**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**«Национальный исследовательский университет ИТМО»**

**(Университет ИТМО)**

**Факультет: ПИиКТ**

**Образовательная программа: Веб-технологии**

**Направление подготовки (специальность) 09.04.04 Программная инженерия**

О Т Ч Е Т

*по* *научно-исследовательской работе*

Тема задания: **Создание и развертывание статического сайта.**

Обучающийся *Механиков Д.И., P4109*

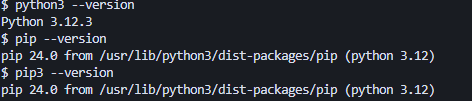
Преподаватель:

Жуков Николай Николаевич, кандидат физико-математических наук, факультет ПИиКТ

Дата **24.09.2025**

Открываем VS Code подключаемся по SSH к серверу.

1. Устанавливаем Python актуальной версии, проверяем работает ли pip (может запускаться из консоли по команде pip или pip3) и virtualenv.

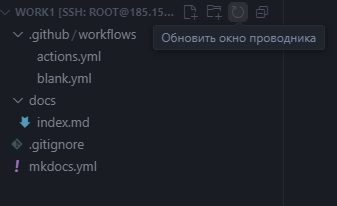


Python и pip установлены, Из-за externally-managed-environment будем использовать встроенный python3 -m venv. Далее создам виртуальное окружение

1. Создаем каталог и там развертываем виртуальное окружение, активируем его.

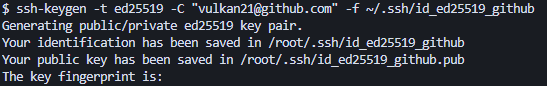


1. Создам файлы, которые были представлены в main из видео



Содержание файлов можно найти в гитхабе в созданном репозитории.

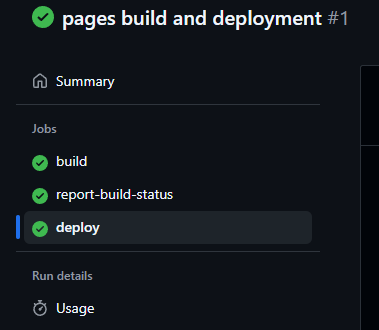
1. Создаем ssh ключ и добавляем публичный ключ в гитхаб



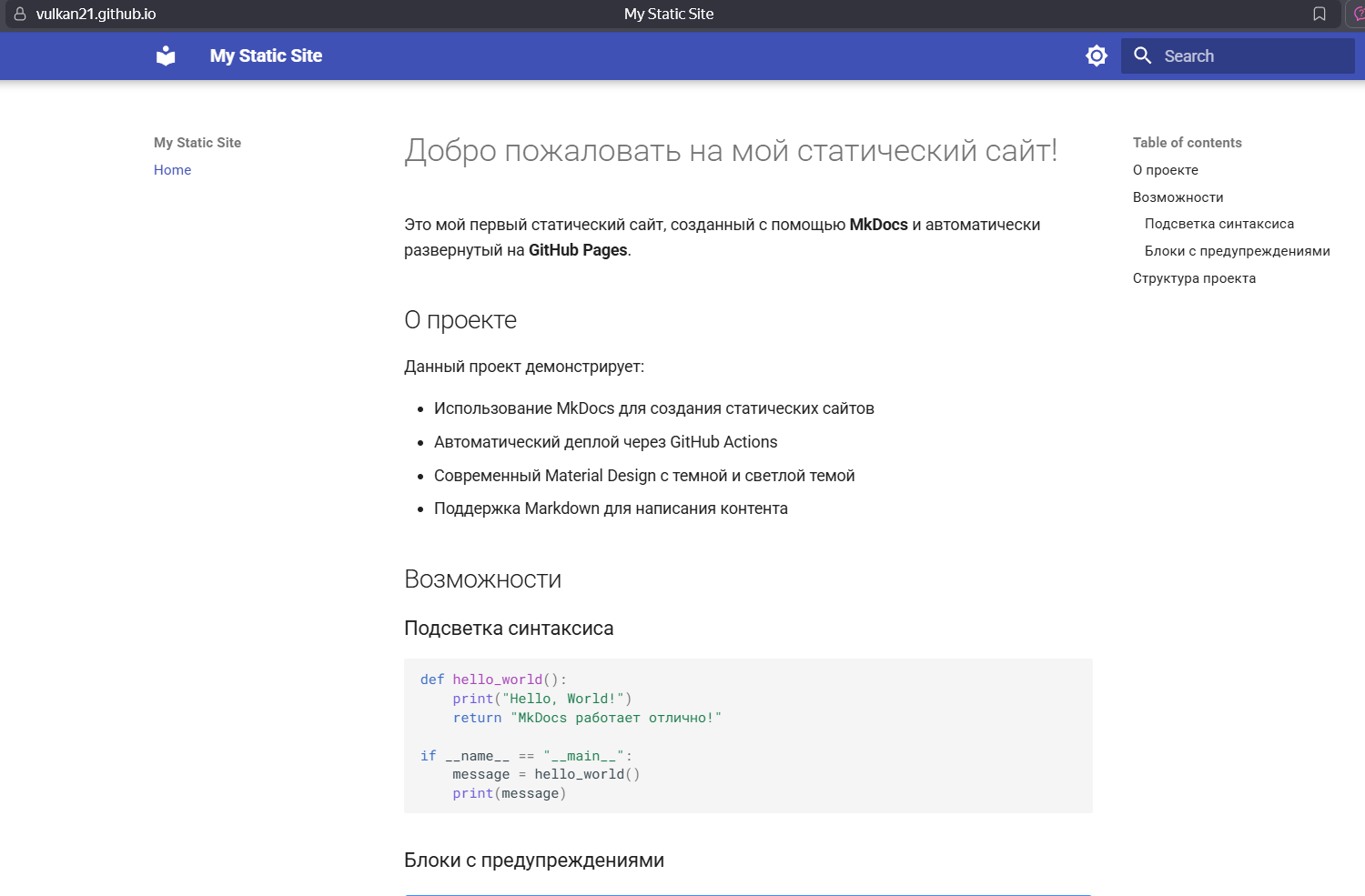
1. Пушим изменения в репозиторий.



1. Обязательно проверяем, что в GitHub Pages в Branch выбрана ветка main, иначе будут ошибки.
2. Проверяем, что все задеплоилось.



1. Сайт готов(https://vulkan21.github.io/site222/)



## Возможности Gitverse для реализации CI/CD.

Одним из ключевых этапов внедрения CI/CD является организация автоматического запуска рабочих процессов (workflow) при изменении состояния репозитория. В GitVerse данная функциональность реализуется посредством встроенного механизма GitVerse Actions, который по своей архитектуре является функциональным аналогом GitHub Actions и поддерживает совместимость с его синтаксисом конфигурационных файлов.

Основным принципом запуска пайплайнов в GitVerse выступает реакция на определённые события (events), происходящие в системе контроля версий. Наиболее распространёнными триггерами являются:

* push — событие, возникающее при фиксации изменений в ветке репозитория. Оно используется для проверки корректности нового кода, его сборки и тестирования на ранних стадиях.
* merge request (pull request) — событие, инициируемое при попытке объединения изменений из одной ветки в другую. Данный триггер применяется для обеспечения качества кода, так как позволяет выполнить автоматизированные проверки до интеграции изменений в основную ветку проекта.

В GitVerse настройка базового workflow осуществляется посредством файла конфигурации в формате YAML, размещаемого в каталоге .gitverse/workflows внутри репозитория. В указанном файле определяются:

* условия срабатывания (on: push, on: merge\_request);
* последовательность выполняемых задач (jobs);
* окружения и зависимости, необходимые для запуска.

Таким образом, даже минимальная конфигурация пайплайна позволяет автоматизировать проверку кода на каждом этапе его интеграции. В образовательной и исследовательской практике это обеспечивает воспроизводимость экспериментов, а в промышленной разработке — снижение вероятности ошибок при ручном выполнении однотипных процедур.

Эффективное использование CI/CD-пайплайнов предполагает работу не только с исходным кодом, но и с внешними параметрами: ключами доступа, токенами, паролями, конфигурационными значениями и URL-адресами. В условиях автоматизации критически важно обеспечить безопасное хранение и передачу этих данных. Для решения данной задачи в GitVerse предусмотрен отдельный механизм управления секретами (*secrets*) и переменными окружения (*variables*).

**Секреты (secrets)** представляют собой конфиденциальные данные, которые могут быть зашифрованы и доступны исключительно в рамках выполнения рабочего процесса (workflow). К типичным примерам относятся:

* учётные данные для доступа к приватным контейнерным реестрам;
* токены API внешних сервисов;
* SSH-ключи для удалённых серверов;
* пароли к базам данных.

Секреты задаются на уровне репозитория или организации и не могут быть прочитаны напрямую через интерфейс или из кода — они доступны исключительно в момент выполнения job. Таким образом, реализуется принцип минимизации экспозиции критических данных.

**Переменные (variables)**, в отличие от секретов, не являются строго конфиденциальными и применяются для хранения параметров конфигурации, которые могут отличаться в зависимости от окружения. К примеру:

* URL тестового и промышленного сервисов;
* номера версий приложения;
* пути до директорий сборки или размещения артефактов.

Использование переменных упрощает сопровождение проектов, так как позволяет централизованно изменять параметры, не внося правки в каждый workflow-файл.

В GitVerse предусмотрена возможность привязки переменных и секретов к конкретным окружениям (*environments*). Это позволяет выстраивать многоступенчатые пайплайны (например, dev → stage → prod), при которых для каждой среды автоматически применяются соответствующие параметры. Такой подход снижает вероятность ошибок, связанных с использованием неверных конфигурационных значений, и повышает уровень безопасности.

С точки зрения безопасности информационных систем, механизм управления секретами и переменными в GitVerse соответствует современным требованиям DevOps-практики. Он минимизирует риск утечки чувствительных данных при интеграции с внешними сервисами, а также обеспечивает воспроизводимость и масштабируемость процессов при переносе приложений между окружениями.

Ключевым преимуществом GitVerse является **локализация и соответствие требованиям российского рынка**. В условиях ограниченного доступа к зарубежным сервисам отечественная платформа обеспечивает юридическую и техническую доступность, что делает её привлекательной для государственных и коммерческих организаций.

Вторая сильная сторона заключается в **интеграции с RuStore** — национальной платформой распространения мобильных приложений. GitVerse предоставляет возможность автоматизации публикации приложений в RuStore, что делает его особенно полезным для разработчиков мобильных решений.

Третьим значимым преимуществом является **поддержка импорта репозиториев с GitHub**, включая перенос истории и workflow-файлов. Благодаря совместимости синтаксиса GitVerse Actions с GitHub Actions процесс миграции проектов минимизирует издержки и позволяет быстро адаптировать уже существующие CI/CD-пайплайны.

Наконец, важным достоинством можно считать **простоту интерфейса и настройки**. GitVerse ориентирован на разработчиков, не обладающих глубокими DevOps-компетенциями, что делает его удобным для учебных целей и внедрения в малых командах.

При всех преимуществах GitVerse обладает рядом ограничений, обусловленных его молодостью как экосистемы. Наиболее заметным является **ограниченное количество готовых экшенов (actions)** по сравнению с GitHub Marketplace. Это вынуждает команды либо разрабатывать собственные действия, либо адаптировать существующие под совместимость с платформой.

Кроме того, **интеграции с внешними сервисами остаются ограниченными**. Если для GitHub/GitLab существует широкий спектр готовых решений (от мониторинга до cloud-провайдеров), то в GitVerse набор интеграций пока невелик, что может осложнить реализацию сложных CI/CD-сценариев.

Также необходимо учитывать, что **экосистема GitVerse относительно молода**, поэтому часть функций находится в стадии доработки. Это может приводить к нестабильности работы отдельных модулей и необходимости дополнительных ручных настроек.

Перспективы развития GitVerse во многом связаны с ростом внутреннего спроса на локализованные DevOps-инструменты. Среди наиболее вероятных направлений можно выделить:

* **Расширение библиотеки готовых экшенов** — создание экосистемы плагинов и действий, адаптированных под популярные стеки разработки (Python, Java, Go, Kubernetes).
* **Углубление интеграций с отечественными облачными провайдерами** (Яндекс Облако, VK Cloud, СберCloud) для упрощённого деплоя и управления инфраструктурой.
* **Развитие инструментов безопасности**: встроенный secret-scanning, SAST/DAST-анализ, контроль лицензий.
* **Формирование сообщества и маркетплейса решений**, аналогичного GitHub Marketplace, что позволит ускорить распространение готовых практик CI/CD.
* **Поддержка гибридных сценариев** — использование GitVerse в связке с зарубежными инструментами при необходимости кросс-платформенной разработки.

Таким образом, несмотря на текущие ограничения, GitVerse обладает значительным потенциалом и в перспективе может стать полноценной альтернативой зарубежным CI/CD-платформам в условиях локального рынка.