





## Ruby LSP

Shopify  [shopify.com](https://shopify.com) |  392 506 установок  
|      (22) | Бесплатно

Плагин VS Code для подключения к Ruby LSP

### Установка

Запустите VS Code Quick Open (Ctrl+P), вставьте следующую команду и нажмите enter.

```
ext install Shopify.ruby-lsp
```

[Копировать](#)[| Подробная информация](#)[Обзор](#)[История версий](#)[Вопросы и ответы](#)[Оценка и обзор](#)

## Ruby LSP (расширение VS Code)

Ruby LSP - это расширение, предоставляющее высокопроизводительные функции для Ruby. Оно подключается к дем-серверу языка [ruby-lsp](#) для анализа кода Ruby и улучшения взаимодействия с пользователем.

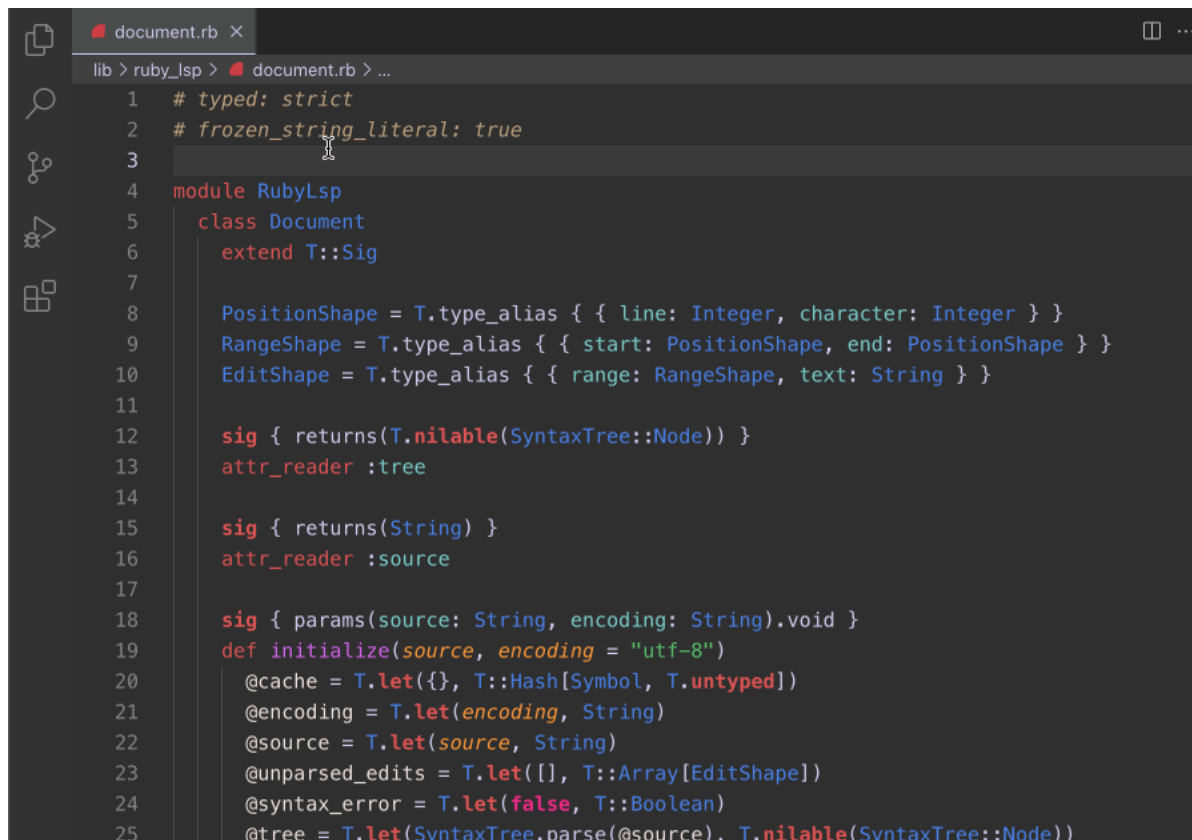
Хотите обсудить опыт разработчиков Ruby? Подумайте о присоединении к общедоступному [рабочему пространству Ruby DX Slack](#).

## Использование

Найдите `Shopify.ruby-lsp` на вкладке расширения и нажмите установить.

По умолчанию Ruby LSP сгенерирует `.ruby-lsp` каталог с пользовательским пакетом, который включает в себя драгоценный камень сервера. Кроме того, он попытается использовать доступные менеджеры версий, чтобы выбрать правильную версию Ruby для любого данного проекта. Дополнительные параметры см. в разделе Конфигурация.

## Характеристики



```
lib > ruby_lsp > document.rb > ...
1 # typed: strict
2 # frozen_string_literal: true
3
4 module RubyLsp
5   class Document
6     extend T::Sig
7
8     PositionShape = T.type_alias { { line: Integer, character: Integer } }
9     RangeShape = T.type_alias { { start: PositionShape, end: PositionShape } }
10    EditShape = T.type_alias { { range: RangeShape, text: String } }
11
12    sig { returns(T.nilable(SyntaxTree::Node)) }
13    attr_reader :tree
14
15    sig { returns(String) }
16    attr_reader :source
17
18    sig { params(source: String, encoding: String).void }
19    def initialize(source, encoding = "utf-8")
20      @cache = T.let({}, T::Hash[Symbol, T.untyped])
21      @encoding = T.let(encoding, String)
22      @source = T.let(source, String)
23      @unparsed_edits = T.let([], T::Array[EditShape])
24      @syntax_error = T.let(false, T::Boolean)
25      @tree = T.let(SyntaxTree.parse(@source), T.nilable(SyntaxTree::Node))
```

Функции Ruby LSP включают в себя

- Семантическое выделение
- Поиск символов и схема кода
- Ошибки и предупреждения RuboCop (диагностика)
- Форматирование при сохранении (с помощью RuboCop или синтаксического дерева)
- Формат зависит от типа
- [Поддержка отладки](#)
- Запуск и отладка тестов через пользовательский интерфейс VS Code
- Перейдите к определению классов, модулей, констант и необходимых файлов
- Отображение документации по классам, модулям и константам при наведении курсора мыши
- Завершение для классов, модулей, констант и требуемых путей
- Нечеткий поиск классов, модулей и констант в любом месте проекта и его зависимостей (символ рабочей области)

Планируется, но еще не завершено добавление поддержки методов определения, завершения, наведения курсора и символа рабочей области.

Смотрите полную информацию о функциях в [документации ruby-lsp\\_server](#).

## Команды

Доступные команды перечислены ниже, и их всегда можно найти, выполнив поиск по Ruby LSP префиксу в палитре команд (горячие клавиши по умолчанию: CMD + SHIFT + P).

Команда	Описание
Ruby LSP: Запуск	Запустите сервер Ruby LSP
Ruby LSP: Перезапуск	Перезапустите сервер Ruby LSP
Ruby LSP: остановить	Остановите сервер Ruby LSP
Ruby LSP: жемчужина языкового сервера обновлений	Обновляет ruby-lsp серверный гем до последней версии

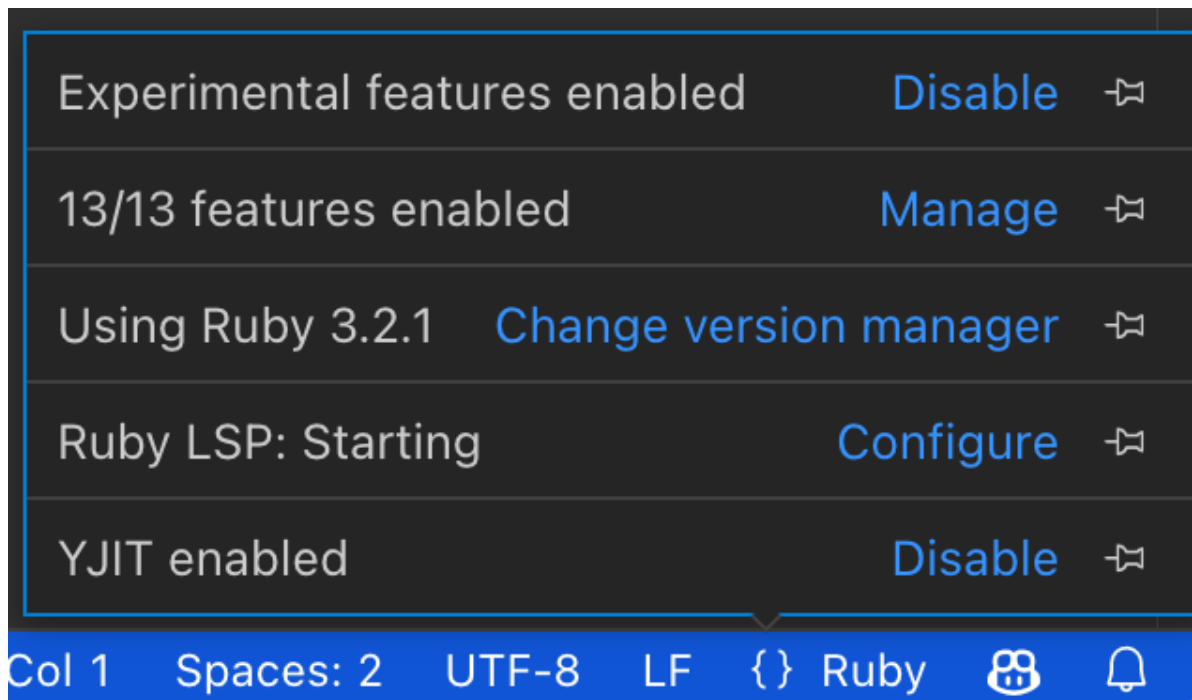
## Фрагменты

Это расширение предоставляет удобные фрагменты для обычных конструкций Ruby, таких как блоки, классы, методы или даже шаблоны модульного тестирования. Полный список можно найти [здесь](#).

## Конфигурация

### Включение или отключение функций

Все его функции по умолчанию включены в Ruby LSP, но поддерживается отключение определенных функций. Для этого откройте центр состояния языка рядом с языковым режимом Ruby и выберите Manage рядом с включенными функциями.



## Менеджеры версий Ruby

Для правильной загрузки сервера Ruby LSP использует диспетчер версий для активации правильных переменных среды, которые указывают Bundler пути к Ruby и gem. Это особенно необходимо при переключении между проектами, использующими разные версии Ruby, поскольку эти пути меняются и их необходимо повторно активировать.

По умолчанию Ruby LSP попытается автоматически определить, какой диспетчер версий ему следует использовать, проверяя, какие из них доступны (auto опция). Если это не удастся, то диспетчер версий необходимо настроить вручную. Вы можете сделать это, щелкнув `Change version manager` в центре состояния языка или изменив пользовательские настройки VS Code.

```
// Available options are
// "auto" (select version manager automatically)
// "none" (do not use a version manager)
// "custom" (use rubyLsp.customRubyCommand for finding/activating Ruby)
// "asdf"
// "chruby"
// "rbenv"
// "rvm"
// "shadowenv"
"rubyLsp.rubyVersionManager": "chruby"
```

Чтобы убедиться, что Руби ЛСП можете найти версию скриптов менеджер, убедитесь, что они загружаются в оболочке скрипт конфигурации (например: `~/.bashrc` следующее, `~/.zshrc`) и что значение переменной окружения `Shell`, расположен и указывая на оболочки по умолчанию.

[!ПРИМЕЧАНИЕ]

Примечания, касающиеся конкретных менеджеров, настройки пользовательской активации для менеджеров, не перечисленных выше, и примеры, предоставленные сообществом, см. в разделе "Менеджеры версий" [.....](#).

## Настройка программы форматирования

Инструмент, который будет использоваться для форматирования файлов, можно настроить с помощью следующих настроек.

```
// Available options
//   auto: automatically detect the formatter based on the app's bundle (default)
//   none: do not use a formatter (disables format on save and related diagnostics)
//   all other options are the name of the formatter (e.g.: rubocop or syntax_tree)
"rubyLsp.formatter": "auto"
```

## Требования к версии Ruby

По умолчанию Ruby LSP использует версию Ruby и пакет Ruby текущего проекта. Это позволяет LSP индексировать правильные версии gem и обеспечивать соответствие поведения при форматировании CI.

Ruby LSP и его основная зависимость [Prism](#) (новый анализатор Ruby) придерживаются одной и той же политики, которая заключается в поддержке только версий Ruby, срок службы которых не истек.

Если вы работаете над проектом с более старой версией Ruby, возможно, удастся установить более старые версии server gem, чтобы получить поддержку более старых rubies, но это также может потребовать использования более старых версий расширения VS Code, поскольку некоторые функциональные возможности требуют реализации как на клиенте, так и на сервере.

Другой альтернативой является использование пользовательского Gemfile отдельно от проекта с другой версией Ruby. Обратите внимание, что некоторые функциональные возможности могут быть ухудшены или требовать настройки вручную, поскольку Ruby LSP не сможет проверить реальный пакет проекта на наличие зависимостей. Пожалуйста, ознакомьтесь с инструкциями ниже.

## Использование пользовательского файла Gemfile

Если вы работаете над проектом, используя более старую версию Ruby, не поддерживаемую Ruby LSP, то вы можете указать отдельный Gemfile для инструментов разработки.

**Примечание:** при использовании этого gems не будут устанавливаться автоматически, как и ruby-lsp обновления.

Создайте каталог для хранения пользовательского пакета вне проекта, использующего старую версию Ruby. Внутри этого каталога добавьте предпочитаемую конфигурацию

диспетчера версий, чтобы выбрать поддерживаемую версию Ruby. Например, при использовании `chruby` это будет выглядеть следующим образом:

```
# the/directory/.ruby-version

3.2.2
```

Создайте `Gemfile` для инструментов разработки внутри этого каталога.

```
# the/directory/Gemfile
# frozen_string_literal: true

source "https://rubygems.org"

gem "ruby-lsp"
gem "rubocop"
```

Запустите `bundle install` внутри этого каталога, чтобы сгенерировать файл блокировки. После того, как каталог будет содержать пользовательскую `Gemfile` и конфигурацию диспетчера версий, используйте следующую конфигурацию в VS Code, чтобы указать Ruby LSP на это `Gemfile`.

```
{
  "rubyLsp.bundleGemfile": "../../relative/path/to/the/directory/Gemfile",
  "rubyLsp.bundleGemfile": "/absolute/path/to/the/directory/Gemfile"
}
```

## Настройка отладчика VS Code

Чтобы настроить отладчик VS Code, вы можете использовать команду "Debug: добавить конфигурацию..." для создания `launch.json` файла в `.vscode` каталоге вашего проекта.

Эта команда сгенерирует следующую конфигурацию:

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "ruby lsp",
      "name": "Debug",
      "request": "launch",
      "program": "ruby ${file}"
    },
    {
      "type": "ruby lsp",
      "request": "launch",
      "name": "Debug test file",
      "program": "ruby -Itest ${relativeFile}"
    },
    {
      "type": "ruby_lsp",
```

```

    "request": "attach",
    "name": "Attach to existing server"
  }
]
}

```

## ПРОТИВ конфигураций кода

В дополнение к собственной конфигурации Ruby LSP, существуют некоторые настройки VS Code, которые, возможно, потребуется изменить, чтобы максимально использовать Ruby LSP. Эти параметры не являются специфичными для Ruby LSP, но они влияют на все языковые серверы и имеют приоритет над любыми другими конфигурациями.

Это параметры, которые могут влиять на поведение Ruby LSP и их объяснения.

```

{
  // All of these settings are scoped only to the Ruby language
  "[ruby]": {
    "editor.defaultFormatter": "Shopify.ruby-lsp", // Use the Ruby LSP as the default
    "editor.formatOnSave": true, // Format files automatically when saving
    "editor.tabSize": 2, // Use 2 spaces for indentation
    "editor.insertSpaces": true, // Use spaces and not tabs for indentation
    "editor.semanticHighlighting.enabled": true, // Enable semantic highlighting
    "editor.formatOnType": true // Enable formatting while typing
  }
}

```

## Устранение неполадок

Чтобы проверить, правильно ли активирован Ruby LSP, вы можете

- Проверьте, работают ли какие-либо функции, такие как форматирование при сохранении или структура файла
- Откройте Output панель VS Code, выберите Ruby LSP канал и проверьте, был ли Ruby LSP is ready напечатан

Если Ruby LSP не запускается, выполните следующие действия

1. Дважды проверьте, настроен ли правильный [Ruby version manager](#)
2. Дважды проверьте, присутствуют ли все требования к менеджеру версий. Например, chruby требуется, чтобы .ruby-version файл существовал на верхнем уровне проекта
3. Если вы используете версию 0.2.0 этого расширения или выше, дважды проверьте, что ruby-lsp драгоценный камень отсутствует в основной версии проекта. Gemfile
4. Перезагрузите окно VS Code, открыв палитру команд и выбрав Developer: Reload window

Если эти шаги не устраняют проблему с инициализацией, попробуйте вручную установить gems, используя пользовательский интерфейс Ruby LSPGemfile, выполнив команду.

```
BUNDLE_GEMFILE=/path/to/your/project/.ruby-lsp/Gemfile bundle install
```

Если после этих шагов Ruby LSP по-прежнему не инициализируется должным образом, пожалуйста, сообщите о проблеме [здесь](#).

## Переход из пакета

---

**Примечание:** при переходе с версии, более ранней, чем v0.2.0, применяется следующее.

Если вы ранее включали ruby-lsp gem в пакет (как часть Gemfile or gemspec проекта), выполните следующие действия для перехода на более новые версии Ruby LSP, для которых gem больше не нужно добавлять в пакет.

1. Предупредите разработчиков, работающих над проектом, что им необходимо обновить расширение Ruby LSP до последней версии (для более старых версий требуется ruby-lsp gem из пакета и, следовательно, он не будет работать, если его удалить)
2. Удалить ruby-lsp из пакета (удалить запись из Gemfile проекта)
3. Запустите пакет, чтобы убедиться, что он обновлен Gemfile.lock
4. [Перезапустите](#) расширение Ruby LSP или перезапустите VS Code, чтобы разрешить Ruby LSP использовать новую настройку

## Телеметрия

---

Сам по себе Ruby LSP по умолчанию не собирает никакой телеметрии, но при желании поддерживает подключение к частной службе показателей.

Чтобы получать запросы метрик, частный плагин должен экспортировать ruby-lsp.getPrivateTelemetryApi команду, которая должна возвращать объект, реализующий TelemetryApi интерфейс, определенный [здесь](#).

Поля, включенные по умолчанию, определены TelemetryEvent [здесь](#). Экспортированный объект API может добавлять любые другие интересные данные и публиковать их в частной службе.

Например,

```
// Create the API class in a private plugin
class MyApi implements TemeletryApi {
  sendEvent(event: TelemetryEvent): Promise<void> {
    // Add timestamp to collected metrics
    const payload = {
      timestamp: Date.now(),
      ...event,
    }
  }
}
```



```
};

// Send metrics to a private service
myFavouriteHttpClient.post("private-metrics-url", payload);
}
}

// Register the command to return an object of the API
vscode.commands.registerCommand(
  "ruby-lsp.getPrivateTelemetryApi",
  () => new MyApi(),
);
```

## Форматирование

---

Когда `rubyLsp.formatter` установлено значение `auto`, Ruby LSP пытается определить, какой форматировщик использовать.

Если пакет имеет **прямую** зависимость от поддерживаемого средства форматирования, такого как `rubocop` или `syntax_tree`, он будет использоваться. В противном случае форматирование будет отключено, и вам нужно будет добавить его в пакет.

## Вклад

---

Сообщения об ошибках и запросы на извлечение приветствуются на GitHub по адресу <https://github.com/Shopify/vscode-ruby-lsp>. Этот проект предназначен, чтобы быть безопасным и гостеприимным пространством для совместной работы, и авторы должны придерживаться [участником пакта](#) кодекса поведения.

Прежде чем вносить свой вклад, пожалуйста, не забудьте [подписать Лицензионное соглашение с участником](#).

## Отладка

Интерактивная отладка работает как для запуска расширения, так и для тестов. На панели "Отладка" выберите, запускать ли расширение в режиме разработки или запускать тесты, установите некоторые точки останова и начните с F5.

## Отслеживание запросов и ответов на LSP

Трассировкой сервера LSP можно управлять с помощью `ruby lsp.trace.server` конфигурационного ключа в `.vscode/settings.json` файле конфигурации.

Возможные значения следующие:

- `off`: трассировка отсутствует
- `messages`: отображать уведомления о запросах и ответах
- `verbose`: отображать каждый запрос и ответ в формате JSON

## Debugging the server using VS Code

The `launch.json` contains a 'Minitest - current file' configuration for the debugger.

1. Add a breakpoint using the VS Code UI.
2. Open the relevant test file.
3. Open the **Run and Debug** panel on the sidebar.
4. Ensure Minitest - current file is selected in the top dropdown.
5. Press F5 OR click the green triangle next to the top dropdown. VS Code will then run the test file with debugger activated.
6. When the breakpoint is triggered, the process will pause and VS Code will connect to the debugger and activate the debugger UI.
7. Open the Debug Console view to use the debugger's REPL.

## License

This extension is available as open source under the terms of the [MIT License](#).

### Категории

[Языки программирования](#)[Фрагменты](#)[Тестирование](#)

### Теги

[отладчики](#)[ошибка](#)[ruby](#)[фрагмент](#)

### Работает с

Универсальный

### Ресурсы

[Проблемы](#)[Репозиторий](#)[Домашняя страница](#)[Лицензия](#)[Журнал изменений](#)[Скачать расширение](#)

### Подробная информация о проекте

 [Shopify/vscode-ruby-lsp](#) Последний коммит: 4 дня назад 11 запросов на извлечение 37 открытых вопросов

### Подробная информация

Версия 0.5.0

Выпущен на 03.05.2022, 01:59:13

Последнее обновление 09.11.2023, 23:21:34

Издатель Shopify

Уникальный идентификатор Shopify.ruby-lsp

Сообщить [Сообщить о проблеме](#)



[Contact us](#) [Jobs](#) [Privacy](#) [Terms of use](#) [Trademarks](#)

© 2023 Microsoft