Version: 3.x

Command Line Interface

mands and the parameters you can pass to them.

Cheat Sheet

Command	Effect
`rasa init`	Creates a new project with example training data, actions, and config files.
`rasa train`	Trains a model using your NLU data and stories, saves trained model in `./models`.
`rasa interactive`	Starts an interactive learning session to create new training data by chatting to your assistant.
`rasa shell`	Loads your trained model and lets you talk to your assistant on the command line.
`rasa run`	Starts a server with your trained model.
`rasa run actions`	Starts an action server using the Rasa SDK.
`rasa visualize`	Generates a visual representation of your stories.
`rasa test`	Tests a trained Rasa model on any files starting with `test_`.
`rasa test e2e`	Runs end-to-end testing fully integrated with the action server that serves as acceptance testing.
`rasa data split nlu`	Performs a 80/20 split of your NLU training data.
`rasa data split stories`	Do the same as `rasa data split nlu`, but for your stories data.
`rasa data convert`	Converts training data between different formats.
`rasa data migrate`	Migrates 2.0 domain to 3.0 format.
`rasa data validate`	Checks the domain, NLU and conversation

Command	Effect
	data for inconsistencies.
`rasa export`	Exports conversations from a tracker store to an event broker.
`rasa evaluate markers`	Extracts markers from an existing tracker store.
`rasa marker upload`	Upload marker configurations to Analytics Data Pipeline
`rasa license`	Display licensing information.
`rasa -h`	Shows all available commands.

NOTE

If you run into character encoding issues on Windows like: `UnicodeEncodeError: 'charmap' codec can't encode character ...` or the terminal is not displaying colored messages properly, prepend `winpty` to the command you would like to run. For example `winpty rasa init` instead of `rasa init`

Log Level

Rasa produces log messages at several different levels (eg. warning, info, error and so on). You can control which level of logs you would like to see with `--verbose` (same as `-v`) or `--debug` (same as `-vv`) as optional command line arguments. See each command below for more explanation on what these arguments mean.

In addition to CLI arguments, several environment variables allow you to control log output in a more granular way. With these environment variables, you can configure log levels for messages created by external libraries such as Matplotlib, Pika, and Kafka. These variables follow **standard logging level in Python**. Currently, following environment variables are supported:

1. LOG_LEVEL_LIBRARIES: This is the general environment variable to configure log level for the main libraries Rasa uses. It covers Tensorflow, `asyncio`, APScheduler, SocketIO, Matplotlib, RabbitMQ, Kafka.
2. LOG_LEVEL_MATPLOTLIB: This is the specialized environment variable to configure log level only for Matplotlib.
3. LOG_LEVEL_RABBITMQ: This is the specialized environment variable to configure log level only for AMQP libraries, at the moment it handles log levels from `aio_pika` and `aiormq`.

4. LOG_LEVEL_KAFKA: This is the specialized environment variable to configure log level only for kafka.
5. LOG_LEVEL_PRESIDIO: This is the specialized environment variable to configure log level only for Presidio, at the moment it handles log levels from `presidio_analyzer` and `presidio_anonymizer`.
6. LOG_LEVEL_FAKER: This is the specialized environment variable to configure log level only for Faker.

General configuration (`LOG_LEVEL_LIBRARIES`) has less priority than library level specific configuration (`LOG_LEVEL_MATPLOTLIB`, `LOG_LEVEL_RABBITMQ` etc); and CLI parameter sets the lowest level log messages which will be handled. This means variables can be used together with a predictable result. As an example:

```
LOG_LEVEL_LIBRARIES=ERROR LOG_LEVEL_MATPLOTLIB=WARNING LOG_LEVEL_KA
```

The above command run will result in showing:

- messages with `DEBUG` level and higher by default (due to `--debug`)
- messages with `WARNING` level and higher for Matplotlib
- messages with `DEBUG` level and higher for kafka
- messages with `ERROR` level and higher for other libraries not configured

Note that CLI config sets the lowest level log messages to be handled, hence the following command will set the log level to `INFO` (due to `--verbose`) and no debug messages will be seen (library level configuration will not have any effect):

```
LOG_LEVEL_LIBRARIES=DEBUG LOG_LEVEL_MATPLOTLIB=DEBUG rasa shell --v
```

As an aside, CLI log level sets the level at the root logger (which has the important handler - `coloredlogs` handler); this means even if an environment variable sets a library logger to a lower level, the root logger will reject messages from that library. If not specified, the CLI log level is set to `INFO` .

Custom logging configuration

> **NEW IN 3.4**

> The Rasa CLI now includes a new argument `--logging-config-file` which accepts a YAML file as value.

You can now configure any logging formatters or handlers in a separate YAML file. The logging config YAML file must follow the **Python built-in dictionary schema**, otherwise it will fail validation. You can pass this file as argument to the `--logging-config-file` CLI option and use it with any of the rasa commands.

---

rasa init

This command sets up a complete assistant for you with some example training data:

```
rasa init
```

It creates the following files:

```
.
├── actions
│   ├── __init__.py
│   └── actions.py
├── config.yml
├── credentials.yml
├── data
│   ├── nlu.yml
│   └── stories.yml
├── domain.yml
├── endpoints.yml
├── models
│   └── <timestamp>.tar.gz
└── tests
    └── test_stories.yml
```

It will ask you if you want to train an initial model using this data. If you answer no, the `models` directory will be empty.

Any of the default CLI commands will expect this project setup, so this is the best way to get started. You can run `rasa train`, `rasa shell` and `rasa test` without any additional configuration.

---

rasa train

The following command trains a Rasa model:

```
rasa train
```

If you have existing models in your directory (under `models/` by default), only the parts of your model that have changed will be re-trained. For example, if you edit your NLU training data and nothing else, only the NLU part will be trained.

If you want to train an NLU or dialogue model individually, you can run `rasa train nlu` or `rasa train core`. If you provide training data only for one one of these, `rasa train` will fall back to one of these commands by default.

`rasa train` will store the trained model in the directory defined by `--out`, `models/` by default. The name of the model by default is `<timestamp>.tar.gz`. If you want to name your model differently, you can specify the name using the `--fixed-model-name` flag.

By default validation is run before training the model. If you want to skip validation, you can use the `--skip-validation` flag. If you want to fail on validation warnings, you can use the `--fail-on-validation-warnings` flag. The `--validation-max-history` is analogous to the `--max-history` argument of `rasa data validate`.

The following arguments can be used to configure the training process:

```
usage: rasa train [-h] [-v] [-vv] [--quiet]
                  [--logging-config-file LOGGING_CONFIG_FILE]
                  [--data DATA [DATA ...]] [-c CONFIG] [-d DOMAIN]
                  [--dry-run] [--skip-validation]
                  [--fail-on-validation-warnings]
                  [--validation-max-history VALIDATION_MAX_HISTORY]
                  [--augmentation AUGMENTATION] [--debug-plots]
                  [--num-threads NUM_THREADS]
                  [--fixed-model-name FIXED_MODEL_NAME] [--persist-
                  [--force] [--finetune [FINETUNE]]
                  [--epoch-fraction EPOCH_FRACTION] [--endpoints EN
                  {core,nlu} ...

positional arguments:
  {core,nlu}
    core                Trains a Rasa Core model using your stories
    nlu                 Trains a Rasa NLU model using your NLU data

options:
  -h, --help            show this help message and exit
  --data DATA [DATA ...]
                        Paths to the Core and NLU data files. (defa
                        ['data'])
  -c CONFIG, --config CONFIG
                        The policy and NLU pipeline configuration o
                        (default: config.yml)
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
```

```
                        specifications in it. The content of these
                        be read and merged together. (default: dom
  --out OUT             Directory where your models should be stor
                        (default: models)
  --dry-run             If enabled, no actual training will be per
                        Instead, it will be determined whether a m
                        be re-trained and this information will be
                        the output. The return code is a 4-bit bit
                        can also be used to determine what exactly
                        retrained: - 0 means that no extensive tra
                        required (note that the responses still mi
                        updating by running 'rasa train'). - 1 mea
                        needs to be retrained - 8 means the traini
                        forced (--force argument is specified) (de
                        False)
  --skip-validation     Skip validation step before training. (def
  --fail-on-validation-warnings
                        Fail on validation warnings. If omitted on
                        will exit with a non zero status code (def
  --validation-max-history VALIDATION_MAX_HISTORY
                        Number of turns taken into account for sto
                        validation. (default: None)
  --augmentation AUGMENTATION
                        How much data augmentation to use during t
                        (default: 50)
  --debug-plots         If enabled, will create plots showing chec
                        their connections between story blocks in
                        called `story_blocks_connections.html`. (d
                        False)
  --num-threads NUM_THREADS
                        Maximum amount of threads to use when trai
                        (default: None)
  --fixed-model-name FIXED_MODEL_NAME
                        If set, the name of the model file/directo
                        set to the given name. (default: None)
  --persist-nlu-data    Persist the NLU training data in the saved
                        (default: False)
  --force               Force a model training even if the data ha
                        changed. (default: False)
  --finetune [FINETUNE]
                        Fine-tune a previously trained model. If n
                        is provided, Rasa Open Source will try to
                        latest trained model from the model direct
                        specified via '--out'. (default: None)
  --epoch-fraction EPOCH_FRACTION
                        Fraction of epochs which are currently spe
                        the model configuration which should be us
                        finetuning a model. (default: None)
  --endpoints ENDPOINTS
                        Configuration file for the connectors as a
                        (default: endpoints.yml)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (d
                        None)
  -vv, --debug          Print lots of debugging statements. Sets l
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (
                        None)
```

```
--logging-config-file LOGGING_CONFIG_FILE
                      If set, the name of the logging configurat
                      will be set to the given name. (default: N
```

See the section on **data augmentation** for info on how data augmentation works and how to choose a value for the flag. Note that `TEDPolicy` is the only policy affected by data augmentation.

See the following section on **incremental training** for more information about the `--epoch-fraction` argument.

Incremental training

> **NEW IN 2.2**
>
> This feature is experimental. We introduce experimental features to get feedback from our community, so we encourage you to try it out! However, the functionality might be changed or removed in the future. If you have feedback (positive or negative) please share it with us on the **Rasa Forum**.

In order to improve the performance of an assistant, it's helpful to practice **CDD** and add new training examples based on how your users have talked to your assistant. You can use `rasa train --finetune` to initialize the pipeline with an already trained model and further finetune it on the new training dataset that includes the additional training examples. This will help reduce the training time of the new model.

By default, the command picks up the latest model in the `models/` directory. If you have a specific model which you want to improve, you may specify the path to this by running `rasa train --finetune <path to model to finetune>`. Finetuning a model usually requires fewer epochs to train machine learning components like `DIETClassifier`, `ResponseSelector` and `TEDPolicy` compared to training from scratch. Either use a model configuration for finetuning which defines fewer epochs than before or use the flag `--epoch-fraction`. `--epoch-fraction` will use a fraction of the epochs specified for each machine learning component in the model configuration file. For example, if `DIETClassifier` is configured to use 100 epochs, specifying `--epoch-fraction 0.5` will only use 50 epochs for finetuning.

You can also finetune an NLU-only or dialogue management-only model by using `rasa train nlu --finetune` and `rasa train core --finetune`

respectively.

To be able to fine tune a model, the following conditions must be met:

1. The configuration supplied should be exactly the same as the configuration used to train the model which is being finetuned. The only parameter that you can change is `epochs` for the individual machine learning components and policies.
2. The set of labels(intents, actions, entities and slots) for which the base model is trained should be exactly the same as the ones present in the training data used for finetuning. This means that you cannot add new intent, action, entity or slot labels to your training data during incremental training. You can still add new training examples for each of the existing labels. If you have added/removed labels in the training data, the pipeline needs to be trained from scratch.
3. The model to be finetuned is trained with `MINIMUM_COMPATIBLE_VERSION` of the currently installed rasa version.

---

rasa interactive

You can start an interactive learning session by running:

```
rasa interactive
```

This will first train a model and then start an interactive shell session. You can then correct your assistants predictions as you talk to it. If `UnexpecTEDIntentPolicy` is included in the pipeline, `action_unlikely_intent` can be triggered at any conversation turn. Subsequently, the following message will be displayed:

```
The bot wants to run 'action_unlikely_intent' to indicate that the
at this point in the conversation. Check out UnexpecTEDIntentPolic
```

As the message states, this is an indication that you have explored a conversation path which is unexpected according to the current set of training stories and hence adding this path to training stories is recommended. Like other bot actions, you can choose to confirm or deny running this action.

If you provide a trained model using the `--model` argument, training is skipped and that model will be loaded instead.

During interactive learning, Rasa will plot the current conversation and a few similar conversations from the training data to help you keep track of where you are. You can view the visualization at http://localhost:5005/visualization.html as soon as the session has started.

This diagram can take some time to generate. To skip the visualization, run `rasa interactive --skip-visualization`.

> **ADD THE `ASSISTANT_ID` KEY INTRODUCED IN 3.5**
>
> Running interactive learning with a pre-trained model whose metadata does not include the `assistant_id` will exit with an error. If this happens, add the required key with a unique identifier value in `config.yml` and re-run training.

The following arguments can be used to configure the interactive learning session:

```
usage: rasa interactive [-h] [-v] [-vv] [--quiet]
                        [--logging-config-file LOGGING_CONFIG_FILE]
                        [-p PORT] [-m MODEL] [--data DATA [DATA ...]]
                        [--skip-visualization]
                        [--conversation-id CONVERSATION_ID]
                        [--endpoints ENDPOINTS] [-c CONFIG] [-d DOM]
                        [--out OUT] [--augmentation AUGMENTATION]
                        [--debug-plots] [--finetune [FINETUNE]]
                        [--epoch-fraction EPOCH_FRACTION] [--force]
                        [--persist-nlu-data]
                        {core} ... [model-as-positional-argument]

positional arguments:
  {core}
    core                Starts an interactive learning session mode
                        new training data for a Rasa Core model by
                        Uses the 'RegexMessageHandler', i.e. `/<int
                        format.
  model-as-positional-argument
                        Path to a trained Rasa model. If a director
                        specified, it will use the latest model in
                        directory. (default: None)

options:
  -h, --help            show this help message and exit
  --e2e                 Save story files in e2e format. In this for
                        messages will be included in the stories. (
                        False)
  -p PORT, --port PORT  Port to run the server at. (default: 5005)
  -m MODEL, --model MODEL
                        Path to a trained Rasa model. If a director
                        specified, it will use the latest model in
                        directory. (default: None)
  --data DATA [DATA ...]
                        Paths to the Core and NLU data files. (defa
                        ['data'])
  --skip-visualization  Disable plotting the visualization during i
                        learning. (default: False)
  --conversation-id CONVERSATION_ID
                        Specify the id of the conversation the mess
                        in. Defaults to a UUID that will be random]
                        generated. (default: 9ba867a8817147f9ae870(
  --endpoints ENDPOINTS
```

```
                        Configuration file for the model server an
                        connectors as a yml file. (default: endpoi

Python Logging Options:
  You can control level of log messages printed. In addition to th
  arguments, a more fine grained configuration can be achieved wit
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (d
                        None)
  -vv, --debug          Print lots of debugging statements. Sets l
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurat
                        will be set to the given name. (default: N

Train Arguments:
  -c CONFIG, --config CONFIG
                        The policy and NLU pipeline configuration
                        (default: config.yml)
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
                        specifications in it. The content of these
                        be read and merged together. (default: dom
  --out OUT             Directory where your models should be stor
                        (default: models)
  --augmentation AUGMENTATION
                        How much data augmentation to use during t
                        (default: 50)
  --debug-plots         If enabled, will create plots showing chec
                        their connections between story blocks in
                        called `story_blocks_connections.html`. (d
                        False)
  --finetune [FINETUNE]
                        Fine-tune a previously trained model. If n
                        is provided, Rasa Open Source will try to
                        latest trained model from the model direct
                        specified via '--out'. (default: None)
  --epoch-fraction EPOCH_FRACTION
                        Fraction of epochs which are currently spe
                        the model configuration which should be us
                        finetuning a model. (default: None)
  --force               Force a model training even if the data ha
                        changed. (default: False)
  --persist-nlu-data    Persist the NLU training data in the saved
                        (default: False)
```

---

rasa shell

You can start a chat session by running:

```
rasa shell
```

By default, this will load up the latest trained model. You can specify a different model to be loaded by using the `--model` flag.

If you start the shell with an NLU-only model, `rasa shell` will output the intents and entities predicted for any message you enter.

If you have trained a combined Rasa model but only want to see what your model extracts as intents and entities from text, you can use the command `rasa shell nlu`.

To increase the logging level for debugging, run:

```
rasa shell --debug
```

> **NOTE**
>
> In order to see the typical greetings and/or session start behavior you might see in an external channel, you will need to explicitly send `/session_start` as the first message. Otherwise, the session start behavior will begin as described in **Session configuration**.

The following arguments can be used to configure the command. Most arguments overlap with `rasa run`; see the **following section** for more info on those arguments.

Note that the `--connector` argument will always be set to `cmdline` when running `rasa shell`. This means all credentials in your credentials file will be ignored, and if you provide your own value for the `--connector` argument it will also be ignored.

```
usage: rasa shell [-h] [-v] [-vv] [--quiet]
                  [--logging-config-file LOGGING_CONFIG_FILE]
                  [--conversation-id CONVERSATION_ID] [-m MODEL]
                  [--log-file LOG_FILE] [--use-syslog]
                  [--syslog-address SYSLOG_ADDRESS]
                  [--syslog-port SYSLOG_PORT]
                  [--syslog-protocol SYSLOG_PROTOCOL] [--endpoints
                  [-i INTERFACE] [-p PORT] [-t AUTH_TOKEN] [--cors
                  [--enable-api] [--response-timeout RESPONSE_TIME(
                  [--request-timeout REQUEST_TIMEOUT]
                  [--remote-storage REMOTE_STORAGE]
                  [--ssl-certificate SSL_CERTIFICATE]
                  [--ssl-keyfile SSL_KEYFILE] [--ssl-ca-file SSL_CA
                  [--ssl-password SSL_PASSWORD] [--credentials CREI
                  [--connector CONNECTOR] [--jwt-secret JWT_SECRET]
                  [--jwt-method JWT_METHOD]
                  [--jwt-private-key JWT_PRIVATE_KEY]
                  {nlu} ... [model-as-positional-argument]

positional arguments:
  {nlu}
```

```
    nlu                 Interprets messages on the command line usi
                        model.
  model-as-positional-argument
                        Path to a trained Rasa model. If a directo
                        specified, it will use the latest model in
                        directory. (default: None)

options:
  -h, --help            show this help message and exit
  --conversation-id CONVERSATION_ID
                        Set the conversation ID. (default:
                        39ef71936f8f41f288953f66100901a9)
  -m MODEL, --model MODEL
                        Path to a trained Rasa model. If a directo
                        specified, it will use the latest model in
                        directory. (default: models)
  --log-file LOG_FILE   Store logs in specified file. (default: Non
  --use-syslog          Add syslog as a log handler (default: False
  --syslog-address SYSLOG_ADDRESS
                        Address of the syslog server. --use-sylog f
                        required (default: localhost)
  --syslog-port SYSLOG_PORT
                        Port of the syslog server. --use-sylog flag
                        required (default: 514)
  --syslog-protocol SYSLOG_PROTOCOL
                        Protocol used with the syslog server. Can b
                        (default) or TCP (default: UDP)
  --endpoints ENDPOINTS
                        Configuration file for the model server and
                        connectors as a yml file. (default: endpoin

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No

Server Settings:
  -i INTERFACE, --interface INTERFACE
                        Network interface to run the server on. (de
                        0.0.0.0)
  -p PORT, --port PORT  Port to run the server at. (default: 5005)
  -t AUTH_TOKEN, --auth-token AUTH_TOKEN
                        Enable token based authentication. Requests
                        provide the token to be accepted. (default:
  --cors [CORS ...]     Enable CORS for the passed origin. Use * to
                        all origins. (default: None)
  --enable-api          Start the web server API in addition to the
                        channel. (default: False)
  --response-timeout RESPONSE_TIMEOUT
                        Maximum time a response can take to process
                        (default: 3600)
  --request-timeout REQUEST_TIMEOUT
                        Maximum time a request can take to process
```

```
                        (default: 300)
  --remote-storage REMOTE_STORAGE
                        Set the remote location where your Rasa mod
                        stored, e.g. on AWS. (default: None)
  --ssl-certificate SSL_CERTIFICATE
                        Set the SSL Certificate to create a TLS sec
                        server. (default: None)
  --ssl-keyfile SSL_KEYFILE
                        Set the SSL Keyfile to create a TLS secured
                        (default: None)
  --ssl-ca-file SSL_CA_FILE
                        If your SSL certificate needs to be verifie
                        specify the CA file using this parameter. (
                        None)
  --ssl-password SSL_PASSWORD
                        If your ssl-keyfile is protected by a passw
                        can specify it using this paramer. (default

Channels:
  --credentials CREDENTIALS
                        Authentication credentials for the connecto
                        file. (default: None)
  --connector CONNECTOR
                        Service to connect to. (default: None)

JWT Authentication:
  --jwt-secret JWT_SECRET
                        Public key for asymmetric JWT methods or sh
                        secretfor symmetric methods. Please also ma
                        use --jwt-method to select the method of th
                        signature, otherwise this argument will be
                        ignored.Note that this key is meant for sec
                        HTTP API. (default: None)
  --jwt-method JWT_METHOD
                        Method used for the signature of the JWT
                        authentication payload. (default: HS256)
  --jwt-private-key JWT_PRIVATE_KEY
                        A private key used for generating web token
                        dependent upon which hashing algorithm is u
                        must be used together with --jwt-secret for
                        the public key. (default: None)
```

---

rasa run

To start a server running your trained model, run:

```
rasa run
```

By default the Rasa server uses HTTP for its communication. To secure the communication with SSL and run the server on HTTPS, you need to provide a valid certificate and the corresponding private key file. You can specify these files as part of the `rasa run` command. If you encrypted your keyfile

with a password during creation, you need to add the `--ssl-password` as well.

```
rasa run --ssl-certificate myssl.crt --ssl-keyfile myssl.key --ssl-
```

Rasa by default listens on each available network interface. You can limit this to a specific network interface using the `-i` command line option.

```
rasa run -i 192.168.69.150
```

Rasa will by default connect to all channels specified in your credentials file. To connect to a single channel and ignore all other channels in your credentials file, specify the name of the channel in the `--connector` argument.

```
rasa run --connector rest
```

The name of the channel should match the name you specify in your credentials file. For supported channels see **the page about messaging and voice channels**.

The following arguments can be used to configure your Rasa server:

```
usage: rasa run [-h] [-v] [-vv] [--quiet]
                [--logging-config-file LOGGING_CONFIG_FILE] [-m MOI
                [--log-file LOG_FILE] [--use-syslog]
                [--syslog-address SYSLOG_ADDRESS] [--syslog-port SY
                [--syslog-protocol SYSLOG_PROTOCOL] [--endpoints EN
                [-i INTERFACE] [-p PORT] [-t AUTH_TOKEN] [--cors [(
                [--enable-api] [--response-timeout RESPONSE_TIMEOUT
                [--request-timeout REQUEST_TIMEOUT]
                [--remote-storage REMOTE_STORAGE]
                [--ssl-certificate SSL_CERTIFICATE]
                [--ssl-keyfile SSL_KEYFILE] [--ssl-ca-file SSL_CA_F
                [--ssl-password SSL_PASSWORD] [--credentials CREDEN
                [--connector CONNECTOR] [--jwt-secret JWT_SECRET]
                [--jwt-method JWT_METHOD] [--jwt-private-key JWT_PR
                {actions} ... [model-as-positional-argument]

positional arguments:
  {actions}
    actions             Runs the action server.
  model-as-positional-argument
                        Path to a trained Rasa model. If a directo
                        specified, it will use the latest model in
                        directory. (default: None)

options:
  -h, --help            show this help message and exit
  -m MODEL, --model MODEL
                        Path to a trained Rasa model. If a directo
```

```
                        specified, it will use the latest model in
                        directory. (default: models)
  --log-file LOG_FILE   Store logs in specified file. (default: Nor
  --use-syslog          Add syslog as a log handler (default: Fals
  --syslog-address SYSLOG_ADDRESS
                        Address of the syslog server. --use-sylog f
                        required (default: localhost)
  --syslog-port SYSLOG_PORT
                        Port of the syslog server. --use-sylog flag
                        required (default: 514)
  --syslog-protocol SYSLOG_PROTOCOL
                        Protocol used with the syslog server. Can b
                        (default) or TCP (default: UDP)
  --endpoints ENDPOINTS
                        Configuration file for the model server and
                        connectors as a yml file. (default: endpoin

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No

Server Settings:
  -i INTERFACE, --interface INTERFACE
                        Network interface to run the server on. (de
                        0.0.0.0)
  -p PORT, --port PORT  Port to run the server at. (default: 5005)
  -t AUTH_TOKEN, --auth-token AUTH_TOKEN
                        Enable token based authentication. Requests
                        provide the token to be accepted. (default:
  --cors [CORS ...]     Enable CORS for the passed origin. Use * to
                        all origins. (default: None)
  --enable-api          Start the web server API in addition to the
                        channel. (default: False)
  --response-timeout RESPONSE_TIMEOUT
                        Maximum time a response can take to process
                        (default: 3600)
  --request-timeout REQUEST_TIMEOUT
                        Maximum time a request can take to process
                        (default: 300)
  --remote-storage REMOTE_STORAGE
                        Set the remote location where your Rasa mod
                        stored, e.g. on AWS. (default: None)
  --ssl-certificate SSL_CERTIFICATE
                        Set the SSL Certificate to create a TLS sec
                        server. (default: None)
  --ssl-keyfile SSL_KEYFILE
                        Set the SSL Keyfile to create a TLS secured
                        (default: None)
  --ssl-ca-file SSL_CA_FILE
                        If your SSL certificate needs to be verifie
                        specify the CA file using this parameter. (
                        None)
```

```
  --ssl-password SSL_PASSWORD
                        If your ssl-keyfile is protected by a passv
                        can specify it using this paramer. (default

Channels:
  --credentials CREDENTIALS
                        Authentication credentials for the connecto
                        file. (default: None)
  --connector CONNECTOR
                        Service to connect to. (default: None)

JWT Authentication:
  --jwt-secret JWT_SECRET
                        Public key for asymmetric JWT methods or sh
                        secretfor symmetric methods. Please also ma
                        use --jwt-method to select the method of th
                        signature, otherwise this argument will be
                        ignored.Note that this key is meant for sec
                        HTTP API. (default: None)
  --jwt-method JWT_METHOD
                        Method used for the signature of the JWT
                        authentication payload. (default: HS256)
  --jwt-private-key JWT_PRIVATE_KEY
                        A private key used for generating web token
                        dependent upon which hashing algorithm is u
                        must be used together with --jwt-secret for
                        the public key. (default: None)
```

For more information on the additional parameters, see **Model Storage**. See the Rasa **HTTP API** page for detailed documentation of all the endpoints.

---

rasa run actions

To start an action server with the Rasa SDK, run:

```
rasa run actions
```

The following arguments can be used to adapt the server settings:

```
usage: rasa run actions [-h] [-v] [-vv] [--quiet]
                        [--logging-config-file LOGGING_CONFIG_FILE]
                        [--cors [CORS ...]] [--actions ACTIONS]
                        [--ssl-keyfile SSL_KEYFILE]
                        [--ssl-certificate SSL_CERTIFICATE]
                        [--ssl-password SSL_PASSWORD] [--auto-reloa

options:
  -h, --help            show this help message and exit
  -p PORT, --port PORT  port to run the server at (default: 5055)
  --cors [CORS ...]     enable CORS for the passed origin. Use * to
                        all origins (default: None)
  --actions ACTIONS     name of action package to be loaded (defaul
```

```
  --ssl-keyfile SSL_KEYFILE
                        Set the SSL certificate to create a TLS sec
                        server. (default: None)
  --ssl-certificate SSL_CERTIFICATE
                        Set the SSL certificate to create a TLS sec
                        server. (default: None)
  --ssl-password SSL_PASSWORD
                        If your ssl-keyfile is protected by a passv
                        can specify it using this paramer. (default
  --auto-reload         Enable auto-reloading of modules containing
                        subclasses. (default: False)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

rasa visualize

To generate a graph of your stories in the browser, run:

```
rasa visualize
```

If your stories are located somewhere other than the default location `data/`, you can specify their location with the `--stories` flag.

The following arguments can be used to configure this command:

```
usage: rasa visualize [-h] [-v] [-vv] [--quiet]
                      [--logging-config-file LOGGING_CONFIG_FILE] |
                      [-s STORIES] [--out OUT] [--max-history MAX_H
                      [-u NLU]

options:
  -h, --help            show this help message and exit
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
                        specifications in it. The content of these
                        be read and merged together. (default: doma
  -s STORIES, --stories STORIES
                        File or folder containing your training st
                        (default: data)
```

```
  --out OUT             Filename of the output path, e.g. 'graph.ht
                        (default: graph.html)
  --max-history MAX_HISTORY
                        Max history to consider when merging paths
                        output graph. (default: 2)
  -u NLU, --nlu NLU     File or folder containing your NLU data, us
                        insert example messages into the graph. (de
                        None)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

---

rasa test

To evaluate a model on your test data, run:

```
rasa test
```

This will test your latest trained model on any end-to-end stories you have defined in files with the `test_` prefix. If you want to use a different model, you can specify it using the `--model` flag.

To evaluate the dialogue and NLU models separately, use the commands below:

```
rasa test core
```

and

```
rasa test nlu
```

You can find more details on specific arguments for each testing type in **Evaluating an NLU Model** and **Evaluating a Dialogue Management Model**.

The following arguments are available for `rasa test`:

```
usage: rasa test [-h] [-v] [-vv] [--quiet]
                 [--logging-config-file LOGGING_CONFIG_FILE] [-m MO
                 [-s STORIES] [--max-stories MAX_STORIES]
                 [--endpoints ENDPOINTS] [--fail-on-prediction-erro
                 [--url URL] [--evaluate-model-directory] [-u NLU]
                 [-c CONFIG [CONFIG ...]] [-d DOMAIN] [--cross-vali
                 [-f FOLDS] [-r RUNS] [-p PERCENTAGES [PERCENTAGES
                 [--no-plot] [--successes] [--no-errors] [--no-warn
                 [--out OUT]
                 {core,nlu} ...

positional arguments:
  {core,nlu}
    core                Tests Rasa Core models using your test stor
    nlu                 Tests Rasa NLU models using your test NLU d

options:
  -h, --help            show this help message and exit
  -m MODEL, --model MODEL
                        Path to a trained Rasa model. If a director
                        specified, it will use the latest model in
                        directory. (default: models)
  --no-plot             Don't render evaluation plots. (default: Fa
  --successes           If set successful predictions will be writt
                        file. (default: False)
  --no-errors           If set incorrect predictions will NOT be wr
                        file. (default: False)
  --no-warnings         If set prediction warnings will NOT be writ
                        file. (default: False)
  --out OUT             Output path for any files created during th
                        evaluation. (default: results)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No

Core Test Arguments:
  -s STORIES, --stories STORIES
                        File or folder containing your test stories
                        .)
  --max-stories MAX_STORIES
                        Maximum number of stories to test on. (defa
  --endpoints ENDPOINTS
                        Configuration file for the connectors as a
                        (default: endpoints.yml)
  --fail-on-prediction-errors
                        If a prediction error is encountered, an ex
                        thrown. This can be used to validate storie
                        tests, e.g. on travis. (default: False)
```

```
  --url URL             If supplied, downloads a story file from a
                        trains on it. Fetches the data by sending a
                        request to the supplied URL. (default: None
  --evaluate-model-directory
                        Should be set to evaluate models trained vi
                        train core --config <config-1> <config-2>'.
                        in the provided directory are evaluated and
                        against each other. (default: False)

NLU Test Arguments:
  -u NLU, --nlu NLU     File or folder containing your NLU data. (d
                        data)
  -c CONFIG [CONFIG ...], --config CONFIG [CONFIG ...]
                        Model configuration file. If a single file
                        and cross validation mode is chosen, cross-
                        is performed, if multiple configs or a fold
                        configs are passed, models will be trained
                        compared directly. (default: None)
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
                        specifications in it. The content of these
                        be read and merged together. (default: doma
```

---

rasa test e2e

Rasa Pro Only

> **RASA PRO LICENSE**
>
> You'll need a license to get started with Rasa Pro. **Talk with Sales**

> **NEW IN 3.5**
>
> You can now use end-to-end testing to test your assistant as a whole, including dialogue management and custom actions.

To run **end-to-end testing** on your trained model, run:

```
rasa test e2e
```

This will test your latest trained model on any end-to-end test cases you have. If you want to use a different model, you can specify it using the `--model` flag.

The following arguments are available for `rasa test e2e`:

```
usage: rasa test e2e [-h] [-v] [-vv] [--quiet] [--logging-config-f

Runs end-to-end testing.

optional arguments:
  -h, --help            show this help message and exit
  -o, --e2e-results     Results file containing end-to-end testing
  --remote-storage REMOTE_STORAGE
                        Set the remote location where your Rasa mo
  -m MODEL, --model MODEL
                        Path to a trained Rasa model. If a directo
  --endpoints ENDPOINTS
                        Configuration file for the model server an

Python Logging Options:
  You can control level of log messages printed. In addition to the

  -v, --verbose         Be verbose. Sets logging level to INFO. (d
  -vv, --debug          Print lots of debugging statements. Sets l
  --quiet               Be quiet! Sets logging level to WARNING. (
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurat

Testing Settings:
  path-to-test-cases    Input file or folder containing end-to-end
  --fail-fast           Fail the test suite as soon as a unit test
```

---

rasa data split

To create a train-test split of your NLU training data, run:

```
rasa data split nlu
```

This will create a 80/20 split of train/test data by default. You can specify the training data, the fraction, and the output directory using the following arguments:

```
usage: rasa data split nlu [-h] [-v] [-vv] [--quiet]
                           [--logging-config-file LOGGING_CONFIG_F
                           [-u NLU] [--training-fraction TRAINING_
                           [--random-seed RANDOM_SEED] [--out OUT]

options:
  -h, --help            show this help message and exit
  -u NLU, --nlu NLU     File or folder containing your NLU data. (
                        data)
  --training-fraction TRAINING_FRACTION
                        Percentage of the data which should be in t
                        data. (default: 0.8)
  --random-seed RANDOM_SEED
                        Seed to generate the same train/test split
```

```
                        None)
  --out OUT             Directory where the split files should be s
                        (default: train_test_split)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

If you have NLG data for retrieval actions, this will be saved to separate files:

```
ls train_test_split

      nlg_test_data.yml     test_data.yml
      nlg_training_data.yml training_data.yml
```

To split your stories, you can use the following command:

```
rasa data split stories
```

It has the same arguments as `split nlu` command, but loads yaml files with stories and perform random splitting. Directory `train_test_split` will contain all yaml files processed with prefixes `train_` or `test_` containing train and test parts.

---

rasa data convert nlu

You can convert NLU data from

- LUIS data format,
- WIT data format,
- Dialogflow data format, or
- JSON

to

- YAML or
- JSON

You can start the converter by running:

```
rasa data convert nlu
```

You can specify the input file or directory, output file or directory, and the output format with the following arguments:

```
usage: rasa data convert nlu [-h] [-v] [-vv] [--quiet]
                             [--logging-config-file LOGGING_CONFIG_
                             [-f {json,yaml}] [--data DATA [DATA ..
                             [--out OUT] [-l LANGUAGE]

options:
  -h, --help            show this help message and exit
  -f {json,yaml}, --format {json,yaml}
                        Output format the training data should be c
                        into. (default: yaml)
  --data DATA [DATA ...]
                        Paths to the files or directories containir
                        data. (default: data)
  --out OUT             File (for `json`) or existing path (for `ya
                        to save training data in Rasa format. (defa
                        converted_data)
  -l LANGUAGE, --language LANGUAGE
                        Language of data. (default: en)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

rasa data migrate

The domain is the only data file whose format changed between 2.0 and 3.0. You can automatically migrate a 2.0 domain to the 3.0 format.

You can start the migration by running:

```
rasa data migrate
```

You can specify the input file or directory and the output file or directory with the following arguments:

```
rasa data migrate -d DOMAIN --out OUT_PATH
```

If no arguments are specified, the default domain path (`domain.yml`) will be used for both input and output files.

This command will also back-up your 2.0 domain file(s) into a different `original_domain.yml` file or directory labeled `original_domain` .

Note that the slots in the migrated domain will contain **mapping conditions** if these slots are part of a form's `required_slots` .

> **CAUTION**
>
> Exceptions will be raised and the migration process terminated if invalid domain files are provided or if they are already in the 3.0 format, if slots or forms are missing from your original files or if the slots or forms sections are spread across multiple domain files. This is done to avoid duplication of migrated sections in your domain files. Please make sure all your slots' or forms' definitions are grouped into a single file.

You can learn more about this command by running:

```
rasa data migrate --help
```

---

rasa data validate

You can check your domain, NLU data, or story data for mistakes and inconsistencies. To validate your data, run this command:

```
rasa data validate
```

The validator searches for errors in the data, e.g. two intents that have some identical training examples. The validator also checks if you have any stories

where different assistant actions follow from the same dialogue history. Conflicts between stories will prevent a model from learning the correct pattern for a dialogue.

> **SEARCHING FOR THE `ASSISTANT_ID` KEY INTRODUCED IN 3.5**
>
> The validator will check whether the `assistant_id` key is present in the config file and will issue a warning if this key is missing or if the default value has not been changed.

If you pass a `max_history` value to one or more policies in your `config.yml` file, provide the smallest of those values in the validator command using the `--max-history <max_history>` flag.

You can also validate only the story structure by running this command:

```
rasa data validate stories
```

> **NOTE**
>
> Running `rasa data validate` does **not** test if your **rules** are consistent with your stories. However, during training, the `RulePolicy` checks for conflicts between rules and stories. Any such conflict will abort training.
>
> Also, if you use end-to-end stories, then this might not capture all conflicts. Specifically, if two user inputs result in different tokens yet exactly the same featurization, then conflicting actions after these inputs may exist but will not be reported by the tool.

To interrupt validation even for minor issues such as unused intents or responses, use the `--fail-on-warnings` flag.

> **CHECK YOUR STORY NAMES**
>
> The `rasa data validate stories` command assumes that all your story names are unique!

You can use `rasa data validate` with additional arguments, e.g. to specify the location of your data and domain files:

```
usage: rasa data validate [-h] [-v] [-vv] [--quiet]
                          [--logging-config-file LOGGING_CONFIG_FIL
                          [--max-history MAX_HISTORY] [-c CONFIG]
                          [--fail-on-warnings] [-d DOMAIN]
                          [--data DATA [DATA ...]]
                          {stories} ...

positional arguments:
  {stories}
    stories             Checks for inconsistencies in the story fil

options:
  -h, --help            show this help message and exit
  --max-history MAX_HISTORY
                        Number of turns taken into account for stor
                        validation. (default: None)
  -c CONFIG, --config CONFIG
                        The policy and NLU pipeline configuration o
                        (default: config.yml)
  --fail-on-warnings    Fail validation on warnings and errors. If
                        only errors will result in a non zero exit
                        (default: False)
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
                        specifications in it. The content of these
                        be read and merged together. (default: doma
  --data DATA [DATA ...]
                        Paths to the files or directories containir
                        data. (default: data)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

rasa export

To export events from a tracker store using an event broker, run:

```
rasa export
```

You can specify the location of the environments file, the minimum and maximum timestamps of events that should be published, as well as the conversation IDs that should be published:

```
usage: rasa export [-h] [-v] [-vv] [--quiet]
                   [--logging-config-file LOGGING_CONFIG_FILE]
                   [--endpoints ENDPOINTS]
                   [--minimum-timestamp MINIMUM_TIMESTAMP]
                   [--maximum-timestamp MAXIMUM_TIMESTAMP]
                   [--offset-timestamps-by-seconds OFFSET_TIMESTAMI
                   [--conversation-ids CONVERSATION_IDS]

options:
  -h, --help            show this help message and exit
  --endpoints ENDPOINTS
                        Endpoint configuration file specifying the
                        store and event broker. (default: endpoints
  --minimum-timestamp MINIMUM_TIMESTAMP
                        Minimum timestamp of events to be exported.
                        constraint is applied in a 'greater than or
                        comparison. (default: None)
  --maximum-timestamp MAXIMUM_TIMESTAMP
                        Maximum timestamp of events to be exported.
                        constraint is applied in a 'less than' comp
                        (default: None)
  --offset-timestamps-by-seconds OFFSET_TIMESTAMPS_BY_SECONDS
                        Offset all event timestamps by the specifie
                        seconds. This won't modify the stored event
                        tracker store, but only change the timestar
                        events exported to the broker. (default: No
  --conversation-ids CONVERSATION_IDS
                        Comma-separated list of conversation IDs to
                        If unset, all available conversation IDs wi
                        exported. (default: None)

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
                        will be set to the given name. (default: No
```

IMPORT CONVERSATIONS INTO RASA X/ENTERPRISE

This command is most commonly used to import old conversations into Rasa X/Enterprise to annotate them. Read more about importing conversations into Rasa X/Enterprise.

rasa evaluate markers

> **⚠ CAUTION**
>
> This feature is currently experimental and might change or be removed in the future. Share your feedback in the forum to help us make it production-ready.

The following command applies the **markers** you defined in your marker configuration file, to pre-existing dialogues stored in your **tracker store**, and produces `.csv` files containing the extracted markers and summary statistics:

```
rasa evaluate markers all extracted_markers.csv
```

Use the following arguments to configure the marker extraction process:

```
usage: rasa evaluate markers [-h] [-v] [-vv] [--quiet] [--config C(

positional arguments:
  output_filename       The filename to write the extracted markers
  {first_n,sample,all}
    first_n             Select trackers sequentially until N are ta
    sample              Select trackers by sampling N.
    all                 Select all trackers.

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       The config file(s) containing marker defini
                        merged together. (default: markers.yml)
  --no-stats            Do not compute summary statistics. (default
  --stats-file-prefix [STATS_FILE_PREFIX]
                        The common file prefix of the files where v
                        `-per-session.csv` will be added automatica
  --endpoints ENDPOINTS
                        Configuration file for the tracker store as
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        domain.yml)

Python Logging Options:
  -v, --verbose         Be verbose. Sets logging level to INFO. (de
  -vv, --debug          Print lots of debugging statements. Sets lo
  --quiet               Be quiet! Sets logging level to WARNING. (d
```

rasa markers upload

Rasa Pro Only

> **RASA PRO LICENSE**
>
> You'll need a license to get started with Rasa Pro. **Talk with Sales**

> **NEW IN 3.6**
>
> This command is available from Rasa Pro 3.6.0 and requires **Rasa Analytics Data Pipeline**

This command applies to **markers** and their **real-time processing**. Running this command validates the marker configuration file against the domain file and uploads the configuration to Analytics Data Pipeline

```
usage: rasa markers upload [-h] [-v] [-vv] [--quiet]
                           [--logging-config-file LOGGING_CONFIG_FI
                           [--config CONFIG]
                           [--rasa-pro-services-url RASA_PRO_SERVIC
                           [-d DOMAIN]

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG       The marker configuration file(s) containing
                        definitions. This can be a single YAML file
                        directory that contains several files with
                        definitions in it. The content of these fil
                        read and merged together. (default: markers
  --rasa-pro-services-url RASA_PRO_SERVICES_URL
                        The URL of the Rasa Pro Services instance t
                        markers to.Specified URL should not contain
                        slash. (default: )
  -d DOMAIN, --domain DOMAIN
                        Domain specification. This can be a single
                        or a directory that contains several files
                        specifications in it. The content of these
                        be read and merged together. (default: doma

Python Logging Options:
  You can control level of log messages printed. In addition to the
  arguments, a more fine grained configuration can be achieved with
  environment variables. See online documentation for more info.

  -v, --verbose         Be verbose. Sets logging level to INFO. (de
                        None)
  -vv, --debug          Print lots of debugging statements. Sets lo
                        to DEBUG. (default: None)
  --quiet               Be quiet! Sets logging level to WARNING. (d
                        None)
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurati
```

```
                        will be set to the given name. (default: N

Description:
  The `rasa markers upload` command allows you to upload markers t

Examples:
  Upload Markers to Rasa Pro Services:
    rasa markers upload --config markers.yml --rasa-pro-services-u
```

rasa license

Rasa Pro Only

> **RASA PRO LICENSE**
>
> You'll need a license to get started with Rasa Pro. **Talk with Sales**

> **NEW IN 3.3**
>
> This command was introduced.

Use `rasa license` to display information about licensing in Rasa Pro, especially information about 3rd party dependencies licenses.

Here is the list of all possible arguments:

```
usage: rasa license [-h] [-v] [-vv] [--quiet] [--logging-config-fi

Display licensing information.

options:
  -h, --help            show this help message and exit

Python Logging Options:
  You can control level of log messages printed. In addition to th

  -v, --verbose         Be verbose. Sets logging level to INFO. (d
  -vv, --debug          Print lots of debugging statements. Sets l
  --quiet               Be quiet! Sets logging level to WARNING. (
  --logging-config-file LOGGING_CONFIG_FILE
                        If set, the name of the logging configurat
```

Previous Next

« Migrate From Other Tools (beta)

Conversation-Driven Development »

*Last updated on **2/23/2024** by **GitHub CI***

Edit this page