WorkFlow Training II
Dependencies and WorkFlow Jobs

Yilian Guo
yguo@cmcrc.com

What are dependencies?

- **Flow of data processing**

Raw data (AWS S3)
→ conversion (AWS S3)
→ Metric generation (AWS S3)
→ Sync to local database

http://docs.aws.cmcrc.com/cmcrc-docs/_images/mqdetlstages.svg

Dependency -- the logic behind data processing in workflow

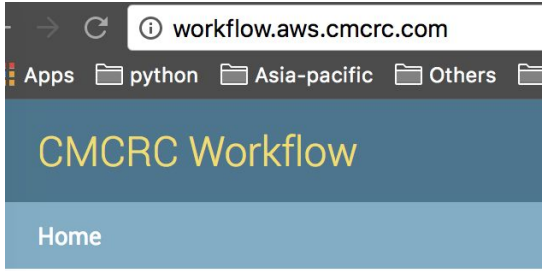
- Set up fixed dependencies in WorkFlow based on logical orders to ensure data processing follow certain steps
- To calculate metrics, we need to manipulate the TAQ (time and sales) data we downloaded from TRTH.
- Some metrics use output from other metrics (e.g. daily_stats use results from daily_trade_summary_generate)

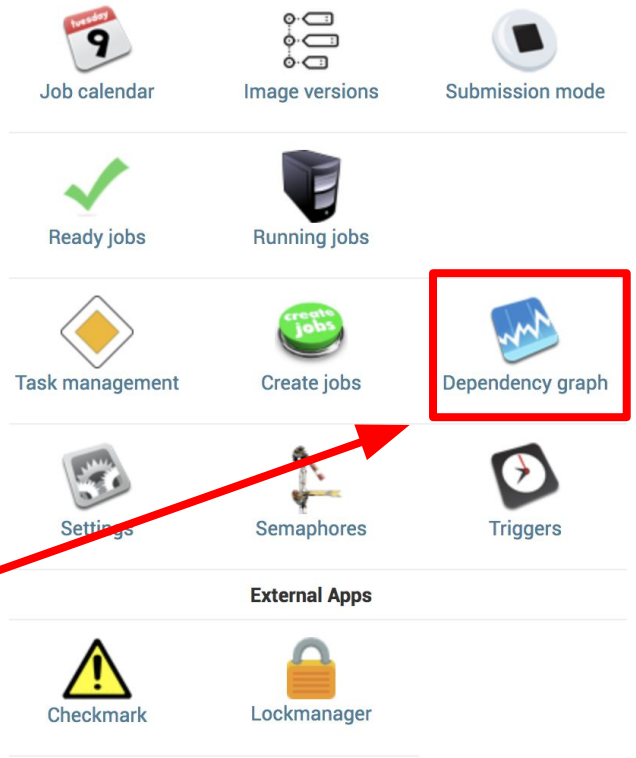
Dependency setup in WorkFlow

- These configure the sequencing between jobs of different tasks on the same date.
- For each dependency, the “parent” job must be SUCCESS (or PRETEND) before the “child” job can run.

Where to create dependencies?

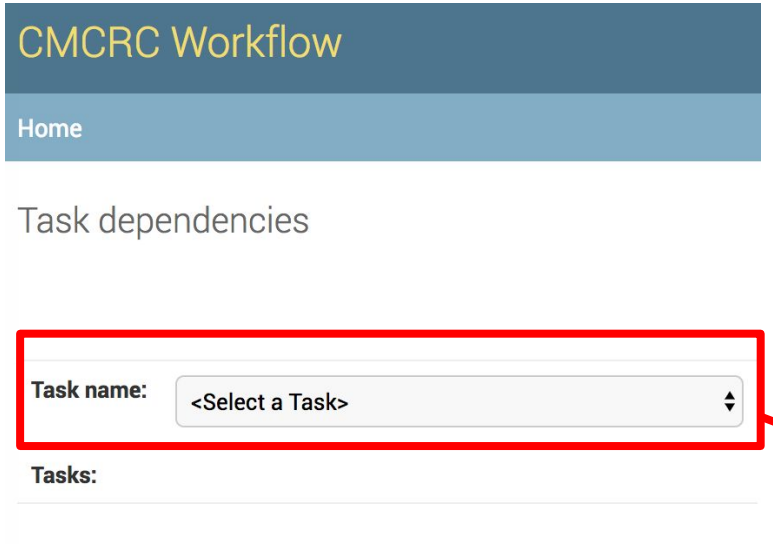
http://workflow.aws.cmcrc.com/





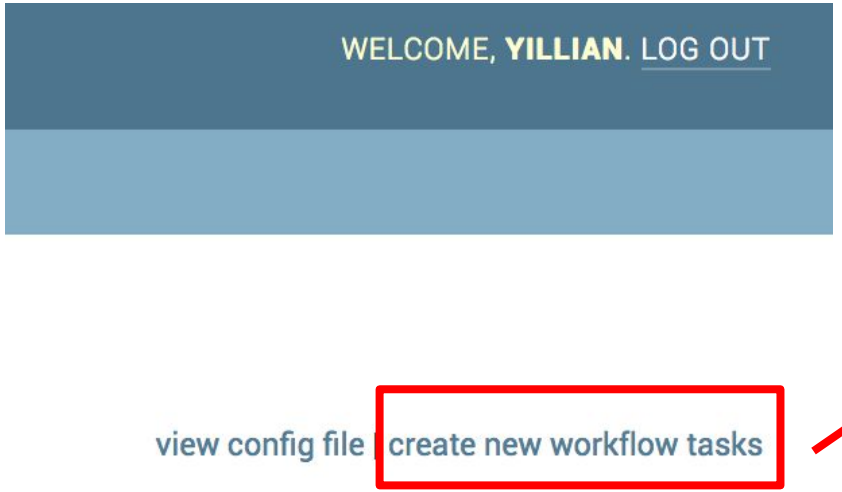
This is where you can see all the existing dependencies in the WorkFlow and create new ones

Where to create dependencies?



This is where you can choose a task and see the existing dependency graphs

Where to create dependencies?



This is where you can create new dependencies

Where to create dependencies?

More details

- Dependencies can be both vertical and horizontal
- Vertical dependency refers to the output of one task can be used as the input of another task
- Horizontal dependency refers to the results from previous trading days

Dependencies:

grandparents		parents		
trth_download/**ASX** trth_download_endofday/**ASX** trth_exchangebyday_download/**exchbyday** dss_venuebyday_download/**venuebyday**	→	convert_taq/**asx_taq**	→	*daily_trade_summary_generate/asx^asx*

Overtaking

This configures the sequencing between jobs in the same task. There are two settings:

- ANY overtaking

  In this mode, jobs can be run in any order; there is no sequencing requirement for the jobs.

- READY overtaking

  In this mode, a job cannot progress to READY until all earlier jobs (that is, jobs for earlier dates) in the same task have also progressed to READY (or beyond READY to RUNNING, SUCCESS/PRETEND, SLEEPING or FAIL).

TRTH data

- TRTH publishes data in one of two ways; (a) in the “exchange by day” mode, or (b) via the REST API.
- Data from API requests will be be retained for up to 45 days, and data from exchange-by-day will typically expire within two weeks.

ETL stage I - Download

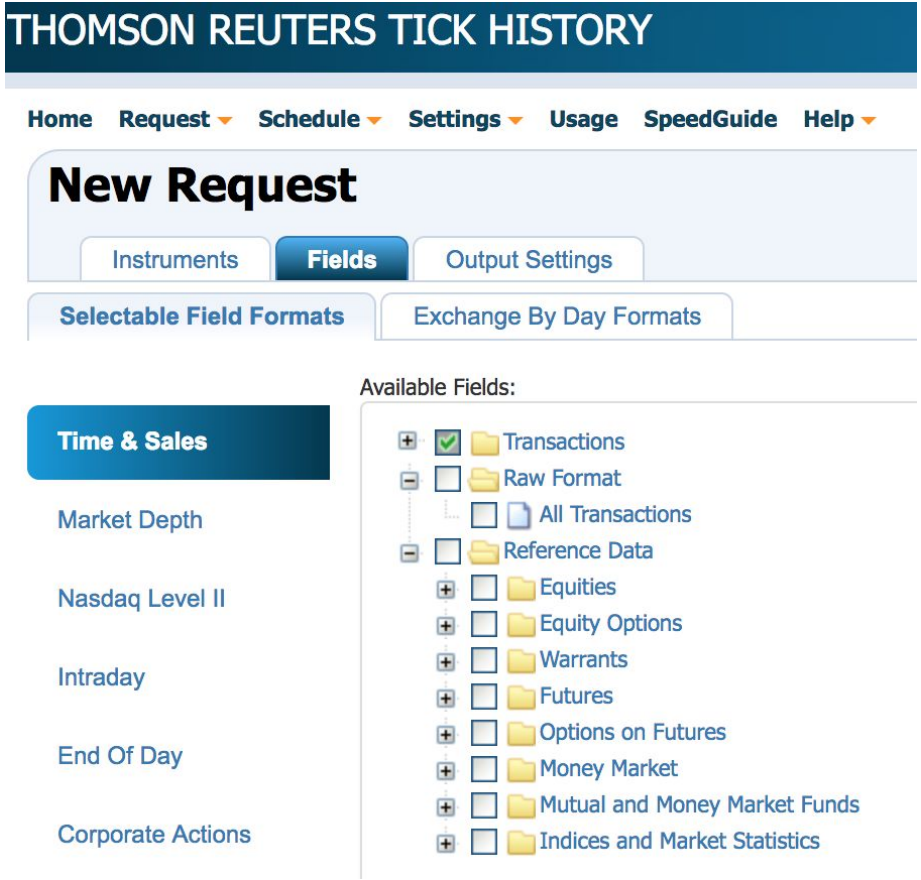
Exchange by day downloading

- For 110+ markets TRTH provides daily dumps of data, which can be accessed through the REST API These need to be requested following a three-stage process:
    1. List the venue files within a date range
    2. Download the files we need
    3. Repeat to collect any updated files that were added during the process
- Venue-by-day downloads are done by the workflow task `trth_exchangebyday_download/exchbyday`
- Since the data is only retained for 2 weeks, this is only suitable for ongoing downloads, not for historical markets. It also does not cover all markets or all data types

ETL stage I - Download

Types of data

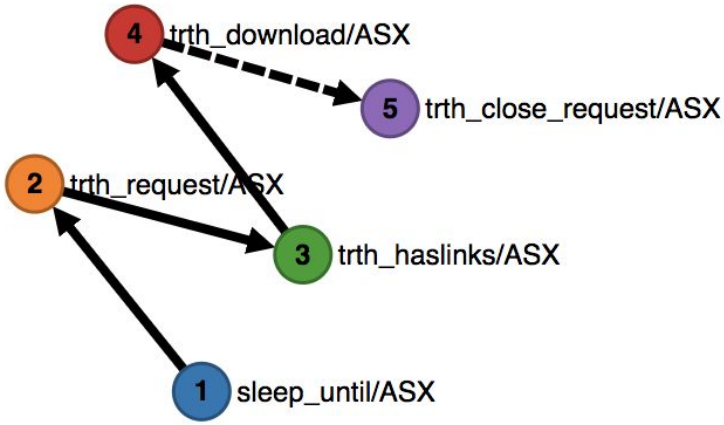
- Time and sales (TAQ -- Trades and quotes)
- End of day (reference and transactions)
- Index
- Corpactions (dividend, shares)
- Depth (10 level)
- Info (global company news)



ETL stage I - Download

Data organisation

- Raw data is organised by stream (previously “feed”), as supplied by TRTH. These may or may not directly correspond to useful concepts of “market”, so the converter needs to rearrange data into “tracks” (trading market + data type).



ETL stage I - Download

API downloading

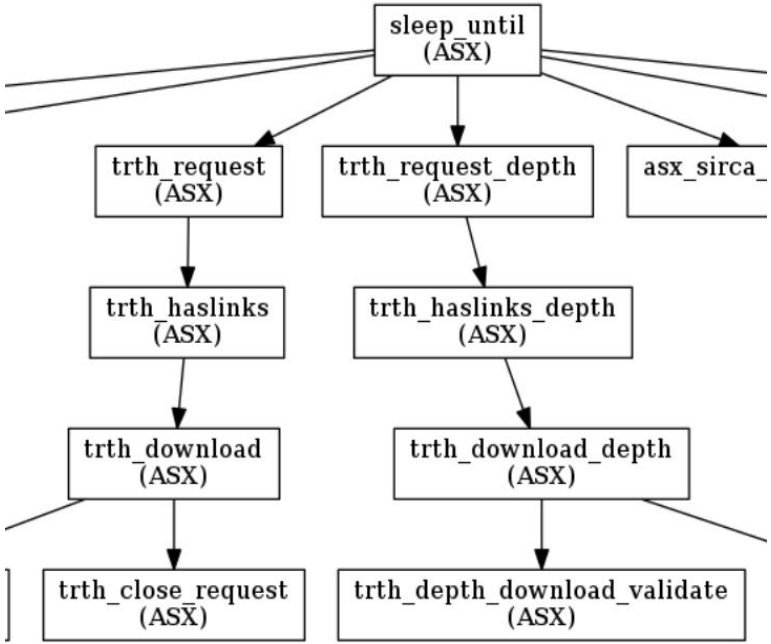
- 4 stages involved

  -- request

  -- haslink

  -- download

  -- close_request



http://workflow.aws.cmcrc.com/graph

Request

- WorkFlow submits requests to TRTH using the credentials provided by TRTH for the cmcrc accounts and several student accounts.
- Successful requests will produce a "job ID" which is added to the output of the request job and read by the following workflow jobs.

Data Type	Task Type in Workflow
TAS	trth_request trth_request_by_chains trth_request_by_isin
Endofday	trth_request_endofday trth_request_endofday_by_chains
EndofDay Transactions	trth_request_endofday_transactions trth_request_endofday_transactions_by_ric_list
Depth	trth_request_depth trth_request_depth_by_isin
Index	trth_request_index
Corpactions	trth_request_corpactions

Haslink

- Haslinks checks for the availability of the data for the submitted requests. Requests go into a queue and may take some time to be processed by TRTH, and large requests may take longer to process.

Data Type	Task Type in Workflow
TAS	trth_haslinks
Endofday	trth_haslinks_endofday
EndofDay Transactions	trth_haslinks_endofday_transactions
Depth	trth_haslinks_depth
Index	trth_haslinks_index
Corpactions	trth_haslinks_corpactions

Download

- Once the haslinks job indicates that the data is ready to download, workflow downloads the data files and stores them in AWS S3. The report files are already generated by the previous haslinks job.

Data Type	Task Type in Workflow
TAS	trth_download
Endofday	trth_download_endofday
EndofDay Transactions	trth_download_endofday_transactions
Depth	trth_download_depth
Index	trth_download_index
Corpactions	trth_download_corpactions

ETL stage II - Convert and data QA

- Validate report RICs

  This is literally a test run of the converter with one invented trade per security, to make sure all the configuration works before running the converter. This saves both our time and processor time, as conversion can take quite a while. (`validate_report_rics/nyse_taq`)

- The converter

  It takes the raw data, as downloaded from TRTH and stored in S3 in stage I, and converts it to Blueshift file format, also stored in S3, ready for stage III (metric generation).

ETL stage II - Convert and data QA

- Track names consist of the “uptick name” of the trading market and a suffix indicating the data type; for instance, `asx_taq` is the “trades and quotes” data for the ASX.
- Task name varies based on data type

-- convert_taq/asx_taq (TAQ)

-- refdata_eod_convert/ASX, import_reference_data/ASX (EOD reference)

-- convert_index/asx_index (index)

-- convert_info/asx_info (info)

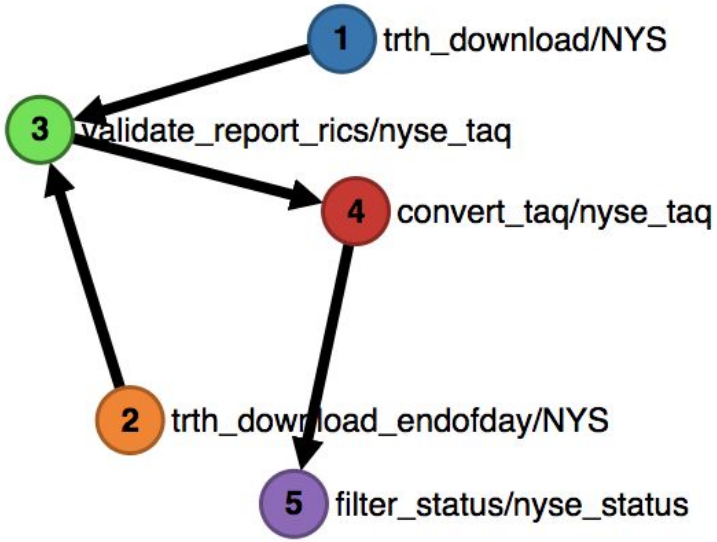
-- refdata_corpactions_convert/ASX (corpactions)

-- convert_endofday_transactions/asx_taq (EODTX)

ETL stage II - Convert and data QA

Filter_status (asx_status)

- filter_status tasks produce a smaller feed for the metrics that don't need trades and quotes (typically because they're only based on the results of another metric). This should make these metrics run much faster. The filter_status tasks depend on the corresponding convert_taq.



ETL stage III - Metrics generate

- The third stage of our ETL process is metric generation. The final product of this stage is metric results in CSV in S3.
- Most metrics are generated for per listing market^trading market.

(e.g. daily_stats_generate/lse^lse,

daily_stats_generate/lse^chi_x,

daily_stats_generate/xetra^chi_x)

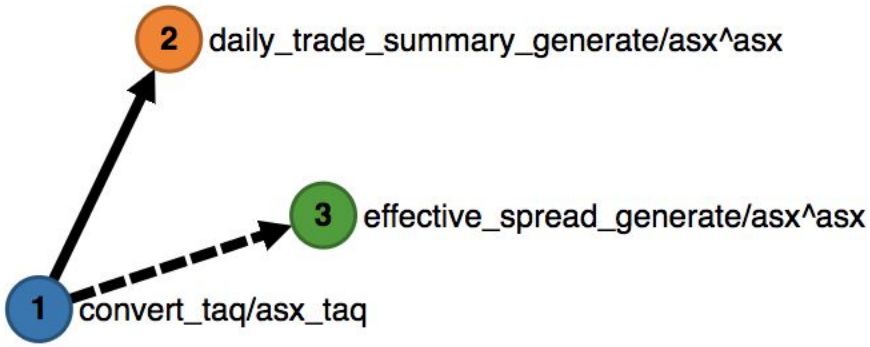
- We also have national metrics, which read convert data from multiple trading markets belong to one listing market.

(e.g. nbbo_generate/lse^, nbbo_generate/xetra^)

ETL stage III - Metrics generate

Input sources:

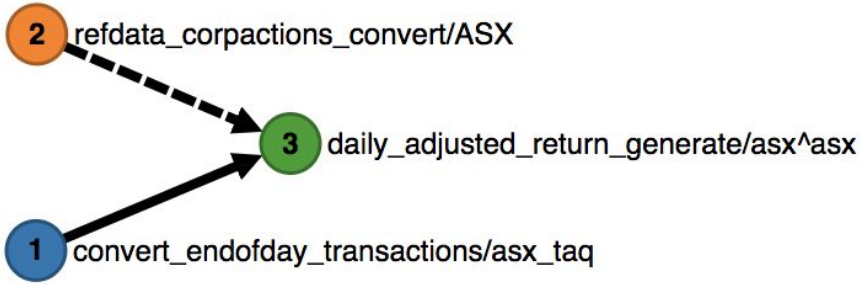
- The converted data in Blueshift. This is the “main” data source and triggers the callbacks in the metric.



ETL stage III - Metrics generate

Input sources:

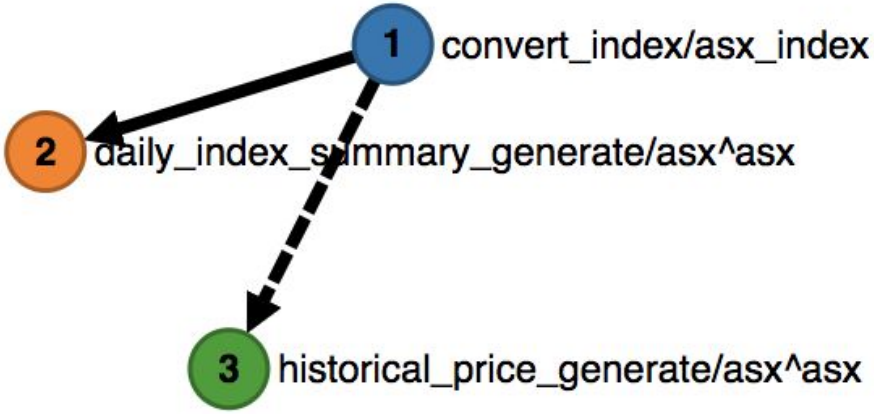
- Refdata. For example, metrics dealing with derivatives will often access refdata in order to determine the underlying security.



ETL stage III - Metrics generate

Input sources:

- Market configuration, for instance to find the main index of the market (such as the [All Ordinaries](#) index on the ASX).



ETL stage III - Metrics generate

Metrics can process data from the following sources (cont.):

The results of other metrics, on the same day or previous days.

- Benchmark

  These metrics are implemented in two stages (technically, as two metrics). In the first stage, they generate a measure for each day separately. In the second stage, which is a separate metric (and therefore a separate task in workflow), they compare the value for today with values for previous days and output a measure or alert based on the degree to which this day is unusual.

E.g

http://workflow.aws.cmcrc.com/job/28262829

ETL stage III - Metrics generate

http://workflow.aws.cmcrc.com/job/28262829

eod_price_dislocation_generate/asx^asx 2018-01-04 job 28262829

Task: eod_price_dislocation_generate/asx^asx

Date: 2018-01-04 (Thursday)

Status: success

restart job

Parents: S eod_window_trade_value_generate/asx^asx , S historical_true_price_generate/asx^asx , S daily_trade_summary_generate/asx^asx , S historical_price_generate/asx^asx , S daily_info_generate/asx^asx

Children: S eod_price_dislocation_sync/asx^asx

ETL stage III - Metrics generate

Metrics can process data from the following sources (cont.):

The results of other metrics, on the same day or previous days.

- Historical data
  Some metrics need access to the previous days' data in order to compute statistics across days; for these, we have "historical price" pseudo-metrics which output prices and other summary statistics on a per-minute basis. The metric itself can then access these prices.

ETL stage III - Metrics generate

http://workflow.aws.cmcrc.com/job/28262829

```
@requires_metric(name='daily_trade_summary', previous_trading_days=30)
@requires_metric(name='historical_price', previous_trading_days=1)
@requires_metric(name='historical_true_price', previous_trading_days=30)
@requires_metric(name='eod_window_trade_value', previous_trading_days=30)
@requires_metric(name='daily_info', previous_trading_days=2)
class EodPriceDislocation(TSSingleMarketMixin, BaseMetric):
```

```python
class AmihudRatio(MaterialisedView, models.Model):
        market = models.ForeignKey(Market)
        date = models.DateField()
        tradable = models.ForeignKey(Tradable, db_constraint=False)
        security = models.CharField(max_length=255)
        daily_trade_value_in_usd = models.DecimalField(max_digits=30, decimal_places=2, null=True, blank=True)
        volume = models.BigIntegerField(null=True, blank=True)
        abs_daily_return_perc = models.DecimalField(max_digits=30, decimal_places=4, null=True, blank=True)
        amihud_ratio = models.FloatField(null=True)
        amivest_measure = models.FloatField(null=True)

        insert_sql = """
            insert into {table_name} (market_id, date, tradable_id, security, daily_trade_value_in_usd,volume,
            abs_daily_return_perc,amihud_ratio, amivest_measure)   (
            select
                ds.market_id,
                ds.date,
                ds.tradable_id,
                ds.security,
                ds.value * ex.usd_per_unit as daily_trade_value_in_usd,
                ds.volume,
                abs(ds.daily_return_perc) as abs_daily_return_perc,
                (abs(ds.daily_return_perc)) / nullif(ds.value * ex.usd_per_unit, 0) as amihud_ratio,
                (ds.volume) / nullif(abs(ds.daily_return_perc),0) as amivest_measure
            from api_dailystats ds
            join api_market m on m.id = ds.market_id
            join api_exchangerate ex on ex.date = ds.date and ex.currency_id = m.trading_currency_id
            join api_tradable t on(t.id = ds.tradable_id)
            join api_listing l on(l.id = t.listing_id)
            where {condition(ds)}
            );
        """
```

ETL stage III - Metrics generate

Metrics can process data from the following sources (cont.):

The results of other metrics, on the same day or previous days.

- Previous result
  Some metrics do not use the converted data at all; they only read the results of a previous day. For instance, there is a metric which calculates the intra-day return based on the close price of today and yesterday; these close prices are retrieved from the result of the “daily trade summary” metric.

ETL stage III - Metrics generate

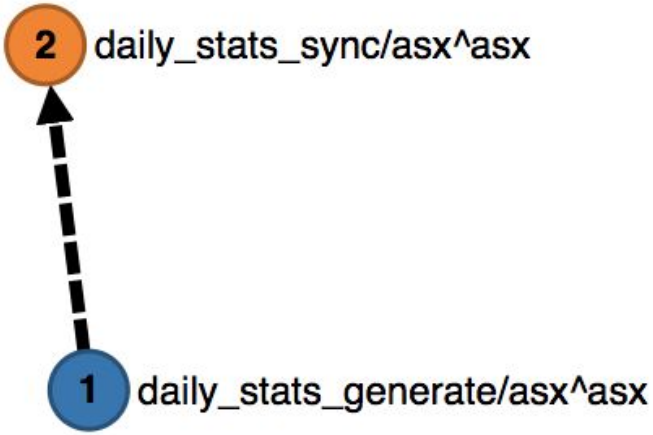
```
@requires_metric(name='daily_trade_summary', previous_trading_days=1)
class DailyStats(TQSingleMarketMixin, SSingleMarketMixin, BaseMetric):
```

ETL stage IV - Metric Sync

- The fourth stage of our ETL process is metric sync. The final product of this stage is instrument-level metric results in the MQDashboard QA database.
- Metrics in MQD can be viewed in on four major levels: instrument level, market (venue) level, group level and index level.
- In this stage, we update the instrument level only; the other levels are updated in ETL stage V - Matview refresh.

ETL stage IV - Metric Sync

- The task type will be the metric name followed by "_sync", e.g. daily_stats_sync. The task name will be the same as metric generate, e.g. daily_stats_sync/lse^lse, nbbo_sync/lse^



ETL stage V - Matview refresh

- The fifth stage of our ETL process is matview refresh. The final product of this stage is market- and index-level metric results in the MQDashboard QA database.
- The task type is **refresh_materialised_views**
- The task name is the trading market name, without any symbols (e.g. refresh_materialised_views/asx)
- The parent jobs are the metric sync jobs

ETL stage V - Matview refresh

refresh_materialised_views/asx 2018-01-04 job 28243695

Task: refresh_materialised_views/asx

Date: 2018-01-04 (Thursday)

Status: success

restart job

Parents: S autocorrelation_sync/asx^asx , S variance_ratio_sync/asx^asx , S daily_crash_risk_sync/asx^asx , S liquidity_group_dailystats_sync/asx^asx , S continuous_trading_manipulation_sync/asx^asx , S deciles_calculate/asx^asx , S effective_spread_sync/asx^asx , S common_liquidity_sync/asx^asx , S daily_stats_sync/asx^asx , S quote_to_trade_ratio_sync/asx^asx , S opening_price_dislocation_sync/asx^asx , S info_leakage_abnormal_volume_sync/asx^asx , S roll_spread_sync/asx^asx , S sqrt_trade_value_sync/asx^asx , S quoted_spread_sync/asx^asx , S quote_volatility_sync/asx^asx , S nbbo_sync/asx^ , S corwin_schultz_spread_sync/asx^asx , S nbbo_effective_spread_sync/asx^ , S nbbo_realised_spread_sync/asx^ , S daily_adjusted_return_sync/asx^asx , S update_index_constituents/all , S realised_spread_multi_interval_sync/asx^asx , S nbbo_implementation_shortfall_sync/asx^ , S implementation_shortfall_sync/asx^asx , S auction_trade_stats_sync/asx^asx , S info_leakage_sync/asx^asx , S eod_price_dislocation_sync/asx^asx , S intraday_sync/asx^asx , S interday_sync/asx^asx

Children: