# Глава 1 Теоретические основы

## 1.1 Общая характеристика курса «Контроль версий с помощью Git»

Система контроля версий (Version Control System, VCS) - это программное обеспечение, которое позволяет отслеживать изменения в исходном коде и других файлах проекта и директориях, в ходе разработки программного продукта. С помощью системы контроля версий разработчики могут работать над проектом одновременно, откатывать изменения, сравнивать версии файлов, а также управлять ветками разработки.

Основные функции VCS:

1. Отслеживание изменений истории развития проекта и изменения файла.
2. Восстановление ранние версии файлов, если в текущей версии возникли проблемы.
3. Упрощает совместную работу нескольких пользователей над проектом.
4. Позволяют создавать ветви для параллельной разработки различных функциональных возможностей или экспериментов.

Преимущества использования системы контроля версий Git:

1. Распределенная архитектура - каждый разработчик имеет полную копию всего репозитория, что повышает отказоустойчивость.  
2. Быстродействие - Git оптимизирован для эффективной работы с большими объемами данных.  
3. Ветвление и слияние - в Git это простые и быстрые операции, что упрощает workflow.  
4. Открытость и популярность - Git является де-факто стандартом в области разработки ПО с огромным сообществом пользователей.  
5. Интеграция с различными сервисами - Git легко интегрируется в современные инструменты разработки.

В рамках прохождения учебной вычислительной практики мною был изучен профильный курс «Version Control with Git», предлагаемый компанией Epam. Ссылка на сайт Epam: https://learn.epam.com. Основная цель курса - подготовить фундамент для изучения программирования, эффективного использования современных инструментов разработчика.   
Данный курс отличается комплексным подходом к освоению этого ключевого инструмента современной разработки программного обеспечения. Учебная программа охватывает не только базовые концепции и команды Git, но и более продвинутые возможности, позволяя участникам курса приобрести системные знания и практические навыки.  
Особое внимание уделяется пониманию ключевых преимуществ Git и его отличительных особенностей среди других VCS. Это позволяет слушателям глубоко осознать потенциал данной системы контроля версий и ее роль в организации эффективного командного взаимодействия.   
Важной особенностью курса является ориентация на долгосрочное применение полученных навыков. Последовательность тем выстроена таким образом, чтобы слушатели могли постепенно наращивать компетенции, формируя прочный фундамент для дальнейшего самостоятельного изучения и совершенствования работы с Git в рамках своих проектов. Курс также рассматривает вопросы интеграции Git с другими инструментами разработки, открывая возможности для более эффективной организации командной работы.  
В ходе обучения рассматривались различные стратегии организации ветвления в проекте, что позволило выработать навыки выбора оптимальной модели ветвления в зависимости от особенностей разрабатываемого программного продукта.

## 1.2 Модули курса

## VCS Concepts Принципиальная концепция системы контроля версий

## Version control types Типы и виды систем версионного контроля: lock-modify-unlock strategy: пока один правит файл, никто другой не может им пользоваться copy-modify-merge strategy: все могут править все файлы в любое время, разрешение конфликта

## Why Git Разбор преимуществ Git перед другими VCS: распределенная архитектура, производительность, простое ветвление, популярность в индустрии.

## Download, install, configure Инструкция по установке и настройке Git на локальную машину. Загрузить Git можно тут: <http://git-scm.com/downloads>

1. Create a github repo and clone it

Cоздание удаленного репозитория на GitHub и клонирование его локально на рабочую машину.

1. Pull from remote

Получение актуальных изменений из удаленного репозитория.

1. Git Gui & gitk  
   Знакомство с графическими интерфейсами для работы с Git(команда git gui& и gitk&)
2. Undoing changes  
   Отмена внесенных изменений и возврат к предыдущим версиям файлов.
3. Git reset

Использование команды git reset для отмены фиксации изменений.

1. Git revert

Отмена коммитов с помощью команды git revert.

1. Branching and merge

Создание, переключение и объединение веток.

1. Conflict solving

Разрешение конфликтов при слиянии веток.

1. Rebase

Интеграция и сонхронизация коммитов из одной ветки в другую с помощью команды git rebase.

1. Cherry-pick

Выборочное применение коммитов из одной ветки в другую.

1. Tags

Создание и работа с метками (тегами) в Git.

1. Stashing

Временное сохранение незафиксированных изменений. Осуществляется с помощью команды git stash.

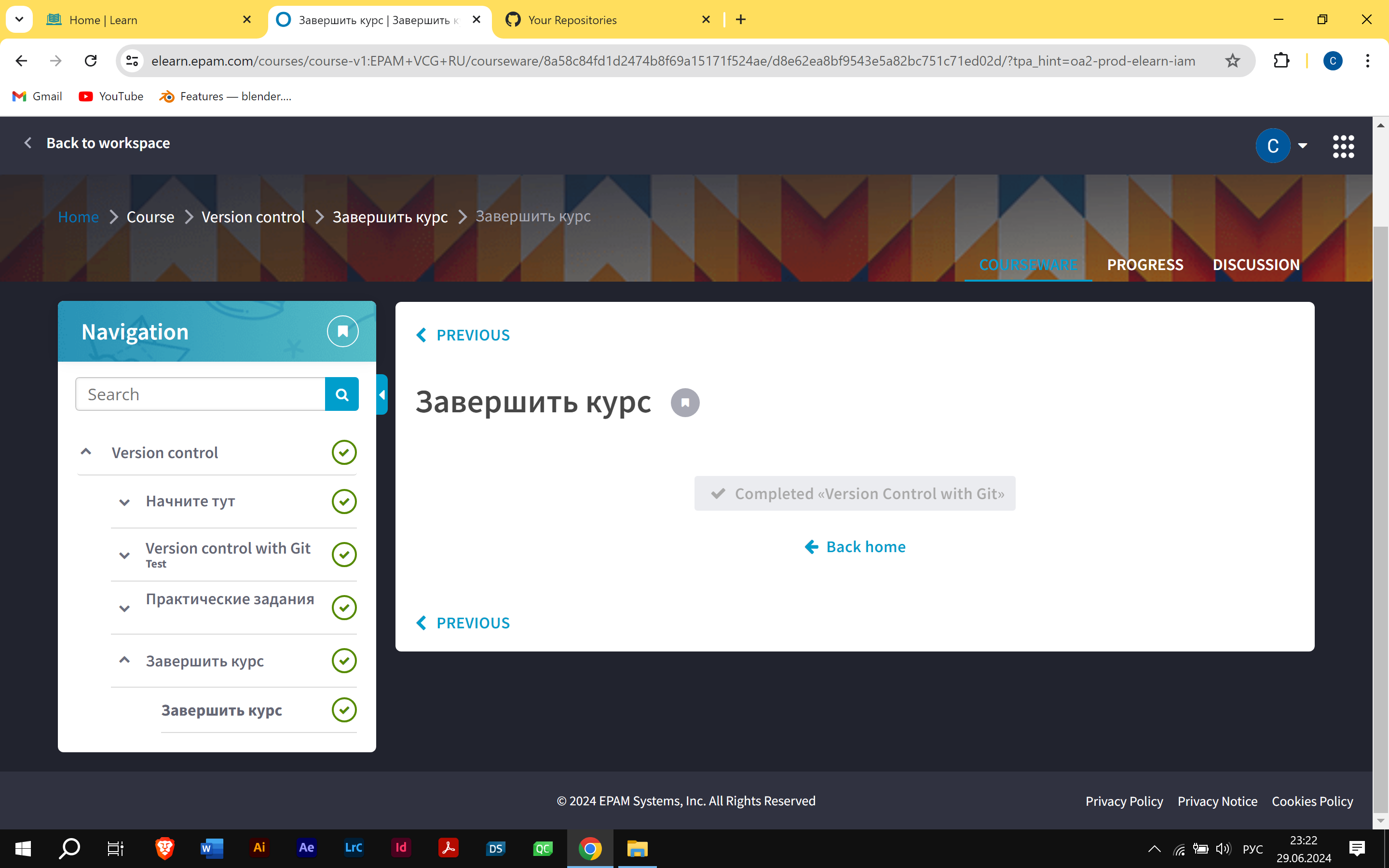
1. Remotes

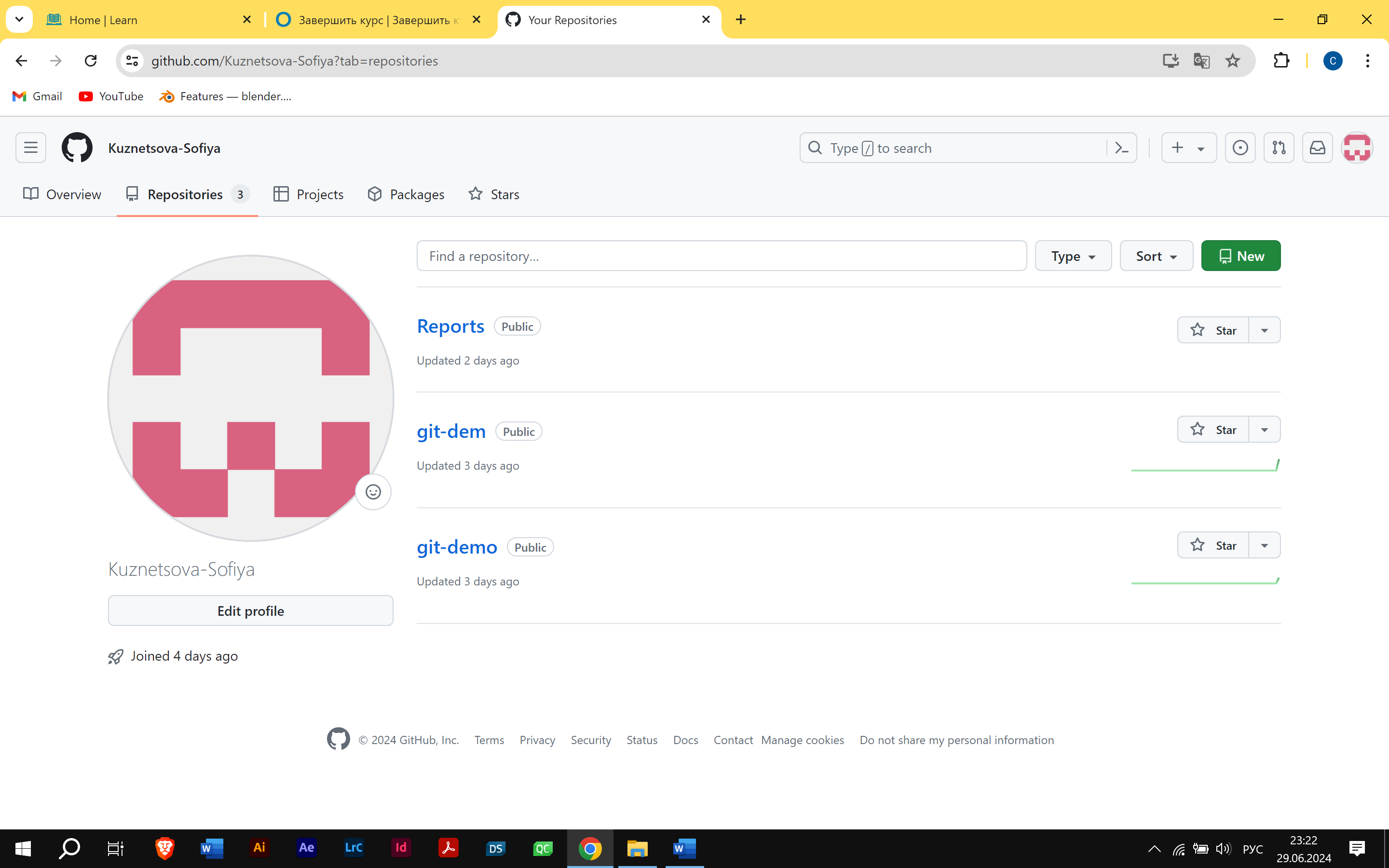
Управление удаленными репозиториями (добавление, удаление, переименование).

1. Branching strategies

Стратегии использования веток.

1. Centralized strategy
2. Feature-branch workflow
3. Gitflow
4. Integration manager workflow
5. Dictator and Lieutenants workflow
6. Forking workflow





## 1.3 Выводы к главе 1

На основе пройденного курса по изучению Git, можно сделать следующие выводы:

* Получены фундаментальные знания о системе контроля версий Git и ее ключевых концепциях.
* Детально изучены возможности Git как наиболее популярной распределенной системы контроля версий. Освоены базовые команды и операции, такие как клонирование, коммиты, ветвление, слияние и отмена изменений.
* Приобретены практические навыки работы с Git, включая создание удаленных репозиториев, интеграцию с GitHub, использование графических интерфейсов и работу с внутренней структурой .git-папки.
* Сформировано понимание ключевых преимуществ Git перед другими VCS, таких как распределенная архитектура, производительность, гибкая система ветвления.
* Изучены продвинутые возможности Git, включая rebasing, cherry-picking, stashing и управление удаленными репозиториями.