Pomodorr

Kamil Daniewski

Jul 05, 2020

CONTENTS:

CHAPTER

ONE

BASIC SETUP

In order to recreate the structure of .envs directory that the project uses, start with creating the .envs directory in the root of the project, as it is done below.

```
pomodorr
├── .envs
│   ├── .local
│   │   ├── .django
│   │   ├── .postgres
│   └── .production
│   │   ├── .django
│   │   ├── .postgres
```

1.1 Configure environment variables

The configurations listed below resemble the default environment setting files needed to set the project up:

- **local envs**:

  pomodorr/.envs/.local/.django

  ```
  # General
  USE_DOCKER=yes
  IPYTHONDIR=/app/.ipython
  DJANGO_SETTINGS_MODULE=config.settings.local


  # Redis
  REDIS_URL=redis://redis:6379/0


  # Celery / Flower
  CELERY_FLOWER_USER=<your_celery_flower_user>
  CELERY_FLOWER_PASSWORD=<your_celery_flower_password>
  ```

  pomodorr/.envs/.local/.postgres

  ```
  # PostgreSQL
  POSTGRES_HOST=postgres
  POSTGRES_PORT=5432
  ```

POSTGRES_DB=pomodorr
POSTGRES_USER=<your_postgres_user>
POSTGRES_PASSWORD=<your_postgres_password>

- **production envs**:

pomodorr/.envs/.production/.django

```
# General
# DJANGO_READ_DOT_ENV_FILE=True
DJANGO_SETTINGS_MODULE=config.settings.production
DJANGO_SECRET_KEY=<your_secret_key>
DJANGO_ADMIN_URL=<your_admin_url>
DJANGO_ALLOWED_HOSTS=<your_hosting_domain>


# Security
# TIP: better off using DNS, however, redirect is OK too
DJANGO_SECURE_SSL_REDIRECT=False


# Email
MAILGUN_API_KEY=<your_mailgun_api_key>
DJANGO_SERVER_EMAIL=<your_server_email>
MAILGUN_DOMAIN=<your_mailgun_domain>


# AWS
DJANGO_AWS_ACCESS_KEY_ID=<your_aws_access_key_id>
DJANGO_AWS_SECRET_ACCESS_KEY=<your_aws_secret_access_key>
DJANGO_AWS_STORAGE_BUCKET_NAME=<your_aws_storage_bucket_name>


# Gunicorn
WEB_CONCURRENCY=4


# Sentry
SENTRY_DSN=<your_sentry_dsn>


# Redis
REDIS_URL=redis://redis:6379/0


# Celery / Flower
CELERY_FLOWER_USER=<your_celery_flower_user>
CELERY_FLOWER_PASSWORD=<your_celery_flower_password>
```

pomodorr/.envs/.production/.postgres

```
# PostgreSQL
```

POSTGRES_HOST=postgres
POSTGRES_PORT=5432
POSTGRES_DB=pomodorr
POSTGRES_USER=<your_postgres_user>
POSTGRES_PASSWORD=<your_postgres_password>

1.2 Building and running containers

The details about getting the project up and running is described on the official django-cookiecutter documentation.

Having configured the environment variables all there is left to do is to build the docker containers and run them. Assuming your current directory is the root directory of the project, type:

```
$ docker-compose -f local.yml build
```

The last step is to get the containers up.

```
$ docker-compose -f local.yml up
```

Note:

You may encounter some problems with already used ports. In that situation, check the `pomodorr/local.yml` configuration file and change the clashing ports.
Likewise, in case of having troubles with setting the project up, please consider having a look at the troubleshooting page of the official django-cookiecutter documentation.
Otherwise feel free to send an email message or report an issue on the github if there is an evidence of a bug.

1.3 Building documentation with sphinx and sphinx-apidoc

Assuming your current directory is `pomodorr/docs`, in order to generate the apidoc out of your docstrings, execute:

```
$ sh apidoc.sh
```

Having done that, in order to generate the html pages, type:

```
$ make html
```

1.4 API documentation

If application is run in DEBUG mode, there is available interactive swagger and redoc API documentation - by default under *http://127.0.0.1:8000/swagger/* and *http://127.0.0.1:8000/redoc/*

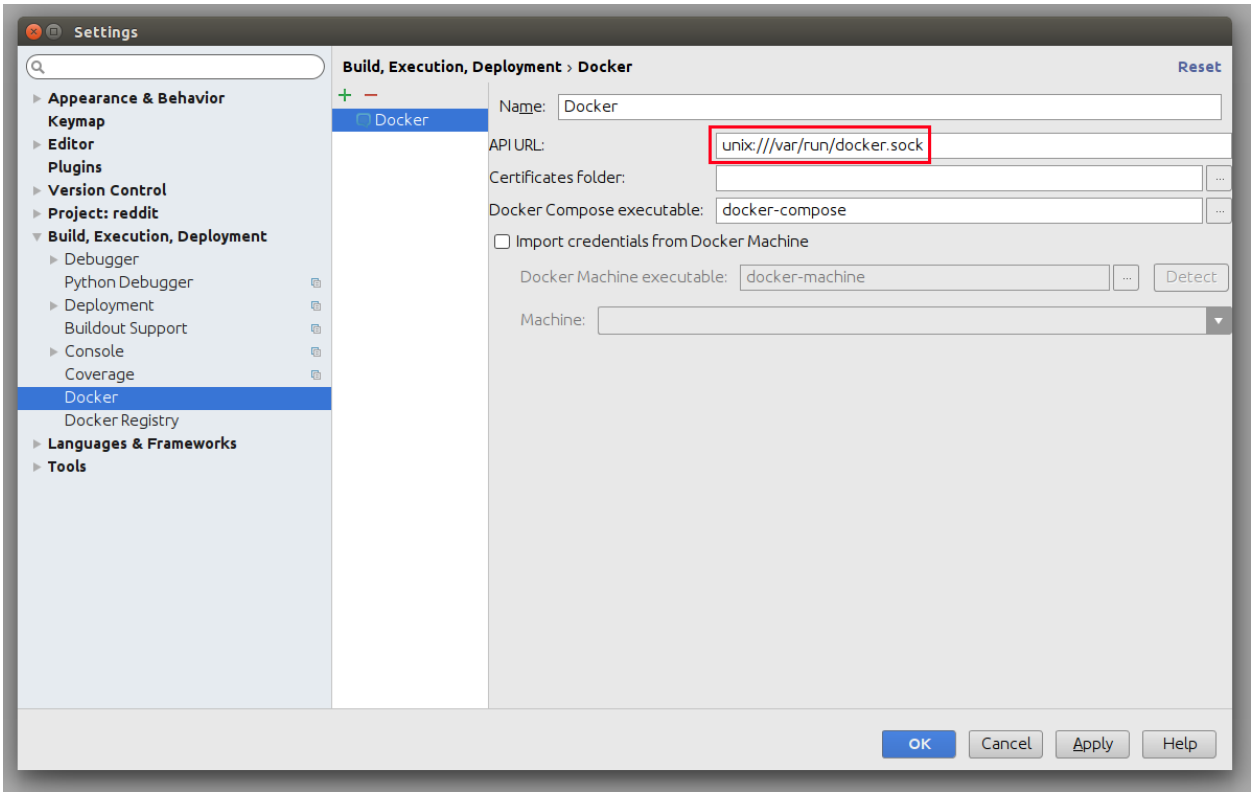
CHAPTER

TWO

DOCKER REMOTE DEBUGGING

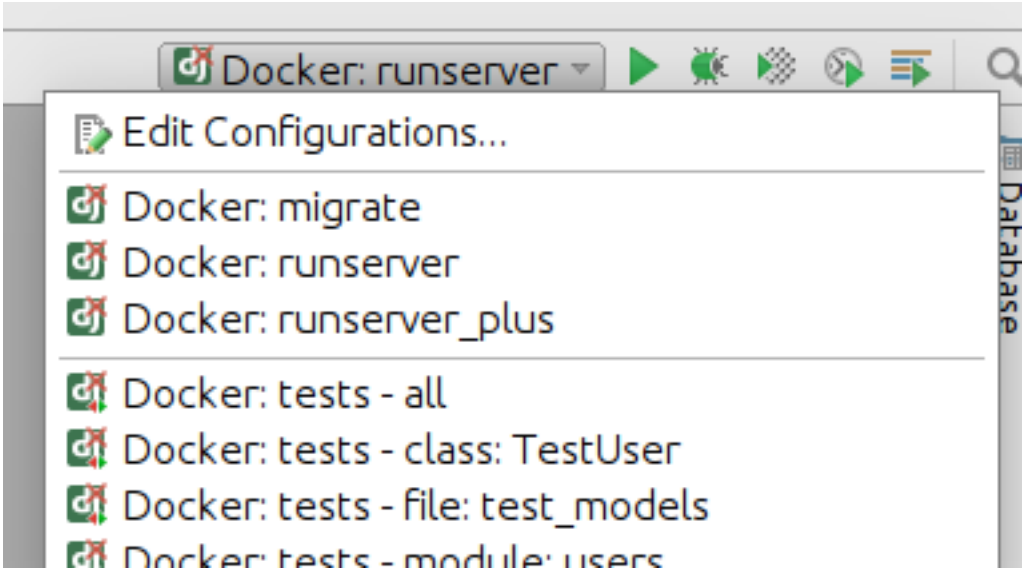
To connect to python remote interpreter inside docker, you have to make sure first, that Pycharm is aware of your docker.

Go to *Settings* > *Build, Execution, Deployment* > *Docker*. If you are on linux, you can use docker directly using its socket *unix:///var/run/docker.sock*, if you are on Windows or Mac, make sure that you have docker-machine installed, then you can simply *Import credentials from Docker Machine*.



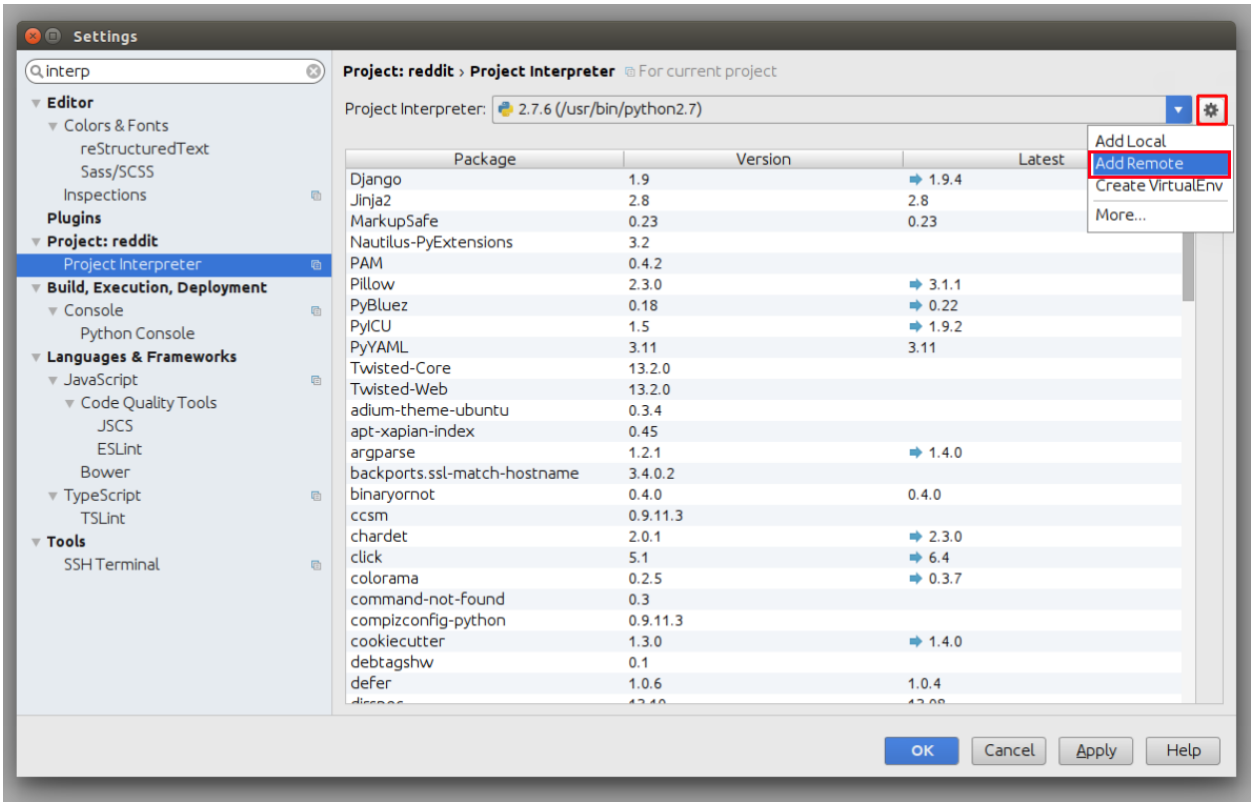
2.1 Configure Remote Python Interpreter

This repository comes with already prepared "Run/Debug Configurations" for docker.

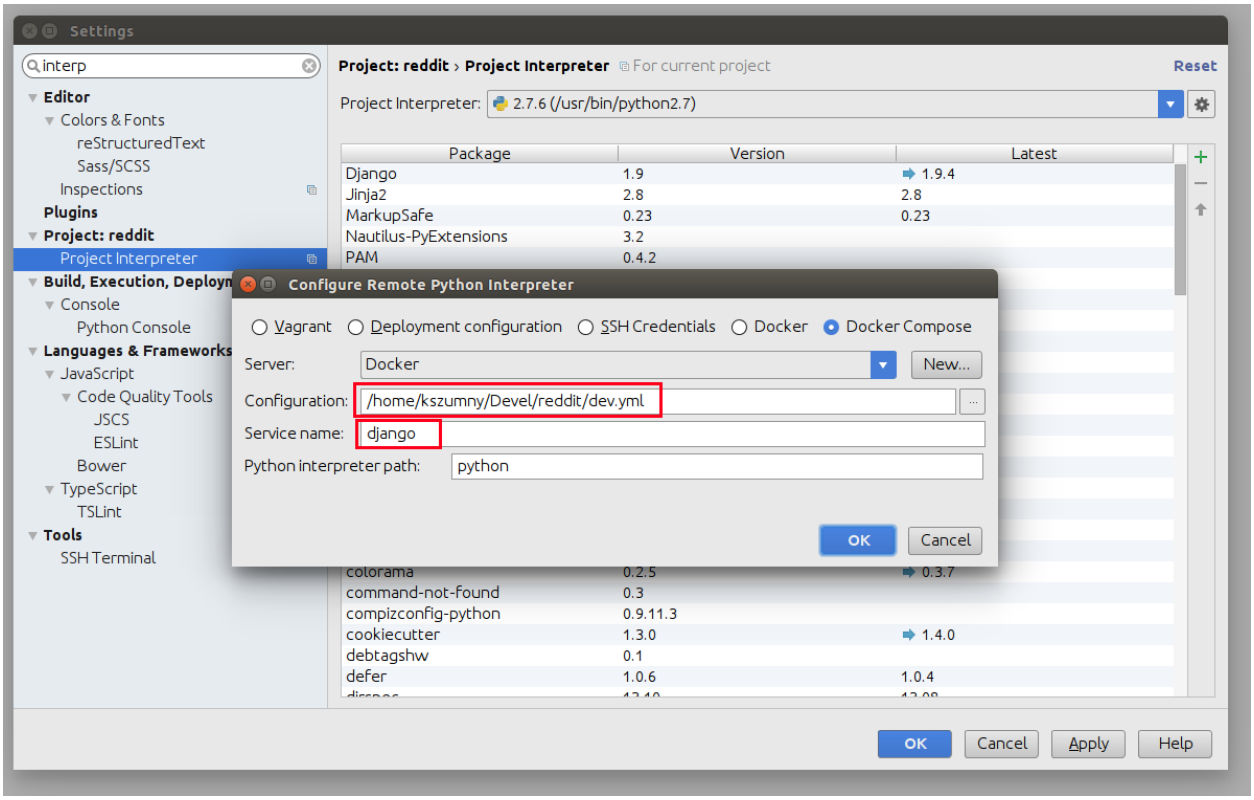


But as you can see, at the beginning there is something wrong with them. They have red X on django icon, and they cannot be used, without configuring remote python interpteter. To do that, you have to go to *Settings* > *Build, Execution, Deployment* first.

Next, you have to add new remote python interpreter, based on already tested deployment settings. Go to *Settings* > *Project* > *Project Interpreter*. Click on the cog icon, and click *Add Remote*.

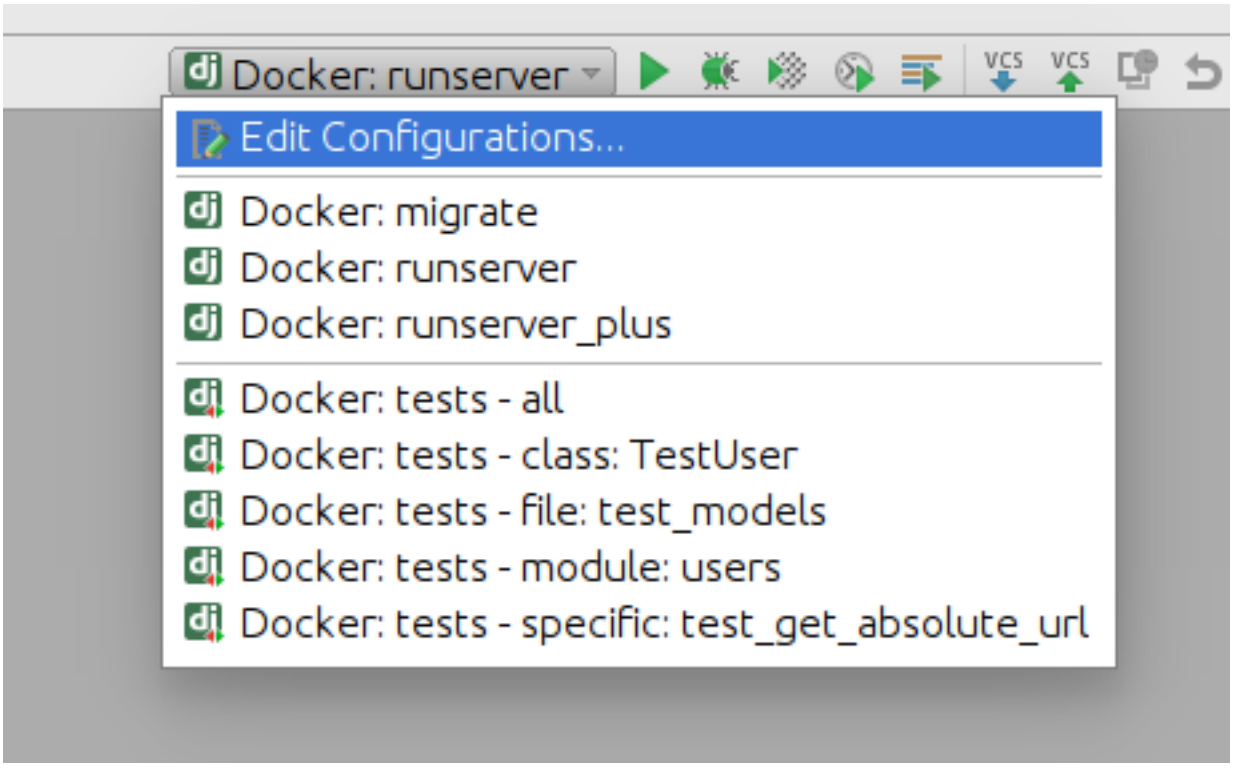


Switch to *Docker Compose* and select *local.yml* file from directory of your project, next set *Service name* to *django*



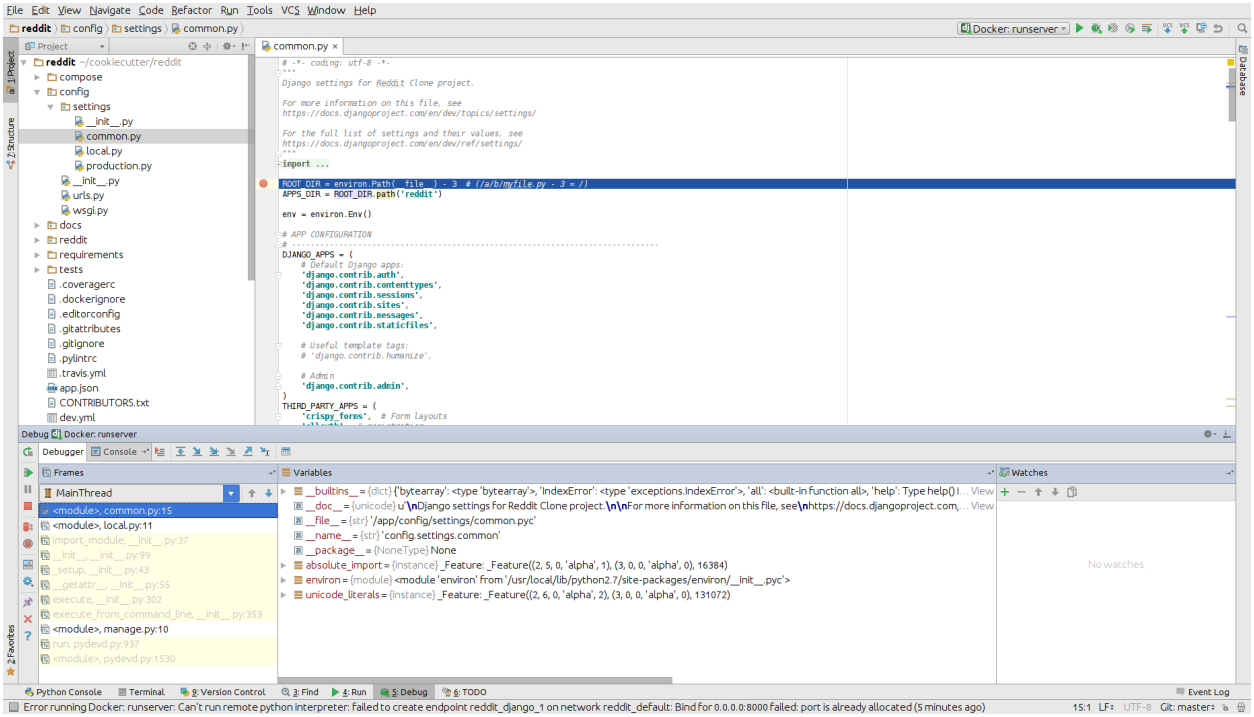
Having that, click *OK*. Close *Settings* panel, and wait few seconds...

After few seconds, all *Run/Debug Configurations* should be ready to use.

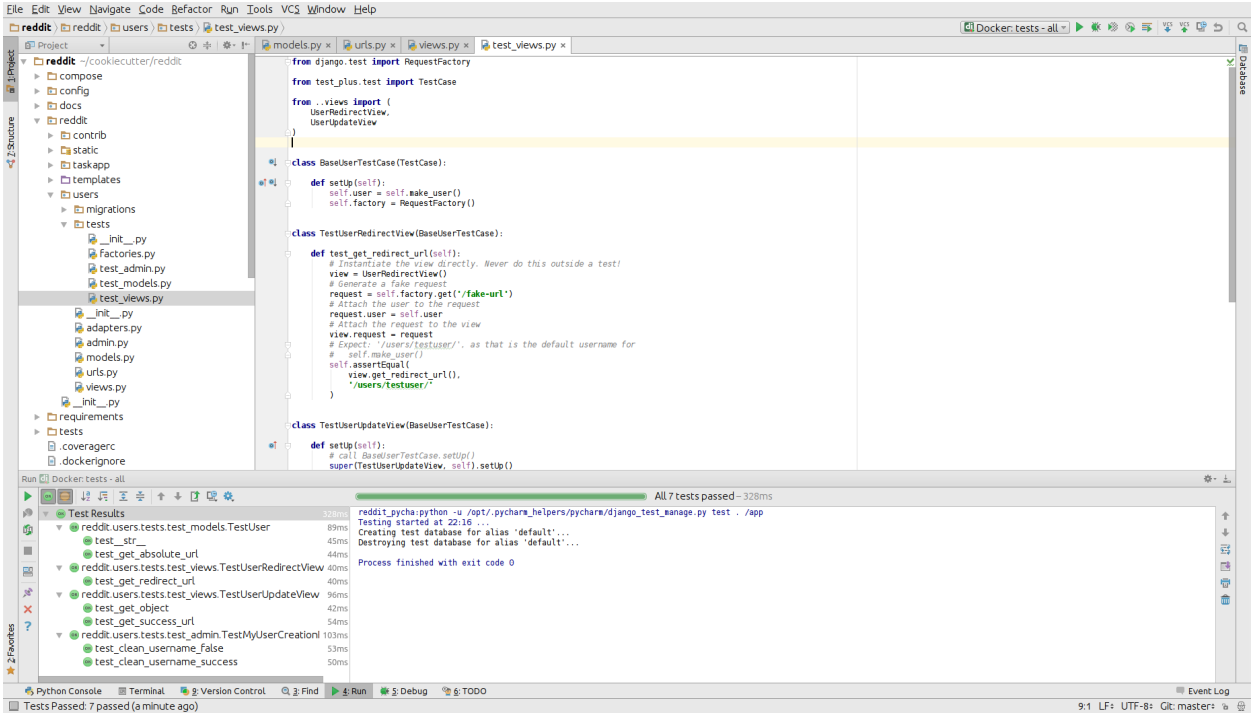


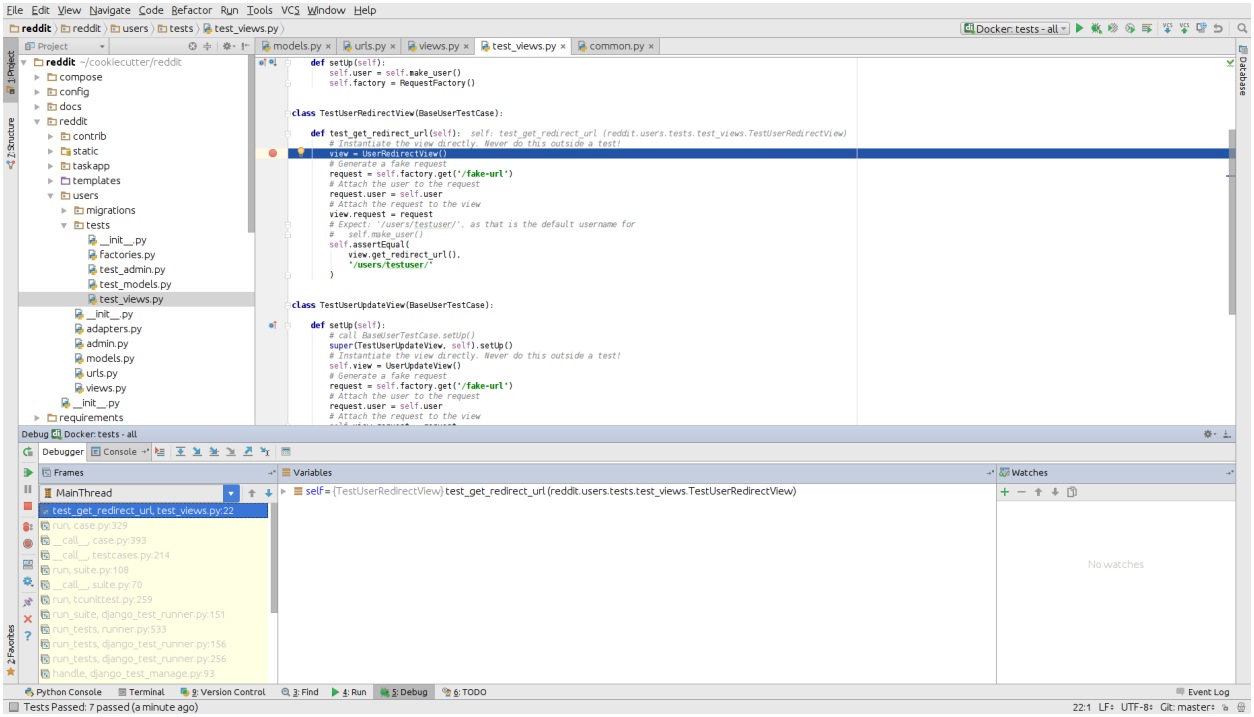
Things you can do with provided configuration:

- run and debug python code

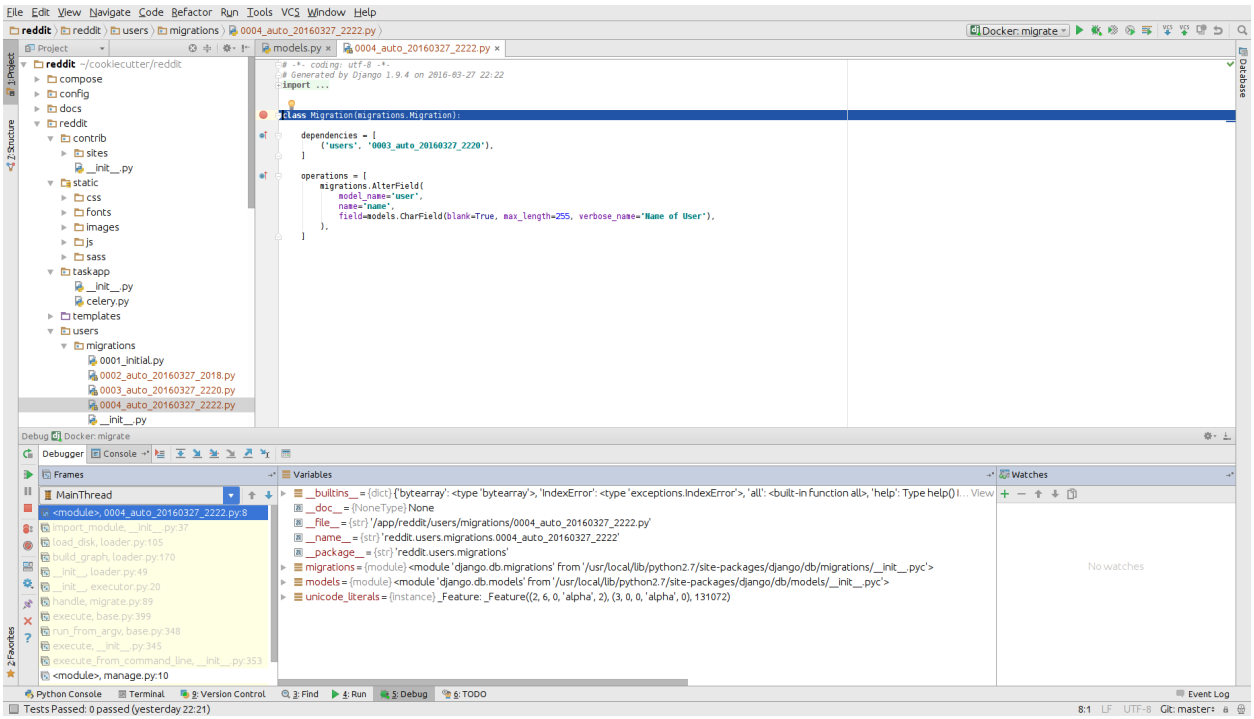


- run and debug tests





- run and debug migrations or different django management commands



- and many others..

2.2 Known issues

- Pycharm hangs on “Connecting to Debugger”

This might be fault of your firewall. Take a look on this ticket - https://youtrack.jetbrains.com/issue/PY-18913

- Modified files in *.idea* directory

Most of the files from *.idea/* were added to *.gitignore* with a few exceptions, which were made, to provide “ready to go” configuration. After adding remote interpreter some of these files are altered by PyCharm:

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .idea/project_name.iml

no changes added to commit (use "git add" and/or "git commit -a")
```

In theory you can remove them from repository, but then, other people will lose a ability to initialize a project from provided configurations as you did. To get rid of this annoying state, you can run command:

```
$ git update-index --assume-unchanged pomodorr.iml
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search