

# RPC PS Benchmark

## How to add your experiment

1. Data
  - Create a data class and add it to the data directory
  - Update `benchmark_class_helper.py` to include your data class in the `data_map`
  - Add configurations to `data_configurations.json` in the configurations directory
2. Model
  - Create a model class and add it to the model directory
  - Update `benchmark_class_helper.py` to include your model class in the `model_map`
  - Add configurations to `model_configurations.json` in the configurations directory
3. Trainer
  - Create a trainer class and add it to the trainer directory
  - Update `benchmark_class_helper.py` to include your trainer class in the `trainer_map`
  - Add configurations to `trainer_configurations.json` in the configurations directory
4. Parameter Server
  - Create a parameter server class and add it to the `parameter_servers` directory
  - Update `benchmark_class_helper.py` to include your `parameter_server` class in the `ps_map`
  - Add configurations to `parameter_server_configurations.json` in the configurations directory
5. Script
  - Create a bash script for your experiment and add it to the `experiment_scripts` directory
6. Testing
  - Add a test method for your script to `test_scripts.py`

## Trainer class

The trainer directory contains base classes to provide a starting point for implementing a trainer. Inherit from a base class and implement your trainer. The benchmark has two requirements for trainers.

1. It must implement a **`init`** method that takes `rank`, `trainer_count`, and `ps_rref` as arguments

```
def __init__(self, rank, trainer_count, ps_rref, backend, use_cuda_rpc):
```
2. It must implement a `train` method that takes `model` and `data` as arguments.

```
def train(self, model, data):
```

## Parameter Server class

The `parameter_server` directory contains base classes to provide a starting point for implementing a parameter server. Inherit from a base class and implement your parameter server. The benchmark has two requirements for parameter servers.

1. It must implement a **init** method that takes `rank` and `ps_trainer_count` as arguments

```
def __init__(self, rank, ps_trainer_count, backend, use_cuda_rpc):
```

2. It must implement a `reset_state` method

```
def reset_state(ps_rref):
```

## Testing

Use `pytest` to run the test methods added to `test_scripts.py`. To test all the scripts added use `pytest test_scripts.py`.