



# Написать плагин Vim

Опубликовано 18 марта 2022 г. Натаниэль [Стикман](#)

Создайте учетную запись [Linode](#) , чтобы  
попробовать это руководство с кредитом в  
размере 100 долларов США .

*Этот кредит будет применен ко всем  
действительным услугам, использованным вами в  
течение первых 60 дней.*

[Зарегистрироваться](#)

Vim — это минималистский текстовый редактор, который легко адаптируется. Помимо конфигурации и скриптов, вы можете расширить Vim с помощью широкого спектра плагинов, разработанных и поддерживаемых большим и активным сообществом.

В этом руководстве вы узнаете, как написать собственный плагин Vim. Руководство проведет вас через создание и развертывание примера плагина. Оно покажет вам, как создавать плагины, использующие скрипт Vim, Python или внешние программы командной строки.

## Прежде чем начать #

1. Ознакомьтесь с нашим руководством [по началу работы с Linode](#) и выполните шаги по настройке имени хоста и часового пояса Linode.
2. Это руководство использует `sudo` везде, где это возможно. Заполните разделы нашего руководства [«Как защитить свой сервер»](#) , чтобы создать стандартную учетную запись пользователя, укрепить доступ по SSH и удалить ненужные сетевые службы.
3. Обновите свою систему.

- В Debian и Ubuntu используйте следующую команду:

```
sudo apt update && sudo apt upgrade
```

- В AlmaLinux , CentOS (8 или более поздней версии) или Fedora используйте следующую команду:

```
sudo dnf upgrade
```

### Примечание

Действия в этом руководстве написаны для пользователей без прав `root`. Команды, требующие повышенных привилегий, имеют префикс `sudo`. Если вы не знакомы с `sudo` командой, см. руководство [Linux Users and Groups](#).

## Как работают плагины Vim #

Vim – это высоконастраиваемый текстовый редактор. По умолчанию Vim поставляется с файлом конфигурации – обычно `~/.vimrc` – который предоставляет вам широкий спектр опций для управления поведением и внешним видом Vim. Вы можете узнать больше о настройке вашего экземпляра Vim в нашем руководстве [Введение в настройку Vim](#).

Используя плагины, Vim становится еще более адаптивным. Редактор имеет широкое и преданное сообщество пользователей. Многие из этих пользователей внесли исключительные инструменты, которые добавляют новые функции или адаптируют существующие функции в Vim.

Когда дело доходит до создания собственного плагина Vim, есть две основные причины для этого:

- Чтобы поделиться своими конфигурациями Vim с более широким сообществом. Плагины – это предпочтительный способ распространения вашего кода Vim для использования другими. Соблюдение некоторых стандартов плагинов и размещение вашего плагина на GitHub делает ваш плагин доступным для других через менеджеры плагинов, такие как [vim-plug](#).
- Чтобы организовать ваши конфигурации Vim. Даже если вы когда-либо будете хранить плагин только для себя, наличие более сложного кода Vim в формате плагина может помочь вам сделать ваши конфигурации Vim более организованными и поддерживаемыми.

## Как написать плагин Vim #

В этом разделе вы узнаете, как создать плагин Vim. Пример плагина отображает время, ищет определения слов и дает предложения по написанию. Инструкции, которые следуют далее, показывают, как настроить предварительные условия и реализовать код для плагина.

Установите Streamlink CLI На Ubuntu И MacOS

Установка И Настройка Supervisor На CentOS 8

Введение В Настройку Vim

Ограничение Доступа С Помощью SFTP-Джейлов В Debian И Ubuntu

Список Заданий Cron В Linux

Нагрузочное Тестирование Веб-Серверов С Помощью Siege

Манипулируйте Списками С Помощью Sort И Uniq

Манипулируйте Текстом Из Командной Строки С Помощью Sed

Изменить Права Доступа К Файлам С Помощью Chmod

Мониторинг И Администрирование Сервера С Помощью Cockpit

Навигация По Emacs С Использованием Evil Mode

Запуск Заданий Или Скриптов С Использованием Crontab При Загрузке

Установка И Изменение Цветовых Схем Vim

Команда Linux Cat

Команда Ping В Linux

Используйте Chroot Для Тестирования В Ubuntu

Используйте CURL С RESTful API

Используйте Glances Для Мониторинга Системы В Linux

Используйте Команды Killall И Kill Для Остановки Процессов В Linux

Используйте Rclone Для Синхронизации Файлов С Хранилищем Объектов Linode

Используйте Tcpdump Для Анализа Сетевого Трафика

Используйте Команду Select Для Обработки Текста В Linux

Используйте Команду Linux Dig Для Поиска Записей DNS

Используйте Команду Nslookup

Используйте Команду Sd Для Поиска И Замены В Linux

Используйте Страницы Tldr В Linux

Использование Cron Для Планирования Задач На Определенное Время Или Интервалы

Использование Команд Текстового Редактора Nano В Linux

Чтобы помочь организовать плагины и сделать их более удобными для обслуживания, в этом руководстве используется менеджер плагинов [vim-plug](#). Однако существуют и другие менеджеры плагинов, поэтому не стесняйтесь выбирать вариант, который лучше всего подходит вам.

Подробности установки vim-plug вы можете узнать в нашем руководстве [Introduction to Vim Customization](#). Однако, если у вас уже установлен cURL, вы можете установить vim-plug с помощью следующей команды:

```
sudo curl -fLo ~/.vim/autoload/plug.vim --create-dirs https://
```

## Анатомия плагина Vim #

Существует множество возможных способов настройки плагина для Vim. Однако метод, показанный здесь, основан на официальной документации и преобладающих тенденциях в сообществе плагинов Vim. Эти передовые методы также позволяют организовать и поддерживать ваш код Vim.

Во-первых, ваш плагин должен иметь основной каталог, который использует имя плагина. В этом каталоге плагин должен иметь `plugin` и `autoload` каталог:

- Каталог `plugin` настраивает плагин. Он определяет команды, которые должен предоставлять плагин, и устанавливает любые сочетания клавиш, которые вы хотите, чтобы плагин имел по умолчанию.
- Каталог `autoload` содержит движок плагина. Сохранение этого кода в `autoload` каталоге позволяет Vim более эффективно его использовать. Vim загружает `autoload` код только в том случае, если вызывается одна из команд, определенных в этой `plugin` части. Это делается для того, чтобы Vim мог загружать только то, что ему нужно, когда это нужно.

Например, если ваш плагин называется `example-plugin`, минимальный каталог плагина может напоминать следующее дерево каталогов:

```
example-plugin/
  autoload/
    example-plugin.vim
  plugin/
    example-plugin.vim
```

В каждом `.vim` файле ваш плагин имеет доступ к трем методам обработки информации:

Использование Команды JQ Для Обработки JSON В Командной Строке	
Просмотр Активных Процессов Linux С Помощью Procs	
Просмотр И Отслеживание Конца Текстовых Файлов С Помощью Tail	
Просмотр Начала Текстовых Файлов С Заголовком	
Инструменты Командной Строки Windows: Руководство Для Начинающих	
Напишите Плагин Neovim С Помощью Lua	
Написать Плагин Vim	
Посмотреть Все 71 Руководство По Инструментам	
Время Безотказной Работы И Аналитика	▼
Руководства По Веб-Серверам	▼
Веб-Сайты Руководства	▼
Референтные Архитектуры	▼
Документы Торговой Площадки	▼
Продукция	↗
API	↗
Блог	↗
Ресурсы Контента	↗

- Использование скрипта Vim, интерпретируемого непосредственно в Vim
- Использование внешнего интерпретатора, например Python, Ruby и т. д.
- Использование вывода другой программы командной строки

Vim не ограничивает ваш плагин только одним из этих методов. Вы можете свободно смешивать и подбирать их в соответствии с потребностями вашего плагина. Обычно лучше всего выбирать подход, который наиболее эффективно достигает целей вашего плагина.

Пример плагина Vim, разработанный в следующих разделах, покажет вам, как реализовать каждый из этих методов.

## Написание плагина Vim #

Плагину требуется некоторая начальная настройка, включая создание его каталогов и файлов скриптов Vim. Эти шаги показывают, как настроить и включить код для `plugin` файла скрипта Vim каталога.

1. Создайте каталог для плагина и перейдите в этот каталог. Это руководство помещает плагин в домашний каталог текущего пользователя.

```
mkdir ~/example-plugin
cd ~/example-plugin
```

В остальной части этого руководства предполагается, что вы находитесь в этом каталоге.

2. Создайте `autoload` и `plugin` каталог.

```
mkdir autoload
mkdir plugin
```

3. Создайте новый `example-plugin.vim` файл в `plugin` каталоге и добавьте в него содержимое `example-plugin.vim`.

Файл: plugin/example-plugin.vim

```
1  " Title:      Example Plugin
2  " Description: A plugin to provide an example for creating Vim plugins.
3  " Last Change: 8 November 2021
4  " Maintainer: Example User <https://github.com/example-user>
5
6  " Prevents the plugin from being loaded multiple times. If the loaded
7  " variable exists, do nothing more. Otherwise, assign the loaded
8  " variable and continue running this instance of the plugin.
9  if exists("g:loaded_example-plugin")
10     finish
```

```

11     endif
12     let g:loaded_example-plugin = 1
13
14     " Exposes the plugin's functions for use as commands in Vim.
15     command! -nargs=0 DisplayTime call example-plugin#DisplayTime
16     command! -nargs=0 DefineWord call example-plugin#DefineWord()
17     command! -nargs=0 AspellCheck call example-plugin#AspellCheck
18

```

4. Создайте новый `example-plugin.vim` файл в `autoload` каталоге. Это файл, который загружается всякий раз, когда вызывается одна из команд вашего плагина:

```
touch autoload/example-plugin.vim
```

В следующих трех разделах показано, как добавлять функции в плагин Vim. Каждый раздел использует свой подход к обработке информации в плагине Vim. В конце концов, у вас есть работающий плагин с тремя полезными командами.

## Использование скрипта Vim #

Добавьте `DisplayTime` функцию к `example-plugin.vim` файлу в `autoload` каталоге. Эта функция отображает дату и время. Она также позволяет пользователю опционально указать флаг, указывающий, хочет ли он видеть только дату ( `d` ) или время ( `t` ).

Файл: `autoload/example-plugin.vim`

```

1  function! example-plugin#DisplayTime(...)
2      if a:0 > 0 && (a:1 == "d" || a:1 == "t")
3          if a:1 == "d"
4              echo strftime("%b %d")
5          elseif a:1 == "t"
6              echo strftime("%H:%M")
7          endif
8      else
9          echo strftime("%b %d %H:%M")
10     endif
11 endfunction
12

```

## Использование переводчика #

1. Установите пакет Vim для Python 3.

```
pip3 install vim
```

2. Добавьте код Python и `DefineWord` функцию Vim в `example-plugin.vim` файл в `autoload` каталоге. Код Python дает вашему плагину функцию для извлечения определений английских слов из

**Викисловаря** . Функция Vim получает слово под курсором пользователя и передает его функции Python.

Файл: autoload/example-plugin.vim



```

1      " [...]
2
3      " Starts a section for Python 3 code.
4      python3 << EOF
5      # Imports Python modules to be used by the plugin.
6      import vim
7      import json, requests
8
9      # Sets up variables for the HTTP requests the
10     # plugin makes to fetch word definitions from
11     # the Wiktionary dictionary.
12     request_headers = { "accept": "application/json" }
13     request_base_url = "https://en.wiktionary.org/api/rest_v1/pag
14     request_url_options = "?redirect=true"
15
16     # Fetches available definitions for a given word.
17     def get_word_definitions(word_to_define):
18         response = requests.get(request_base_url + word_to_define
19
20         if (response.status_code != 200):
21             print(response.status_code + ": " + response.reason)
22             return
23
24         definition_json = json.loads(response.text)
25
26         for definition_item in definition_json["en"]:
27             print(definition_item["partOfSpeech"])
28
29             for definition in definition_item["definitions"]:
30                 print(" - " + definition["definition"])
31     EOF
32
33     " Calls the Python 3 function.
34     function! example-plugin#DefineWord()
35         let cursorWord = expand('<word>')
36         python3 get_word_definitions(vim.eval('cursorWord'))
37     endfunction
38

```

## Использование программы командной строки **#**

1. Установите **aspell** , инструмент командной строки для проверки орфографии. Vim имеет встроенную проверку орфографии, но эта дает вам преимущества использования внешнего инструмента и стандартного формата.

- В Debian и Ubuntu используйте следующую команду:

```
sudo apt install aspell
```

- В AlmaLinux , CentOS , Fedora используйте следующую команду:

```
sudo dnf install aspell aspell-en
```

2. Добавьте `AspellCheck` функцию в `example-plugin.vim` файл в `autoload` каталоге. `system` Функция, используемая здесь, позволяет плагину выполнять команды в командной строке системы. Вы также можете использовать `exec` функцию вместе с `!` символом для запуска системных команд.

Файл: `autoload/example-plugin.vim`



```
1 | " [...]  
2 |  
3 | function! example-plugin#AspellCheck()  
4 |     let cursorWord = expand('<word>')  
5 |     let aspellSuggestions = system("echo '" . cursorWord . "'  
6 |     let aspellSuggestions = substitute(aspellSuggestions, "&  
7 |     let aspellSuggestions = substitute(aspellSuggestions, "  
8 |     echo aspellSuggestions  
9 | endfunction  
10 |
```

## Установить плагин #

Последний шаг для начала использования вашего плагина – это добавление его в ваш менеджер плагинов. Для этого добавьте строку, подобную той, что указана ниже, в конфигурацию плагина в вашем файле конфигурации Vim. Эта строка работает с `vim-plug` и местоположением плагина, используемым в шагах выше. Однако вам нужно изменить строку в зависимости от используемого вами менеджера плагинов и фактического местоположения и имени вашего плагина.

Файл: `~/.vimrc`



```
1 | " [...]  
2 | Plug '~/example-plugin'  
3 | " [...]
```

Либо снова откройте Vim, либо снова загрузите файл конфигурации, и вы готовы начать использовать плагин. Если вы хотите сделать свой плагин доступным для более широкого сообщества Vim, следуйте инструкциям в следующих разделах.

## Как развернуть плагин Vim #

Большинство менеджеров плагинов Vim автоматически извлекают плагины из GitHub. Это дает вам удобный способ распространения вашего плагина. Ниже вы можете увидеть, как загрузить ваш плагин в репозиторий GitHub. Вы также можете получить представление о том, какую дополнительную информацию вы можете предоставить для руководства вашими пользователями.

## Добавьте README для вашего плагина Vim #

Обычно хорошей практикой является включение файла Readme при распространении плагина Vim. В файле Readme должны быть указаны инструкции по установке и некоторые положения о том, как использовать плагин. В вашем файле Readme также должны быть указаны любые дополнительные системные требования вашего плагина. Например, созданный выше пример плагина требует, чтобы у пользователя был `aspell` установлен Python 3.

Создайте `README.md` файл в базовом каталоге плагина. GitHub автоматически визуализирует и отображает содержимое этого файла любому, кто посещает главную страницу вашего репозитория.

Взгляните на наш [пример файла README](#) для идей о том, какую информацию вы можете захотеть предоставить. Этот пример соответствует примеру плагина, разработанному в разделах выше.

## Создать Git-репозиторий #

1. В каталоге вашего плагина используйте следующую команду для инициализации репозитория Git.

```
git init
```

2. Создайте `.gitignore` файл. Если есть файлы или каталоги, которые вы не хотите добавлять в удаленный репозиторий Git, добавьте шаблоны, соответствующие этим файлам/каталогам, в `.gitignore` файл.

Вот простой пример, который игнорирует `.DS_STORE` файлы:

Файл: `.gitignore`



```
1 | .DS_STORE
```

3. Добавьте файлы вашего плагина для подготовки к первому коммиту Git.

```
git add .
```

4. Зафиксируйте файлы. Рекомендуется добавлять краткий описательный комментарий к каждому сделанному вами коммиту, как показано ниже:

```
git commit -m "Initial commit."
```

5. Добавьте удаленный репозиторий. Замените URL в примере ниже на URL вашего удаленного репозитория.



```
git remote add origin https://github.com/example-user/exa
```

☰

🔍

Создать Git-репозиторий

⬆

☰

Облачные Руководства И Руководства

⬆

Устранение неполадок Виртуальных Сетевых Подключений

Смарт + Линод

Приложения

Базы Данных

Разработка

Руководства По Почтовым Серверам

Игровые Серверы

Кubernetes

IP-Адреса, Сети И Домены

Платформа Линод

Быстрые Ответы

Безопасность, Обновления И Резервное Копирование

Инструменты И Справочники

Основы Linux

Пользовательские Ядра И Дистрибутивы

Передача Файлов

Управление Пакетами Linux

Инструменты

Архивирование И Сжатие Файлов С Помощью GNU Tar И GNU Zip

Проверьте Использование Диска В Linux С Помощью Команды Df

Создание Ссылок Файловой Системы С Помощью Ln

Загрузка Ресурсов Из Командной Строки С Помощью Wget

Более Быстрая Навигация По Файлам С Помощью Автоматического Перехода

Поиск Файлов В Linux С Помощью Командной Строки

Найти Файлы С Помощью Команды Fd

Начало Работы С Vi И Vim

Как Разделить Файлы С Помощью Split

Как Установить И Использовать

git push -u origin master

Теперь ваш плагин доступен другим пользователям в их локальных экземплярах.

Дополнительная информация

Вы можете обратиться к следующим ресурсам для получения дополнительной информации по этой теме. Мы надеемся, что они будут полезны вам. Мы можем ручаться за точность или содержание этих внешних ресурсов материалов.

• [Советы по Vim Wiki: Как написать плагин](#)

• [IBM Developer: написание скриптов](#)

• [Справочное руководство Vim: Инициализация](#)

Эта страница была первоначально опубликована 11 ноября 2024 года.

НЕО ПОТОМУ ЧТО

🔗

🔗

Ваше мнение

Дайте нам знать, было ли это полезно.

Оставить комментарий

Присоединяйтесь к обсуждению.

Прочитайте другие комментарии или комментарии ниже. Комментарии должны быть уместными по теме руководства или рекламе. Перед публикацией пожалуйста обратитесь к нашей [службе поддержки](#) или [сайте сообщества](#).

На этой странице

Прежде чем начать

Как работают плагины Vim

Как написать плагин Vim

А. Анатомия плагина Vim

В. Написание плагина Vim

Как развернуть плагин Vim

А. Добавьте README для вашего плагина

В. Создать Git-репозиторий

Дополнительная информация

ГЗТ В Linux

Как Установить И Использовать  
Zoxide На Linux

Как Установить NeoVim И Плагины  
С Помощью Vim-Plug



© 2003-2024 Linode LLC.  
Все права защищены.

[Управление предпочтениями](#)



[Карта сайта](#)

[Поддерживать](#)

[Партнеры](#)

[Обязательство по обеспечению  
доступности](#)

[Юридический Центр](#)

[Состояние системы](#)