Arvados - Story #17667

Install docs explains InternalURL / ExternalURL, private networks & split DNS

05/12/2021 03:35 PM - Peter Amstutz

Status:	In Progress	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:	Ward Vandewege	**% Done:**	0%
Category:	Documentation	**Estimated time:**	0.00 hour
Target version:	2021-10-27 sprint		

Description

The externalURL is not explained on the config reference page, should probably be added there. The InternalURL is explained accurately there.

Defaults for internalURL should be added to the config reference, with our standard ports for the services. This will also mean we have defaults for this, for the first time!

The private networks/split DNS explanation could at the bottom of the 'planning and prerequesites' page in the docs.

Subtasks:

Task # 17669: Review **New**

History

#1 - 05/12/2021 04:07 PM - Peter Amstutz

- Assigned To set to Nico César

#2 - 05/25/2021 02:31 PM - Nico César

- Status changed from New to In Progress

#3 - 05/25/2021 04:13 PM - Ward Vandewege

- Description updated

#4 - 05/26/2021 04:32 PM - Peter Amstutz

- Target version changed from 2021-05-26 sprint to 2021-06-09 sprint

#5 - 06/09/2021 03:48 PM - Peter Amstutz

- Assigned To changed from Nico César to Ward Vandewege

#6 - 06/09/2021 03:48 PM - Peter Amstutz

- Target version changed from 2021-06-09 sprint to 2021-06-23 sprint

#7 - 06/19/2021 03:06 PM - Ward Vandewege

- Target version changed from 2021-06-23 sprint to 2021-07-07 sprint

#8 - 07/07/2021 03:33 PM - Peter Amstutz

- Target version changed from 2021-07-07 sprint to 2021-07-21 sprint

#9 - 07/07/2021 07:18 PM - Ward Vandewege

From our documentation:

```
# In each of the service sections below, the keys under
# InternalURLs are the endpoints where the service should be
# listening, and reachable from other hosts in the cluster.
```

InternalURLs is used by the Arvados services to determine the IP/port they should listen on. But it is also used by the other services in the cluster as the address of the relevant service.

ExternalURL is used by the service to advertise the IP address/port or hostname/port combination it listens at, for access from outside the cluster.

For example:

```
Services:
  Controller:
    InternalURLs: "http://localhost:9004": {}
    ExternalURL: "https://xxxx1.arvadosapi.com"
  RailsAPI:
    InternalURLs: "http://localhost:8000": {}
    ExternalURL: "-"
  Keepstore:
    InternalURLs:
      "http://keep0.xxxx1.arvadosapi.com:25107":
        Rendezvous: rendezvousvalu1
      "http://keep1.xxxx1.arvadosapi.com:25107":
        Rendezvous: rendezvousvalu2
  Workbench1:
    ExternalURL: "https://workbench.xxxx1.arvadosapi.com"
  Workbench2:
    ExternalURL: "https://workbench2.xxxx1.arvadosapi.com"
  Keepbalance:
    InternalURLs:
      "http://xxxx1.arvadosapi.com:9005/": {}
  DispatchCloud:
    ExternalURL: "-"
    InternalURLs:
      "http://xxxx1.arvadosapi.com:9006": {}
  Keepproxy:
    InternalURLs:
     "http://localhost:25107": {}
    ExternalURL: "https://keep.xxxx1.arvadosapi.com"
  Websocket:
    InternalURLs:
      "http://127.0.0.1:9003": {}
    ExternalURL: "wss://ws.xxxx1.arvadosapi.com/websocket"
  WebDAV:
    InternalURLs:
      "http://127.0.0.1:9200": {}
    ExternalURL: "https://collections.xxxx1.arvadosapi.com/"
  WebDAVDownload:
    ExternalURL: "https://download.xxxx1.arvadosapi.com/"
  WebShell:
    InternalURLs: {}
    ExternalURL: "https://webshell.xxxx1.arvadosapi.com/"
```

In this configuration example, arvados-controller listens on localhost port 9004, and it is externally accessible at https://xxxx1.arvadosapi.com. This is possible because there is an nginx reverse proxy that sits in front of the `arvados-controller` process and maps the external hostname to localhost:9004 on the machine that runs arvados-controller.

The xxxx1.arvadosapi.com hostname must resolve from outside the cluster, **but also from the inside**. A split DNS setup can be handy here; routing internal traffic to the external IP address for the API server tends to be bad from a security perspective. In a cloud environment, it can also become expensive as all traffic to official IP addresses tends to cost money, even when it originates from inside the same cloud provider.

RailsAPI is configured differently: the ExternalURL is set to - which means it is not applicable (the RailsAPI process is not directly accessible from outside). The InternalURL is set to http://localhost:8000, and this hostname:port combination is used by arvados-controller to access RailsAPI. Note that because localhost is used in this example, this implies that arvados-controller and RailsAPI live on the same host in this configuration. RailsAPI is a rails process, and the hostname/port combination it listens on is actually defined in the nginx/Passenger connection, so in this case InternalURL is only used by e.g. arvados-controller to find the RailsAPI.

The Keepstore entry has no ExternalURL defined, which is equivalent to it being set to -. This is because keepstores should not be directly accessible from outside the cluster. The InternalURL values are set to keep0.xxxx1.arvadosapi.com:25107 and keep1.xxxx1.arvadosapi.com:25107, which implies that those 2 hostnames must resolve to the correct hosts **inside the Arvados cluster**. The keepstore processes on those hosts will resolve that DNS entry to an IP address and listen on that address at port 25107. Simultaneously, any services **inside the Arvados cluster** that need to talk to the keepstores will resolve the keep0.xxxx1.arvadosapi.com hostname to an IP address that must be routable to the host that runs that keepstore process.

The Workbench1 and Workbench2 entries have no InternalURL defined, because they don't need one. Workbench1 is a Rails application; the IP/port it listens on is configured in the nginx/Passenger configuration. Workbench2 is a set of flat files served by nginx.
The values for ExternalURL must resolve to the correct host **from outside the cluster** so that clients can access these services. Typically these hostnames will point to an nginx reverse proxy that sits in front of Workbench1, and that can serve the files for Workbench2.

The Keepbalance entry has no ExternalURL because this service should not be accessible from outside the cluster. The InternalURL is set to http://xxxx1.arvadosapi.com:9005/, and is only used to export Prometheus metrics. The keep-balance program will resolve xxxx1.arvadosapi.com to an IP address and listen on that IP address at port 9005. The process that polls those metrics must be able to resolve xxxx1.arvadosapi.com to an IP address local to the Arvados cluster, and contact that IP address at port 9005.

The Dispatchcloud entry follows the same logic as Keepbalance.

The Keepproxy entry lists http://localhost:25107 as its InteralURL. The keepproxy process listens on that hostname/port. There is an nginx reverse process that sits in front of it that maps the https://keep.xxxx1.arvadosapi.com hostname to localhost:25107. This means that keep.xxxx1.arvadosapi.com must resolve to the correct IP for this nginx reverse proxy **from outside the cluster**.

The Websocket entry is similar to the Keepproxy entry. Note that the ExternalURL is a websockets URL. The ws.xxxx1.arvadosapi.com hostname must resolve from outside the cluster. To make the websocket service accessible from inside (handy from shell nodes, for example), the ws.xxxx1.arvadosapi.com must also resolve correctly from inside. See the node about split DNS, above.

The WebDAV entry is similar to the Keepproxy entry. It is used by the keep-web program to provide a WebDAV endpoint.

The WebDAVDownload entry only has an ExternalURL defined. This is an additional endpoint provided by keep-web for file download. The download.xxxx1.arvadosapi.com hostname must resolve **from outside the cluster**.

The WebShell entry defines an empty list for InternalURLs; this is the same as omitting the InternalURLs key. It is not needed because the webshell functionality is provided by third party tools, and configured separately from the Arvados configuration in the nginx and shellinabox configuration files. The ExternalURL field is set to https://webshell.xxxx1.arvadosapi.com/ which must resolve from outside the cluster.

#10 - 07/12/2021 04:00 PM - Tom Clegg

ExternalURL

- **Instructs clients/applications how to connect to Arvados services**
- when used outside the cluster, must resolve to [a load balancer or proxy which calls through to] the host where the service is running
- when used inside the cluster, should resolve to the **internal** IP address of the host where the service is running
  - otherwise traffic is slower / more expensive
  - this only matters for controller and websocket (and webdav, if any applications are using it inside the cluster)
  - shouldn't matter for keepproxy because internal clients should always use keepstore directly
- not needed for certain services (railsapi, keep-balance, arvados-dispatch-cloud) because clients/applications do not connect to them
- not needed for keepstore because "internal" clients are told to connect to keepstore's internalURLs, and "external" clients are told to connect to keepproxy

InternalURLs

- **Instructs server components how to connect to other server components within the cluster**
- when used inside the cluster, must resolve to the host where the service is running
- does not need to be resolvable/routable from outside the cluster
- completely ignored in some cases because we don't (yet) use the config file to control routing -- instead we require the operator to type the same info into an nginx.conf file
- can be "localhost" (or 0.0.0.0) in some cases
  - railsapi, because it only accepts requests from controller, which runs on the same host
  - workbench1 + 2, because it only accepts requests forwarded by the nginx proxy which runs on the same host
- keepstore is unusual in that clients/applications that are identified as "internal" are told to connect to the InternalURLs

Currently, several services use InternalURLs to figure out the desired listening address at startup, which has a couple of problems that (I think) are fixable like so:

- If ListenAddress is set, listen on that address
  - If keepstore, get the ARVADOS_SERVICE_INTERNAL_URL env var and choose the corresponding entry from InternalURLs to determine which keepstore volumes, to use (if not set, and there's more than one keepstore in the config, then fail)
- Otherwise, if one of the InternalURLs has a host:port part that is usable as a listening address, listen on that host:port.
  - If ARVADOS_SERVICE_INTERNAL_URL env var is set, choose the corresponding InternalURLs entry. (This makes it possible to run two keepstore services on a single node, e.g., for testing/demo.)
- Otherwise, don't start the service.

tbd: Still seems like there's a better way to handle keepstore in the kubernetes scenario where the hostname isn't known when creating the config file.

- perhaps: If ListenAddress is set, and ARVADOS_SERVICE_INTERNAL_URL is not set, and there's more than one keepstore in the config *but AccessViaHosts is empty for all volumes,* then it doesn't matter which InternalURLs entry we think is "ours" -- so listen on ListenAddress and use all volumes.

tbd:

- Would it be an error to use ListenAddress without ARVADOS_SERVICE_INTERNAL_URL if InternalURLs has both http and https entries? With this config we wouldn't know whether to listen for plain http or https:

```
Keepstore:
  ListenAddress: "0.0.0.0:1234"
  InternalURLs:
    "http://host1:1234/": {}
    "https://host2:1234/": {}
```

#11 - 07/13/2021 09:21 PM - Tom Clegg

17667-listen-address @ 57f5d50c3315348e3b5be06b5f620532e850566b -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2577/

- If ListenAddress config is set ("addr:port"), bind to that addr:port (and choose http/https scheme indicated by InternalURLs entries, which must all use the same scheme).
- Otherwise, if ARVADOS_SERVICE_INTERNAL_URL is set in environment, bind to that addr:port.
- Otherwise, try binding each InternalURLs addr:port until one works.
- When a service needs to know its own InternalURL (e.g., interpreting AccessViaHosts):
  - If ARVADOS_SERVICE_INTERNAL_URL is set, use that
  - Otherwise, if ListenAddress is set, use that (even if it doesn't match any entry in InternalURLs) -- note this means if you want to use both AccessViaHosts and ListenAddress, you also need to set ARVADOS_SERVICE_INTERNAL_URL.
  - Otherwise, use the InternalURLs entry that worked

This only changes the lib/service module, which is used by controller, dispatchcloud, health, keep-balance, keepstore, and ws. We also need to update arv-git-http, keepproxy, and keep-web to use lib/service so they work the same way.

#12 - 07/21/2021 03:11 PM - Peter Amstutz

- Target version changed from 2021-07-21 sprint to 2021-08-04 sprint

#13 - 07/22/2021 07:49 PM - Ward Vandewege

Tom Clegg wrote:

> 17667-listen-address @ 57f5d50c3315348e3b5be06b5f620532e850566b -- https://ci.arvados.org/view/Developer/job/developer-run-tests/2577/
>
> - If ListenAddress config is set ("addr:port"), bind to that addr:port

Great.

> (and choose http/https scheme indicated by InternalURLs entries, which must all use the same scheme).

This feels complicated/brittle. Can we be more explicit? If we need to define a scheme, why not put it in ListenAddress? I can easily see a scenario where InternalURL would have a different scheme than what we want the service itself to do (e.g. SSL termination at something that sits in front of the service).

> - Otherwise, if ARVADOS_SERVICE_INTERNAL_URL is set in environment, bind to that addr:port.

This is for backwards compatibility only, right?

> - Otherwise, try binding each InternalURLs addr:port until one works.

This is for backwards compatibility only, right? And only necessary for Keepstore (or any future service where we have multiple instances with unique configuration)?

> - When a service needs to know its own InternalURL (e.g., interpreting AccessViaHosts):

So this is for Keepstores only, then?

> - If ARVADOS_SERVICE_INTERNAL_URL is set, use that
> - Otherwise, if ListenAddress is set, use that (even if it doesn't match any entry in InternalURLs) -- note this means if you want to use both AccessViaHosts and ListenAddress, you also need to set ARVADOS_SERVICE_INTERNAL_URL.
> - Otherwise, use the InternalURLs entry that worked

This feels very complicated. How much of this is because we don't have another way for a keepstore to discover which part of the config to use? Could we solve that problem differently and greatly simplify this logic?

> This only changes the lib/service module, which is used by controller, dispatchcloud, health, keep-balance, keepstore, and ws. We also need to update arv-git-http, keepproxy, and keep-web to use lib/service so they work the same way.

Let's talk a bit more about this. Is there a way to simplify things? I'm particularly confused about the utility of InternalURL (in the non-Keepstore case) after we introduce ListenAddress. Can we explicitly list the ways in which that variable would be used for each of our services?

It seems to me that InternalURL would be valuable if split DNS is not an option, but as we have learned with the whole nginx geo thing, it may not be so easy or practical for a service to determine if it needs to use the internal address vs the external address, in which case, ... does InternalURL serve a purpose anymore?

#14 - 07/29/2021 07:51 PM - Tom Clegg

Ward Vandewege wrote:

> Tom Clegg wrote:
>
> > (and choose http/https scheme indicated by InternalURLs entries, which must all use the same scheme).
>
> This feels complicated/brittle. Can we be more explicit? If we need to define a scheme, why not put it in ListenAddress? I can easily see a scenario where InternalURL would have a different scheme than what we want the service itself to do (e.g. SSL termination at something that sits in front of the service).

We could call it ListenURL: "http://localhost:1234" ...

> - Otherwise, if ARVADOS_SERVICE_INTERNAL_URL is set in environment, bind to that addr:port.
>
> This is for backwards compatibility only, right?

It makes it possible to run multiple keepstore processes on the same host that access different volumes, which we do in tests.

> - Otherwise, try binding each InternalURLs addr:port until one works.
>
> This is for backwards compatibility only, right? And only necessary for Keepstore (or any future service where we have multiple instances with unique configuration)?

It makes it possible to run keepstore out of the box (at least in setups where vhostnames resolve to interface addrs) without having to arrange custom env vars like "ARVADOS_SERVICE_INTERNAL_URL=http://this-vhost-name:1234" in each host's keepstore startup script.

It also makes it possible for each host to have all the programs installed but only run the ones that are actually used, like arvados-server-easy / arvados-server boot try to do. (If ListenAddress is set for a service, we need to run that service on all hosts because we don't know whether InternalURLs actually get routed to us.)

> - When a service needs to know its own InternalURL (e.g., interpreting AccessViaHosts):
>
> So this is for Keepstores only, then?

Yes.

> - If ARVADOS_SERVICE_INTERNAL_URL is set, use that
> - Otherwise, if ListenAddress is set, use that (even if it doesn't match any entry in InternalURLs) -- note this means if you want to use both AccessViaHosts and ListenAddress, you also need to set ARVADOS_SERVICE_INTERNAL_URL.
> - Otherwise, use the InternalURLs entry that worked
>
> This feels very complicated. How much of this is because we don't have another way for a keepstore to discover which part of the config to use? Could we solve that problem differently and greatly simplify this logic?

Yes, this is 100% about keepstore processes figuring out which AccessViaHosts entries they're supposed to use. (We might need something similar when we want to support running multiple dispatch processes, since they will also need to know which one is "me".)

Another possibility is for each service to try connecting to all the InternalURLs at startup and see which one loops back to itself -- but that could be a bit of a nightmare if there's an additional proxy involved that isn't necessarily configured/running yet while keepstore is starting up.

I guess it's complicated because we're trying to accommodate all possible setups (like multiple keepstores on same host) but also avoid extra configuration tasks in easy cases (one keepstore per host, hostnames resolve to network interface addresses).

Perhaps we should draft the admin-facing docs and see if it still seems complicated. Something like

- If you have DNS (or /etc/hosts) entries that resolve to your cluster hosts' network interface addresses (e.g., keep0 has IP addr 10.10.10.10 and keep0.zzzzz.example.com resolves to 10.10.10.10 on all cluster hosts), and you aren't trying to insert any gateways/proxies between Arvados components, just put an entry like http://keep0.zzzzz.example.com:25107/ in InternalURLs for each host where you run the service, and you're done.
- Otherwise, customize your setup:
  - InternalURLs lists all the URLs internal Arvados components can use to connect to instances of the service
  - ListenAddress (or ListenURL) specifies the addr:port the service process should listen on -- typically 0.0.0.0:port, or, if you're inserting your own gateways/proxies in front of the internal Arvados services, perhaps localhost:port or 0.0.0.0:differentport.

- If you use ListenAddress for keepstore, you must also set the ARVADOS_SERVICE_INTERNAL_URL env var to match one of the InternalURLs entries, so keepstore knows which AccessViaHosts entries to use.

  This only changes the lib/service module, which is used by controller, dispatchcloud, health, keep-balance, keepstore, and ws. We also need to update arv-git-http, keepproxy, and keep-web to use lib/service so they work the same way.

Let's talk a bit more about this. Is there a way to simplify things? I'm particularly confused about the utility of InternalURL (in the non-Keepstore case) after we introduce ListenAddress. Can we explicitly list the ways in which that variable would be used for each of our services?

InternalURLs tells Arvados server components how to connect to other Arvados server components:

- when keepproxy connects to keepstore, it connects to the addresses given in Services.Keepstore.InternalURLs.
- when controller connects to railsAPI, it connects to the address given in Services.RailsAPI.InternalURLs.
- when Nginx connects to keep-web, it connects to the address given in Services.WebDAV.InternalURLs (only in arvados-server boot, though -- the install guide doesn't have unified config for Nginx yet, it still instructs the operator to type the same information into an Nginx config file).

  It seems to me that InternalURL would be valuable if split DNS is not an option, but as we have learned with the whole nginx geo thing, it may not be so easy or practical for a service to determine if it needs to use the internal address vs the external address, in which case, ... does InternalURL serve a purpose anymore?

There are two different contexts with two different (and related) meanings for external/internal, which is confusing.

- InternalURLs / ExternalURL -- "internal" refers to internal communication between Arvados services, "external" refers to how clients connect to services
- Internal client / external client -- "internal" refers to clients that are able to bypass keepproxy and connect directly to keepstore InternalURLs, which we special-case to avoid having to scale keepproxy.

#15 - 08/04/2021 03:29 PM - Peter Amstutz

- Target version changed from 2021-08-04 sprint to 2021-08-18 sprint

#16 - 08/18/2021 03:02 PM - Peter Amstutz

- Target version changed from 2021-08-18 sprint to 2021-09-01 sprint

#17 - 09/01/2021 03:15 PM - Peter Amstutz

- Target version changed from 2021-09-01 sprint to 2021-09-15 sprint

#18 - 09/15/2021 03:03 PM - Peter Amstutz

- Target version changed from 2021-09-15 sprint to 2021-09-29 sprint

#19 - 09/28/2021 07:16 PM - Peter Amstutz

- Release set to 42

#20 - 09/29/2021 03:21 PM - Peter Amstutz

- Target version changed from 2021-09-29 sprint to 2021-10-13 sprint

#21 - 10/07/2021 03:42 PM - Peter Amstutz

- Release deleted (42)

#22 - 10/13/2021 03:48 PM - Peter Amstutz

- Target version changed from 2021-10-13 sprint to 2021-10-27 sprint