Item 271
git_comments:

1. don't consider runs that are executed in the future unless specified by config and schedule_interval is None
2. This allows allow_trigger_in_future config to take affect, rather than mandating exec_date <= UTC
3. don't consider runs that are executed in the future unless specified by config and schedule_interval is None

git_commits:

1. **summary:** [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038)
   message: [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038)

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates
   body: Allow externally triggered dags to run for future exec dates --- Link to JIRA issue: https://issues.apache.org/jira/browse/AIRFLOW-4495 - [X] Description above provides context of the change - [X] Commit message starts with `[AIRFLOW-4495]`, where AIRFLOW-NNNN = JIRA ID* - [X] Unit tests coverage for changes (not needed for documentation changes) - [X] Commits follow "[How to write a good git commit message](http://chris.beams.io/posts/git-commit/)" - [X] Relevant documentation is updated including usage instructions. - [X] I will engage committers as explained in [Contribution Workflow Example](https://github.com/apache/airflow/blob/master/CONTRIBUTING.rst#contribution-workflow-example). (*) For document-only changes, no JIRA issue is needed. Commit message starts `[AIRFLOW-XXXX]`. --- In case of fundamental code change, Airflow Improvement Proposal ([AIP](https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+Improvements+Proposals)) is needed. In case of a new dependency, check compliance with the [ASF 3rd Party License Policy](https://www.apache.org/legal/resolved.html#category-x). In case of backwards incompatible changes please leave a note in [UPDATING.md](https://github.com/apache/airflow/blob/master/UPDATING.md). Read the [Pull Request Guidelines](https://github.com/apache/airflow/blob/master/CONTRIBUTING.rst#pull-request-guidelines) for more information.

github_pulls_comments:

1. # [Codecov](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=h1) Report > :exclamation: No coverage uploaded for pull request base (`master@b5bd9ab`). [Click here to learn what that means](https://docs.codecov.io/docs/error-reference#section-missing-base-commit). > The diff coverage is `97.77%`. [![Impacted file tree graph](https://codecov.io/gh/apache/airflow/pull/7038/graphs/tree.svg?width=650&token=WdLKlKHOAU&height=150&src=pr)](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=tree) ```diff @@ Coverage Diff @@ ## master #7038 +/- ## == Coverage ? 83.36% == Files ? 791 Lines ? 41468 Branches ? 0 == Hits ? 34568 Misses ? 6900 Partials ? 0 ``` | [Impacted Files](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=tree) | Coverage Δ | | |---|---|---| | [airflow/models/dag.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9tb2RlbHMvZGFnLnB5) | `90.9% <ø> (ø)` | | | [...irflow/contrib/operators/emr_add_steps_operator.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9jb250cmliL29wZXJhdG9ycy9lbXJfYWRkX3N0ZXBzX29wZXJhdG9yLnB5) | `100% <ø> (ø)` | | | [airflow/sensors/base_sensor_operator.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9zZW5zb3JzL2Jhc2Vfc2Vuc29yX29wZXJhdG9yLnB5) | `98.61% <ø> (ø)` | | | [airflow/sensors/external_task_sensor.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9zZW5zb3JzL2V4dGVybmFsX3Rhc2tfc2Vuc29yLnB5) | `85.52% <ø> (ø)` | | | [...ib/operators/azure_container_instances_operator.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9jb250cmliL29wZXJhdG9ycy9henVyZV9jb250YWluZXJfaW5zdGFuY2VzX29wZXJhdG9yLnB5) | `100% <ø> (ø)` | | | [airflow/models/baseoperator.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9tb2RlbHMvYmFzZW9wZXJhdG9yLnB5) | `96.28% <ø> (ø)` | | | [airflow/ti_deps/deps/prev_dagrun_dep.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy90aV9kZXBzL2RlcHMvcHJldl9kYWdydW5fZGVwLnB5) | `80.55% <ø>

(ø)` | | | [airflow/gcp/hooks/bigquery.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9nY3AvaG9va3MvYmlncXVlcnkucHk=) | `91.81% <0%> (ø)` | | | [airflow/cli/commands/celery\_command.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9jbGkvY29tbWFuZHMvY2VsZXJ5X2NvbW1hbmQucHk=) | `52.5% <0%> (ø)` | | | [airflow/jobs/scheduler\_job.py](https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree#diff-YWlyZmxvdy9qb2JzL3NjaGVkdWxlcl9qb2IucHk=) | `89.34% <100%> (ø)` | | | ... and [4 more] (https://codecov.io/gh/apache/airflow/pull/7038/diff?src=pr&el=tree-more) | | ------ [Continue to review full report at Codecov](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=continue). > **Legend** - [Click here to learn more](https://docs.codecov.io/docs/codecov-delta) > `Δ = absolute <relative> (impact)`, `ø = not affected`, `? = missing data` > Powered by [Codecov](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=footer). Last update [b5bd9ab...40d17ba](https://codecov.io/gh/apache/airflow/pull/7038?src=pr&el=lastupdated). Read the [comment docs](https://docs.codecov.io/docs/pull-request-comments).
2. @yuqian90 pls review
3. @ashb @kaxil @mik-laj @nuclearpinguin -> WDYT ?
4. I don't think we should allow this. This will create a problem for the following case: DAG is scheduled to run every day at 5 PM 1) Dag is triggered for a future date (ex: 10 Jan 2020 17:00:00) and succeeds 2) 10 Jan arrives, Scheduler tries to run it but fails because it violates unique key constraint
5. @kaxil but this change only applies to dag's with no schedule_interval. I can put another guard on apply if use_job_schedule =false. I solely use external triggers in my environment
6. > I don't think we should allow this. This will create a problem for the following case: > > DAG is scheduled to run every day at 5 PM > > 1. Dag is triggered for a future date (ex: 10 Jan 2020 17:00:00) and succeeds > 2. 10 Jan arrives, Scheduler tries to run it but fails because it violates unique key constraint Hi, @kaxil. Thanks for pointing out. I agree with @tooptoop4 that his change is only meant for DAGs with `schedule_interval` equal to None, i.e. those that are triggered externally using the CLI or other means. The ones scheduled by Airflow should not be affected.
7. Oh. Looks like some tests failed. Pls fix. @tooptoop4
8. passed
9. @kaxil pls merge
10. @potiuk @kaxil I want to bring it to the attention of committers. I have brought this up in an older PR previously, let me explain again why this PR is useful. In our use case of Airflow, we have to download some files with file name some_name_20190801.txt at 20190801 08:00 Tokyo time. So intuitively, we want to put this task on the DAG with execution_date 20190801. But at the moment, this is not possible with Airflow, even with externally triggered DAGs (i.e. `schedule_interval=None`). The reason is because Airflow scheduler does not consider tasks on the 20190801 DAG for execution until 20190801 00:00 UTC, which is 20190801 09:00 Tokyo time. So the earliest time we can run this task is at 09:00 Tokyo time. With this constraint in mind, we came up with our own temp workaround. What we did was to put the downloader for some_name_20190801.txt on the execution_date 20190731 DAG. This works, but it’s not great because in the Web UI the DAG execution_date is shown as 20190731. Whoever looking at the DAG always gets confused because they would think the DAG is downloading files for 20190731. With this PR by @tooptoop4, we will finally be able to put the downloader for some_name_20190801.txt on the 20190801 DAG. This greatly simplifies things. There are other examples why this is useful. What i brought up here is only a simple one for illustration. This PR should be pretty safe to merge because it only changes the behaviour if the DAG has `schedule_interval` set to `None` and with `run_future_exec_dates` set to True in `airflow.cfg`. And it only makes a difference if the user externally triggers the DAG before execution_date midnight in UTC timezone.
11. updates made
12. @kaxil pls merge
13. @yuqian90 do u think the test u suggested caused other test failures?
14. > @yuqian90 do u think the test u suggested caused other test failures? The test failures in Travis are in seemingly unrelated tests such as testviews.py. it's very unlikely to be caused by the new tests you added. How about trying to rerun or ask if anyone else see similar failure on slack?
15. > The test failures in Travis are in seemingly unrelated tests such as testviews.py. it's very unlikely to be caused by the new tests you added. How about trying to rerun or ask if anyone else see similar failure on slack? I restarted the tests to check.
16. @kaxil pls merge :)
17. Sorry for the comment spam but I thought it would be important for @yuqian90 We have the exact same problem you have, we solve it by setting the DAG with a start date of the timezones it should be scheduled in, e.g. Tokyo time. We then create a macro which is a modified version of this: https://stackoverflow.com/a/54922776/2958068 The macro above provides the execution date in the same time zone as the DAG. Further we modify it to provide the actual scheduled date e.g. if scheduled to run on 2019-01-01 08:00 that's the datetime it provides (which in the default Airlow variables is the 'next execution date').
18. > Sorry for the comment spam but I thought it would be important for @yuqian90 > > We have the exact same problem you have, we solve it by setting the DAG with a start date of the timezones it should be scheduled in, e.g. Tokyo time. We then create a macro which is a modified version of this:

https://stackoverflow.com/a/54922776/2958068 > > The macro above provides the execution date in the same time zone as the DAG. Further we modify it to provide the actual scheduled date e.g. if scheduled to run on 2019-01-01 08:00 that's the datetime it provides (which in the default Airlow variables is the 'next execution date'). @notatallshaw thanks for the comment. Did setting `start_date` to the timezone you want work? We definitely remember trying something like that. But the airflow scheduler still did not run tasks before UTC midnight. Looking at the original code before this PR, it's not hard to see why. No matter what timezone we set the `start_date` to, this line was always skipping tasks as long as the execution_date is greater than `timezone.utcnow()`. So if the timezone we need is in Asia, this does not work till UTC midnight. But this PR fixed / improved this. Now we no longer have to put tasks for date T on the DAG for execution_date T-1. ``` airflow/jobs/scheduler_job.py ... # don't consider runs that are executed in the future if run.execution_date > timezone.utcnow(): self.log.error("Execution date is in future: %s", run.execution_date) continue ``` ```

19. @yuqian90 "Did setting start_date to the timezone you want work?" Yes, for example if you let this code run for a week: import pendulum from datetime import datetime from airflow import DAG from airflow.operators.dummy_operator import DummyOperator START_DATE = datetime(2020, 1, 23, tzinfo=pendulum.timezone('Pacific/Auckland')) SCHEDULE = '00 03 * * *' dag = DAG('nz_schedule_test', schedule_interval=SCHEDULE, start_date=START_DATE, catchup=True) dummy_task = DummyOperator(dag=dag, task_id='nz_schedule_test',) It successfully runs at the correct 3am NZ time, which is way before midnight UTC. But you get weird execution dates and ds_dates because Airflow normalizes the execution date to UTC before rendering the template or providing the context. So where you would normally expect Airflow to provide 2020-01-23 as the ds date it provides the previous day 2020-01-22 (day not date, it literally provides yesterday not the previous schedule date because this is a timezone issue not a schedule issue) That's why we use the above mentioned macro system to provide dates in the correct timezones. With this we can happily use Airflow to schedule DAGs in whatever timezones is required. The only thing you still need to watch for are start dates and end dates and the execution date shown in the UI. Hope that helps! We've been running this successfully on 1.10.3 and 1.10.6.
20. Sorry for the comment spam on a merged PR, but a quick follow-up to @kaxil's comment above: > DAG is scheduled to run every day at 5 PM > 1. Dag is triggered for a future date (ex: 10 Jan 2020 17:00:00) and succeeds > 2. 10 Jan arrives, Scheduler tries to run it but fails because it violates unique key constraint What if this is the use case we _want_ to support? i.e.: DAG is scheduled to run every day at 5 PM, and let's say I did some upstream data changes. I'm impatient and want to run the DAG at 4pm for one particular day instead of 5pm. Ideally: - Trigger dag with a future date (ex: 3 Feb 2020 17:00:00) at 4pm via airflow CLI; succeeds - 5pm arrives, and the scheduler handles the unique key constraint gracefully and does not run the DAG @kaxil @yuqian90 et al, thoughts on the above? Said another way, the proposal is to extend `allow_trigger_in_future` to all DAGs, including ones with a `schedule_interval`, by handling the unique constraint gracefully.
21. @pbotros if we turn on allow_trigger_in_future to all DAGs (i.e. make it the default), and we trigger the DAG for a future date because we are inpatient, it probably makes more sense for the scheduler to raise an error when it tries to re-trigger at the original scheduled time? What you want to do sounds like a one-off scenario that can be dealt with via communication instead of letting the scheduler handle it silently? However, if you constantly want to trigger something earlier than it's original schedule, is there any change that can be made to the system? E..g why not consider changing to external trigger altogether so you can have full control over what time the DAG is triggered?
22. @yuqian90 I see your point - since our use case is a one-off, this extra communication of a failed pipeline could be okay. However, for others I wonder if this might not be a one-off case; it might be something a team does once a week, or once a month, or every time a particular ticket comes in, etc. The extra step of handling the failure at re-triggering time could be burdensome and yet another thing that goes into an oncall handbook or something that causes an oncall page. Just to clarify as well - the suggestion isn't to make `allow_trigger_in_future` be _enabled_ for all DAGs, the suggestion is to make `allow_trigger_in_future` *respected* by all DAGs. An alternative to changing the default behavior of the scheduler is adding another configuration option, something like `handle_duplicate_runs_gracefully`, that would allow for handling the re-triggering scenario for those that opt-in. My naive feeling is that this isn't a very intuitive option and would be difficult to explain, but open to alternatives as well. Thoughts?
23. > @yuqian90 I see your point - since our use case is a one-off, this extra communication of a failed pipeline could be okay. However, for others I wonder if this might not be a one-off case; it might be something a team does once a week, or once a month, or every time a particular ticket comes in, etc. The extra step of handling the failure at re-triggering time could be burdensome and yet another thing that goes into an oncall handbook or something that causes an oncall page. > > Just to clarify as well - the suggestion isn't to make `allow_trigger_in_future` be _enabled_ for all DAGs, the suggestion is to make `allow_trigger_in_future` _respected_ by all DAGs. > > An alternative to changing the default behavior of the scheduler is adding another configuration option, something like `handle_duplicate_runs_gracefully`, that would allow for handling the re-triggering scenario for those that opt-in. My naive feeling is that this isn't a very intuitive option and would be difficult to explain, but open to alternatives as well. > > Thoughts? You mentioned some DAGs that are usually scheduled daily at a specific time, but need to be triggered earlier than the scheduled time once every week regularly. For complicated triggering logic like this, did you consider making another DAG that triggers this DAG with `TriggerDagRunOperator` ?

Then the DAG itself can be set to `schedule_interval=None` and relies on another DAG to do the triggering. It'll also respect `allow_trigger_in_future` once `schedule_interval=None` is set. Having Airflow silently handling trigger dag with duplicated key sounds a bit dangerous to me. But maybe others have different opinions.

24. > You mentioned some DAGs that are usually scheduled daily at a specific time, but need to be triggered earlier than the scheduled time once every week regularly. For complicated triggering logic like this, did you consider making another DAG that triggers this DAG with TriggerDagRunOperator ? Then the DAG itself can be set to schedule_interval=None and relies on another DAG to do the triggering. It'll also respect allow_trigger_in_future once schedule_interval=None is set. I was not familiar with that; that would definitely suit the bill but does seem a little less nice than handling everything within a single DAG. Definitely something we can fallback to if these changes aren't liked. > Having Airflow silently handling trigger dag with duplicated key sounds a bit dangerous to me. But maybe others have different opinions. True - this is an existing codebase with a lot of existing users; a change like this could potentially break things. @kaxil || @nuclearpinguin || others: thoughts on this? TL;DR of the proposal is to (a) make this `allow_trigger_in_future` respected in all DAGs regardless of their `scheduled_interval`, and (b) gracefully handle the unique constraint violation when inserting a DAG run. Thanks!
25. Ping @potiuk || @mik-laj || others on thoughts?

github_pulls_reviews:

1. I think if we want to keep the previous behaviour (do we?) then we should add back the original behaviour when end_date is not specified and `run_future_exec_dates == False`. When `run_future_exec_dates == False`, it will behave slightly differently than before - before it was trying to find all task before utcnow, but in this case it will also find future runs (if there are any). Unless it is all handled by the filtering in runnable_exec_date_dep.py below that is.
2. fixed
3. This change to `get_task_instances()` seems to be newly included from the previously closed PR? This is going to change the TI returned to include those considered in the "future". Why is this change necessary? If it is indeed needed, you should probably consider doing the same as what you did in `scheduler_job.py`, i.e. only include "future" tasks if `RUN_FUTURE_EXEC_DATES` is True **and** `dag.schedule_interval` is None
4. Okay this does look necessary. And it's easy to fix. How about adding this property to `class DAG`? ```python class DAG: ... @property def allow_future_exec_dates(self): return conf.getboolean('scheduler', 'RUN_FUTURE_EXEC_DATES', fallback=False) and self.schedule_interval is None ``` This line can become: ```python tis = session.query(TaskInstance).filter(TaskInstance.dag_id == self.dag_id, TaskInstance.execution_date >= start_date, TaskInstance.task_id.in_([t.task_id for t in self.tasks])) if end_date or not self.allow_future_exec_dates: end_date = end_date or timezone.utcnow() tis = tis.filter(TaskInstance.execution_date <= end_date) ``` And you may want to use `dag.allow_future_exec_dates` in the two other places that you checked the same conditions too, i.e. scheduler_job.py and runnable_exec_date_dep.py
5. updated
6. This can now be: ```python if run.execution_date > timezone.utcnow() and not run.dag.allow_future_exec_dates: self.log.error("Execution date is in future: %s", run.execution_date) continue ```
7. This can be: ```python if not ti.task.dag.allow_future_exec_dates: ... ```
8. done
9. ```suggestion If you want to use 'external trigger' to run future-dated execution dates, set ``allow_trigger_in_future = True`` in ``scheduler`` section in ``airflow.cfg``. ```
10. PLEASE do not add todo comments in the code. Either find out why and fix it or document it, or don't add the comment. This doesn't help anyone.
11. ```suggestion - name: allow_trigger_in_future ```
12. ```suggestion allow_trigger_in_future = False ```
13. ```suggestion ```
14. :interrobang: This shouldn't need changing - your change is meant to only affect dags in a particular state. This dep should respect the same setting/flag. If it doesn't that is a bug.
15. Can you add more description here to tell users they changed it to False and the Dag had already run for future date, what would happen
16. Can you also add a test to check the new feature to test the behavior when config is enabled?
17. @tooptoop4 , in some sense what you discovered is a flaw in the test itself. Since the dag in the test is a Mock object, `bool(dag.allow_future_exec_dates)` is always True (that's how Mock objects behave). So you can easily fix the test, and also like @kaxil suggested, you should add a test for the scenarios where allow_future_exec_dates is True and False. Here's what I suggest. In the following snippet, `test_exec_date_after_end_date` is what the original test tries to test. You should also add `test_exec_date_after_end_date_allow_future_exec_dates` which sets `allow_future_exec_dates` to True to test the new scenario your PR creates. Note that difference in the comments and in the assertTrue vs assertFalse. ```python class TestRunnableExecDateDep(unittest.TestCase): def _get_task_instance(self, execution_date,

dag_end_date=None, task_end_date=None, allow_future_exec_dates=False): dag = Mock(end_date=dag_end_date) dag.allow_future_exec_dates = allow_future_exec_dates task = Mock(dag=dag, end_date=task_end_date) return TaskInstance(task=task, execution_date=execution_date) @freeze_time('2016-01-01') def test_exec_date_after_end_date(self): """ If the dag's execution date is in the future this dep should fail """ ti = self._get_task_instance(dag_end_date=datetime(2016, 1, 3), task_end_date=datetime(2016, 1, 3), execution_date=datetime(2016, 1, 2),) self.assertFalse(RunnableExecDateDep().is_met(ti=ti)) @freeze_time('2016-01-01') def test_exec_date_after_end_date_allow_future_exec_dates(self): """ If the dag's execution date is in the future and allow_future_exec_dates is True, this dep should pass """ ti = self._get_task_instance(dag_end_date=datetime(2016, 1, 3), task_end_date=datetime(2016, 1, 3), execution_date=datetime(2016, 1, 2), allow_future_exec_dates=True) self.assertTrue(RunnableExecDateDep().is_met(ti=ti)) ```
18. Just realized this line needs to be changed. Note that run.dag is not yet set at this point. You need to use the `dag` variable. Probably worth considering adding a test for `run.execution_date > timezone.utcnow()` in test_scheduler_job.py to detect this. ``` if run.execution_date > timezone.utcnow() and not dag.allow_future_exec_dates: ```
19. Let's include dag_end_date too in the test
20. Maybe you could use `parameterized` as both the tests share almost the same code. You could parameterize `schedule_interval` and `execution_date` if you would like. tbh same execution date should work too
21. Let's add a test for this piece of code.
22. I am ' end_date=datetime(2016, 11, 5),' line 47
23. ```suggestion Only has effect if schedule_interval is set to None in DAG ``` The default value for `schedule_interval` is actually `timedelta(days=1)` so I thought `is set to None` is a more accurate description of this requirement here.
24. Same here ```suggestion # Only has effect if schedule_interval is set to None in DAG ```
25. It's not clear to me what the convention is right now. But there are more code using lower case. So let's make this lower case. ```suggestion 'allow_trigger_in_future', ```
26. ```suggestion # This allows allow_trigger_in_future config to take affect, rather than mandating exec_date <= UTC ```
27. @tooptoop4 you may consider replacing both `test_exec_date_after_end_date_with_allow_config_but_sched ` and `test_exec_date_after_end_date_with_allow_config ` with something like this. It's just a free function, does not have to be part of `class TestRunnableExecDateDep`. ```python import pytest @freeze_time('2016-11-01') @pytest.mark.parametrize("allow_trigger_in_future,schedule_interval,execution_date,is_met", [('True', None, datetime(2016, 11, 3), True), ('True', "@daily", datetime(2016, 11, 3), False), ('False', None, datetime(2016, 11, 3), False), ('False', "@daily", datetime(2016, 11, 3), False), ('False', "@daily", datetime(2016, 11, 1), True), ('False', None, datetime(2016, 11, 1), True)]) def test_exec_date_dep(allow_trigger_in_future, schedule_interval, execution_date, is_met): """ If the dag's execution date is in the future this dep should fail """ with conf_vars({('scheduler', 'allow_trigger_in_future'): allow_trigger_in_future}): dag = DAG('test_localtaskjob_heartbeat', start_date=datetime(2015, 1, 1), end_date=datetime(2016, 11, 5), schedule_interval=schedule_interval) with dag: op1 = DummyOperator(task_id='op1') ti = TaskInstance(task=op1, execution_date=execution_date) assert RunnableExecDateDep().is_met(ti=ti) == is_met ```
28. @tooptoop4 , maybe add a test for this code to `test_taskinstance.py`, unless you find some better place for the test.
29. Good Point @yuqian90

jira_issues:

1. **summary:** allow externally triggered dags to run for future 'Execution date'
description: 1. useful to handle future date for externally triggered batch process where ingesting 'forecast' data where filename date is in the future 2. this error is just in the scheduler log and not propagated up, so the dag stays in 'running' state forever (or for 1 year waiting for the time to pass :)) ERROR - Execution date is in future: 2020-01-01 00:00:00+00:00 fix below works if u only have externally triggered DAGs: commenting below ti_deps\deps\runnable_exec_date_dep.py #if ti.execution_date > cur_date: # yield self._failing_status(# reason="Execution date \{0} is in the future (the current " # "date is \ {1}).".format(ti.execution_date.isoformat(), # cur_date.isoformat())) commenting below jobs.py # don't consider runs that are executed in the future #if run.execution_date > timezone.utcnow(): # self.log.error(# "Execution date is in future: %s", # run.execution_date #) # continue

jira_issues_comments:

1. tooptoop4 commented on pull request #5458: [AIRFLOW-4495] allow externally triggered dags to run for future 'Exe… URL: https://github.com/apache/airflow/pull/5458 …cution date' Make sure you have checked _all_ steps

below. ### Jira - [ ] My PR addresses the following [Airflow Jira] (https://issues.apache.org/jira/browse/AIRFLOW/) issues and references them in the PR title. For example, "\[AIRFLOW-XXX\] My Airflow PR" - https://issues.apache.org/jira/browse/AIRFLOW-XXX - In case you are fixing a typo in the documentation you can prepend your commit with \[AIRFLOW-XXX\], code changes always need a Jira issue. - In case you are proposing a fundamental code change, you need to create an Airflow Improvement Proposal ([AIP] (https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+Improvements+Proposals)). - In case you are adding a dependency, check if the license complies with the [ASF 3rd Party License Policy] (https://www.apache.org/legal/resolved.html#category-x). ### Description - [ ] Here are some details about my PR, including screenshots of any UI changes: ### Tests - [ ] My PR adds the following unit tests __OR__ does not need testing for this extremely good reason: ### Commits - [ ] My commits all reference Jira issues in their subject lines, and I have squashed multiple commits if they address the same issue. In addition, my commits follow the guidelines from "[How to write a good git commit message](http://chris.beams.io/posts/git-commit/)": 1. Subject is separated from body by a blank line 1. Subject is limited to 50 characters (not including Jira issue reference) 1. Subject does not end with a period 1. Subject uses the imperative mood ("add", not "adding") 1. Body wraps at 72 characters 1. Body explains "what" and "why", not "how" ### Documentation - [ ] In case of new functionality, my PR adds documentation that describes how to use it. - All the public functions and the classes in the PR contain docstrings that explain what it does - If you implement backwards incompatible changes, please leave a note in the [Updating.md] (https://github.com/apache/airflow/blob/master/UPDATING.md) so we can assign it to a appropriate release ### Code Quality - [ ] Passes `flake8` ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

2. tooptoop4 commented on pull request #6943: [AIRFLOW-4495] allow externally triggered dags to run for future exec… URL: https://github.com/apache/airflow/pull/6943 …_date Make sure you have checked _all_ steps below. ### Jira - [ ] My PR addresses the following [Airflow Jira] (https://issues.apache.org/jira/browse/AIRFLOW/) issues and references them in the PR title. For example, "\[AIRFLOW-XXX\] My Airflow PR" - https://issues.apache.org/jira/browse/AIRFLOW-XXX - In case you are fixing a typo in the documentation you can prepend your commit with \[AIRFLOW-XXX\], code changes always need a Jira issue. - In case you are proposing a fundamental code change, you need to create an Airflow Improvement Proposal ([AIP] (https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+Improvements+Proposals)). - In case you are adding a dependency, check if the license complies with the [ASF 3rd Party License Policy] (https://www.apache.org/legal/resolved.html#category-x). ### Description - [ ] Here are some details about my PR, including screenshots of any UI changes: ### Tests - [ ] My PR adds the following unit tests __OR__ does not need testing for this extremely good reason: ### Commits - [ ] My commits all reference Jira issues in their subject lines, and I have squashed multiple commits if they address the same issue. In addition, my commits follow the guidelines from "[How to write a good git commit message](http://chris.beams.io/posts/git-commit/)": 1. Subject is separated from body by a blank line 1. Subject is limited to 50 characters (not including Jira issue reference) 1. Subject does not end with a period 1. Subject uses the imperative mood ("add", not "adding") 1. Body wraps at 72 characters 1. Body explains "what" and "why", not "how" ### Documentation - [ ] In case of new functionality, my PR adds documentation that describes how to use it. - All the public functions and the classes in the PR contain docstrings that explain what it does - If you implement backwards incompatible changes, please leave a note in the [Updating.md] (https://github.com/apache/airflow/blob/master/UPDATING.md) so we can assign it to a appropriate release ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
3. tooptoop4 commented on pull request #5458: [AIRFLOW-4495] allow externally triggered dags to run for future 'Exe… URL: https://github.com/apache/airflow/pull/5458 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
4. tooptoop4 commented on pull request #7038: [AIRFLOW-4495] allow externally triggered dags to run for future exec dates URL: https://github.com/apache/airflow/pull/7038 --- Link to JIRA issue: https://issues.apache.org/jira/browse/AIRFLOW-4495 - [X] Description above provides context of the change - [X] Commit message starts with `[AIRFLOW-4495]`, where AIRFLOW-NNNN = JIRA ID* - [X] Unit tests coverage for changes (not needed for documentation changes) - [X] Commits follow "[How to write a good git commit message](http://chris.beams.io/posts/git-commit/)" - [X] Relevant documentation is updated including usage instructions. - [X] I will engage committers as explained in [Contribution Workflow Example] (https://github.com/apache/airflow/blob/master/CONTRIBUTING.rst#contribution-workflow-example). (*) For document-only changes, no JIRA issue is needed. Commit message starts `[AIRFLOW-XXXX]`. --- In case of

fundamental code change, Airflow Improvement Proposal ([AIP](https://cwiki.apache.org/confluence/display/AIRFLOW/Airflow+Improvements+Proposals)) is needed. In case of a new dependency, check compliance with the [ASF 3rd Party License Policy](https://www.apache.org/legal/resolved.html#category-x). In case of backwards incompatible changes please leave a note in [UPDATING.md](https://github.com/apache/airflow/blob/master/UPDATING.md). Read the [Pull Request Guidelines](https://github.com/apache/airflow/blob/master/CONTRIBUTING.rst#pull-request-guidelines) for more information. ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org

5. tooptoop4 commented on pull request #6943: [AIRFLOW-4495] allow externally triggered dags to run for future exec… URL: https://github.com/apache/airflow/pull/6943 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
6. kaxil commented on pull request #7038: [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates URL: https://github.com/apache/airflow/pull/7038 ---------- This is an automated message from the Apache Git Service. To respond to the message, please log on to GitHub and use the URL above to go to the specific comment. For queries about this service, please contact Infrastructure at: users@infra.apache.org
7. Commit 6414d38a04d74b04bd3c0add1dd61fff3df740c1 in airflow's branch refs/heads/master from tooptoop4 [https://gitbox.apache.org/repos/asf?p=airflow.git;h=6414d38] [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038)
8. Commit a002d5b7583b07ccb172a417221a5ad430670fca in airflow's branch refs/heads/v1-10-test from tooptoop4 [https://gitbox.apache.org/repos/asf?p=airflow.git;h=a002d5b] [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038) (cherry-picked from 6414d38a0)
9. Commit 8cb0a30ae88700468f20dc80517e65056b642a54 in airflow's branch refs/heads/v1-10-test from tooptoop4 [https://gitbox.apache.org/repos/asf?p=airflow.git;h=8cb0a30] [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038) (cherry-picked from 6414d38a0)
10. Commit 675c7a4139518bc01b7bbac5d25896d3f12d4752 in airflow's branch refs/heads/v1-10-test from tooptoop4 [https://gitbox.apache.org/repos/asf?p=airflow.git;h=675c7a4] [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038) (cherry-picked from 6414d38a0)
11. Commit 675c7a4139518bc01b7bbac5d25896d3f12d4752 in airflow's branch refs/heads/v1-10-stable from tooptoop4 [https://gitbox.apache.org/repos/asf?p=airflow.git;h=675c7a4] [AIRFLOW-4495] Allow externally triggered dags to run for future exec dates (#7038) (cherry-picked from 6414d38a0)