



{KODE{KLOUD

just enough





GIT

- What is GIT?
- Install GIT
- GIT Repositories
- Remote Repositories
- Clone, Pull & Push
- GIT vs GITHUB

- my-application
- LICENSE
- README.md
- requirements.txt
- main.py 

main.py

```
from flask import Flask

app = Flask(__name__)

app_color = 'Blue'

@app.route('/')
def hello():
    return 'Hello, World!'

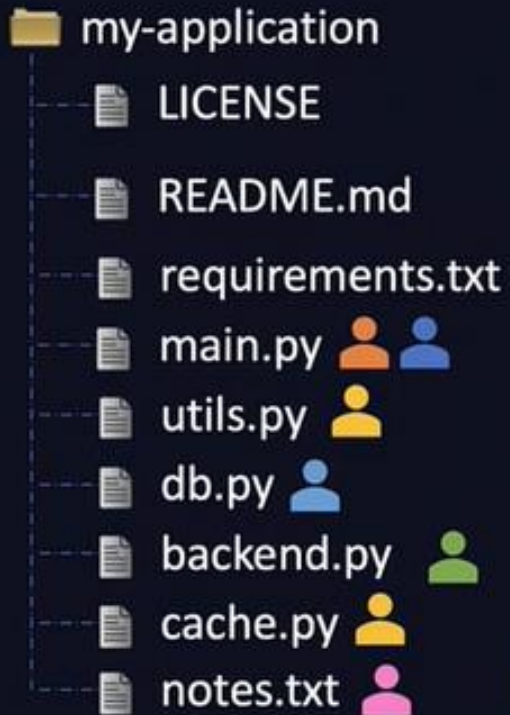
@app.route('/user/<username>')
def show_user_profile(username):
    return 'User %s' % escape(username)

@app.route('/post/<int:post_id>')
def show_post(post_id):
    return 'Post %d' % post_id

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    # show the subpath after /path/
    return 'Subpath %s' % escape(subpath)
```

Source Control Management (SCM)

Version Control Systems (VCS)



What changes were made?

When where they made?

By whom where they made?

main.py

```
from flask import Flask
```

```
app = Flask(__name__)
```

```
app_color = 'Green'
```

```
@app.route('/')
```

```
def hello():
```

```
    return 'Welcome to the community!'
```

```
@app.route('/user/<username>')
```

```
def show_user_profile(username):
```

```
    return 'User %s' % escape(username)
```

```
@app.route('/post/<int:post_id>')
```

```
def show_post(post_id):
```

```
    return 'Post Number %d' % post_id
```

```
@app.route('/path/<path:subpath>')
```

```
def show_subpath(subpath):
```

```
    # show the subpath after /path/
```

```
    return 'Subpath %s' % escape(subpath)
```

Git



1. Initialize a GIT Repository

2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes

5. Commit Changes

Git

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

GIT Repository

```
▶ yum install git
```

```
▶ git version
```

```
git version 1.8.3.1
```

```
▶ git init
```

```
Initialized empty Git repository in my-application/.git/
```

0. Install GIT

1. Initialize a GIT Repository

2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes

5. Commit Changes

Git

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

GIT Repository

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Untracked files:
```

```
LICENSE
README.md
requirements.txt
main.py
utils.py
db.py
backend.py
cache.py
notes.txt
```

```
nothing added to commit but untracked files present
```

```
git add LICENSE README.md main.py ...
```

0. Install GIT

1. Initialize a GIT Repository

2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes

5. Commit Changes

Git

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

GIT Repository

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
new file:   LICENSE
new file:   README.md
new file:   backend.py
new file:   cache.py
new file:   db.py
new file:   main.py
new file:   requirements.txt
new file:   utils.py
```

```
Untracked files:
```

```
notes.txt
```

0. Install GIT

1. Initialize a GIT Repository

2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes

5. Commit Changes

Git

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

Git Repository

```
git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
new file:   LICENSE
new file:   README.md
new file:   backend.py
new file:   cache.py
new file:   db.py
new file:   requirements.txt
new file:   utils.py
```

```
Changes not staged for commit:
```

```
modified:   main.py
```

```
Untracked files:
```

```
notes.txt
```

```
git add main.py
```

0. Install GIT

1. Initialize a GIT Repository

2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes

5. Commit Changes

Git

```
▶ git commit -m "Initial Commit"
```

```
[master (root-commit) f5cdb03] Initial commit
8 files changed, 1 insertion(+)
create mode 100644 LICENSE
create mode 100644 README.md
create mode 100644 backend.py
create mode 100644 cache.py
create mode 100644 db.py
create mode 100644 main.py
create mode 100644 requirements.txt
create mode 100644 utils.py
```

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

GIT Repository

0. Install GIT

1. Initialize a GIT Repository


2. Configure Files To Track

3. Git Tracks Changes

4. Stage Changes



5. Commit Changes

My Laptop

- my-application
 - LICENSE
 - README.md
 - requirements.txt
 - main.py 
 - utils.py
 - db.py
 - backend.py
 - cache.py
 - notes.txt


Local GIT Repository

Mark's Laptop

- my-application
 - LICENSE
 - README.md
 - requirements.txt
 - main.py 
 - utils.py
 - db.py 
 - backend.py
 - cache.py
 - notes.txt


Local GIT Repository

Aditi's Laptop

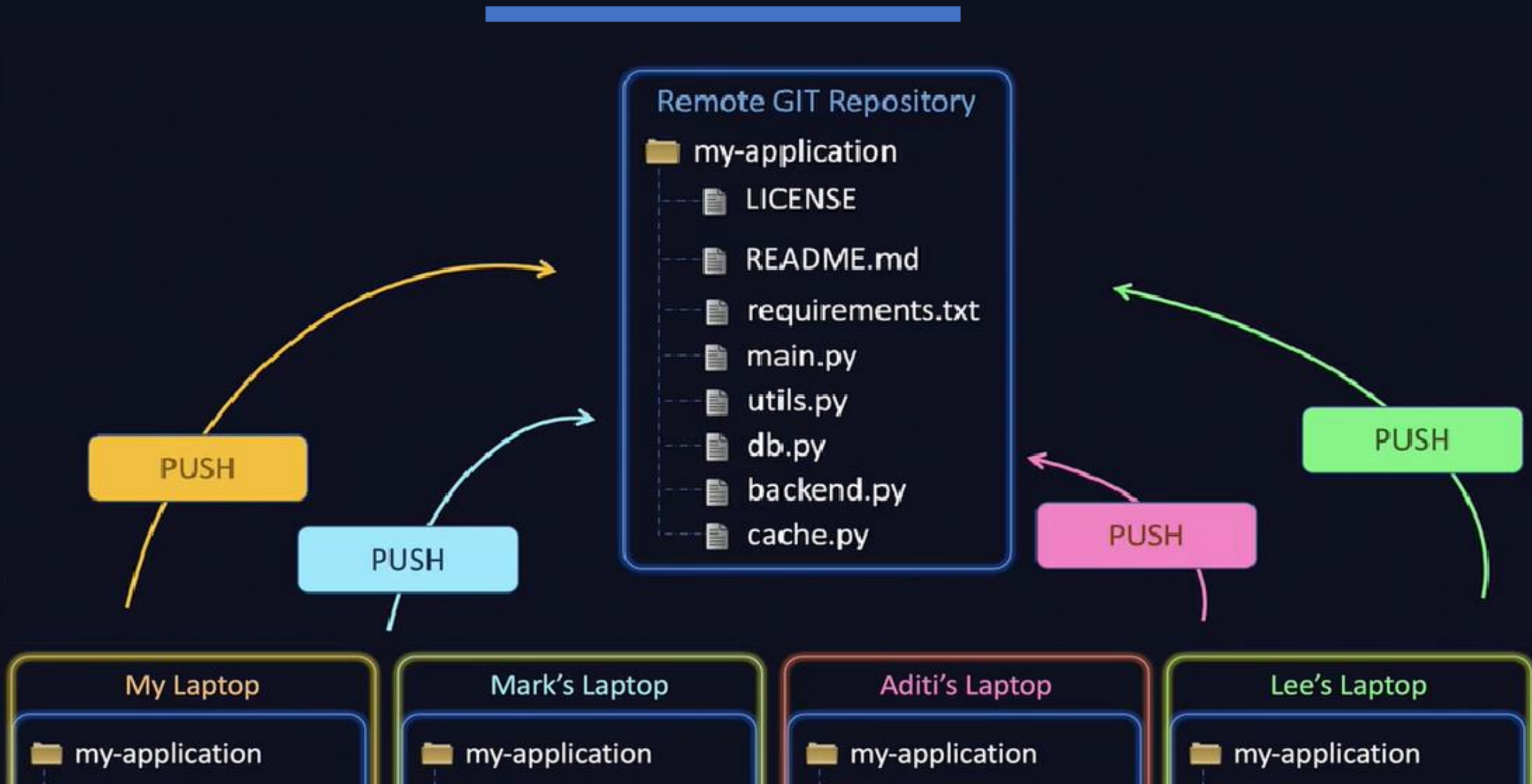
- my-application
 - LICENSE
 - README.md
 - requirements.txt
 - main.py
 - utils.py
 - db.py
 - backend.py
 - cache.py 
 - notes.txt

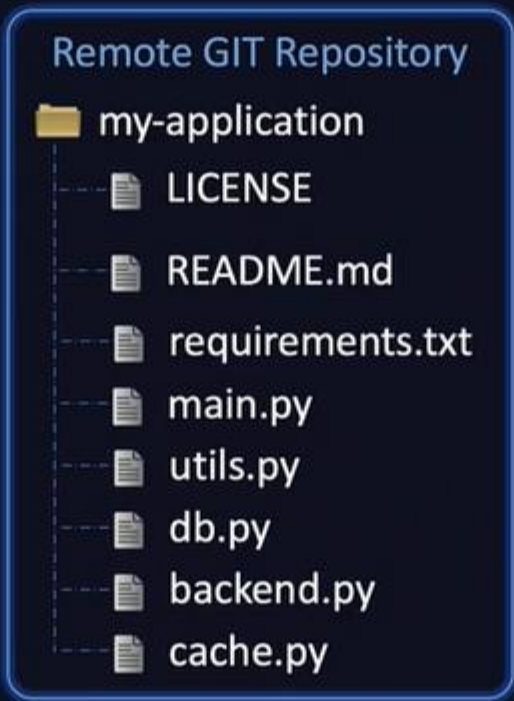
Local GIT Repository

Lee's Laptop

- my-application
 - LICENSE
 - README.md
 - requirements.txt
 - main.py
 - utils.py
 - db.py
 - backend.py 
 - cache.py
 - notes.txt

Local GIT Repository



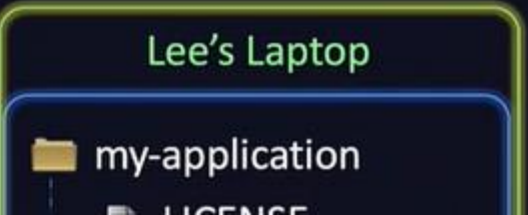
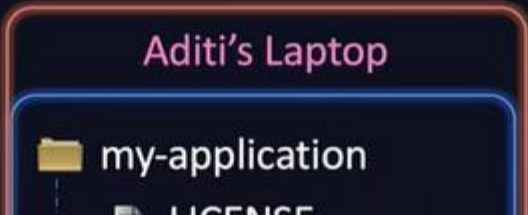
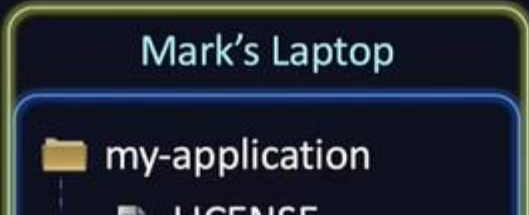


PULL

PULL

PULL

PULL





Search or jump to...



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)



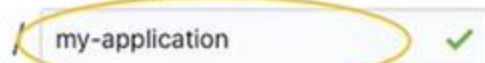
Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner



Repository name *



Great repository names are short and memorable. Need inspiration? How about [fluffy-telegram](#)?

Description (optional)

My application is a python based application



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer.

Add .gitignore: **None** ▾

Add a license: **None** ▾



Create repository

Push

<https://github.com/mmumshad/my-application.git>

My Laptop

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

Local GIT Repository

```
git init
```

```
git add .
```

```
git commit -m "Initial Commit"
```

```
git remote add <name> <url>
```

```
git remote add github
https://github.com/mmumshad/my-application.git
```

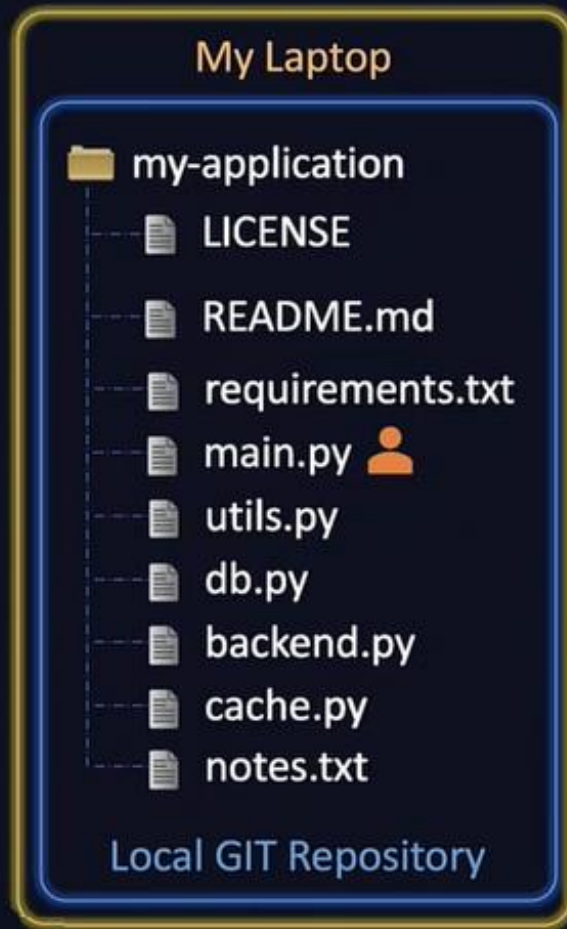
```
git push <name> <branch>
```

```
git push github master
```

GitHub

Remote GIT Repository

Clone



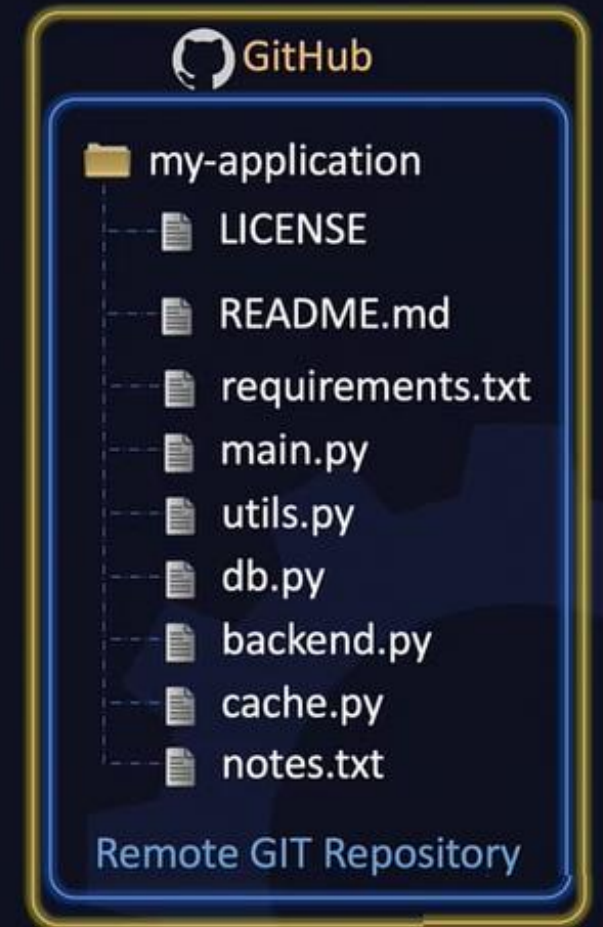
```
git remote add <name> <url>
```

```
git remote add github  
https://github.com/mmumshad/my-application.git
```

```
git push <name> <branch>
```

```
git push -u github master
```

<https://github.com/mmumshad/my-application.git>



Pull

<https://github.com/mmumshad/my-application.git>

My Laptop

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

Local GIT Repository

```
git remote add <name> <url>
```

```
git remote add github
https://github.com/mmumshad/my-application.git
```

```
git push <name> <branch>
```

```
git push -u github master
```

PUSH

```
mark$ git clone https://github.../my-application.git
```

```
mark$ git remote -v
```

```
origin https://github.com/mmumshad/my-application.git (fetch)
origin https://github.com/mmumshad/my-application.git (push)
```

```
mark$ git pull
```

PULL

GitHub

```
my-application
├── LICENSE
├── README.md
├── requirements.txt
├── main.py
├── utils.py
├── db.py
├── backend.py
├── cache.py
└── notes.txt
```

Remote GIT Repository

Mark's Laptop



{KODE{KLOUD