Plan

Задачи на урок:

1. Установка git
2. Проверка IntelIJ
3. Создание локальных репозиториев при помощи IntelIJ
4. Создание удаленных репозиториев
5. Подключение remote через интрефейс IntelIJ
6. push, pull, add, commit через интрефейс IntelIJ

Theory

Создание пустого проекта и подключение удаленного репозитория

- Создали удаленный репозиторий
- Создали локальный и вводим команды
- git init
- создаем .gitignore
- git remote add origin git@github.com:AlisherKhamidov/example_05.git
- git add .
- git commit -m 'init'
- git push origin main или git push и выбираем из предложенных вариантов
- git pull origin main - подтянуть изменения из удаленного репозитория из ветки main

Если существует удаленный репозиторий и вы хотите его склонировать

(допустим, что ваш репозиторий или тот, где вы участник)

Настройка `git`

Перед началом работы необходимо сообщить `git`, кто вы и как вас представлять другим участникам распределенной системы контроля версий. Пока вы не подпишетесь, система не даст вам регистрировать снимки проекта, коммиты.

Все ваши изменения должны быть подписаны вашим именем и электронным адресом, чтобы другие участники проекта знали, чьи это правки и как с вами связаться. Сделать это нужно один раз, как правило, сразу после установки git, если вы переустановите систему, процедуру потребуется повторить.

Чтобы задать ваши имя и электронный адрес, следует воспользоваться командой `git config`:

```
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com
```

Убедиться в том, что настройки успешно установлены, можно, запросив их список при помощи команды `git config --list`.

Она выдает множество настроек, в том числе и только что установленные значения.

```
user.name=John Doe
user.email=johndoe@example.com
```

Файл `.gitignore`

Файл `.gitignore` для работы с проектом в `IntelliJ Idea` должен включать в себя следующие строчки:

```
.idea/
*.iml
```

Работа с GitHub и IntelliJ Idea

Работа с GitHub в нужном нам количестве подробно описана на видеозаписи занятия.

Документация по настройке IntelliJ Idea

Базовые понятия

- **проект** - каталог с файлами исходного кода (кодовая база)
- **репозиторий** - хранилище истории разработки проекта
- **клонирование (репо)** - скачивание репо на компьютер

Базовый порядок работы

1. **Инициализация нового репо**
    1. `git init`
    2. создается “скрытое хранилище” - каталог `.git/`
2. **Сохранение**
    - Индексация файлов (добавить в очередь на сохранение)
        1. `git add .`
    - Выполнить сохранение
        - `git commit -m 'update'`
3. **Привязка репо**
4. **Выгрузка ветки на GitHub**
    1. `git push -u origin main` (`main` или другое название)

2. `git push` (если ветку уже выгружал)

Файл `README.md`

- использует формат `Markdown`
- описание репо на GitHub
- помещается в корень проекта, как правило

Пример

```
## Test Project
```

Работа с привязкой

Удалить старую привязку

```
git remote rm origin
```

Добавить привязку

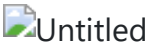
```
git remote add origin скопированная_ссылка
```

Просмотр текущей привязки

```
git remote -v
```

Ветвление в Git

- Ветка - еще одна версия проекта (изолированный поток разработки)



Стратегии ветвления в Git

1. **Git Flow**
    1. `main/master/stable` - long-term (только для проверенного, оттестированного кода - “**священный грааль**”)
    2. `develop/current` - long-term (для тестирования, текущая разработка)
    3. `login/bugfix1/payments` - short-term
2. **GitHub Flow**
    1. `main/master/stable` - long-term
    2. `login/bugfix1/payments` - short-term

Базовые команды по работе с ветками в Git

- `git branch` просмотр веток
  - `git branch -avv` подробный вывод
  - выйти из просмотра - `q`
- `git branch новая_ветка` создать ветку
  - `git branch новая_ветка старая_ветка`
- `git checkout ветка` переключиться на ветку
  - `git checkout -` переключиться на предыдущую ветку
  - **ПЕРЕКЛЮЧАТЬСЯ НЕОБХОДИМО С “ЧИСТЫМ СТАТУСОМ”**
- `git checkout -b новая_ветка` создать и переключиться
  - `git checkout -b новая_ветка старая_ветка`
- `git branch -D ветка` удалить ветку (локально)
- `git merge название_ветки` слияние веток
- `git push origin --delete ветка` удалить ветку (дистанционную)
- `git branch -m новое_название` переименовать ветку (локально)

Слияние веток

- перенос (интеграция) изменений из одной ветки в другую
- выполняется командой
  - `git merge название_ветки`
- перед слиянием необходимо переключиться в целевую ветку

Пример

```
git checkout -b login
# внести правки
git checkout master
git merge login
git branch -D login
```

Клонирование репо

1. Открыть репо на **GitHub**
2. Решить, куда его скачать
3. Скопировать SSH-ссылку
4. Выполнить команду
    1. `git clone скопированная_ссылка`

Ссылки

- клонирование
- ветвление
- клонирование через IDEA

Homework

Homework

Задание 1

https://github.com/AlisherKhamidov/example_06

Code

./code/example_02/README.md

```
# Задание 1

Создайте класс Coffee с полями:
- boolean isMilky;
- String title; (например, cappuccino)

Создайте два объекта класса кофе:
- капучино и эспрессо
```

./code/example_02/.gitignore

```
.DS_STORE
/out/
```

./code/example_02/src/Main.java

```
public class Main {
  public static void main(String[] args) {

  }
}
```

./code/example_06/.gitignore

```
### IntelliJ IDEA ###
out/
!**/src/main/**/out/
!**/src/test/**/out/

### Eclipse ###
```

```
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache
bin/
!**/src/main/**/bin/
!**/src/test/**/bin/

### NetBeans ###
/nbproject/private/
/nbbuild/
/dist/
/nbdist/
/.nb-gradle/

### VS Code ###
.vscode/

### Mac OS ###
.DS_Store
```

./code/example_06/src/Main.java

```
public class Main {
  public static void main(String[] args) {
    int x = 10;
    int y = 15;

  }
}
```

./code/example_06/README.md

```
# Задание

У вас есть две переменные x и y, создайте метод, который бы возвращал сумму
Вызовите этот метод для x и y.

Не забудьте, сделать в начале форк, а в конце pull request.
```