails server -p 3001 RAILS_ENV=development`.

Что с этим делать мы уже знаем:

```
server:
  bundle exec bin/rails server -p 3001 RAILS_ENV=development
```

Продолжаем разговор:

```
logs:
  tail -f log/development.log
```

В общем идею вы поняли. Дополнительная прелесть в том что нет никакой необходимости это все делать за один раз. Поймал себя на том что опять долго вспоминал длинную команду? Добавляй ее в мэйкфайл!

2. Overcoming Make weirdness

Вова добрался до тестов, радостно добавляет в Makefile строчку для прогона тестов:

```
test:
  MINITEST_REPORTER=SpecReporter bundle exec bin/rails test
```

Запускает, и получает привет:

```
$: make test
make: `test' is up to date.
```

– "Эээ, похоже что-то с вашим Make не так" - думает Владимир.

Надо понимать что Make придумывался чтобы билдить всякое, и изначальная семантика команды `make test` заключается в том чтобы сгенерировать папку `test`.

Соответственно, раз такая папка есть, то запускать ничего уже не надо! (*так думает Make, Вова то как раз с этим не согласен*).

Переубедить Make довольно просто. Надо добавить в Makefile магическую строчку:

`.PHONY: test`

Если таких команд несколько, то просто пишем их все через пробел:

`.PHONY: app test log doc`

Вторая вредность Make заключается в том что он отказывается работать, если, о боже мой, для отступа использованы пробелы, а не символ табуляции:

```
$: make test
Makefile:13: *** missing separator.  Stop.
```

Ну такое, да. Если ваш редактор умеет понимать типы файлов, то скорее всего он уже сам догадался использовать табы. Если нет, придется его немного подконфигурить.

Теперь Вова знает чего опасаться и как этого избегать, так что можно ехать дальше.

3. Running multiple commands at once

Хотим прогнать тесты, и если все ок, то задеплоить? No problemos:

`make test deploy`

Да, команды можно составлять в длинные цепочки, и если какая-то из команд фэйлится, то остальные запущены не будут.

4. Subcommands

В какой-то момент в команде решили что негоже деплоить без запуска тестов и прогон тестов просто захардкодили внутрь команды деплой:

```
deploy: test
  ansible-playbook -i inventory/production --tags 'deploy' # ...
```

Т.е. при использовании `make deploy` до деплоя дойдет дело только если пройдут тесты.

5. Aliases

Семен добавил в Makefile команду для прогона миграций, а Вова все никак не может запомнить как она называется. Иногда он пишет `make dbmigrate`, иногда `make db_migrate`, иногда по привычке вообще `make db:migrate`.

Увы, с последним ничего не поделать. Двоеточия в названиях комманд не поддерживаются. Зато можно смело нафигачить себе алиасов на остальные варианты! Чем Вова и занялся.

Для этого не нужно копипастить нашу длинную команду несколько раз, достаточно записать вызов оригинала после двоеточия:

```
db-migrate:
  bundle exec bin/rails db:migrate

db_migrate: db-migrate
dbmigrate: db-migrate
```

Придумать годное название для шортката сходу бывает непросто. В таких случая как раз можно насоздавать сразу несколько алиасов, и оставить тот который приживется со временем.

6. Multiline commands

Со временем Вове надоело набирать такую длинную команду, и он заменил ее на `make db` (не забыв добавить `db` в `PHONY:`)

И все бы классно, но в какой-то момент разработчики на проекте договорились не коммитить в проект `db/schema.rb` (который авто-обновляется после прогона миграций), а это значит что каждый раз после прогона миграций приходилось выполнять команду `make schema-reset`:

```
schema-reset:
  git checkout HEAD -- db/schema.rb
```

К счастью, никто не запрещает запускать несколько команд под одним шорткатом, и команды Make можно вызывать из Makefile:

```
db:
  bundle exec bin/rails db:migrate
  make schema-reset

schema-reset:
  git checkout HEAD -- db/schema.rb
```

Единственный минус в том, что Make по умолчанию многословен, и печатает каждую команду прежде чем выполнить:

```
$: make db
bundle exec bin/rails db:migrate
# ...
make schema-reset
git checkout HEAD -- db/schema.rb
```

К счатью это легко забороть.

7. Suppressing output

Все что нужно сделать чтобы Make не выводил саму команду, а просто её выполнял, это добавить перед ней символ "@".

Например, так:

```
hello:
  @echo "Привет, Вова!"
```

Соответственно лишний вывод "make schema-reset" прячем так:

```
db:
  bundle exec bin/rails db:migrate
  @make schema-reset
```

Ура!

8. Ignoring errors

При деплое на staging разрабы тоже решили прогонять тесты:

```
staging-deploy:
  @make test
  ansible-playbook -i inventory/staging --tags 'deploy' #...
```

Правда быстро выяснилось, что иногда надо задеплоить, даже если тесты падают!

Чтобы не выпиливать тесты из сценария, но деплоить несмотря на их результат, можно использовать магический префикс "-":

```
staging-deploy:
  -@make test
  ansible-playbook -i inventory/staging --tags 'deploy' #...
```

9. Running command only if another one fails

А можно было поступить по другому.

```
staging-deploy:
  @make test || echo "Опять Вова поломал тесты!!"
  ansible-playbook -i inventory/staging --tags 'deploy' #...
```

Это даже не фишка Make, это обычный Bash scripting. В результате программа будет каждый раз журить Вову если тесты упали, но и от деплоя отказываться не будет.

10. Passing arguments

Однажды Вове понадобилось стянуть дамп базы со стэйджинга на свой комп. Пришлось напрячь остатки пямяти и разродиться скриптом:

```
staging-fetch-dump:
  scp app@staging-server.dev:/path/to/app/db/dump.tgz ./
```

Только вот неплохо бы его сделать чуть более полезным. Вдруг понадобится какой-то еще файл стягивать.

По такому случаю можно передать название файла в качестве аргумента:

```
staging-fetch:
  scp app@staging-server.dev:/path/to/app/$(F)/ ./
```

Вызов команды теперь будет выглядеть так:

`make staging-fetch F=db/dump.tgz`

11. Seamless arguments

Однако в случае когда аргумент всего один, было бы классно избавиться от необходимости запоминать название этого самого аргумента и вызывать команду прямо так: `make staging-fetch db/dump.tgz`

Этого можно добиться, но только с помочью черной магии. Надо добавить в Makefile вот такую конструкцию:

```
ARGS = $(filter-out $@,$(MAKECMDGOALS))
%:
  @:
```

Шорткат при таких раскладах выглядит вот так:

```
staging-fetch:
  scp app@staging-server.dev:/path/to/app/$(ARGS)/ ./
```

Но магия на то и черная, что у неё есть неприятный спецеффект – уже после того как все успешно выполнится, прилетает вот такое сообщение:

```
make: *** No rule to make target 'db/dump.tgz'.  Stop.
```

Можете почитать подробности того как это работает, чуть ниже вроде даже показывают как победить проблему с ошибкой, но мне это победить не удалось.

Короче говоря, Вова смирился, и решил что можно заплатить такую цену за такую фичу, ну а вам решать самим.

12. Advanced scripting

Внезапно админы запилили все так, что теперь на каждый фичебранч поднимается по отдельному стэйджингу.

И все бы классно, но теперь в наши крутые шорткаты для работы со стэйджингом придется добавлять по еще одной переменной - имени сервера:

```
ssh:
  ssh app@$(S)

staging-fetch:
  scp app@$(S):/path/to/app/$(F)/ ./
```

... TO BE CONTINUED IN PRO VERSION: ...