Arvados - Feature #7816

[Crunch2] Execute minimal container spec with logging

11/17/2015 07:12 AM - Tom Clegg

Status:	Resolved	**Start date:**	11/17/2015
Priority:	Normal	**Due date:**	
Assigned To:	Peter Amstutz	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2016-01-20 Sprint		

Description

This program runs on a worker host. It will be invoked by the future Crunch2 dispatcher. Its responsibilities include:

1. Retrieve the Container record with the given UUID
2. Load the specified docker image from Keep
3. Translate the Container specification to a "docker run" command line or invoke the container via Docker SDK
4. Forward the executor's logging and container's stdout, stderr to the Arvados API as Log entries and to a collection
5. Update the Container record to reflect the actual state of the container (progress/completion), exit code, logs
6. Testing for all above features

Not included in this story:

- Start arv-mount on the host (if required by the container) and shutting it down when the container exits (#8015)
- Impose runtime constraints, using "docker run" arguments and other facilities (#8017)
- Report resources used by the container while it runs (presumably using crunchstat) (#8016)
- Save the container's output in the Container record (except logs) (#8020)
- log throttling (#8019)
- retry failure (#8018)

See Containers API

Subtasks:

Task # 8089: Review 7816-exit-code	**Resolved**
Task # 8044: Review 7816-crunch2-exec	**Resolved**
Task # 8046: Review testing pseudocode	**Resolved**
Task # 8043: Write tests	**Resolved**
Task # 8042: Implement executor	**Resolved**
Task # 8045: Define test cases	**Resolved**

Related issues:

Related to Arvados - Feature #6518: [Crunch] [Crunch2] Dispatch containers vi...	**Resolved**	**07/08/2015**
Blocked by Arvados - Story #6429: [API] [Crunch2] Implement "containers" and ...	**Resolved**	**12/03/2015**

Associated revisions

Revision 3344f5b9 - 01/04/2016 07:34 PM - Peter Amstutz

Merge branch '7816-exit-code' refs #7816

Revision ef3e45fc - 01/20/2016 04:04 PM - Peter Amstutz

Merge branch '7816-crunch2-exec' closes #7816

Revision 22e538ac - 01/20/2016 04:40 PM - Peter Amstutz

Add crunch-run to run-tests refs #7816

Revision 22e538ac - 01/20/2016 04:40 PM - Peter Amstutz

Add crunch-run to run-tests refs #7816

History

#1 - 11/17/2015 03:09 PM - Brett Smith

- Target version set to Arvados Future Sprints

#2 - 12/14/2015 02:55 PM - Peter Amstutz

(Notes from #8001)

Implement an "arv-container" command. This will run on the compute node to actually implement container execution.

It has the following responsibility:

```
Initialize from uuid of a Queued container record and API token
    Redirect stdout & stderr to logs table & keep
    Fetch and install Docker image as needed
    Set up mount points as described by container record. May include fetching from git and setting up keep mo
unt points.
    Run docker container
    Finalize log, output & mark container record as Complete.
```

#3 - 12/15/2015 03:41 PM - Brett Smith

- Target version deleted (Arvados Future Sprints)

- Release set to 11

#4 - 12/15/2015 08:58 PM - Peter Amstutz

- Description updated

#5 - 12/15/2015 08:58 PM - Peter Amstutz

- Description updated

- Story points set to 3.0

#6 - 12/15/2015 09:00 PM - Peter Amstutz

- Description updated

#7 - 12/15/2015 09:48 PM - Brett Smith

- Target version set to 2016-01-06 sprint

#8 - 12/16/2015 07:24 PM - Peter Amstutz

- Subject changed from [Crunch2] Execute a container with specified mounts and constraints to [Crunch2] Execute minimal container spec with logging

#9 - 12/16/2015 08:33 PM - Peter Amstutz

- Assigned To set to Peter Amstutz

#10 - 12/16/2015 09:42 PM - Peter Amstutz

Test cases:

```
{
    "command": ["echo", "hello world"],
    "container_image": "debian:8",
    "cwd": ".",
    "environment": {},
    "mounts": {},
    "output_path": "/tmp",
    "priority": 1,
    "runtime_constraints": {}
}
```

- Test that image is downloaded (test may need to delete image if it's already present on the local system)
- Test that running the image doesn't produce any Docker errors
- Test that container record changes from "Queued" to "Running" to "Complete"
- Test that exit_code changes from "null" to "0"
- Test that "hello world" is logged to logs table
- Test that "hello world" is logged to collection & set on "log" field
- Test that logs table is cleaned up after the "log" collection is created and the "log" field is set

```
{
    "command": ["false"],
    "container_image": "debian:8",
    "cwd": ".",
```

```
    "environment": {},
    "mounts": {},
    "output_path": "/tmp",
    "priority": 1,
    "runtime_constraints": {}
}
```

- Test that exit_code is "1"

```
{
    "command": ["pwd"],
    "container_image": "debian:8",
    "cwd": ".",
    "environment": {},
    "mounts": {},
    "output_path": "/tmp",
    "priority": 1,
    "runtime_constraints": {}
}
```

- Test that it prints the CWD of the image

Failure conditions to test:

- Test that failing to download the docker image is logged
- Test that failing to execute the image is logged
- Test that failing to save the logs to a collection is logged and logs table is left alone

Suggest that we add an "Error" state to container that means some kind of infrastructure failure happened. Containers that run and complete as normal are put in "Complete" state with their exit code set.

(we could also just say that "Complete" with a null exit code implies an Error state, but it seems better to be explicit)

#11 - 12/17/2015 06:24 PM - Tom Clegg

container_image is a PDH here, not a docker image name:tag. (I guess we need a "download docker images to keep" component before we can support specifying a docker image name:tag in a ContainerRequest; until then, the caller has to get the image into Keep before making the ContainerRequest.)

Should add a test case that emits stderr (better yet: both stderr and stdout).

For infrastructure errors, either exit_code=NULL or exit_code>0 would be fine for this version. (An "Error" state might be useful but we should discuss further before we go there. Let's not wedge this story behind too many container API improvements if we can help it.)

Test that log timestamps in the log file are RFC3339 UTC with subsecond precision.

Deleting the docker image ("docker rmi") during test cleanup seems sketchy if we're using debian:8 -- it means downloading it many times, and could easily conflict with other uses of docker on the dev/test box. How about loading it by docker image hash instead (so we don't accumulate unlimited versions over time as debian:8 gets updated), and skipping the delete?

#12 - 12/29/2015 03:12 PM - Peter Amstutz

Looks like we can get stats directly from Docker: https://godoc.org/github.com/samalba/dockerclient#DockerClient.StartMonitorStats

#13 - 01/04/2016 02:54 PM - Tom Clegg

In source:services/api/test/unit/container_test.rb assert_raises() should wrap only c.save!, not c.reload etc.; other than that, and an extra blank line near the end of that same file, 7816-exit-code LGTM.

#14 - 01/05/2016 09:29 PM - Tom Clegg

+1 syncing locator regexp with the one on the wiki, thanks

FileToken seems a somewhat confusing name for the data structure that results from unmarshalling a file token. I suppose these are StreamSegments? Other suggestions?

instead of copypasting expectStringSlicesEqual to expectFileTokensEqual, just use reflect.DeepEqual()

Instead of RFC3339Timestamp(), how about now.Format(RFC3339Fixed) (with const RFC3339Fixed = "2006-01-02T15:04:05.000000000Z07:00")

Seems we can't have busybox in the source tree / git history, it's redistributable only as GPLv2 (unless we pin to <= 1.2.2) ... I'd suggest we stub docker with binstubs in PATH (like crunch-job tests) to do unit testing, and let docker download images from the internet for real (and store in Keep for real) to do an integration test.

ThrottledLogger seems more complicated than it needs to be. Suggestion to simplify:

- add a chan bytes.Buffer to ThrottledLogger
- in Write(), append to the current buf, then send the current buf to the bufs channel if the channel is ready

```
select {
case this.bufs <- this.buf:
  this.buf = &bytes.Buffer{}
default:
}
```

- combine flusher() and goWriter() into something like

```
ticker := time.NewTicker(time.Second)
for b := range this.bufs {
  this.writer.Write(b.Bytes())
  <- ticker.C
}
```

- in Stop() send this.buf if non-empty, then close the bufs channel

ThrottledLogger Stop() should be called Close() and return an error, then ThrottledLogger will be an io.WriteCloser

I'm finding the logging stuff generally a bit hard to follow. I wonder if it would be better to split the various features (add timestamp to each line, throttle, write to arvados logs table) into different types that implement WriteCloser (or ReadCloser) by wrapping another writer/reader? io.MultiWriter or io.TeeReader can be used to tee to stderr, logs, and Keep at the appropriate points. Or maybe I just need some comments to summarize how the plumbing makes the right flavor of log go to each destination?

While copypasting and modifying crunchrunner's ManifestStreamWriter/ManifestWriter to CollectionFileWriter/CollectionWriter, maybe document the connection/differences between the two in some comments at the top? Obviously they should be consolidated into a supported SDK feature at some point (but I don't want to block on that now).

RuntimeConstraints should probably be a map[string]interface{} -- some values are going to be numbers, arrays, etc.

The use of the ContainerRunner.Cancelled bool seems a bit unsafe: SetupSignals() makes it possible for Cancelled to change to true at any time, like between CommitLogs() and UpdateContainerRecordComplete(), which would cause the log to say Complete but the end state to be Cancelled. Perhaps sync.Once would be a good way to ensure finalState stays stable once we get to the top of the defer func in Run()?

- [redacted]

```
  ...
  finalize sync.Once
}

this.finalize(func() { this.finalState = "Cancelled" })
```

Instead of err.Error() "Cancelled" we can have a var ErrCancelled = errors.New("Cancelled") and then check err ErrCancelled

Go style (try "golint *.go"):

- Don't use "this" or "self"
- Uuid should be UUID, Api should be API (or Client, since the object isn't really an API), etc.
- Add comments on non-trivial methods saying what they're expected to do

(Haven't looked at the tests yet.)

#15 - 01/06/2016 04:49 PM - Tom Clegg

- File crunch-exec-test-log-b702c49.txt added

running tests takes 97s (why is it slow?) and one test fails (log attached).

Not obvious what FullRunHelper() does, a comment would be good, maybe even a more descriptive name.

(*ClosableBuffer)Write() is superfluous -- if you delete it, ClosableBuffer will get bytes.Buffer's Write method automatically.

You might be able to replace ErrorReader with something like iotest.TimeoutReader(bytes.NewBufferString("foo"))

#16 - 01/06/2016 06:57 PM - Ward Vandewege

- Status changed from New to In Progress

#17 - 01/06/2016 08:05 PM - Brett Smith

- Target version changed from 2016-01-06 sprint to 2016-01-20 Sprint

- Story points changed from 3.0 to 1.0

#18 - 01/15/2016 05:05 PM - Peter Amstutz

Tom Clegg wrote:

> +1 syncing locator regexp with the one on the wiki, thanks
>
> FileToken seems a somewhat confusing name for the data structure that results from unmarshalling a file token. I suppose these are StreamSegments? Other suggestions?

Now "FileStreamSegments".

> instead of copypasting expectStringSlicesEqual to expectFileTokensEqual, just use reflect.DeepEqual()

Gone.

> Instead of RFC3339Timestamp(), how about now.Format(RFC3339Fixed) (with const RFC3339Fixed = "2006-01-02T15:04:05.000000000Z07:00")

Done.

> Seems we can't have busybox in the source tree / git history, it's redistributable only as GPLv2 (unless we pin to <= 1.2.2) ... I'd suggest we stub docker with binstubs in PATH (like crunch-job tests) to do unit testing, and let docker download images from the internet for real (and store in Keep for real) to do an integration test.

Busybox is gone, testing is now against a stub Docker service instead of a real one.

> I'm finding the logging stuff generally a bit hard to follow. I wonder if it would be better to split the various features (add timestamp to each line, throttle, write to arvados logs table) into different types that implement WriteCloser (or ReadCloser) by wrapping another writer/reader? io.MultiWriter or io.TeeReader can be used to tee to stderr, logs, and Keep at the appropriate points. Or maybe I just need some comments to summarize how the plumbing makes the right flavor of log go to each destination?

I added comments. The motivation for the design is to move data through in a way that's threadsafe, asynchronous, and efficient.

> ThrottledLogger Stop() should be called Close() and return an error, then ThrottledLogger will be an io.WriteCloser

Fixed.

> While copypasting and modifying crunchrunner's ManifestStreamWriter/ManifestWriter to CollectionFileWriter/CollectionWriter, maybe document the connection/differences between the two in some comments at the top? Obviously they should be consolidated into a supported SDK feature at some point (but I don't want to block on that now).

Agree, a comprehensive Collection object would be ideal, for the time being I added a note.

> RuntimeConstraints should probably be a map[string]interface{} -- some values are going to be numbers, arrays, etc.

Fixed.

> The use of the ContainerRunner.Cancelled bool seems a bit unsafe: SetupSignals() makes it possible for Cancelled to change to true at any time, like between CommitLogs() and UpdateContainerRecordComplete(), which would cause the log to say Complete but the end state to be Cancelled. Perhaps sync.Once would be a good way to ensure finalState stays stable once we get to the top of the defer func in Run()?
>
> - [...]

I think sync.Once is overkill but you're right there's a race there. It now decided if it is Complete or Cancelled before doing the finalization.

> Instead of err.Error() "Cancelled" we can have a var ErrCancelled = errors.New("Cancelled") and then check err ErrCancelled

Good idea. Fixed.

> Go style (try "golint *.go"):
>
> - Don't use "this" or "self"
> - Uuid should be UUID, Api should be API (or Client, since the object isn't really an API), etc.

- Add comments on non-trivial methods saying what they're expected to do

Golint is't part of the main go package? I guess I can install it...

#19 - 01/18/2016 11:28 PM - Tom Clegg

> instead of copypasting expectStringSlicesEqual to expectFileTokensEqual, just use reflect.DeepEqual()

Gone.

Not gone at eff4b3c...?

Pls rename to crunch-run (cf. IRC... match up with "docker run")

It appears that the plumbing goes docker -> CopyReaderToLog -> ThrottledLogger -> ArvLogWriter, where

- CopyReaderToLog splits long lines into foo[...]\n [...]bar\n etc.
- ThrottledLogger adds timestamps, then batches lines in order to write only once per second
- ArvLogWriter tees the batched-up writes to arvados.v1.logs.create() and another writer (a CollectionFileWriter)

Am I reverse-engineering correctly?

This seems to mean we're munging long lines with "[...]" even in the collection we write to Keep. Is this an intentional change from the current Crunch1 behavior?

(Using exactly the same timestamp in both logs is certainly a welcome change from the current Crunch1 behavior...)

Is there any reason for communicating via (*logging.Logger)Print() instead of just calling (*ThrottledLogger)Write() (or io.WriteString()) from CopyReaderToLog? It seems like CopyReaderToLog is the only thing that writes to a ThrottledLogger, and it always writes one line at a time, which seems to make ThrottledLogger's use of Scanner redundant. (I feel like I'm probably missing something here...)

Anyway, most of this stuff can be dealt with later.

I'd like to get it renamed to crunch-run before merging, to avoid digging the Rename Hole deeper in test/packaging/docs, though.

#20 - 01/19/2016 09:44 PM - Peter Amstutz

Tom Clegg wrote:

> > instead of copypasting expectStringSlicesEqual to expectFileTokensEqual, just use reflect.DeepEqual()
>
> Gone.
>
> Not gone at eff4b3c...?

Oops, found it and fixed it.

> Pls rename to crunch-run (cf. IRC... match up with "docker run")

Done.

> It appears that the plumbing goes docker -> CopyReaderToLog -> ThrottledLogger -> ArvLogWriter, where
>
> - CopyReaderToLog splits long lines into foo[...]\n [...]bar\n etc.
> - ThrottledLogger adds timestamps, then batches lines in order to write only once per second
> - ArvLogWriter tees the batched-up writes to arvados.v1.logs.create() and another writer (a CollectionFileWriter)
>
> Am I reverse-engineering correctly?

Yes, that's right.

> This seems to mean we're munging long lines with "[...]" even in the collection we write to Keep. Is this an intentional change from the current Crunch1 behavior?

> (Using exactly the same timestamp in both logs is certainly a welcome change from the current Crunch1 behavior...)

Yes, I think it's desirable for the logs table and keep logs to match up exactly (except for whatever throttling we apply). Not splitting lines (or doing it differently) makes that a lot harder.

> Is there any reason for communicating via (*logging.Logger)Print() instead of just calling (*ThrottledLogger)Write() (or io.WriteString()) from CopyReaderToLog? It seems like CopyReaderToLog is the only thing that writes to a ThrottledLogger, and it always writes one line at a time, which seems to make ThrottledLogger's use of Scanner redundant. (I feel like I'm probably missing something here...)

It's because the line splitting does Print(prefix, string(line), suffix). Also it's passing in a log.Logger interface, not a Writer interface. I don't know if there is what internal buffering is in the logger, it may be possible to avoid a copy by writing through.

> Anyway, most of this stuff can be dealt with later.
>
> I'd like to get it renamed to crunch-run before merging, to avoid digging the Rename Hole deeper in test/packaging/docs, though.

Done

#21 - 01/20/2016 03:50 PM - Peter Amstutz

One further consideration, I renamed CopyReaderToLog to ReadWriteLines updated it to use the Writer directly.

#22 - 01/20/2016 04:10 PM - Peter Amstutz

- Status changed from In Progress to Resolved

- % Done changed from 83 to 100

Applied in changeset arvados|commit:ef3e45fcc338a85432c207685567385972f79ee6.

Files

crunch-exec-test-log-b702c49.txt	1.55 KB	01/06/2016	Tom Clegg