

5. Documentation for the Performance interface [Electronic resource] URL: <https://developer.mozilla.org/ru/docs/Web/API/Performance> (accessed: 05.11.2022).

6. Examples of using the Performance interface [Electronic resource] URL: <https://habr.com/ru/company/skillfactory/blog/538334/> (accessed: 11/20/2022).

*Статья поступила в редакцию 03.11.2022, одобрена после рецензирования 25.11.2022, принята к публикации 25.11.2022.*

*The article was submitted 03.11.2022; approved after reviewing 25.11.2022; accepted for publication 25.11.2022.*

Научная статья

УДК 004.4

## ОРГАНИЗАЦИЯ ПРОЦЕССА СБОРКИ И ПУБЛИКАЦИИ СТАТИЧЕСКОГО ВЕБ-САЙТА В СООТВЕТСТВИИ С ПРАКТИКОЙ CI/CD

**Федор Владиславович Мельников**

Российский государственный педагогический университет им. А. И. Герцена,

Санкт-Петербург, Россия

balrun.dev@gmail.com

**Аннотация.** В статье рассматриваются методы организации непрерывной интеграции и непрерывного развёртывания веб-сайтов, разработанных с помощью генераторов статических сайтов. Выделены преимущества их применения. Рассмотрены стадии процесса сборки и тестирования сайта. Выделены различные методы автоматизированного тестирования, которые применимы к статическим сайтам и могут быть включены в процесс интеграции. Приведён пример реализации процесса сборки и публикации сайта, разработанного с использованием генератора статических сайтов MkDocs, в среде GitHub Actions.

**Ключевые слова:** генератор статических сайтов, непрерывная интеграция, непрерывное развёртывание, система контроля версий, инструменты разработки сайтов

**Для цитирования:** Мельников Ф. В. Организация процесса сборки и публикации статического веб-сайта в соответствии с практикой CI/CD // Научно-технические инновации и веб-технологии. 2022. № 2. С. 34-40.

Original article

## ORGANIZING THE BUILD AND PUBLISHING PROCESS OF A STATIC WEBSITE ACCORDING TO CI/CD PRINCIPLES

**Fedor V. Melnikov**

The Herzen State Pedagogical University of Russia, Saint-Petersburg, Russia

balrun.dev@gmail.com

**Abstract.** The article discusses methods for organizing continuous integration and continuous deployment of websites developed using static site generators. The advantages of their application are highlighted. The stages of the process of building and testing the site are considered. Various automated testing methods that are applicable to static sites and can be included in the integration process, are identified. An example of the implementation of the process of building and publishing a site, developed using the MkDocs static site generator, in the GitHub Actions environment is given.

**Keywords:** static site generator, continuous integration, continuous deployment, version control system, site development tools

**For citation:** Melnikov F. V. Organizing the build and publishing process of a static website according to CI/CD principles // Scientific and technical innovations and web technologies. 2022. no. 2. P. 34-40.

Статические веб-сайты имеют широкую сферу применения. В случаях, когда разработка динамического сайта не является целесообразной, применение технологий статических сайтов позволяет упростить процесс разработки. Динамические сайты, как правило, редактируются с помощью системы управления контентом, изменения отображаются сразу после сохранения. Публикация статических сайтов предполагает предварительную генерацию страниц и может производиться как в ручном, так и в автоматическом режиме. Автоматизация процесса публикации позволяет упростить разработку сайта и уменьшить время, необходимое для обновления информации на сервере. В связи с этим, является актуальным исследование методов автоматизированной сборки и публикации статических сайтов.

В. М. Никинтинская, Ю. А. Сокуренок, И. Б. Государев отмечают, что ресурсы, выполненные в виде статических сайтов, направлены на предоставление информации пользователю, и контент обновляется сравнительно нечасто [1].

Статический сайт состоит из HTML-страниц, каскадных таблиц стилей, кода на языке JavaScript.

Разработка статического сайта «вручную» является нецелесообразной. Данный подход имеет следующие недостатки:

- необходимость ручного копирования шаблона страницы, что не только приводит к необходимости выполнения повторяющихся операций, но и существенно затрудняет внесение изменений в шаблон (при этом должна быть изменена каждая страница);

- необходимость ручной разметки контента;

- невозможность обеспечения консистентности данных, т.к. не обеспечивается правильность гиперссылок.

В современной веб-разработке под статическим сайтом чаще всего понимается сайт, разработанный с помощью генератора статических сайтов (static site generator) – системы, которая производит генерацию HTML-страниц на основе исходных файлов. Контент сайта описывается на языках разметки (например, с помощью Markdown).

Статические сайты показывают меньшее время загрузки страницы по сравнению с динамическими. Также они являются более безопасными. Это связано с тем, что сервер отдаёт клиенту готовые файлы, при этом не производится обращение к серверной части, базе данных [2].

Статические сайты являются переносимыми, их можно просматривать локально и копировать на другие серверы без изменений.

Другой важной особенностью является независимость сайта по сравнению с ресурсами, разрабатываемыми и размещаемыми на публичных платформах.

В классификации инструментов генерации статических сайтов, предложенной В. М. Никинтинской и И. Б. Государевым, в классе «Язык и экосистема» выделяются генераторы, написанные на языке JavaScript и генераторы, написанные на других языках [3]. Как правило, инструменты первой группы интегрированы с такими фреймворками, как React, Vue, Svelte. Их применение часто предполагает использование инструмента транспилляции и системы сборки. Инструменты второй группы осуществляют сборку из файлов разметки без использования JavaScript, пользователь редактирует текстовые файлы.

Использование генератора статических сайтов позволяет автоматизировать процесс сборки страниц. Вместе с тем, не все генераторы позволяют публиковать сайт в автоматическом режиме. Ручная загрузка сайта на сервер увеличивает время, необходимое для публикации обновлённой версии.

В самом простом случае разработка сайта производится локально. После редактирования страниц готовый сайт проверяется в ручном режиме, после чего файлы копируются на удалённый сервер. Этот способ не предполагает автоматизации процесса публикации.

Некоторые генераторы, например, Lektor и MkDocs, предоставляют возможность загрузки готового сайта в репозиторий GitHub для публикации с использованием GitHub Pages. Важно отметить, что при этом разработка может также производиться локально, в т.ч. без использования системы контроля версий. Такой подход не предполагает редактирование страниц на разных устройствах или разными пользователями.

Встроенные средства генераторов предлагают ограниченный набор способов автоматической публикации.

Важно отметить, что целесообразной может быть разработка сайта с применением системы контроля версий. Размещение исходных файлов в GitHub или GitLab позволяет не только использовать возможности Git, но и автоматизировать процесс сборки и публикации сайта.

Автоматизация процесса сборки и публикации сайта может производиться с помощью систем непрерывной интеграции и непрерывного развёртывания. К ним относятся, например, GitHub Actions и GitLab CI/CD. Применение указанных систем в данном случае может быть более предпочтительным по сравнению с другими системами, такими как Jenkins, Circle CI и др. Это связано с тем, что они интегрированы с платформой для размещения кода и по умолчанию доступны для использования. Вместе с тем, сборка статического сайта является сравнительно простым процессом, не требующим специальных функций, которые поддерживают другие системы CI/CD.

Непрерывная интеграция (continuous integration, CI) в данном контексте заключается в выполнении автоматизированных сборок проекта. При этом исходные файлы хранятся в репозитории системы контроля версий.

Непрерывное развёртывание (continuous deployment, CD) заключается в обновлении сайта короткими итерациями, при этом изменения загружаются на сервер в автоматизированном режиме.

Непрерывная интеграция и развёртывание предполагают выполнение различных стадий в автоматическом режиме после загрузки изменений в репозиторий. Пример организации CI/CD для разработки программного продукта представлен на рис. 1.



Рисунок 1 – пример стадий CI/CD для программного продукта

Разработка статического сайта с использованием инструментов, которые не требуют использования JavaScript, не предполагает написания и отладки программного кода. В связи с этим, процесс сборки и публикации может состоять из меньшего количества стадий (рис. 2).



Рисунок 2 – пример стадий CI/CD для статического сайта

Логическое разделение стадий может не отражаться в системе CI/CD, они могут быть запущены в одной среде выполнения.

В данном примере отсутствует стадия тестирования. Для небольших сайтов иногда достаточно визуальной проверки страниц собранного сайта перед отправкой изменений в репозиторий. Во время редактирования страниц может быть запущен встроенный веб-сервер генератора, который обновляет редактируемые страницы в режиме реального

времени. После внесения изменений производится загрузка в репозиторий, затем запускается процесс сборки и публикации сайта на сервере.

Визуальное тестирование сайта при применении CI/CD может быть организовано с помощью специальных инструментов [4]. К ним относятся, например, Percy и Applitools.

Процесс сборки статического сайта, разработанного с использованием инструментов первой группы (в соответствии с рассмотренной выше классификацией), может включать в себя линтинг (проверку исходного кода), транспиляцию, тестирование JavaScript-кода (например, с помощью Selenium или Cypress). Для сборки статического сайта, разработанного с использованием инструментов второй группы, не требуется выполнения перечисленных операций.

Тестирование исходных файлов сайта (для инструментов второй группы), как правило, не выделяется как отдельная стадия. Проверка валидности файлов, содержащих текст с разметкой на языках Markdown, reStructuredText, AsciiDoc и других, до начала сборки не имеет смысла, т.к. в случае наличия ошибок в исходных файлах сборка будет прервана генератором, и последующие стадии не будут выполнены.

Можно выделить следующие виды автоматизированного тестирования статических сайтов:

- проверка правописания,
- тестирование JavaScript,
- проверка гиперссылок (поиск неправильных ссылок),
- в некоторых случаях может быть целесообразным выполнение тестов содержимого страниц (content-based tests).

Существует большое количество способов размещения статических сайтов: хостинги, виртуальные (VPS) и выделенные серверы, облачные платформы, некоторые объектные хранилища, сети доставки содержимого (CDN), специальные сервисы (например, GitHub Pages). Организация процесса публикации сайта зависит от выбранного способа его размещения.

Рассмотрим процесс сборки и публикации статического сайта с использованием GitHub Actions.

Рабочий процесс (workflow) состоит из одного или нескольких заданий (jobs), которые выполняются последовательно или параллельно. Каждое задание (стадия) включает определённые шаги (steps) и выполняется в отдельном контейнере (изолированной среде). Запуск производится при наступлении определённого события. Схема рабочего процесса представлена на рис. 3.

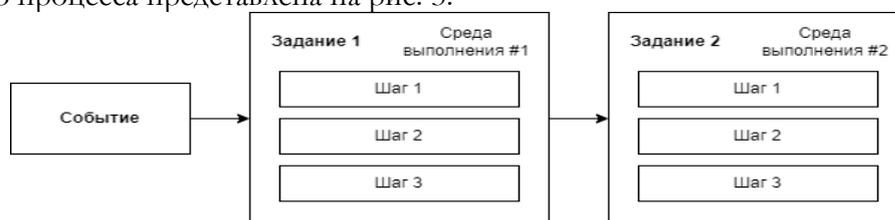


Рисунок 3 – рабочий процесс в среде GitHub Actions

Процесс описывается в файле YAML, который добавляется в репозиторий. Файл состоит из нескольких разделов: название процесса, событие, при котором выполняется запуск, описание заданий.

Сборка и публикация статического сайта может выполняться внутри одного задания, в одном контейнере, подобно тому, как она осуществляется локально.

Можно выделить следующие этапы процесса, которые необходимы для сборки и публикации сайта, разработанного с использованием любого генератора: подготовка среды выполнения, выполнение сборки, развёртывание на сервере.

Подготовка среды выполнения включает в себя копирование файлов репозитория в среду выполнения (checkout) и установку генератора статических сайтов.

Сборка осуществляется генератором аналогично тому, как это выполняется на локальном компьютере.

Развёртывание, как правило, включает подготовку (например, установку SSH-ключей) и копирование файлов на сервер.

Выделение стадий в разные задания не имеет смысла, т.к. это не только приводит к необходимости повторного выполнения копирования, но и увеличивает время сборки.

Выбор генератора не имеет существенного влияния на организацию процесса сборки. Он обусловлен потребностями и предпочтениями пользователя. По мнению автора данной статьи, для небольших сайтов (до 100 страниц) производительность генератора не оказывает значимого влияния на время выполнения процесса сборки и развёртывания, т.к. подготовка среды выполнения занимает значительно больше времени, чем работа генератора. Однако, существуют публикации, в которых отмечается, что на скорость сборки напрямую влияет количество файлов сайта [5]. Тем не менее, время сборки небольших сайтов довольно мало по отношению ко времени, которое необходимо для подготовки среды системы CI/CD.

Процессы CI/CD небольшого сайта без автоматизированного тестирования являются аналогичными для большинства генераторов. Пример организации процесса сборки и публикации сайта, разработанного с помощью генератора MkDocs, в среде GitHub Actions представлен на рис. 4.

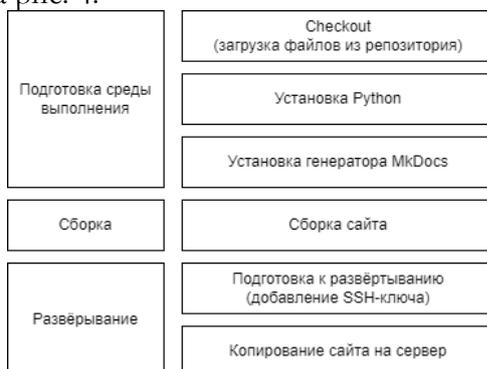


Рисунок 4 – организация процессов CI/CD для статического сайта, разработанного с помощью MkDocs

Рассмотрим реализацию данного процесса (рис. 5). Процесс выполняется при загрузке изменений в репозиторий. Он содержит одно задание – Build and Deploy и выполняется в одной изолированной среде (виртуальной машине Ubuntu). Собранный сайт копируется на удалённый сервер.

```
name: Build and Deploy

on:
  push:

jobs:
  build:
    name: Build and Deploy
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v3

      - name: Python Setup
        uses: actions/setup-python@v4
        with:
          python-version: 3.11.1

      - name: MkDocs Setup
        run: pip install mkdocs

      - name: Build
        run: mkdocs build

      - name: Install SSH key
        run: |
          install -m 600 -D /dev/null ~/.ssh/id_rsa
          echo "${{ secrets.PRIVATE_SSH_KEY }}" > ~/.ssh/id_rsa
          echo "${{ secrets.KNOWN_HOSTS }}" > ~/.ssh/known_hosts

      - name: Deploy
        run: rsync --archive --stats site/ ${{ secrets.REMOTE_DEST }}
```

Рисунок 5 – содержимое файла с описанием рабочего процесса

Задание состоит из 6 этапов:

- Checkout – копирование файлов репозитория в среду выполнения;
- Python Setup – установка интерпретатора Python;
- MkDocs Setup – установка генератора статических сайтов MkDocs (применяется приложение setup-python);
- Build – выполнение сборки сайта;
- Install SSH key – копирование SSH ключа и добавление удалённого сервера в список известных;
- Deploy – развёртывание сайта на удалённом сервере.

Перед первым развёртыванием необходимо произвести настройку удалённого сервера – установить веб-сервер (например, Apache или Nginx), сгенерировать SSH-ключ для доступа без пароля.

В файле используются зашифрованные переменные окружения – Secrets, значения которых задаются в настройках репозитория. Указывать значения переменных напрямую в файле описания рабочего процесса не рекомендуется по соображениям безопасности.

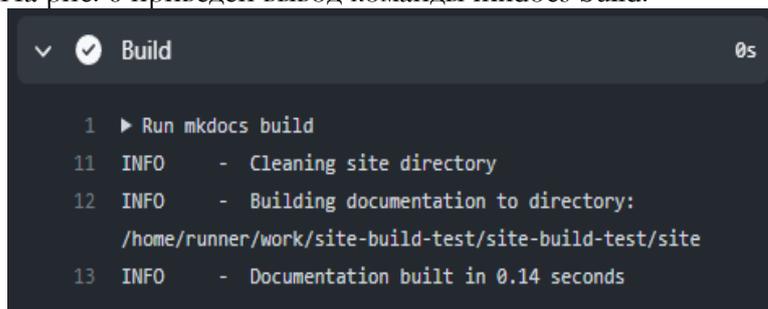
В переменной PRIVATE\_SSH\_KEY задаётся приватный ключ. Важно отметить, что генерация ключа без парольной фразы позволяет не использовать дополнительные утилиты, например, ssh-agent. Ручной ввод парольной фразы невозможен, т.к. рабочий процесс выполняется в автоматическом режиме.

Значение переменной KNOWN\_HOSTS представляет собой список открытых ключей известных серверов. При настройке рабочего процесса необходимо получить ключ сервера с помощью утилиты ssh-keyscan.

В переменной REMOTE\_DEST указывается путь для загрузки файлов на удалённом сервере. Он включает имя пользователя, адрес сервера и путь к директории.

Копирование сайта на удалённый сервер осуществляется с помощью утилиты rsync. Она передаёт на сервер только изменения файлов, что положительно сказывается на скорости копирования.

GitHub Actions позволяет отслеживать процесс выполнения. Отображается вывод команд всех шагов. На рис. 6 приведён вывод команды mkdocs build.



```
Build 0s
1  ▶ Run mkdocs build
11 INFO - Cleaning site directory
12 INFO - Building documentation to directory:
/home/runner/work/site-build-test/site-build-test/site
13 INFO - Documentation built in 0.14 seconds
```

Рисунок 6 – информация о сборке сайта

Сборка сайта, разработанного с помощью MkDocs и содержащего 10 страниц, осуществляется в среднем за 150 мс. Время выполнения рабочего процесса для данного сайта составляет в среднем 25 секунд.

Применение инструментов CI/CD позволяет ускорить и упростить процесс внесения изменений при работе со статическими сайтами. При этом возможна не только автоматизация процесса сборки и публикации, но и организация различных видов тестирования. Методы организации CI/CD схожи при применении разных генераторов статических сайтов.

## Список источников

1) Никитинская В. М., Сокуренок Ю. А., Государев И. Б. Обзор современных практик создания статических веб-сайтов // Альманах научных работ молодых ученых Университета ИТМО. Том 7. – СПб: Университет ИТМО, 2018. – С. 229-231.

- 2) Rinaldi B. Static Site Generators. – O’Reilly Media, Inc., 2015. – URL: <https://www.oreilly.com/content/static-site-generators/>.
- 3) Никитинская В. М., Государев И. Б. Классификация инструментов генерации статических веб-сайтов // Альманах научных работ молодых учёных Университета ИТМО. Том 1. – СПб: Университет ИТМО, 2019. – С. 216-219.
- 4) Visual Testing | Cypress Documentation [Электронный ресурс]. URL: <https://docs.cypress.io/guides/tooling/visual-testing> (дата обращения: 05.12.2022).
- 5) Comparing Static Site Generator Build Times [Электронный ресурс]. URL: <https://css-tricks.com/comparing-static-site-generator-build-times/> (дата обращения: 05.12.2022).

## References

- 1) Nikitinskaya V. M., Sokurenko Yu. A., Gosudarev I. B. Review of modern practices for creating static websites // Almanac of scientific works of young scientists of ITMO University. Volume 7. – St. Petersburg: ITMO University, 2018. – pp. 229-231.
- 2) Rinaldi B. Static Site Generators. – O’Reilly Media, Inc., 2015. – URL: <https://www.oreilly.com/content/static-site-generators/>.
- 3) Nikitinskaya V. M., Gosudarev I. B. Classification of tools for generating static websites // Almanac of Scientific Works of Young Scientists of ITMO University. Volume 1. – St. Petersburg: ITMO University, 2019. – pp. 216-219.
- 4) Visual Testing | Cypress Documentation [Electronic resource]. URL: <https://docs.cypress.io/guides/tooling/visual-testing> (date of application: 12/05/2022).
- 5) Comparing Static Site Generator Build Times [Electronic resource]. URL: <https://css-tricks.com/comparing-static-site-generator-build-times/> (date of application: 12/05/2022).

*Статья поступила в редакцию 03.11.2022, одобрена после рецензирования 25.11.2022, принята к публикации 25.11.2022.*

*The article was submitted 03.11.2022; approved after reviewing 25.11.2022; accepted for publication 25.11.2022.*

Научная статья  
УДК 004.05

## СРАВНЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СЕРВЕРНЫХ РЕШЕНИЙ НА БАЗЕ PHP И NODE.JS

**Павел Игоревич Сигети**

Федеральное государственное автономное образовательное  
учреждение высшего образования  
«Национальный исследовательский университет ИТМО»,  
Санкт-Петербург, Россия,  
[sigeti385@gmail.com](mailto:sigeti385@gmail.com)

**Аннотация.** В статье сравнивается производительность двух веб-серверных технологий: PHP и Node.js. Полученные результаты могут быть использованы в качестве рекомендации при выборе технологии.

**Ключевые слова:** серверные веб-технологии, backend, PHP, Node.js, JavaScript

**Для цитирования:** Сигети П.И. Сравнение производительности серверных решений на базе PHP и Node.js // Научно-технические инновации и веб-технологии. 2022. № 2. С. 40-44.