Arvados - Story #14807

[arvados-dispatch-cloud] Features/fixes needed before first production deploy

02/04/2019 10:17 PM - Tom Clegg

Status:	Resolved	**Start date:**	01/29/2019
Priority:	Normal	**Due date:**	
Assigned To:	Tom Clegg	**% Done:**	100%
Category:	Crunch	**Estimated time:**	0.00 hour
Target version:	2019-03-27 Sprint		

Description

Issues encountered & fixed/worked around during dev deploy:

- Include instance address (host or IP) in logs and management API responses
- Ensure crunch-run --list works even if /var/lock is a symlink
- Log full instance ID, not (Instance)String(), which might be an abbreviated name
- Fix management API endpoints to allow specifying instance IDs that have slashes
- Pass SSH public key to Azure so it doesn't crash (Azure refuses to create a node without adding an admin account)
- Fix host part of SSH target address being dropped
- Allow driver to specify a login username
- Send ARVADOS_API_* values on stdin instead of environment vars (typical SSH server is configured to refuse these env vars)
- If ProviderType is not given in an instance type in the cluster config, default to the type name (not the empty string)
- Pass a random string to Azure driver as "node-token" (or fix Azure driver so it doesn't expect that)

Further improvements necessary to run in production:

- Send detached crunch-run stdout+stderr to systemd journal so sysadmin can make subsequent arrangements if needed
- Metrics: total cost of nodes in idle or booting state
- Metrics: total cost of nodes with admin-hold flag set
- Log when an instance goes down unexpectedly (i.e., state != Shutdown when deleted from list)
- Log when a container is added to or dropped from the queue
- Obey logging format in cluster config file (as of #14325, HTTP request logs were JSON, operational logs were text)
- Drain node if container process still running after several SIGTERM attempts
- Provide a "mark node as broken" callback mechanism for crunch-run (drain node, unless it's already marked "hold" -- see #14807#note-20)
- Configurable rate limit for Create and Destroy calls to cloud API (background: reaching API call rate limits can cause penalties; also, when multiple instance types are created concurrently, the cloud might create the lower-priority types but then reach quota before creating the higher-priority types; see #14360#note-36)
- Metrics: number of containers, split by state and instance type
- Load API host & token from cluster config file instead of env vars
- Ensure crunch-run exits instead of hanging if ARVADOS_API_HOST/TOKEN is empty or broken
- Kill containers (or at least log a warning) if a worker is kept busy by a container whose UUID does not exist according to the API server's queue (e.g., container deleted from database) #14977
- "Kill instance now" management API
- (Azure) error out if AddedScratch>0 because that isn't implemented yet

Other improvements that were made here even though not necessary to run in production:

- crunch-run --detach: send logs to journal
- Move "cat .../node-token" host key verification mechanism out of Azure driver (instead, have the dispatcher do this itself if the driver returns cloud.ErrNotImplemented)

Dispatching containers to cloud VMs

Subtasks:

Task # 14868: Review 14807-dispatch-cloud-fixes	**Resolved**
Task # 14962: Review 14807-escalate-sigterm	**Resolved**
Task # 15006: Review 14807-prod-blockers	**Resolved**
Task # 15016: Review 14807-prod-blockers (last 2 items)	**Resolved**

Related issues:

Related to Arvados - Story #13908: [Epic] Replace SLURM for cloud job schedul...	**Resolved**	
Related to Arvados - Bug #14844: [dispatch-cloud] Azure driver bugs discovere...	**Resolved**	**02/28/2019**

Related to Arvados - Bug #15045: [arvados-cloud-dispatch] commit 115cbd648263...	**Resolved**	
Blocked by Arvados - Bug #14977: [arvados-dispatch-cloud] kill crunch-run pro...	**Resolved**	**03/18/2019**
Follows Arvados - Feature #14325: [crunch-dispatch-cloud] Dispatch containers...	**Resolved**	**01/28/2019**

Associated revisions

Revision ca2d9469 - 02/21/2019 07:41 PM - Tom Clegg

Merge branch '14807-dispatch-cloud-fixes'

refs #14807

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision d4da0c6a - 03/05/2019 07:08 PM - Tom Clegg

Merge branch '14807-zero-metrics'

refs #14807

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision 1c21d98f - 03/20/2019 07:24 PM - Tom Clegg

Merge branch '14807-escalate-sigterm'

refs #14807

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision b715e06b - 03/21/2019 08:50 PM - Tom Clegg

Merge branch '14807-prod-blockers'

refs #14807

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

Revision 004cf8a6 - 03/26/2019 02:28 PM - Tom Clegg

Merge branch '14807-prod-blockers'

refs #14807

Arvados-DCO-1.1-Signed-off-by: Tom Clegg <tclegg@veritasgenetics.com>

History

#1 - 02/04/2019 10:17 PM - Tom Clegg

- Related to Story #13908: [Epic] Replace SLURM for cloud job scheduling/dispatching added

#2 - 02/04/2019 10:17 PM - Tom Clegg

- Due date set to 01/29/2019

- Start date set to 01/29/2019

- Follows Feature #14325: [crunch-dispatch-cloud] Dispatch containers to cloud VMs directly, without slurm or nodemanager added

#3 - 02/06/2019 08:59 PM - Tom Clegg

- Description updated

#4 - 02/06/2019 09:26 PM - Tom Clegg

- Description updated

#5 - 02/12/2019 03:43 PM - Tom Clegg

- Description updated

#6 - 02/14/2019 03:00 PM - Tom Clegg

- Description updated

#7 - 02/14/2019 06:45 PM - Tom Clegg

14807-dispatch-cloud-fixes @ 16589cd93e6780db6a07d7cc110724dea4c19e3e fixes a number of issues we ran into while deploying #14325 on a dev cluster.

- 16589cd93 14807: Include more detail in errors.
- 2873d55ea 14807: Fix crunch-run --list output when /var/lock is a symlink.
- 3a1c03950 14807: Always set node-token tag.
- 6e4237de7 14807: Log full instance ID.
- d63c18fb8 14807: Expose instance IP addresses in logs and management API.
- 55c07b5b9 14807: Fix SSH target address.
- 286e41383 14807: Accept .../instances/_/drain?instance_id=X.
- 554d1808c 14807: Pass SSH public key to driver.
- e57e3e19b 14807: Allow driver to specify SSH username.
- 45113a215 14807: When ProviderType is unspecified, default to Arvados type.
- ddcb0fb32 14807: Pass env vars on stdin instead of using SSH feature.
- ae31f1897 14807: Match systemd description to component name.

This will need to be rebased after #14745 merges, though.

#8 - 02/14/2019 07:00 PM - Tom Clegg

- Description updated

#9 - 02/15/2019 09:34 PM - Tom Clegg

- Status changed from New to In Progress

Rebased.

- 36e1f63fd 14807: Send detached crunch-run logs to journal via systemd-cat.
- 970af93af 14807: Remove errant rm.
- 79693e508 14807: Update API endpoints: instance_id is always a query param.
- e1e0f6789 14807: Log idle time in seconds instead of nanoseconds.
- 87bf45c8c 14807: Cancel or requeue container when priority drops to zero.
- 91b39ff3f 14807: Use context to pass a suitable logger to all service commands.
- 80c48b78f 14807: Log when a container is added/removed from the queue.
- abd21f165 14807: Split instance count/size/cost metrics by idle/hold status.
- d6fbaeba4 14807: Fix up azure log message.
- 30ca2a11c 14807: Move secret-tag host key verify mechanism out of Azure driver.
- 3d662ef38 14807: Don't delete existing tags when updating.
- f6d551a68 14807: Load API host/token directly from stdin without shell hack.
- 3d7b91541 14807: Wait at least 1 second between retries on initial queue poll.
- d3cef2f89 14807: Include more detail in errors.
- de9a5e270 14807: Fix crunch-run --list output when /var/lock is a symlink.
- 0de109fe6 14807: Always set node-token tag.
- e96f8774c 14807: Log full instance ID.
- 832235d35 14807: Expose instance IP addresses in logs and management API.
- 601eeec89 14807: Fix SSH target address.
- 97a1babd7 14807: Accept .../instances/_/drain?instance_id=X.
- c4c77dc1e 14807: Pass SSH public key to driver.
- efe3cb087 14807: Allow driver to specify SSH username.
- ed317e6b2 14807: When ProviderType is unspecified, default to Arvados type.
- d2bdc5af9 14807: Pass env vars on stdin instead of using SSH feature.
- bcabab96d 14807: Match systemd description to component name.

#10 - 02/19/2019 06:40 PM - Tom Clegg

- Description updated

#11 - 02/19/2019 06:50 PM - Tom Clegg

- Description updated

Addressed in this branch:

- Include instance address (host or IP) in logs and management API responses
- Ensure crunch-run --list works even if /var/lock is a symlink
- Log full instance ID, not (Instance)String(), which might be an abbreviated name
- Fix management API endpoints to allow specifying instance IDs that have slashes
- Pass SSH public key to Azure so it doesn't crash (Azure refuses to create a node without adding an admin account)
- Fix host part of SSH target address being dropped
- Allow driver to specify a login username
- Send ARVADOS_API_* values on stdin instead of environment vars (typical SSH server is configured to refuse these env vars)

- If ProviderType is not given in an instance type in the cluster config, default to the type name (not the empty string)
- Pass a random string to Azure driver as "node-token" (or fix Azure driver so it doesn't expect that)

(The "node-token" stuff is moved out of the Azure driver entirely, see below)

- Send detached crunch-run stdout+stderr to systemd journal so sysadmin can make subsequent arrangements if needed
- Metrics: total cost of nodes in idle or booting state
- Metrics: total cost of nodes with admin-hold flag set
- Log when an instance goes down unexpectedly (i.e., state != Shutdown when deleted from list)
- Log when a container is added to or dropped from the queue
- Obey logging format in cluster config file (as of #14325, HTTP request logs were JSON, operational logs were text)

- Move "cat .../node-token" host key verification mechanism out of Azure driver (instead, have the dispatcher do this itself if the driver returns cloud.ErrNotImplemented)

Still left to do:

- Load API host & token from cluster config file instead of env vars
- Ensure crunch-run exits instead of hanging if ARVADOS_API_HOST/TOKEN is empty or broken
- Configurable rate limit for Create and Destroy calls to cloud API (background: reaching API call rate limits can cause penalties; also, when multiple instance types are created concurrently, the cloud might create the lower-priority types but then reach quota before creating the higher-priority types; see #14360#note-36)
- Send SIGKILL if container process still running after several SIGTERM attempts / N seconds after first SIGTERM
- Shutdown node if container process still running after several SIGKILL attempts
- Propagate configured "check for broken node" script name to crunch-run
- Metrics: number of containers, split by state (and instance type?)

desired, but not necessary to run in production

- Metrics that indicate cloud failure (time we've spent trying but failing to create a new instance)
- Test suite that uses a real cloud provider
- Test activity/resource usage metrics
- Multiple cloud drivers
- Generic driver test suite
- Performance metrics for dispatching (e.g., time between seeing a container in the queue and starting its crunch-run process on a worker)
- Optimize worker VM deployment (e.g., automatically install a matching version of crunch-run on each worker)
- Configurable spending limits
- Update runtime_status field when cancelling containers after crunch-run crashes or the cloud VM dies without finalizing the container (already done for the "no suitable instance type" case)
- If present, use VM image ID given in runtime_constraints instead of image ID from cluster config file
- (API) Allow admin users to specify image ID in runtime_constraints
- Metrics: count unexpected shutdowns, split by instance type
- Atomically install correct version of crunch-run (perhaps /proc/self/exe) to worker VM as part of boot probe

#12 - 02/19/2019 09:03 PM - Tom Clegg

- Description updated

#13 - 02/20/2019 05:17 AM - Tom Clegg

- Assigned To set to Tom Clegg

- Target version changed from To Be Groomed to 2019-02-27 Sprint

#14 - 02/20/2019 01:59 PM - Tom Clegg

14807-dispatch-cloud-fixes @ 6d852fc2b60e140decfafc92a971a5d1f027e854https://ci.curoverse.com/job/developer-run-tests/1079/

#15 - 02/20/2019 02:01 PM - Tom Clegg

- Description updated

#16 - 02/20/2019 04:41 PM - Tom Clegg

- Related to Bug #14844: [dispatch-cloud] Azure driver bugs discovered in trial run added

#17 - 02/20/2019 04:52 PM - Tom Clegg

14807-dispatch-cloud-fixes @ d97388bdbfeb6a43cb86996012a1db0ba4a8871fhttps://ci.curoverse.com/job/developer-run-tests/1080/

#18 - 02/21/2019 02:35 PM - Tom Clegg

Added one more commit:

- 9559f6df3 14807: Avoid doing concurrent update requests per container.

Ward noticed during testing on c97qk that the dispatcher was making lots of overlapping "cancel" API requests for the same container, instead of waiting for the first attempt to succeed. This change avoids launching more than one API call at a time for any one container. (Rate-limiting API requests and/or pausing after errors might be even better, but this seems like a good start.)

#19 - 02/21/2019 03:51 PM - Peter Amstutz

```
cmd := "crunch-run --detach '" + ctr.UUID + "'"
```

Why does crunch-run need to run as root? Is it to access Docker, /var/lock etc? On slurm, we don't run as root, but the crunch user is in the Docker group. I guess for our purposes, it already has root access, it might as well use it.

```
    tagKeyNodeToken     = "node-token" // deprecated, but required by Azure driver
```

The "deprecated" comment suggests a legacy feature that has been superceded by a new feature, which is a weird thing to write in a brand-new component. (Should at least expand the explanation, the comment is cryptic).

The ssh_executor tests fail for me:

```
FAIL: executor_test.go:93: ExecutorSuite.TestExecute

target address "::"
executor_test.go:142:
    c.Check(err, check.ErrorMatches, `.*(unable to authenticate|connection refused).*`)
... error string = "dial tcp [::]:22: connect: cannot assign requested address"
... regex string = ".*(unable to authenticate|connection refused).*"

executor_test.go:148:
    c.Check(err, check.ErrorMatches, `.*connection refused.*`)
... error string = "dial tcp [::]:0: connect: cannot assign requested address"
... regex string = ".*connection refused.*"

target address "::"
executor_test.go:142:
    c.Check(err, check.ErrorMatches, `.*(unable to authenticate|connection refused).*`)
... error string = "dial tcp [::]:22: connect: cannot assign requested address"
... regex string = ".*(unable to authenticate|connection refused).*"

executor_test.go:148:
    c.Check(err, check.ErrorMatches, `.*connection refused.*`)
... error string = "dial tcp [::]:0: connect: cannot assign requested address"
... regex string = ".*connection refused.*"

target address "::"
executor_test.go:142:
    c.Check(err, check.ErrorMatches, `.*(unable to authenticate|connection refused).*`)
... error string = "dial tcp [::]:22: connect: cannot assign requested address"
... regex string = ".*(unable to authenticate|connection refused).*"

executor_test.go:148:
    c.Check(err, check.ErrorMatches, `.*connection refused.*`)
... error string = "dial tcp [::]:0: connect: cannot assign requested address"
... regex string = ".*connection refused.*"

OOPS: 1 passed, 1 FAILED
--- FAIL: Test (0.03s)
FAIL
coverage: 82.9% of statements
FAIL    git.curoverse.com/arvados.git/lib/dispatchcloud/ssh_executor    0.036s
```

#20 - 02/21/2019 04:01 PM - Peter Amstutz

> Propagate configured "check for broken node" script name to crunch-run

This is slightly backwards, the broken node hook doesn't test for a broken node, it is the action to take once crunch-run has decided the node is broken (eg certain types of Docker failures indicating the Docker daemon is unhealthy). We do need to propagate that to crunch-run, but we also need a mechanism by which the compute node can indicate to c-d-c that it shouldn't be used (on slurm, we run scontrol state=FAILED).

#21 - 02/21/2019 04:24 PM - Peter Amstutz

On the theme of simplifying, I also bumped #12900 up near the top of the to be groomed list

#22 - 02/21/2019 05:34 PM - Tom Clegg

- Description updated

Peter Amstutz wrote:

> [...]
>
> Why does crunch-run need to run as root? Is it to access Docker, /var/lock etc? On slurm, we don't run as root, but the crunch user is in the Docker group. I guess for our purposes, it already has root access, it might as well use it.

Indeed, we should be running crunch-run (and arv-mount) in a non-root account -- I think all we really need is docker and fuse. I've added that to the "not a blocker for production" list for now -- perhaps we will move it up to "blocker".

> The "deprecated" comment suggests a legacy feature that has been superceded by a new feature, which is a weird thing to write in a brand-new component. (Should at least expand the explanation, the comment is cryptic).
>
> The ssh_executor tests fail for me:
>
> [...]

30ca2a1 removed the "deprecated" comment, so I suspect you're not testing the current branch...?

#23 - 02/21/2019 06:33 PM - Tom Clegg

If the tests are failing on 9559f6df3 then I suspect it's a test env thing (localhost resolving to an IPv6 addr?) since tests pass for me & jenkins. I'll try to reproduce.

#24 - 02/21/2019 07:04 PM - Tom Clegg

It seems net.Listen("tcp", ":") in a basic docker container ("docker run debian:9") where there is no IPv6 address returns a listener whose address is reported as [::]:12345, even though it's impossible to connect to that address. The previous version passed the test because of a bug: it ignored the address completely, and connected to :12345, which works.

Changed the test suite to use 127.0.0.1 -- this will break in an IPv6-only host, but at least it should work in the more likely case of an IPv4-only host.

#25 - 02/21/2019 07:09 PM - Peter Amstutz

Tom Clegg wrote:

> It seems net.Listen("tcp", ":") in a basic docker container ("docker run debian:9") where there is no IPv6 address returns a listener whose address is reported as [::]:12345, even though it's impossible to connect to that address. The previous version passed the test because of a bug: it ignored the address completely, and connected to :12345, which works.
>
> Changed the test suite to use 127.0.0.1 -- this will break in an IPv6-only host, but at least it should work in the more likely case of an IPv4-only host.

Passes now. LGTM.

#26 - 02/22/2019 02:56 AM - Tom Clegg

- Description updated

#27 - 02/22/2019 02:58 AM - Tom Clegg

- Target version changed from 2019-02-27 Sprint to Arvados Future Sprints

#28 - 03/04/2019 06:46 PM - Tom Clegg

- Description updated

#29 - 03/04/2019 06:49 PM - Tom Clegg

- Description updated

#30 - 03/08/2019 08:18 PM - Tom Clegg

- Description updated

#31 - 03/13/2019 03:27 PM - Tom Morris

- Target version changed from Arvados Future Sprints to 2019-03-27 Sprint

#32 - 03/15/2019 11:53 PM - Tom Clegg

- Description updated

#33 - 03/18/2019 01:42 PM - Tom Clegg

- Blocked by Bug #14977: [arvados-dispatch-cloud] kill crunch-run procs for containers that are deleted or have state=Cancelled when dispatcher starts up added

#34 - 03/18/2019 04:06 PM - Tom Clegg

14807-escalate-sigterm @ 96f73954851d44ab715fa0b34861a93c00aa0519https://ci.curoverse.com/view/Developer/job/developer-run-tests/1133/

- Send SIGKILL if container process still running after several SIGTERM attempts / N seconds after first SIGTERM
- Shutdown node if container process still running after several SIGKILL attempts

This rearranges the code a little, adding a new remoteRunner struct that

- corresponds to a crunch-run process on a worker node
- owns the "start" and "kill" code
- owns the (new) goroutine that sends term/kill signals repeatedly until the remote process dies (or it gives up on SIGKILL and instructs the worker to drain/shutdown)

...rather than add multiple map[string]whatever fields to worker to track "already started a kill process", "gave up on SIGTERM so shouldn't keep the node alive", etc.

The branch also includes some test improvements:

- Report "container runs more than once" as a dispatcher bug (before, the stub driver's fake crunch-run would sometimes make changes after "--kill" or "--list" had declared it dead -- with that fixed, "container runs more than once" can be detected reliably)
- Fix (unrelated) unreliable test that expected (cloud.InstanceSet)Create() to be synchronous

#35 - 03/19/2019 03:20 PM - Peter Amstutz

Let me see if I understand the logic. This should probably be documented in the code.

1. First send TERM. Successful TERM means graceful shutdown and mostly leaves the worker in known in a known state suitable for reuse.
2. If TERM doesn't succeed after some time (success seems to be effectively defined as "the file at lockprefix+uuid+locksuffix was cleaned up") then escalate to KILL
3. If the process is killed, it will leave a dangling lockfile (now unlocked) with a stale PID
4. os.FindProcess always returns success even if the process doesn't exist
5. Sending a signal to a non-existent process returns the internal error os.errFinished = errors.New("os: process already finished"), not nil. This contradicts the comment on KillProcess() "It returns 0 if the process is successfully killed or didn't exist in the first place."
6. after killDeadline is exceeded onUnkillable() is called which set the worker to draining
7. the worker probe determines what containers are running by trying to lock each lockfile it finds. A successful kill will leave the lockfile unlocked, so it will report nothing running on the worker, this will call closeRunner() which abandons any further escalation
8. an empty, draining node will shut down normally
9. however, if crunch-run is in an uninterruptable sleep (conceivable if FUSE driver falls over and then crunch-run tries to interact with the mount) it can't be killed, so a node which is in draining state and consists only of containers for which the sentKILL flag is set, is also a candidate for shutdown

Since the worker is in an undefined state crunch-run has been killed, I think that once it has given up sending TERM and is about to escalate to KILL, it should immediately mark the worker as draining to prevent any more work being scheduled on the worker. Whether it is able to successfully send a KILL or not, a worker in an undefined state shouldn't be reused. However, we don't want to interrupt other containers still running successfully.

#36 - 03/19/2019 04:18 PM - Tom Clegg

Peter Amstutz wrote:

> 1. First send TERM. Successful TERM means graceful shutdown and mostly leaves the worker in known in a known state suitable for reuse.
> 2. If TERM doesn't succeed after some time (success seems to be effectively defined as "the file at lockprefix+uuid+locksuffix was cleaned up") then escalate to KILL

Success is defined as "process has ended".

Lockfile cleanup happens later but doesn't affect this. It just saves time checking ancient stale ones, and avoid filling /var/lock/ with cruft.

> 1. If the process is killed, it will leave a dangling lockfile (now unlocked) with a stale PID
> 2. os.FindProcess always returns success even if the process doesn't exist
> 3. Sending a signal to a non-existent process returns the internal error os.errFinished = errors.New("os: process already finished"), not nil. This contradicts the comment on KillProcess() "It returns 0 if the process is successfully killed or didn't exist in the first place."

Your reading of os.FindProcess() and proc.Signal() agree with mine, but I think KillProcess() is working as advertised. Perhaps you didn't see the "err != nil means kill succeeded" logic at the end of kill()? I've added some comments there to draw attention to it.

1. after killDeadline is exceeded onUnkillable() is called which set the worker to draining
2. the worker probe determines what containers are running by trying to lock each lockfile it finds. A successful kill will leave the lockfile unlocked, so it will report nothing running on the worker, this will call closeRunner() which abandons any further escalation
3. an empty, draining node will shut down normally
4. however, if crunch-run is in an uninterruptable sleep (conceivable if FUSE driver falls over and then crunch-run tries to interact with the mount) it can't be killed, so a node which is in draining state and consists only of containers for which the sentKILL flag is set, is also a candidate for shutdown

> Since the worker is in an undefined state crunch-run has been killed, I think that once it has given up sending TERM and is about to escalate to KILL, it should immediately mark the worker as draining to prevent any more work being scheduled on the worker. Whether it is able to successfully send a KILL or not, a worker in an undefined state shouldn't be reused. However, we don't want to interrupt other containers still running successfully.

Indeed, it seems better to drain the worker when we give up on SIGTERM, since we can assume the user's container and/or arv-mount is still running (and other things on the node are awry) whether or not SIGKILL succeeds in killing crunch-run.

In that case, sending SIGKILL seems pointless, and perhaps counterproductive since it would interfere with debugging. Would it be better to keep sending SIGTERM while waiting for the remaining containers to drain?

#37 - 03/19/2019 05:33 PM - Peter Amstutz

Tom Clegg wrote:

1. If the process is killed, it will leave a dangling lockfile (now unlocked) with a stale PID
2. os.FindProcess always returns success even if the process doesn't exist
3. Sending a signal to a non-existent process returns the internal error os.errFinished = errors.New("os: process already finished"), not nil. This contradicts the comment on KillProcess() "It returns 0 if the process is successfully killed or didn't exist in the first place."

> Your reading of os.FindProcess() and proc.Signal() agree with mine, but I think KillProcess() is working as advertised. Perhaps you didn't see the "err != nil means kill succeeded" logic at the end of kill()? I've added some comments there to draw attention to it.

Oh, I see the logic flips. If proc.Signal() **doesn't** return an error, that's an error.

Technically proc.Signal() could also return EPERM, but I think we currently assume c-d-c logs in as root or uses sudo, so that shouldn't happen.

> Indeed, it seems better to drain the worker when we give up on SIGTERM, since we can assume the user's container and/or arv-mount is still running (and other things on the node are awry) whether or not SIGKILL succeeds in killing crunch-run.
>
> In that case, sending SIGKILL seems pointless, and perhaps counterproductive since it would interfere with debugging. Would it be better to keep sending SIGTERM while waiting for the remaining containers to drain?

Agree about not sending KILL so there's at least some opportunity for debugging. I don't think continuing to send TERM makes a difference one way or the other. If crunch-run didn't get the message the first time it gets TERM, sending TERM forty more times isn't going to make it behave any differently.

#38 - 03/19/2019 05:36 PM - Tom Clegg

14807-escalate-sigterm @ 41365ac598721e31fc88c462934e0a06cafe2aaehttps://ci.curoverse.com/view/Developer/job/developer-run-tests/1136/

```
-// Kill starts a background task to kill the remote process,
-// escalating from SIGTERM to SIGKILL to onUnkillable() according to
-// the configured timeouts.
+// Kill starts a background task to kill the remote process, first
+// trying SIGTERM until reaching timeoutTERM, then calling
+// onUnkillable().
+//
+// SIGKILL is not used. It would merely kill the crunch-run supervisor
+// and thereby make the docker container, arv-mount, etc. invisible to
+// us without actually stopping them.
```

#39 - 03/19/2019 05:56 PM - Peter Amstutz

Another thought:

crunch-run only waits 1 second for the process to exit. However, it is likely for it to take longer than that even when everything is behaving, . Asking docker to stop the container sends TERM and escalates to KILL after 10 seconds. Stopping arv-mount has an 8 second timeout. Cancelling a container still uploads results, which also takes time. This means we're likely to log "kill failed" repeatedly. I think this is harmless (eventually

crunch-run does go away, or c-d-c drains the node) but it will create logging noise and ops doesn't like to see errors being logged as part of normal operation.

I suggest either extending crunch-run's "wait for process to go away" period to 15 or 20 seconds, or suppressing logging these errors until termDeadline has passed.

#40 - 03/19/2019 07:00 PM - Tom Clegg

> ops doesn't like to see errors being logged as part of normal operation
>
> suggest ... suppressing logging these errors until termDeadline has passed

I agree, in fact this distinction is already implemented using log levels:

- "kill failed" is level=info because it's part of normal operation (but it's helpful when troubleshooting to know about the attempt and see the error message)
- "unkillable container, draining worker" is level=warn because it suggests a problem somewhere

#41 - 03/20/2019 06:52 PM - Peter Amstutz

Tom Clegg wrote:

> > ops doesn't like to see errors being logged as part of normal operation
> >
> > suggest ... suppressing logging these errors until termDeadline has passed
>
> I agree, in fact this distinction is already implemented using log levels:
>
> - "kill failed" is level=info because it's part of normal operation (but it's helpful when troubleshooting to know about the attempt and see the error message)
> - "unkillable container, draining worker" is level=warn because it suggests a problem somewhere

Ah, I overlooked the log levels. Although, assuming everything logs at "info", the message "kill failed" will still appear in the logs. I'm just trying to avoid a future debug session where an operator looking at the logs gets hung up on an incorrect diagnosis based on the word "failed". It is really important to distinguish for anyone looking at the logs between expected errors (part of normal operation, but sometimes still interesting) and unexpected errors, I don't know if the log level totally captures that.

Maybe tweak the message so that instead of Info("kill failed") it uses softer language like Info("crunch-run --kill has not succeeded yet")

Otherwise LGTM.

#42 - 03/20/2019 07:26 PM - Tom Clegg

OK, changed "kill failed" → "kill attempt unsuccessful". Info log includes any stderr coming from crunch-run, like "%s: pid %d: sent signal %d (%s) but process is still alive" so hopefully it won't be too hard to interpret.

#43 - 03/20/2019 07:33 PM - Tom Clegg

- Description updated

14807-prod-blockers @ 743680ab176697218aa41839149e02a160786bdf

- 743680ab1 14807: Give up if initial keep services list isn't loaded in 60s. ("Ensure crunch-run exits instead of hanging")
- b0efcaab2 14807: Report queue metrics.
- 0e1e46da3 14807: Add "kill instance" management API.
- a614135cd 14807: Configurable rate limit for cloud provider API calls.
- ef288ca2c 14807: Drain instances that crunch-run reports broken.

#44 - 03/21/2019 01:59 PM - Peter Amstutz

- What is the rational for having the dispatcher API look like

POST /arvados/v1/dispatch/instances/kill?instance_id=xxx

instead of

POST /arvados/v1/dispatch/instances/xxx/kill

which would be more consistent with the existing arvados API?

- How hard would it be to have a metric "age of container at the head of the queue" or "age of containers when they start running" as a way of measuring average latency from when a container enters the queue to when it starts running, this would be helpful in diagnosing situations like "is something wrong or is the cluster just heavily loaded"

#45 - 03/21/2019 02:16 PM - Tom Clegg

- Description updated

Peter Amstutz wrote:

- What is the rational for having the dispatcher API look like

POST /arvados/v1/dispatch/instances/kill?instance_id=xxx

instead of

POST /arvados/v1/dispatch/instances/xxx/kill

Azure instance IDs contain slashes. 97a1babd776addd419fb5c050157786bdbd6232f

- How hard would it be to have a metric "age of container at the head of the queue" or "age of containers when they start running" as a way of measuring average latency from when a container enters the queue to when it starts running

Not hard. This is on the "not included here" list, but I noticed it's missing from Dispatching containers to cloud VMs so I added it there too.

#46 - 03/21/2019 04:35 PM - Peter Amstutz

Tom Clegg wrote:

> Peter Amstutz wrote:
>
> - What is the rational for having the dispatcher API look like
>
> POST /arvados/v1/dispatch/instances/kill?instance_id=xxx
>
> instead of
>
> POST /arvados/v1/dispatch/instances/xxx/kill
>
> Azure instance IDs contain slashes. 97a1babd776addd419fb5c050157786bdbd6232f
>
> - How hard would it be to have a metric "age of container at the head of the queue" or "age of containers when they start running" as a way of measuring average latency from when a container enters the queue to when it starts running
>
> Not hard. This is on the "not included here" list, but I noticed it's missing from Dispatching containers to cloud VMs so I added it there too.

Thanks.

14807-prod-blockers LGTM

#47 - 03/21/2019 04:41 PM - Peter Amstutz

Peter Amstutz wrote:

> Tom Clegg wrote:
>
> > Peter Amstutz wrote:
> >
> > - What is the rational for having the dispatcher API look like
> >
> > POST /arvados/v1/dispatch/instances/kill?instance_id=xxx
> >
> > instead of
> >
> > POST /arvados/v1/dispatch/instances/xxx/kill
> >
> > Azure instance IDs contain slashes. 97a1babd776addd419fb5c050157786bdbd6232f
> >
> > - How hard would it be to have a metric "age of container at the head of the queue" or "age of containers when they start running"

as a way of measuring average latency from when a container enters the queue to when it starts running

Not hard. This is on the "not included here" list, but I noticed it's missing from Dispatching containers to cloud VMs so I added it there too.

Thanks.

14807-prod-blockers LGTM

Actually, I just realized I looked at the code but didn't run tests. Tests here:

https://ci.curoverse.com/view/Developer/job/developer-run-tests/1141/

#48 - 03/21/2019 05:31 PM - Peter Amstutz

14807-prod-blockers LGTM

Actually, I just realized I looked at the code but didn't run tests. Tests here:

https://ci.curoverse.com/view/Developer/job/developer-run-tests/1141/

.... aaand that's failing a dispatchcloud test so I'll have to take back that premature LGTM...

#49 - 03/21/2019 05:49 PM - Peter Amstutz

There's a very large amount of log output. Here's just the failure.

```
    c.Fatalf("timed out; still waiting for %d containers: %q", len(waiting), waiting)
... Error: timed out; still waiting for 98 containers: map["zzzzz-dz642-000000000000066":{} "
zzzzz-dz642-000000000000196":{} "zzzzz-dz642-000000000000123":{} "zzzzz-dz642-000000000000018":{} "
zzzzz-dz642-000000000000057":{} "zzzzz-dz642-000000000000092":{} "zzzzz-dz642-000000000000015":{} "
zzzzz-dz642-000000000000035":{} "zzzzz-dz642-000000000000162":{} "zzzzz-dz642-000000000000002":{} "
zzzzz-dz642-000000000000026":{} "zzzzz-dz642-000000000000099":{} "zzzzz-dz642-000000000000017":{} "
zzzzz-dz642-000000000000081":{} "zzzzz-dz642-000000000000187":{} "zzzzz-dz642-000000000000138":{} "
zzzzz-dz642-000000000000163":{} "zzzzz-dz642-000000000000011":{} "zzzzz-dz642-000000000000058":{} "
zzzzz-dz642-000000000000145":{} "zzzzz-dz642-000000000000188":{} "zzzzz-dz642-000000000000105":{} "
zzzzz-dz642-000000000000107":{} "zzzzz-dz642-000000000000148":{} "zzzzz-dz642-000000000000153":{} "
zzzzz-dz642-000000000000186":{} "zzzzz-dz642-000000000000028":{} "zzzzz-dz642-000000000000051":{} "
zzzzz-dz642-000000000000154":{} "zzzzz-dz642-000000000000185":{} "zzzzz-dz642-000000000000195":{} "
zzzzz-dz642-000000000000170":{} "zzzzz-dz642-000000000000001":{} "zzzzz-dz642-000000000000012":{} "
zzzzz-dz642-000000000000033":{} "zzzzz-dz642-000000000000044":{} "zzzzz-dz642-000000000000059":{} "
zzzzz-dz642-000000000000073":{} "zzzzz-dz642-000000000000003":{} "zzzzz-dz642-000000000000139":{} "
zzzzz-dz642-000000000000155":{} "zzzzz-dz642-000000000000137":{} "zzzzz-dz642-000000000000115":{} "
zzzzz-dz642-000000000000171":{} "zzzzz-dz642-000000000000194":{} "zzzzz-dz642-000000000000100":{} "
zzzzz-dz642-000000000000116":{} "zzzzz-dz642-000000000000177":{} "zzzzz-dz642-000000000000114":{} "
zzzzz-dz642-000000000000067":{} "zzzzz-dz642-000000000000113":{} "zzzzz-dz642-000000000000146":{} "
zzzzz-dz642-000000000000178":{} "zzzzz-dz642-000000000000036":{} "zzzzz-dz642-000000000000082":{} "
zzzzz-dz642-000000000000122":{} "zzzzz-dz642-000000000000130":{} "zzzzz-dz642-000000000000025":{} "
zzzzz-dz642-000000000000074":{} "zzzzz-dz642-000000000000161":{} "zzzzz-dz642-000000000000052":{} "
zzzzz-dz642-000000000000034":{} "zzzzz-dz642-000000000000106":{} "zzzzz-dz642-000000000000179":{} "
zzzzz-dz642-000000000000098":{} "zzzzz-dz642-000000000000124":{} "zzzzz-dz642-000000000000050":{} "
zzzzz-dz642-000000000000089":{} "zzzzz-dz642-000000000000042":{} "zzzzz-dz642-000000000000065":{} "
zzzzz-dz642-000000000000075":{} "zzzzz-dz642-000000000000193":{} "zzzzz-dz642-000000000000043":{} "
zzzzz-dz642-000000000000083":{} "zzzzz-dz642-000000000000132":{} "zzzzz-dz642-000000000000027":{} "
zzzzz-dz642-000000000000091":{} "zzzzz-dz642-000000000000147":{} "zzzzz-dz642-000000000000019":{} "
zzzzz-dz642-000000000000164":{} "zzzzz-dz642-000000000000169":{} "zzzzz-dz642-000000000000009":{} "
zzzzz-dz642-000000000000090":{} "zzzzz-dz642-000000000000129":{} "zzzzz-dz642-000000000000049":{} "
zzzzz-dz642-000000000000068":{} "zzzzz-dz642-000000000000084":{} "zzzzz-dz642-000000000000121":{} "
zzzzz-dz642-000000000000041":{} "zzzzz-dz642-000000000000131":{} "zzzzz-dz642-000000000000172":{} "
zzzzz-dz642-000000000000108":{} "zzzzz-dz642-000000000000076":{} "zzzzz-dz642-000000000000097":{} "
zzzzz-dz642-000000000000010":{} "zzzzz-dz642-000000000000020":{} "zzzzz-dz642-000000000000156":{} "
zzzzz-dz642-000000000000004":{}]

OOPS: 8 passed, 1 FAILED
--- FAIL: Test (10.03s)
```

#50 - 03/21/2019 05:52 PM - Peter Amstutz

I see "probe reported broken instance" appearing a lot, that may be delaying completion past the test deadline, either there's a bug in the code, the

test harness, or the failure rate in the test is too high.

#51 - 03/21/2019 06:02 PM - Peter Amstutz

- File 14807-fail-log.txt added

Here's the whole log

#52 - 03/21/2019 06:16 PM - Tom Clegg

14807-prod-blockers @ d2aa25400fe0b6b8fc231b1c5e0d32085e38b6bchttps://ci.curoverse.com/job/developer-run-tests/1144/

If every fake node hits a fake failure mode in 200-400 ms, throughput can fall below 20 containers per second and fail the test.

#53 - 03/21/2019 08:58 PM - Tom Clegg

- Description updated

Another branch with the last two items

14807-prod-blockers @ 4d9a5c4f0f19c2e6d394dca6a1de903dc09c43e5https://ci.curoverse.com/view/Developer/job/developer-run-tests/1145/

- 4d9a5c4f0 14807: Refuse to create instance with AddedScratch>0.
- 115cbd648 14807: Load API host/token from cluster config if present.

#54 - 03/22/2019 03:45 PM - Peter Amstutz

Tom Clegg wrote:

> - 115cbd648 14807: Load API host/token from cluster config if present.

I'm a little confused by this one.

lib/service/cmd.go gets cluster.SystemRootToken, or falls back to ARVADOS_API_TOKEN, then sets it as context[serviceToken]

The dispatcher gets the context, saves a copy of the context, but also sets the token using "AuthToken: service.Token(ctx)"

dispatcher.initialize() calls arvados.NewClientFromConfig(disp.Cluster) (which doesn't set AuthToken) or falls back to arvados.NewClientFromEnv() (which does)

Then client.AuthToken is overwritten with dispatcher.AuthToken

I see what you're doing but it seems slightly convoluted, essentially there's at least two places someone needs to look to understand where the client config comes from. Also storing the token in both context[serviceToken] and dispatcher.AuthToken is redundant. I don't know if there's a way to refactor it to be a bit more straightforward? (If that's going to be hard, If I don't want to hold up the branch on this, though).

#55 - 03/22/2019 06:48 PM - Tom Clegg

Falling back to env vars (for a sane transition) does make it a bit more complicated. Setting that aside, the idea is

- The dispatcher code shouldn't be responsible for finding an appropriate token; it should just be passed in. Currently, it could just take SystemRootToken from the cluster config, but that's not a long term solution so I'm avoiding the path where we bake that assumption into individual components.
- The service.Command wrapper creates a dispatcher, supplying an appropriate cluster config and token.
- Other callers (like tests) can also create a dispatcher, supplying an appropriate cluster config and token.

As for having to look in two places to get config, this is a goal. I'm trying to get away from the idea of bundling the auth token with client config. A single client should be able to do work (concurrently) on behalf of various clients/tokens. Ideally cluster configs, service discovery, and auth secrets are distinct things.

Embedding the token in a context does seem like an unnecessary complication. I've deleted that, and added it as an argument to service.NewHandlerFunc instead.

14807-prod-blockers @ 433d10b31924631f5b4c18b828301a4fe45bbf0chttps://ci.curoverse.com/view/Developer/job/developer-run-tests/1146/

(branch also adds a test case for lib/service)

#56 - 03/25/2019 01:40 PM - Peter Amstutz

Tom Clegg wrote:

> Falling back to env vars (for a sane transition) does make it a bit more complicated. Setting that aside, the idea is

- The dispatcher code shouldn't be responsible for finding an appropriate token; it should just be passed in. Currently, it could just take SystemRootToken from the cluster config, but that's not a long term solution so I'm avoiding the path where we bake that assumption into individual components.
- The service.Command wrapper creates a dispatcher, supplying an appropriate cluster config and token.
- Other callers (like tests) can also create a dispatcher, supplying an appropriate cluster config and token.

> As for having to look in two places to get config, this is a goal. I'm trying to get away from the idea of bundling the auth token with client config. A single client should be able to do work (concurrently) on behalf of various clients/tokens. Ideally cluster configs, service discovery, and auth secrets are distinct things.

By "two places" I was referring to two places in the code (service/cmd.go and dispatchcloud/dispatcher.go both have logic to first look at the config and then fall back to environment), not two sources of configuration. Making the token passed in explicitly makes it a bit clearer that the intention is the service framework provides the token. What if services/cmd.go also set cluster.Services.Controller.ExternalURL from ARVADOS_API_HOST as necessary, so dispatchcloud can assume the configuration is filled in and only needs to call NewClientFromConfig() ?

> Embedding the token in a context does seem like an unnecessary complication. I've deleted that, and added it as an argument to service.NewHandlerFunc instead.

Thanks.

> 14807-prod-blockers @ 433d10b31924631f5b4c18b828301a4fe45bbf0chttps://ci.curoverse.com/view/Developer/job/developer-run-tests/1146/
>
> (branch also adds a test case for lib/service)

Aside from the (optional) suggestion above, this LGTM.

#57 - 03/25/2019 02:58 PM - Tom Clegg

Peter Amstutz wrote:

> What if services/cmd.go also set cluster.Services.Controller.ExternalURL from ARVADOS_API_HOST as necessary, so dispatchcloud can assume the configuration is filled in and only needs to call NewClientFromConfig() ?

Ah, I see. Yes, I've moved the "fall back to env var for token" code into lib/service so it doesn't clutter dispatcher. (It's on its way to a separate "config migration/defaults" piece, but I don't want to creep too far into #13648 here.)

14807-prod-blockers @ e281a42ac126952960838e0aae826e00091a8404https://ci.curoverse.com/view/Developer/job/developer-run-tests/1147/

#58 - 03/26/2019 02:26 PM - Peter Amstutz

Tom Clegg wrote:

> Peter Amstutz wrote:
>
> > What if services/cmd.go also set cluster.Services.Controller.ExternalURL from ARVADOS_API_HOST as necessary, so dispatchcloud can assume the configuration is filled in and only needs to call NewClientFromConfig() ?
>
> Ah, I see. Yes, I've moved the "fall back to env var for token" code into lib/service so it doesn't clutter dispatcher. (It's on its way to a separate "config migration/defaults" piece, but I don't want to creep too far into #13648 here.)
>
> 14807-prod-blockers @ e281a42ac126952960838e0aae826e00091a8404https://ci.curoverse.com/view/Developer/job/developer-run-tests/1147/

This LGTM.

#59 - 03/26/2019 06:01 PM - Tom Clegg

- Description updated

#60 - 03/26/2019 07:27 PM - Tom Clegg

- Status changed from In Progress to Resolved

#61 - 03/29/2019 02:00 PM - Ward Vandewege

- Related to Bug #15045: [arvados-cloud-dispatch] commit 115cbd6482632c47fdcbbbe4abc9543e7e8e30ec breaks API host loading added

#62 - 03/29/2019 02:08 PM - Ward Vandewege

- Subject changed from [crunch-dispatch-cloud] Features/fixes needed before first production deploy to [arvados-dispatch-cloud] Features/fixes

needed before first production deploy

#63 - 05/21/2019 10:27 PM - Tom Morris

- Release set to 15

Files

14807-fail-log.txt	3.17 MB	03/21/2019	Peter Amstutz