Arvados - Bug #12764

[Crunch2] Support writable file staging

12/06/2017 03:50 PM - Peter Amstutz

Status:	Resolved	**Start date:**	01/12/2018
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	2018-01-31 Sprint		

Description

> 2. Mount points underneath output_path must not use "writable":true. If any of them are set as writable, the API will refuse to create/update the container request, and crunch-run will fail the container.

Proposed alternate behavior, to support CWL writable file staging.

> 2. Mount points underneath output_path which have "writable":true will be copied into the target output_path and may be updated by the container.

The implementation should be easy - if a file or directory is marked writable and mounted under the output path, it should be copied into place after arv-mount is set up but before the container starts.

Subtasks:

Task # 12952: Review 12764-writable-file	**Resolved**
Task # 13017: crunch-run uses restricted permission parent directory	**Resolved**
Task # 13019: Review 12764-secure-tmp-parent	**Resolved**

Associated revisions

Revision 69b7576d - 01/31/2018 02:22 AM - Peter Amstutz

Merge branch '12764-writable-file' refs #12764

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

Revision 5857ebe3 - 01/31/2018 07:34 PM - Peter Amstutz

Writable staged files/dirs need to be 0777 to match outdir mode. refs #12764

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

Revision 5f6738f1 - 02/01/2018 04:35 PM - Peter Amstutz

Merge branch '12764-secure-tmp-parent' refs #12764

Arvados-DCO-1.1-Signed-off-by: Peter Amstutz <pamstutz@veritasgenetics.com>

History

#1 - 12/06/2017 03:50 PM - Peter Amstutz

- Status changed from New to In Progress

#2 - 12/06/2017 04:01 PM - Peter Amstutz

- Subject changed from [CWL] Attempt to mount file mounted from empty collection PDH to [CWL] writable files not supported, not detected

- Description updated

#3 - 12/06/2017 07:39 PM - Peter Amstutz

- Subject changed from [CWL] writable files not supported, not detected to [Crunch2] Support writable file staging

- Description updated

#4 - 12/06/2017 07:41 PM - Peter Amstutz

- Description updated

#5 - 12/06/2017 08:12 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

- Target version set to 2017-12-20 Sprint

#6 - 12/06/2017 08:28 PM - Peter Amstutz

- Status changed from In Progress to New

- Target version changed from 2017-12-20 Sprint to Arvados Future Sprints

#7 - 01/12/2018 02:19 PM - Peter Amstutz

- Status changed from New to In Progress

#8 - 01/12/2018 08:46 PM - Peter Amstutz

- Target version changed from Arvados Future Sprints to 2018-01-17 Sprint

#9 - 01/17/2018 07:36 PM - Peter Amstutz

- Target version changed from 2018-01-17 Sprint to 2018-01-31 Sprint

#10 - 01/26/2018 06:06 PM - Tom Clegg

AFAICT the current documented+enforced restriction ("must not be writable") comes from #9397 and its sole purpose was to get the then-required feature out faster by deferring the "copy" implementation.

> Mount points underneath output_path which have "writable":true will be copied into the target output_path and may be updated by the container

This sounds like a sensible implementation.

Just a suggestion, we might want to rephrase that a bit for the docs so it describes the user-visible behavior rather than the implementation. The meaning of writable=true ("can modify files, but modifications are not saved") is already mentioned elsewhere but might bear repeating. I wonder if it would be better to mention the consequences of the "copied into the target" part rather than just stating how it's implemented -- e.g., if your output path is in a tmpfs, you need to make sure it's big enough to accommodate copies of the inner writable mounts.

#11 - 01/29/2018 08:05 PM - Peter Amstutz

Tom Clegg wrote:

> Just a suggestion, we might want to rephrase that a bit for the docs so it describes the user-visible behavior rather than the implementation. The meaning of writable=true ("can modify files, but modifications are not saved") is already mentioned elsewhere but might bear repeating. I wonder if it would be better to mention the consequences of the "copied into the target" part rather than just stating how it's implemented -- e.g., if your output path is in a tmpfs, you need to make sure it's big enough to accommodate copies of the inner writable mounts.

Updated text:

"Mount points underneath output_path which have "writable":true are copied into output_path during container initialization and may be updated, renamed, or deleted by the running container. The original collection is not modified. On container completion, files remaining in the output are saved to the output collection. The mount at output_path must be big enough to accommodate copies of the inner writable mounts."

I've rebased on latest and am going to re-run tests, assuming they pass, any other comments prior to merge?

#12 - 01/29/2018 08:09 PM - Peter Amstutz

12764-writable-file @ 3e51fd869e600fd3d60e892346896fd9ce3d8c9c

#13 - 01/29/2018 08:39 PM - Tom Clegg

I'm not keen on this pattern

```
err = something()
if err == nil {
    // happy path
    if foo {
        err = somethingelse()
    } else {
        err = somethingdifferent()
```

```
    }
}
if err != nil {
    return err
}
```

Probably better to stick with the Go style "don't pyramid / handle errors first", even if that means using the same error fmt string twice.

Might help debugging if the error message mentions which mount was being processed. I guess that would involve adding "bind" to the copyFile struct?

The error handling in the WalkFunc seems a bit suspect. Is it intentional that it silently ignores non-nil incoming walkerr, and non-regular (pipe/socket/device/symlink) files? I would think all of those should be fatal errors, since they are unexpected/"impossible".

#14 - 01/29/2018 09:11 PM - Peter Amstutz

Tom Clegg wrote:

> Might help debugging if the error message mentions which mount was being processed. I guess that would involve adding "bind" to the copyFile struct?
>
> The error handling in the WalkFunc seems a bit suspect. Is it intentional that it silently ignores non-nil incoming walkerr, and non-regular (pipe/socket/device/symlink) files? I would think all of those should be fatal errors, since they are unexpected/"impossible".

Cleaned up error handling a bit:

12764-writable-file @ c8891a5eeec540404c7aff02a200fd95f6dbe64b

#15 - 01/29/2018 09:28 PM - Tom Clegg

LGTM

#16 - 01/29/2018 09:52 PM - Peter Amstutz

Tests at https://ci.curoverse.com/job/developer-run-tests/560/

... looks like there's a couple things left to fix.

#17 - 01/31/2018 09:48 PM - Tom Clegg

reviewing 12764-secure-tmp-parent @ 2eb85f70168d331e49e9d38bfbd2292d586dfcc1

Looks like runner.ParentTemp makes all the runner.CleanupTempDir stuff superfluous, so we should remove it.

apart from that, LGTM

I wonder if it would be worthwhile to call the tempdir "zzzzz-dz642-asdfasdfasdfasd-123456" instead of "crunch-run123456"?

#18 - 02/01/2018 08:31 PM - Peter Amstutz

- Target version changed from 2018-01-31 Sprint to 2018-02-14 Sprint

#19 - 02/01/2018 09:32 PM - Peter Amstutz

- Status changed from In Progress to Resolved

- Target version changed from 2018-02-14 Sprint to 2018-01-31 Sprint