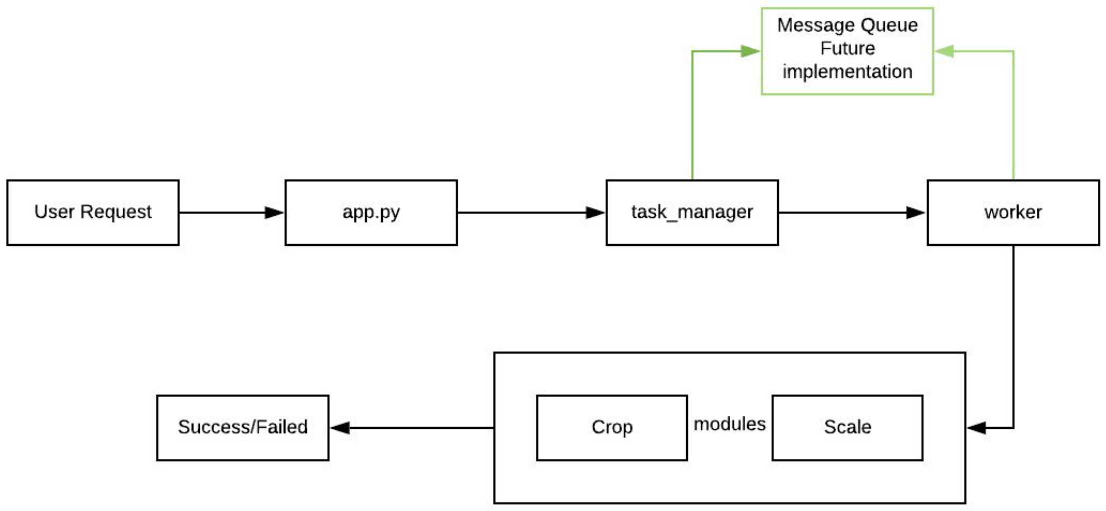
MotoGp Data Preparation Pipeline:

The MotoGp data pipeline is an API based microservice solution, this framework is being developed and provides the below features to the user.

1. Easy access to the user through an API endpoint.
2. Users can send the request in a form of json, with a flexible number of parameters.
3. Highly scalable, it can be easily scale up and down based on the volume of images it's going to process.
4. Easy to add or remove new data processing functionality.
5. Easy to register new data sources, today we have for MotoGp tomorrow it could be soccer.
6. Backend processing engine is running in containers, so it's easy to deploy and manage dependencies.
7. Processing engine will be triggered on demand with a mix of spot and on demand ec2, so it's cost effective.
8. Easy to maintain, for lots of enhancement it just needs a small configuration file change.
9. This framework will be deployed in AWS private network and we can make sure data is also secure at any state with encryption keys and ssl.

Architecture:



Usage:

1. Unzip the code.
2. Run docker build --tag motogpaug:latest .
3. docker run -it -p 8888:8888 -p 5000:5000 motogpaug:latest
4. API endpoint will be available http://localhost:5000/
5. Do a post call in http://localhost:5000/data_prep
6. Sample user request

```
[
        {
                "name": "MotoGP",
                "input_image_size": [480,270],
                "augmentations": [
                        {
                                "CROP":{
                                         "crop_dimensions": [
                                                [270,270]
                                        ],
                                        "crop_type": "left_down_center"
                                 },
                                "RANDOM_CROP": {
                                        "crop_dimensions": [80,80],
                                        "target_count": 3
                                }
                        }
                 ],
                "train_test_split": {
                        "train_ratio": 80,
                        "test_ratio": 20
                }
        }
]
```

7. The highlighted parameters are optional, like if you want to resize the images you can provide or else it will take the image size as input, same for train test split. These are optional because sometime users might be interested only in augmentation not to split it in train tests.
8. A jupyter notebook will also be available in port 8888, you can use the jupyter notebook also to access the data pipeline api.

Not Included:

1. Logging will just display the logs in the console and a separate module is kept for future improvement.
2. Logger is just a placeholder in the module, we need to develop this part to deliver the log to the enterprise central logging like cloudwatch, logz.io, elastic search etc.
3. MQ between tasks manager and worker is not implemented to decouple the process for scalability.
4. Flask is being used here, which is not a production ready API framework, Need to change it to WSGI while running it in production.