The Interface into Atomic App

Dusty Mabe
Software Container Engineer
2015-11-11

What is the goal of AtomicApp ?

- Main goal is to run nuleculized applications
- Most likely multi-container
  - Supports many different orchestration engines (providers)
- More features in the future?
  - Possibly
  - For now let's concentrate on today

Checkpoint

- Problems we are trying to solve
- Atomic CLI + Atomic App
- Pain Points
- 1st possibility of the day
  - Use AtomicApp software in a more direct manner (at least from the user perspective)
- 2nd possibility of the day
  - Don't embed AtomicApp software in Nuleculized applications.

Problems we are trying to solve: UX

- Atomic App upstream developers are running into decisions we need to make based on the user experience.
- These decisions need to improve user experience, but also incorporate Red Hat's strategy around container management.
- Who is the user?
  - Consumers of Nulecule Applications
  - Developers of Nulecule Applications
  - Developers of Atomic App software (upstream)

UX: Consumers of Nulecule Applications

- Consumers need to run/configure apps:
  - atomic run --opt3=`--answers=/path/to/answers.conf mariadb-app
- Do they care what is going on underneath?
  - Maybe, but probably not
- Priority as target audience for UX decisions:
  - High; need to make it easy to deploy/configure complex applications

UX: Developers of Nulecule Applications

- ISV developers need to:
  - Understand how the software works
  - Be able to develop/test/debug
  - Report bugs
  - `atomicapp install/run/stop/crazyverb helloapache`
- Do they care what is going on underneath?
  - Yes, but right now it's not easy to understand
- Priority as target audience for UX decisions:
  - High; need to make it easy to understand/develop nulecule/AtomicApps

UX: Developers of Atomic App software

- Atomic App developers need to:
  - Understand how the software works
  - Be able to develop/test/debug
  - Report/Fix Bugs
- Do they care what is going on underneath?
  - Absolutely, but right now it's not easy to understand
- Priority as target audience for UX decisions:
  - Low, but may affect community involvement

An example UX decision we've made

- Don't unpack files to or rely on cwd
  - Not obvious to user that files are going to get put in their current working directory when they run atomic run app
  - Fixed by now putting files into either
    - Specified destination
      - mkdir /tmp/fooapp &&
        atomic install --opt3=”--destination=/tmp/fooapp” <image>
    - Automatically generated destination
      - atomic install <image> --> files in /var/lib/atomicapp/$RANDOM
  - Has implications on usage via Atomic CLI
  - Since you always have to pass an image to Atomic CLI
    - To run “already installed/modified app”:
      - atomic run --opt3=”--destination=/tmp/fooapp” <image>
      - vs atomicapp run /tmp/fooapp

Trade one bad UX for Another

- Before – unexpected things happen to system
  - unpack files into and use cwd
    - atomic install helloapache
    - atomic run hellopache
  - Bad: puts the files in whatever directory you are in
  - Bad: If user already provided image name then why provide it again?
- After – more complex command line
  - unpack files to generated or user defined directory
    - atomic install --opt3=”--destination=/tmp/foo” helloapache
    - atomic run --opt3=”--destination=/tmp/foo” helloapache
  - Bad: confusing: what is opt3? why provide dest twice?
  - Still Bad: Why provide image name again?

Problems we are trying to solve: confusion

- Understanding what is going on behind the scenes is confusing.
  - Atomic App software is embedded in ISV containers
    - people get confused on why.. one ISV was basing all of their binary application containers on the Atomic App base container, rather than just having the single metadata container be based on it
  - Atomic CLI is used to execute docker run command against container provided by ISV
  - Atomic App is completely hidden
    - this is perceived as a benefit(app knows how to run itself)
    - but, really confuses people when they are first wrapping their heads around it

Checkpoint

- Problems we are trying to solve
- Atomic CLI + Atomic App
- Pain Points
- 1st possibility of the day
  - Use AtomicApp software in a more direct manner (at least from the user perspective)
- 2nd possibility of the day
  - Don't embed AtomicApp software in Nuleculized applications.

Where does Atomic CLI fit in?

- Atomic CLI
  - Used to execute run labels on container images
    - RUN/INSTALL/STOP/UNINSTALL labels
    - Ex: `atomic run cockpit/ws` works pretty well
- In this context:
  - Used as a gateway into running Atomic Apps
  - A lot of similarities in “verbage” with Atomic App

Should we be running AtomicApp through Atomic CLI?

- It depends
  - Is Atomic App simple enough (and will it stay simple enough) that the Atomic CLI workflow will be good enough
  - Are Atomic App and Atomic CLI similar enough that all of Atomic App's workflow can be modeled through Atomic CLI? Or are they too different?
    - Let's compare them and see

Comparison - INSTALL

- Atomic CLI Install
  - Run an install script (typically create systemd unit file to start a service on boot)
- Atomic App CLI Install
  - Pull down metadata from container and evaluate Nulecule to find and pull down all dependencies (Nested Nulecules)
  - Probably should be called “unpack”

Comparison - RUN

- Atomic CLI Run
  - Run a container via an image RUN label.
- AtomicApp CLI Run
  - Download metadata, evaluate Nulecule and deploy containers in chosen provider (kube,openshift,etc)

Comparison - STOP

Atomic CLI Stop

- Docs say that it will simply run STOP label from provided container image.
- Code actually expects the arg provided to be the name of a running container.
  - bug?

- Atomic App CLI Stop
  - Evaluate Nulecule and determine which deployed artifacts need to be stopped.

Are they that different?

- They are very similar but are also different
- Is it better to use AtomicApp software on it's own or to have Atomic be the universal interface into containerized applications?
  - Let's explore some pain points first

Checkpoint

- Problems we are trying to solve
- Atomic CLI + Atomic App
- Pain Points
- 1st possibility of the day
  - Use AtomicApp software in a more direct manner (at least from the user perspective)
- 2nd possibility of the day
  - Don't embed AtomicApp software in Nuleculized applications.

Pain Points – Labels

- Atomic CLI supports some execution labels (INSTALL,RUN,STOP,etc).
- Adding features in the form of new “verbs” is not optimal
  - We are somewhat limited in the labels that exist
  - Possible solution: generic execution labels in Atomic CLI
    - upstream is open to accept PR for this, something like:
      - atomic exe --label=UNPACK <image>

Pain Points – Passing options

- If you want to provide options to Atomic App via Atomic CLI you must work through the “OPT” variables in order to pull it off.
  - atomic run --opt3=”--provider=docker --answers=/tmp/answers.conf” helloapache
  - atomicapp run --provider=docker --answers=/tmp/answers.conf helloapache
- OPT* variables are odd for users
- AtomicApp itself could improve here by making options not be positional

Pain Points – Developer Workflow

- For developing in Atomic App upstream, to develop a change and test you must:
  - Edit Code
  - Build AtomicApp base container
  - Build app Nulecule container from base container
  - Use `atomic` to run/test it and repeat.
- This is a long process, leads to shortcuts:
  - Running Atomic App directly from source
  - Running Atomic App base container against Nulecule
- We can and will improve by enhancing our automated tests.

Checkpoint

- Problems we are trying to solve
- Atomic CLI + Atomic App
- Pain Points
- 1st possibility of the day
  - Use AtomicApp software in a more direct manner (at least from the user perspective)
- 2nd possibility of the day
  - Don't embed AtomicApp software in Nuleculized applications.

Possibility #1 - use Atomic App CLI more directly (install atomicapp via atomic install)

- With the pain points mentioned there are things we can do to make them better in the existing model, but is it worth considering using Atomic App more directly?
  - atomic install projectatomic/atomicapp
  - atomicapp install --destination=./ helloapache
  - cp answers.conf.sample answers.conf
  - sed -i s/kubernetes/docker/ answers.conf
  - atomicapp run ./
  - atomicapp stop ./

Possibility #1 - use Atomic App CLI more directly (via an alias, if `atomic` not installed)

- OR we could use an alias (NOT PRETTY):
  - alias atomicapp='docker run -it --rm --privileged -v $(pwd):/atomicapp -v /run:/run -v /:/host --net=host --name atomicapp projectatomic/atomicapp:${ATOMICAPPVERSION-latest}'
  - alias=`atomic run atomicapp`
  - ATOMICAPPVERSION=0.2.1
  - atomicapp install --destination=./ helloapache
  - cp answers.conf.sample answers.conf
  - sed -i s/kubernetes/docker/ answers.conf
  - atomicapp run ./
  - atomicapp stop ./

Possibility #1 - use Atomic App CLI more directly (via first class support in `atomic`)

- OR we could add first class support for Atomic App in Atomic CLI
  - atomic app install --destination=./ helloapache
    - app submodule will run atomicapp base container against helloapache image
  - cp answers.conf.sample answers.conf
  - sed -i s/kubernetes/docker/ answers.conf
  - atomic app run ./
  - atomic app stop ./
  - atomic app crazyverb ./

Possibility #1 - use Atomic App CLI more directly (comparison to today)

- For comparison this is how it looks with `atomic`
  - atomic install --opt3=”--destination=./” helloapache
  - cp answers.conf.sample answers.conf
  - sed -i s/kubernetes/docker/ answers.conf
  - atomic run --opt3=”--destination=./” helloapache
  - atomic stop --opt3=”--destination=./” helloapache
    - ^^^ THERE IS A BUG HERE IF WE WANT STOP TO WORK

Checkpoint

- Problems we are trying to solve
- Atomic CLI + Atomic App
- Pain Points
- 1st possibility of the day
  - Use AtomicApp software in a more direct manner (at least from the user perspective)
- 2nd possibility of the day
  - Don't embed AtomicApp software in Nuleculized applications.

Possibility #2 – don't embed in nulecule containers

- If we adopt a model of using Atomic App “directly” then we could stop embedding Atomic App in the metadata containers that are known as Atomic Apps today
  - No more “FROM projectatomic/atomicapp:0.2.1”

Possibility #2 – don't embed in nulecule containers

- **Benefits:**
  - The containers now just contain Nulecule files and artifacts
    - smaller footprint=less concern for security vulnerabilities
    - Don't have to rebuild these nulecule containers when a new version of atomicapp comes out.
  - The delivered container image is now just a standalone Nulecule. It is no longer an AtomicApp
    - This means that any implementation of the Nulecule specification can consume these containers and deploy a nuleculeized application.
    - If we come up with a go version of atomicapp we can consume the same ISV artifacts.

Possibility #2 – don't embed in nulecule containers

- **Benefits:**
  - I want to emphasize this point from previous slide:
    - ***“This means that any implementation of the Nulecule specification can consume these containers and deploy a nuleculeized application.”***
      - If we come up with a go version of atomicapp we can consume the same ISV artifacts.
      - If we come up with a libnulecule or libatomicapp library then openshift might be able to use that in the future.
      - A single Nulecule metadata container can be used by Atomic App base containers from RHEL, CentOS, Fedora, Ubuntu etc..

Possibility #2 – don't embed in nulecule containers

- **Benefits:**
  - The version of atomicapp that is running is no longer dependent on the upstream app developer.
    - In other words the Nulecules don't “rot”. If you have the latest version of atomicapp on your system then you will be using the latest version to launch all of your atomicapps.
    - People who are scripting the cli (like cockpit right now) don't have to worry about somebody throwing an old atomicapp at them with incompatible cli interface.
  - Less confusion – right now people often get confused about what is actually running

Possibility #2 – don't embed in nulecule containers

- **Negatives:**
  - The version of atomicapp that is running is no longer dependent on the upstream app developer.
    - Depends on what version of atomicapp is called by the user's system.
      - We can workaround this – 1 user can easily choose what version of atomicapp to run OR 2 – we can embed requested version information into the Nulecule containers and have atomicapp respect that.
    - Can no longer do `atomic run helloapache` :(

Summary

- The Atomic App user/developer experience can be improved.
- The architecture of how Nulecules/AtomicApps are built and executed often leads to confusion among developers
- How can we improve?
  - Fixing the way it is now?
  - Using Atomic App more directly?
  - Run Atomic App separately from Nulecules.
    - Nulecules are now independent of implementation

What can Atomic App project do better?

- We need to improve our communication within the platform as well as to our users.
- We need to work with Atomic CLI closer upstream to get changes in that will help our user's experiences.

Thoughts?