Meeshkan Documentation

The Great and Mighty Meeshkan Developers

Mar 08, 2019

Contents

`meeshkan` is a Python package providing control to your machine learning jobs.

CHAPTER 1

Main Features

Here are just a few of the things `meeshkan` can do:

- Notify you of your job's progress at fixed intervals.
- Notify you when certain events happen
- Schedule machine learning jobs for execution
- Allow you to control training jobs remotely
- Allow monitoring Amazon SageMaker jobs

CHAPTER 2

Quick start

We recommend running all command-line commands below in a new Python virtual environment.

2.1 Sign-up

Sign up at meeshkan.com and you will get your **API key**, also referred to as *token*.

2.2 Installation

Install `meeshkan` with pip:

```
$ pip install meeshkan
```

If install fails, your Python version may be too old. Please try again with **Python >= 3.6.2**.

2.3 Setup

Setup your credentials:

```
$ meeshkan setup
```

You are prompted for your **API key** that you should have received when signing up, so fill that in. If you'd like to be able to run code from private GitHub repositories, you may also fill an optional GitHub Access Token (only read access is required) when running `meeshkan setup`. You may skip that prompt otherwise. The command creates the folder `.meeshkan` in your home directory. The folder contains your credentials, agent logs and outputs from your submitted jobs.

Start the agent:

```
$ meeshkan start
```

If starting the agent fails, check that your credentials are properly setup. Also check known issues.

2.4 Running jobs from the command line

Download example script called report.py from meeshkan-client examples folder to your current directory:

```
$ wget https://raw.githubusercontent.com/Meeshkan/meeshkan-client/dev/examples/report.
↪py
```

The script uses *meeshkan.report_scalar()* to report scalar values to the agent. These values are included in the job notifications sent at fixed intervals.

Submit the example job with 10 second reporting interval:

```
$ meeshkan submit --name report-example --report-interval 10 report.py
```

The command schedules the script for execution. As there is nothing else in the queue, execution starts immediately.

If you setup Slack integration at meeshkan.com, you should receive a notification for job being started. You should get notifications every ten seconds. The script runs for 20 seconds, so you should get one notification containing scalar values.

The script uses *meeshkan.report_scalar()* to report scalar values to the agent. These scalar values are included in the job notifications sent at fixed intervals.

You can list the submitted jobs with:

```
$ meeshkan list
```

Retrieve logs for the job named `report-example`:

```
$ meeshkan logs report-example
```

Stop the agent:

```
$ meeshkan stop
```

2.5 Running jobs from Python

Download example script called blocking_job.py:

```
$ wget https://raw.githubusercontent.com/Meeshkan/meeshkan-client/dev/examples/
↪blocking_job.py
```

Execute the script:

```
$ python blocking_job.py
```

If you setup Slack integration at meeshkan.com, you should again receive a notification for a job being started.

Note that unlike `meeshkan submit` used above, this example uses *meeshkan.as_blocking_job()* to notify Meeshkan agent of the job context. The decorated function is executed immediately in the calling process, thereby

blocking the terminal until the script finishes execution. Running blocking jobs in this manner is a simple way to run Python scripts with Meeshkan notifications if you do not need the agent's scheduling capabilities.

2.6 PyTorch example

You can use Meeshkan with any Python machine learning framework. As an example, let us use PyTorch to train a convolution neural network on MNIST.

First install `torch` and `torchvision`:

```
$ pip install torch torchvision
```

Then download the PyTorch example:

```
$ wget https://raw.githubusercontent.com/Meeshkan/meeshkan-client/dev/examples/
↪pytorch_mnist.py
```

Ensure that the agent is running:

```
$ meeshkan start
```

Submit the PyTorch example with a one-minute report interval:

```
$ meeshkan submit --name pytorch-example --report-interval 60 pytorch_mnist.py
```

CHAPTER 3

Meeshkan Python API

meeshkan.**save_token**(*token: str*)

Save Meeshkan API key to `~/.meeshkan/credentials`. Unlike *meeshkan.init()*, does not start or restart the agent. Creates also the required directories if they do not exist.

Parameters **token** – Meeshkan API key

meeshkan.**submit_git**(*repo: str*, *entry_point: str*, *branch_or_commit: str = None*, *job_name: Optional[str] = None*, *report_interval_secs: Optional[float] = None*)

Submits a GitHub repository as a job to the agent. The agent must be running.

Example:

```
# A basic call would pull the repository at its current state (default branch) and
# run the entry point file.
meeshkan.git.submit(repo="Meeshkan/meeshkan-client",
                    entry_point="examples/pytorch_mnist.py",
                    job_name="example #1", report_interval_secs=60)

# A call with branch name would run the given branch from its most updated commit.
meeshkan.git.submit(repo="Meeshkan/meeshkan-client",
                    entry_point="examples/pytorch_mnist.py",
                    branch_or_commit="dev",
                    job_name="example #2",
                    report_interval_secs=10)

# A call with a given commit will locally reset the repository to the state for
# that commit. Commit_sha can be either short SHA or the full one.
# In this case, either "61657d7" or "1657d79bfd92fda7c19d6ec273b09068f96a777"
# would be fine.
meeshkan.git.submit(repo="Meeshkan/meeshkan-client",
                    entry_point="examples/pytorch_mnist.py",
                    branch_or_commit="61657d7",
                    job_name="example #3",
                    report_interval_secs=30)
```

Parameters

- **repo** – A string describing a **GitHub** repository in plain <user or organization>/<repo name> format
- **entry_point** – A path (relevant to the repository) for the entry point (or file to run)
- **branch_or_commit** – Optional string, describing the either the branch to fork (checkout the branch when cloning locally), or the commit SHA to use.
- **job_name** – Optional name to give to the job
- **report_interval_secs** – Optional float, notification report interval in seconds.

Returns `Job` object

`meeshkan.`**`report_scalar`**(*val_name: str*, *value: float*, **vals*) → bool

Reports scalars to the Meeshkan agent. Reported scalars are included in the sent notifications.

Requires Meeshkan agent to be running and aware of the job context. The job context can be defined in multiple ways:

1. By submitting the script or notebook for execution to the agent with `meeshkan submit`.
2. By decorating a function with *`meeshkan.as_blocking_job()`*
3. By using the job context manager *`meeshkan.create_blocking_job()`*

Example of `train.py` script submitted with `meeshkan submit --name my-job train.py`:

```
import meeshkan

EPOCHS = 10

for epoch in range(EPOCHS):
    # Compute loss
    loss = ...
    # Report loss to the Meeshkan agent
    meeshkan.report_scalar("loss", loss)
```

Parameters

- **val_name** – The name of the scalar to report
- **value** – The value of the scalar
- **vals** – Any additional (val_name, value) pairs to add.

Return bool True if job was found, False if not.

`meeshkan.`**`add_condition`**(**vals*, *condition*, *only_reported=False*)

Adds a condition to send notification when scalars fulfill a condition. Requires Meeshkan agent to be running and aware of the job context as for *`meeshkan.report_scalar()`*.

Example:

```
# Add a condition to notify when training loss is less than 0.8
meeshkan.add_condition("train_loss", lambda v: v < 0.8)

# Add another condition to notify when `val_loss` and `val_acc` are smaller and␣
↪greater
# than given values, respectively
```

(continues on next page)

(continued from previous page)

```
meeshkan.add_condition("val_loss", "val_acc", lambda loss, acc: loss < 0.5 and␣
↪acc > 0.95)

for epoch in range(EPOCHS):
    # Compute `train_loss`
    train_loss = ...
    # Report the value to the agent.
    # If the added condition is fulfilled, notification is sent.
    meeshkan.report_scalar("train_loss", train_loss)

    # Report validation results
    if epoch % VALIDATION_INTERVAL == 0:
        val_loss = ...
        val_acc = ...
        meeshkan.report_scalar("val_loss", val_loss, "val_acc", val_acc)
```

Parameters

- **vals** – List of scalar names to include in the condition definition.
- **condition** – A callable accepting as many arguments as listed values and returning boolean.
- **only_reported** – Report all scalars in a job if True, only report the ones relevant to the condition if False. Defaults to False.

meeshkan.**submit_notebook**(*job_name: str = None*, *report_interval: Optional[float] = None*, *notebook_password: str = None*)

Submits the whole Jupyter notebook for execution. Requires the agent to be running. Can only be called from within a notebook instance. On password-protected notebooks, `notebook_password` must be supplied.

Parameters

- **job_name** – Name of the job.
- **report_interval** – Report interval in seconds.
- **notebook_password** – Notebook password, required for password-protected notebooks.

meeshkan.**submit_function**(*func*, *job_name: str = None*, *report_interval: Optional[float] = None*, *args: List[Any] = None*, *kwargs: Dict[str, Any] = None*)

Submits a given function for separate execution. Relevant global variables are serialized in the process and are not updated when the job is done. Requires the agent to be running. This can be invoked from a script, IPython terminal, or Jupyter Notebook, allowing fast prototyping.

Example:

```
>>> def train():
    ... # Does stuff with globally-defined `model`, `train_set`, `optimizer`, etc.
↪..
    ... # Send notification when "loss" is less than 0.8
    ... meeshkan.add_condition("loss", lambda v: v < 0.8)
    ... # Enter training loop
    ... for i in range(EPOCHS):
    ...     # Compute loss
    ...     loss = ...
    ...     # Report loss to the Meeshkan agent
    ...     meeshkan.report_scalar("loss", loss)
```

(continues on next page)

(continued from previous page)

```
>>> meeshkan.submit_function(train)  # Submit default training
>>> optimizer.learning_rate = 0.001
>>> meeshkan.submit_function(train, job_name="lr=0.001")  # Submit with lower␣
↪learning rate
>>> EPOCHS = 1
>>> meeshkan.submit_function(train, job_name="run for one epoch")  # etc...
```

Parameters

- **func** – A function to run
- **job_name** – An optional name for the job.
- **report_interval** – An optional report interval for the job.
- **args** – An optional list of arguments to send to the function.
- **kwargs** – An optional dictionary of keyword arguments to send to the function

Raises **InvalidTypeForFunctionSubmission** – if *func* is not a well defined function.

meeshkan.**create_blocking_job**(*name: str*, *report_interval_secs: Optional[float] = None*) → meeshkan.api.external_job.ExternalJobWrapper

Create a blocking Meeshkan job used as context manager. The job is called blocking because it is not scheduled to the agent for execution. The job can be reused as context manager, ensuring that scalars reported earlier with *meeshkan.report_scalar()* are still included in the notifications.

Example:

```
meeshkan_job = meeshkan.create_blocking_job(name="my-job", report_interval_
↪secs=60)
with meeshkan_job:
    # Send notification when "loss" is less than 0.8
    meeshkan.add_condition("loss", lambda v: v < 0.8)
    # Enter training loop
    for i in range(EPOCHS):
        # Compute loss
        loss = ...
        # Report loss to the Meeshkan agent
        meeshkan.report_scalar("loss", loss)
```

Parameters

- **name** – Name of the job
- **report_interval_secs** – Notification report interval in seconds

Returns Meeshkan blocking job

meeshkan.**as_blocking_job**(*job_name*, *report_interval_secs*)

Mark a function as Meeshkan job: notifications are sent when the function execution begins and ends. If the function reports scalar values with *meeshkan.report_scalar()*, notifications are sent also at the given report intervals.

The function execution blocks the calling process, i.e., execution is not scheduled to the *meeshkan* agent for execution.

Example:

```
@meeshkan.as_blocking_job(job_name="my-job", report_interval_secs=60)
def train():
    # Send notification when "loss" is less than 0.8
    meeshkan.add_condition("loss", lambda v: v < 0.8)
    # Enter training loop
    for i in range(EPOCHS):
        # Compute loss
        loss = ...
        # Report loss to the Meeshkan agent
        meeshkan.report_scalar("loss", loss)
```

Parameters

- **job_name** – Name of the job
- **report_interval_secs** – Notification report interval in seconds.

Returns Function decorator

meeshkan.**start**() → bool
Start the Meeshkan agent.

Return bool True if agent was started, False if agent was already running.

meeshkan.**init**(*token: Optional[str] = None*)
Initialize the Meeshkan agent, optionally with the provided credentials.

- `meeshkan.init()` without the token is equivalent to `meeshkan.restart()`.
- `meeshkan.init(token=...)` with the token is equivalent to `meeshkan.save_token(token=...)` followed by `meeshkan.restart()`.

Parameters **token** – Meeshkan API key, optional. Only required if credentials have not been setup before.

meeshkan.**stop**()
Stop the agent.

meeshkan.**restart**()
Restart the agent. Also reload configuration variables such as credentials.

meeshkan.**is_running**() → bool
Check if the agent is running.

meeshkan.sagemaker.**monitor**(*job_name: str*, *poll_interval: Optional[float] = None*)
Start monitoring a SageMaker training job. Requires the agent to be running.

The agent periodically reads the metrics reported by the job from the SageMaker API and sends Meeshkan notifications.

Requires `sagemaker` Python SDK to be installed. The required AWS credentials are automatically read using the standard Boto credential chain.

Example:

```
job_name = "sagemaker-job"
sagemaker_estimator.fit({'training': inputs}, job_name=job_name, wait=False)
meeshkan.sagemaker.monitor(job_name=job_name, poll_interval=600)
```

Parameters

- **job_name** – SageMaker training job name
- **poll_interval** – Polling interval in seconds, optional. Defaults to one hour.

CHAPTER 4

Command-line interface

4.1 meeshkan

Command-line interface for working with the Meeshkan agent. If no `COMMAND` is given, it is assumed to be `submit`.

Use `meeshkan COMMAND -h` to get help for given `COMMAND`.

```
meeshkan [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Show the version and exit.

--debug

--silent

4.1.1 cancel

Cancel a queued/running job.

`JOB_IDENTIFIER` is either the job's ID, number, name, or pattern to match against the job's name.

```
meeshkan cancel [OPTIONS] JOB_IDENTIFIER
```

Arguments

JOB_IDENTIFIER
Required argument

4.1.2 clean

Alias for `meeshkan clear`.

```
meeshkan clean [OPTIONS]
```

4.1.3 clear

Clear Meeshkan log and job directories in `~/.meeshkan`.

```
meeshkan clear [OPTIONS]
```

4.1.4 help

Show this message and exit.

```
meeshkan help [OPTIONS]
```

4.1.5 im-bored

???

```
meeshkan im-bored [OPTIONS]
```

4.1.6 list

List the job queue and status for each job.

```
meeshkan list [OPTIONS]
```

4.1.7 logs

Retrieve the logs for a given job.

`JOB_IDENTIFIER` is either the job's ID, number, name, or pattern to match against the job's name.

```
meeshkan logs [OPTIONS] JOB_IDENTIFIER
```

Arguments

JOB_IDENTIFIER
Required argument

4.1.8 notifications

Retrieve notification history for a given job.

`JOB_IDENTIFIER` is either the job's ID, number, name, or pattern to match against the job's name.

```
meeshkan notifications [OPTIONS] JOB_IDENTIFIER
```

Arguments

JOB_IDENTIFIER
Required argument

4.1.9 report

Print the latest scalars reported for a job.

`JOB_IDENTIFIER` is either the job's ID, number, name, or pattern to match against the job's name.

```
meeshkan report [OPTIONS] JOB_IDENTIFIER
```

Arguments

JOB_IDENTIFIER
Required argument

4.1.10 setup

Configure the Meeshkan agent.

```
meeshkan setup [OPTIONS]
```

4.1.11 sorry

Send error logs to Meeshkan HQ. Sorry for inconvenience!

```
meeshkan sorry [OPTIONS]
```

4.1.12 start

Start the agent.

```
meeshkan start [OPTIONS]
```

4.1.13 status

Print the agent status.

```
meeshkan status [OPTIONS]
```

4.1.14 stop

Stop the agent.

```
meeshkan stop [OPTIONS]
```

4.1.15 submit

Submit a new job to the agent.

`ARGS` can be either a single file (extension `.py`, `.ipynb`, or `.sh`) or a shell command such as `echo Hello`.

```
meeshkan submit [OPTIONS] [ARGS]...
```

Options

-n, --name <name>
Job name

-r, --report-interval <report_interval>
Number of seconds between each report for this job. [default: 3600.0]

Arguments

ARGS
Optional argument(s)

Python Module Index

m

Index

Symbols

A

C

I

J

M

R

S