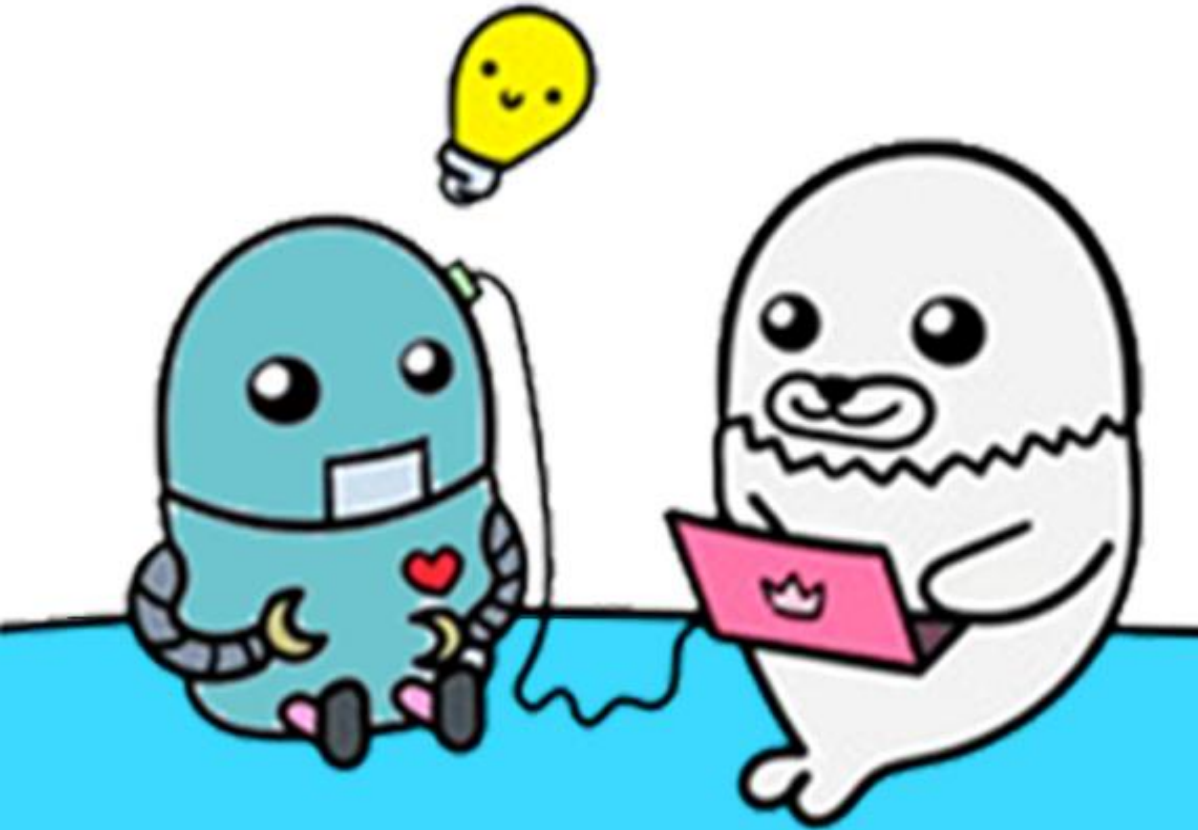
Яндекс Кью

Open Data Science

MLOps и production подход к ML исследованиям



📅 28 марта - 28 мая

Яндекс Кью

Open Data Science

MLOps и production подход к ML исследованиям

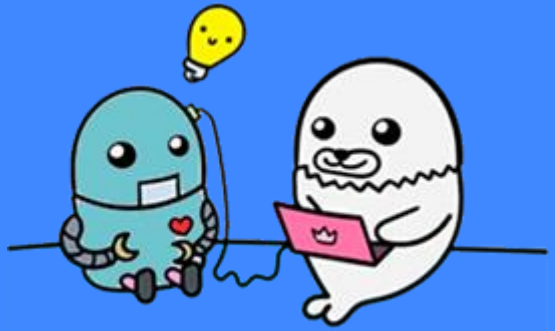
Шаблонизация. Python пакеты и CLI. Snakemake

Павел Кикин

Газпромнефть ЦР

Руководитель направления NLP

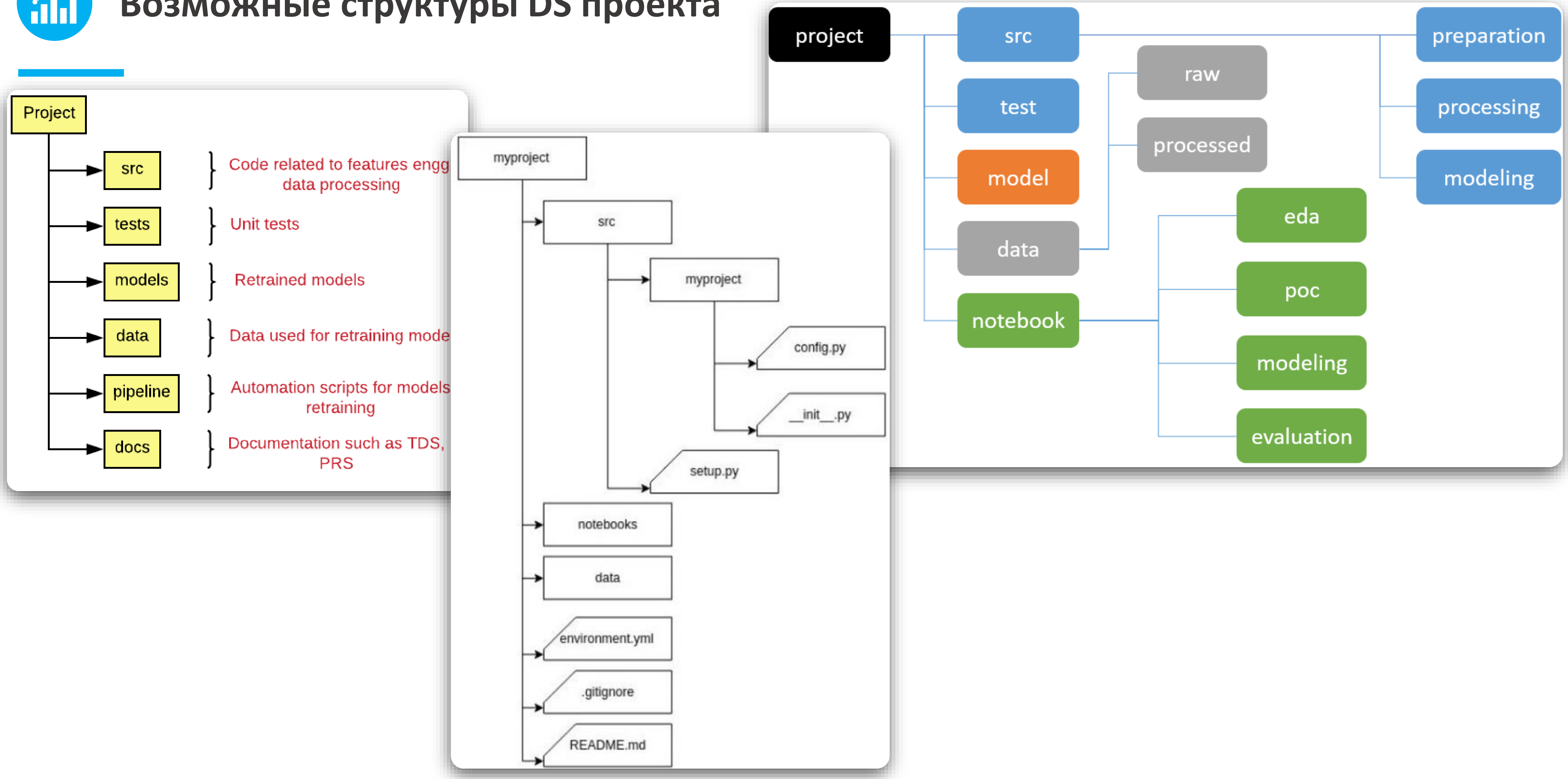
t.me/pavel_kikin



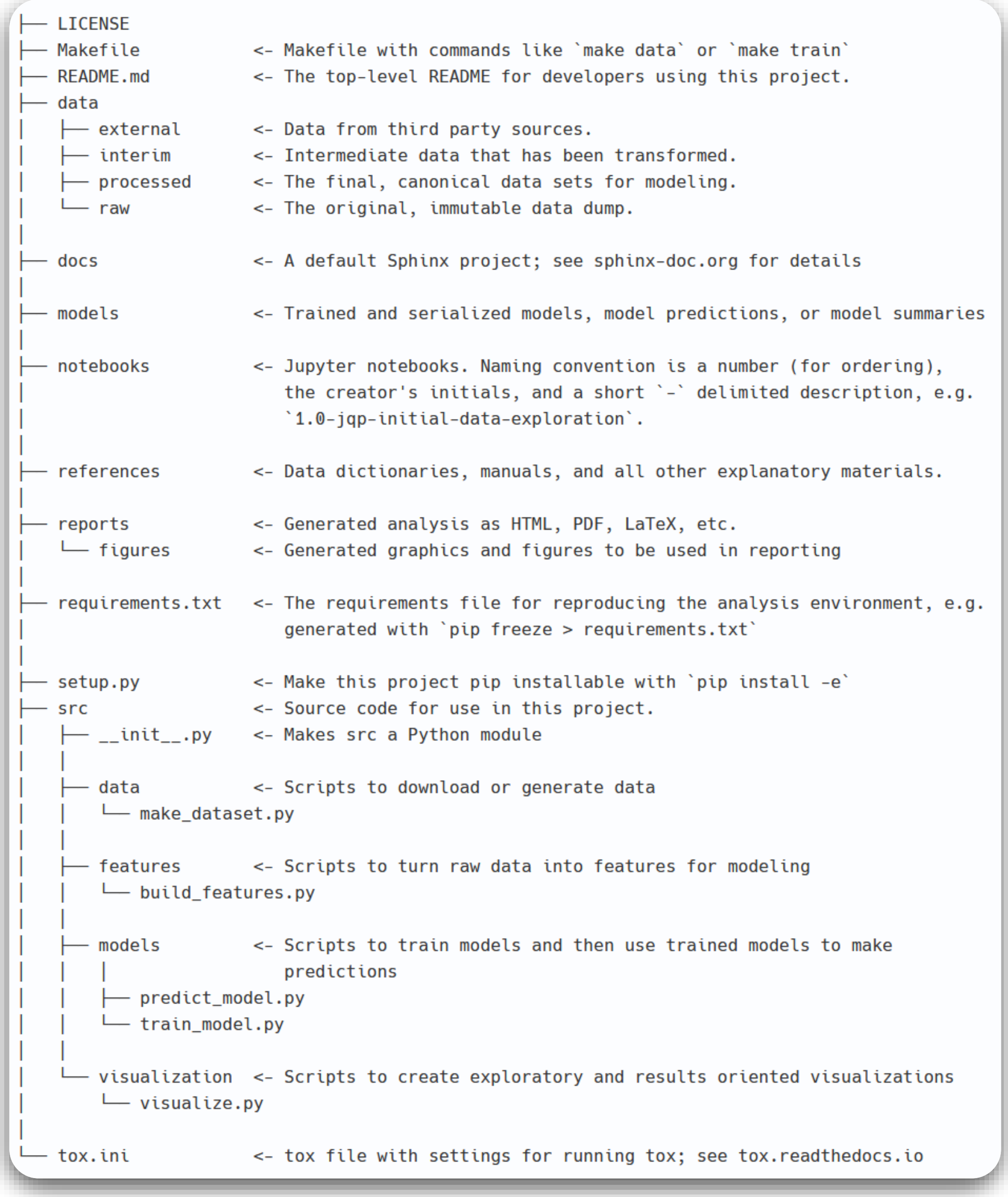
Артефакты DS проекта

- Данные
- EDA, Preprocessing
- Описания, отчеты (MD, Jupyter, latex), графики
- Feature engineering
- Алгоритмы предсказания
- Модели

Возможные структуры DS проекта



Cookiecutter DS

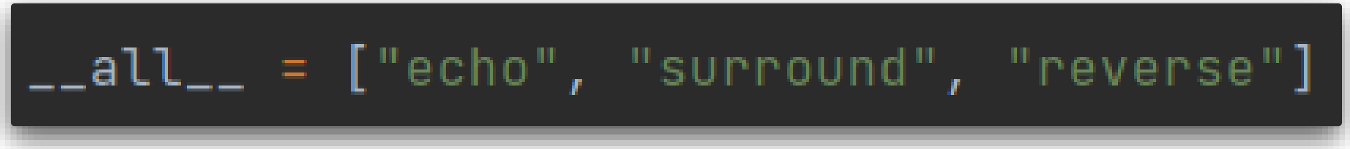


```
├── LICENSE
├── Makefile           <- Makefile with commands like `make data` or `make train`
├── README.md          <- The top-level README for developers using this project.
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been transformed.
│   ├── processed      <- The final, canonical data sets for modeling.
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphinx-doc.org for details
│
├── models             <- Trained and serialized models, model predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming convention is a number (for ordering),
│                         the creator's initials, and a short `-` delimited description, e.g.
│                         `1.0-jqp-initial-data-exploration`.
│
├── references         <- Data dictionaries, manuals, and all other explanatory materials.
│
├── reports            <- Generated analysis as HTML, PDF, LaTeX, etc.
│   └── figures        <- Generated graphics and figures to be used in reporting
│
├── requirements.txt   <- The requirements file for reproducing the analysis environment, e.g.
│                         generated with `pip freeze > requirements.txt`
│
├── setup.py           <- Make this project pip installable with `pip install -e`
├── src                <- Source code for use in this project.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate data
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into features for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then use trained models to make
│   │   │                 predictions
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and results oriented visualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running tox; see tox.readthedocs.io
```

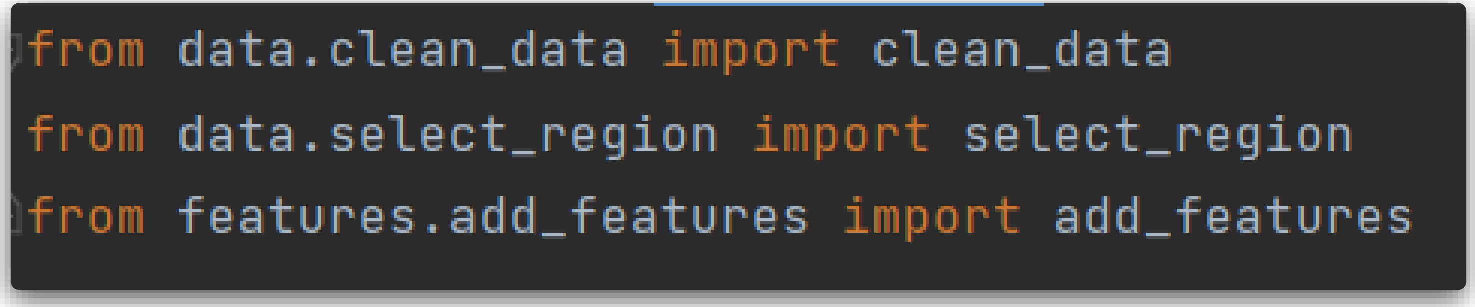
Создание Cookiecutter шаблона

```
python -m venv ./venv
call ./venv/scripts/activate
cookiecutter https://github.com/drivendata/cookiecutter-data-science
```

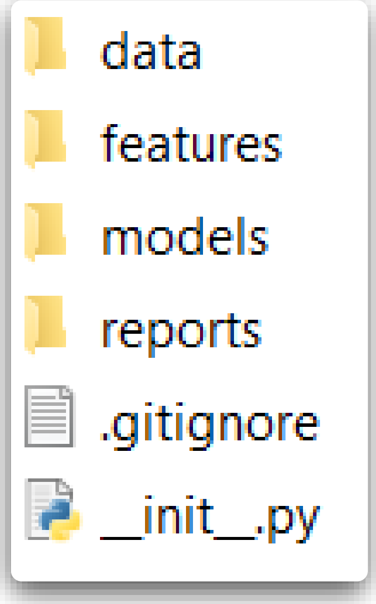
SRC как пакеты



```python
__all__ = ["echo", "surround", "reverse"]
```

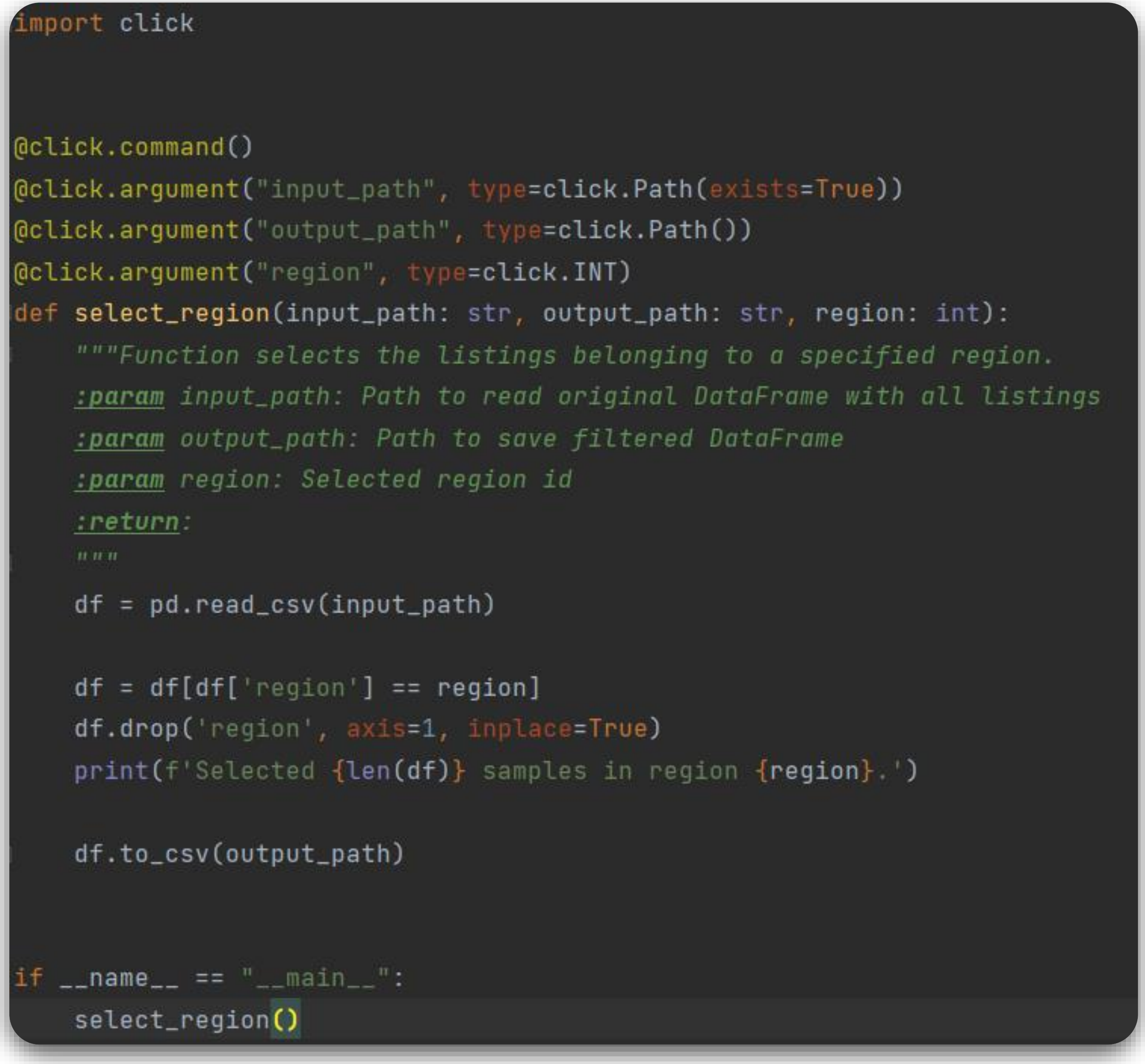


```python
from data.clean_data import clean_data
from data.select_region import select_region
from features.add_features import add_features
```



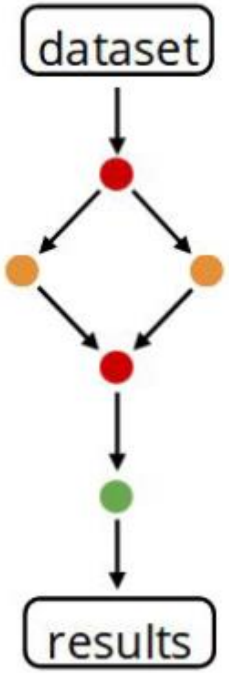
data
features
models
reports
.gitignore
__init__.py

CLI



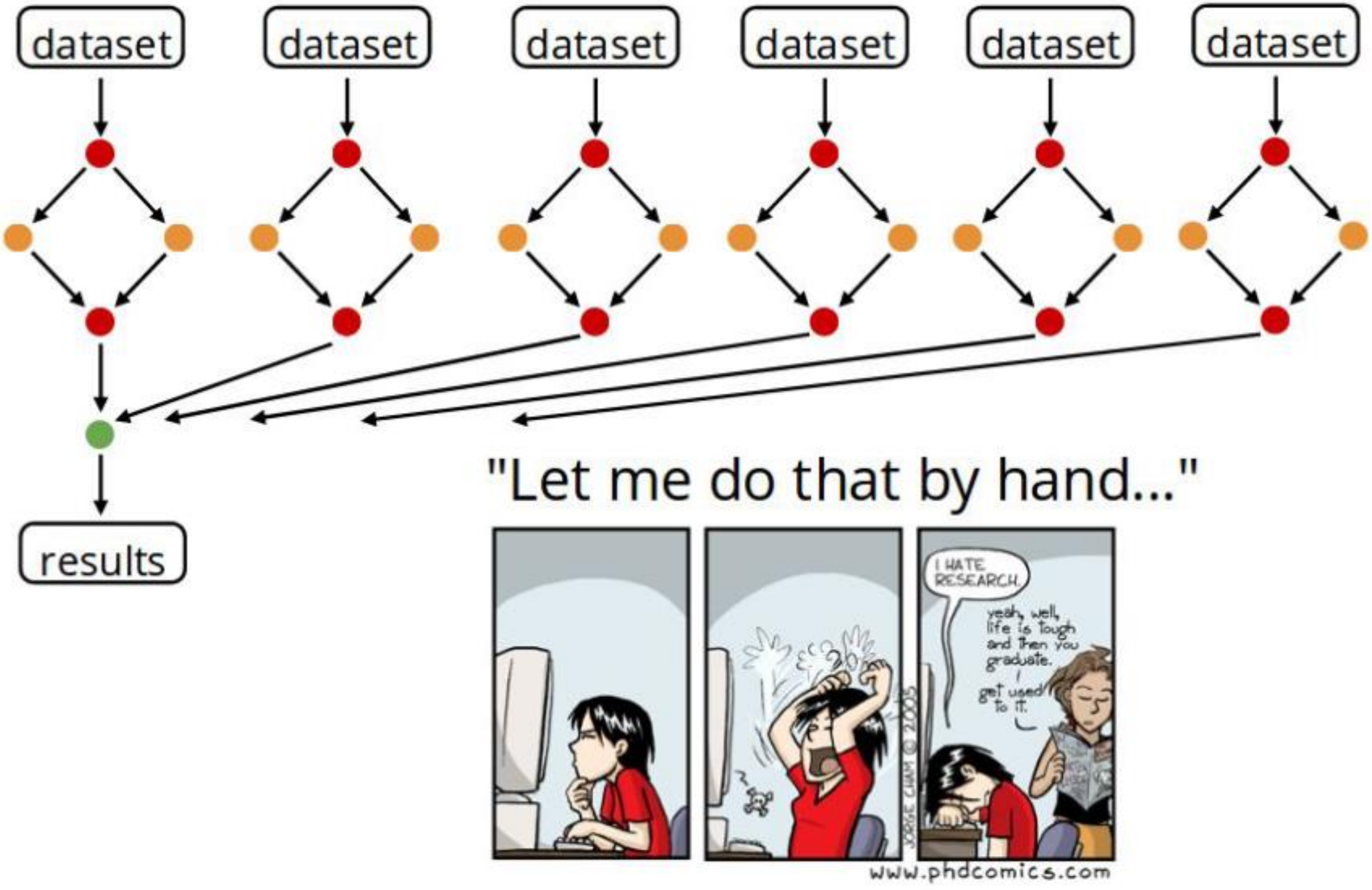
```python
import click


@click.command()
@click.argument("input_path", type=click.Path(exists=True))
@click.argument("output_path", type=click.Path())
@click.argument("region", type=click.INT)
def select_region(input_path: str, output_path: str, region: int):
    """Function selects the listings belonging to a specified region.
    :param input_path: Path to read original DataFrame with all listings
    :param output_path: Path to save filtered DataFrame
    :param region: Selected region id
    :return:
    """
    df = pd.read_csv(input_path)

    df = df[df['region'] == region]
    df.drop('region', axis=1, inplace=True)
    print(f'Selected {len(df)} samples in region {region}.')

    df.to_csv(output_path)


if __name__ == "__main__":
    select_region()
```

Workflow менеджеры

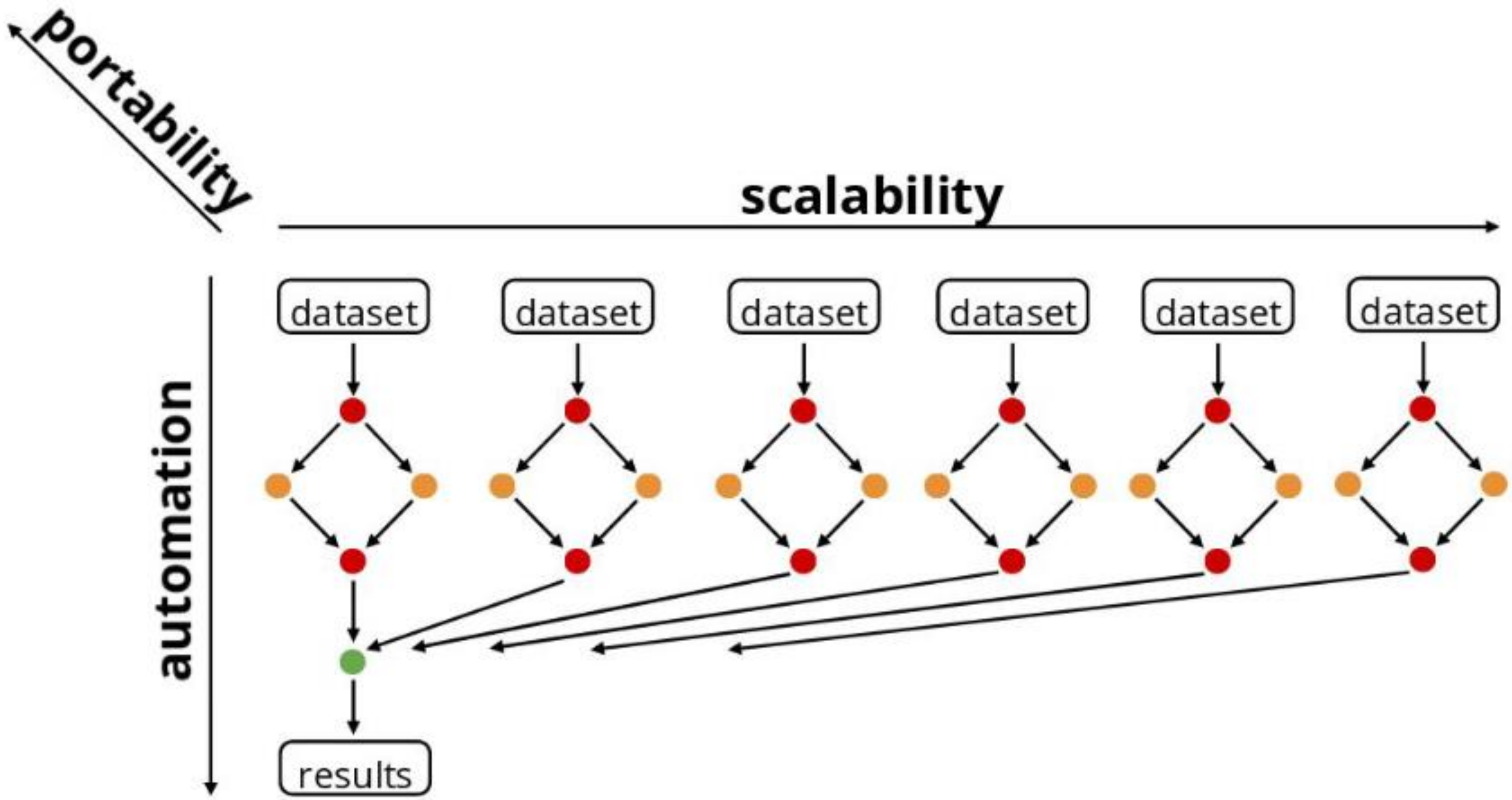


"Let me do that by hand..."

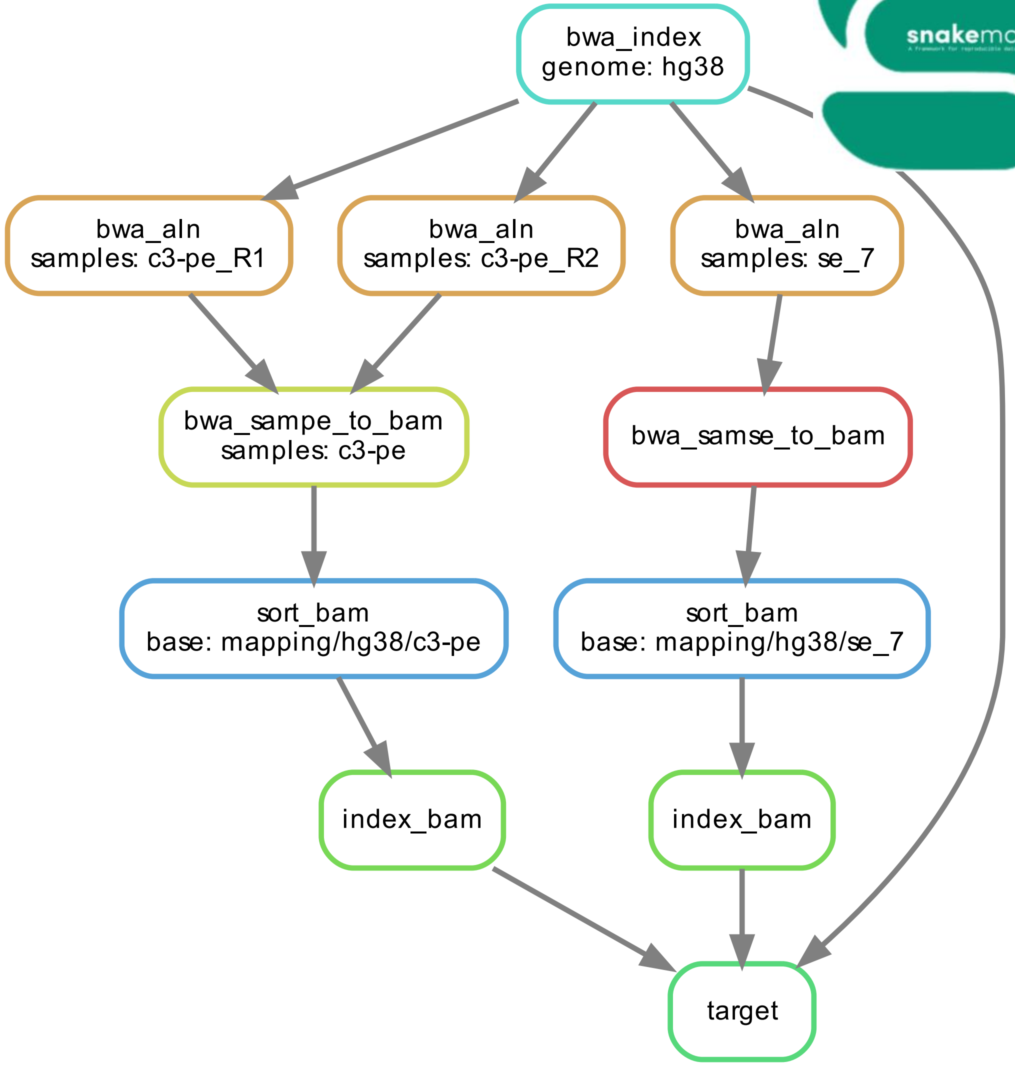
Workflow менеджеры



Workflow менеджеры

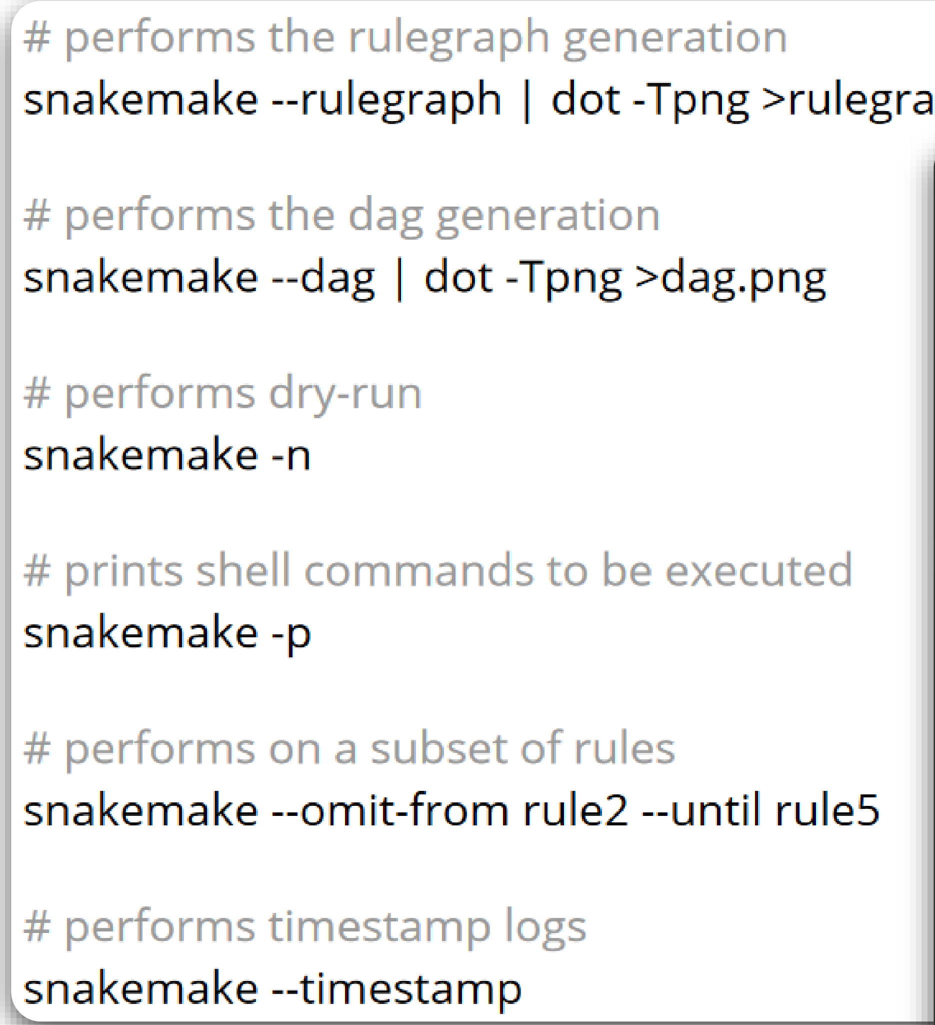


SnakeMake

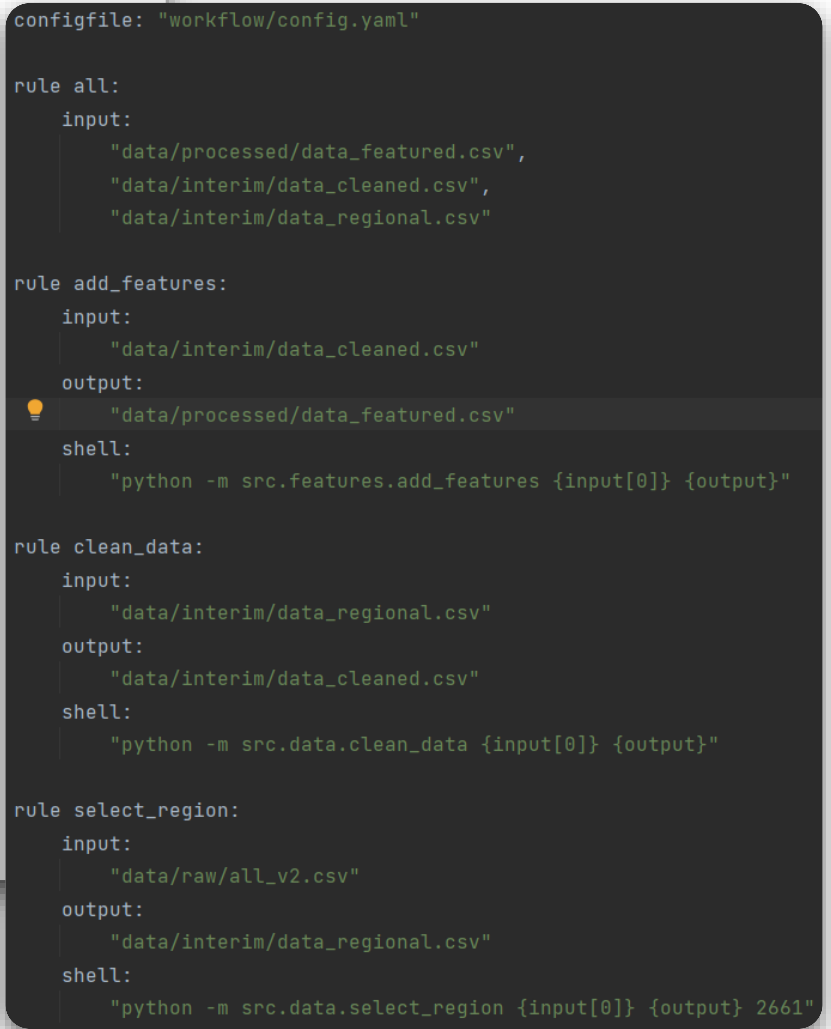


- Декомпозирует анализ на правила, описанные на Python
- Правила определяют как получить выходной файл из входного
- SnakeMake определеяет зависимости и порядок исполнения в виде DAG (направленный ацикличный граф)
- Строит авто-отчеты

SnakeMake



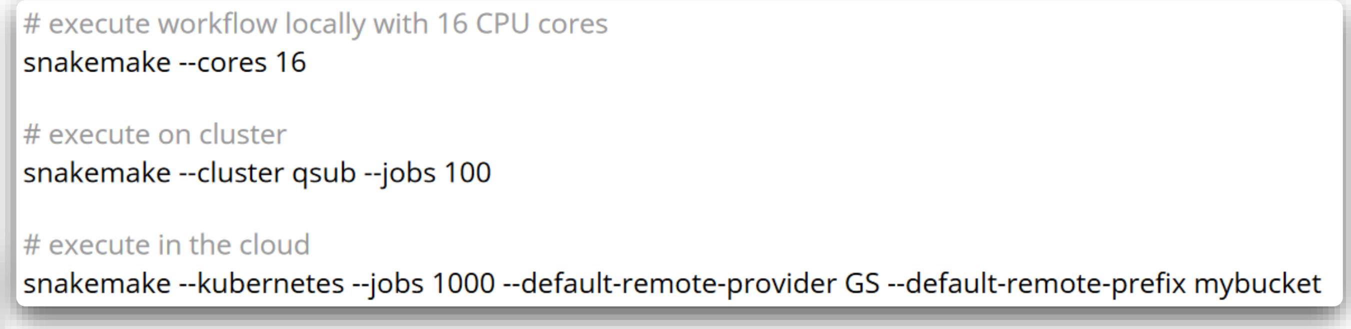
```
# performs the rulegraph generation
snakemake --rulegraph | dot -Tpng >rulegraph.png

# performs the dag generation
snakemake --dag | dot -Tpng >dag.png

# performs dry-run
snakemake -n

# prints shell commands to be executed
snakemake -p

# performs on a subset of rules
snakemake --omit-from rule2 --until rule5

# performs timestamp logs
snakemake --timestamp
```



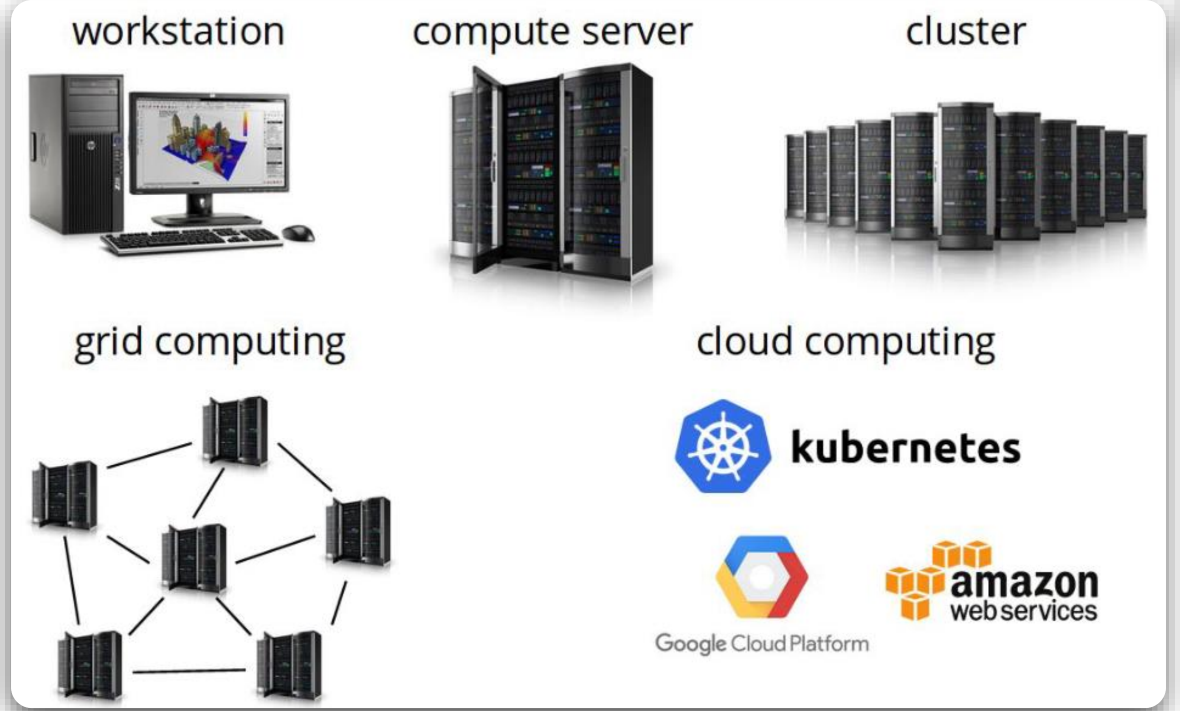
```
configfile: "workflow/config.yaml"

rule all:
    input:
        "data/processed/data_featured.csv",
        "data/interim/data_cleaned.csv",
        "data/interim/data_regional.csv"

rule add_features:
    input:
        "data/interim/data_cleaned.csv"
    output:
        "data/processed/data_featured.csv"
    shell:
        "python -m src.features.add_features {input[0]} {output}"

rule clean_data:
    input:
        "data/interim/data_regional.csv"
    output:
        "data/interim/data_cleaned.csv"
    shell:
        "python -m src.data.clean_data {input[0]} {output}"

rule select_region:
    input:
        "data/raw/all_v2.csv"
    output:
        "data/interim/data_regional.csv"
    shell:
        "python -m src.data.select_region {input[0]} {output} 2661"
```

SnakeMake

- Независимые части DAG могут быть исполнены параллельно
- Максимизирует параллелизм с учетом ресурсов на любое количество:
  - Компьютеров
  - Кластеров
  - Облаков

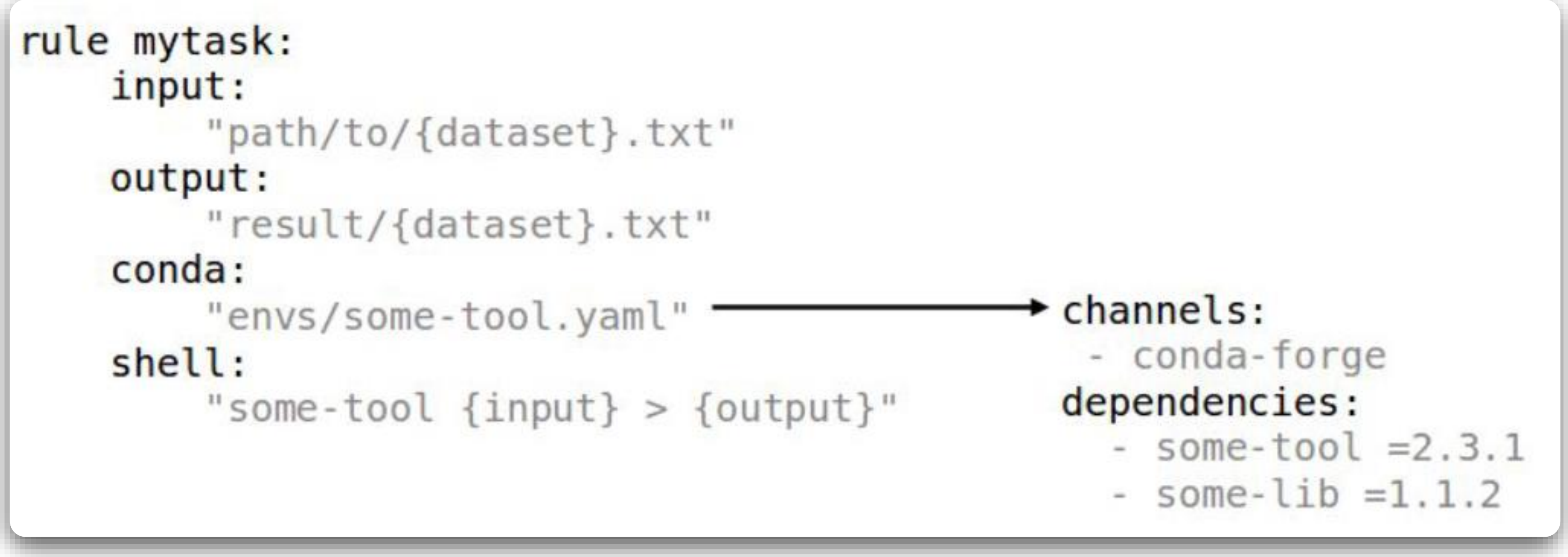


```
# execute workflow locally with 16 CPU cores
snakemake --cores 16

# execute on cluster
snakemake --cluster qsub --jobs 100

# execute in the cloud
snakemake --kubernetes --jobs 1000 --default-remote-provider GS --default-remote-prefix mybucket
```



SnakeMake

Каждое отдельное задание может использовать свое окружение



```
rule mytask:
    input:
        "path/to/{dataset}.txt"
    output:
        "result/{dataset}.txt"
    conda:
        "envs/some-tool.yaml"
    shell:
        "some-tool {input} > {output}"
```

```
channels:
 - conda-forge
dependencies:
  - some-tool =2.3.1
  - some-lib =1.1.2
```

Яндекс Кью

Open Data Science

MLOps и production подход к ML исследованиям

Шаблонизация. Python пакеты и CLI. Snakemake

Павел Кикин
Газпромнефть ЦР
Руководитель направления NLP
t.me/pavel_kikin