Arcade: An Interactive Science Platform in CANFAR

Brian Major, JJ Kavelaars, Sebastien Fabbro, Daniel Durand, and Helena Jeeves
CADC - National Research Council Canada, Victoria, B.C., Canada;
brian.major@nrc-cnrc.gc.ca

Abstract. For a number of years, CANFAR (Canadian Advanced Network for Astronomy Research) has offered virtual machines (VMs) as a way to do both interactive computing and batch processing in the cloud. A VM is a general and flexible base offering that can suit nearly any given astronomy compute project, but demands of users a thorough understanding of the intricacies of software installation and maintenance, and requires significant effort to achieve initial benefits. Arcade is an effort by the Canadian Astronomy Data Centre (CADC) to offer the CANFAR community an astronomy-focused, easier-to-use, and more intuitive science platform for conducting reproducible science.

1. Introduction

Arcade is composed of a number of pre-built, specialized application bundles (Docker\textsuperscript{1} containers) that appear in a graphical desktop environment. They run independently of each other in the cloud, allowing Arcade to optimize container execution to their particular resource requirements. Each container has user-specific access to a shared file system and other CANFAR cloud services, outfitting users with a variety of computing, analysis and storage tools.

At the CADC we see a lot of potential in Arcade and will be planning its evolution based on a number of key questions. How can scalability be best achieved? How can we reduce the burden of software and infrastructure maintenance for users and operators? Can we allow users to customize their Arcade experience? How can consoles be launched in batch processing? How can we best leverage open source technology and development from other projects? We shall discuss the core concepts of Arcade and explore its potential in respect to these questions and from feedback from the astronomy computing community.

2. Everything is a Container

Arcade is composed of a number of cooperating containers that expose an API (Application Programming Interface) to users and programs. These cooperating containers are called Arcade’s system containers. This API to Arcade allows for the creation and management of more containers: session containers and software containers.

\textsuperscript{1}https://www.docker.com/
Session containers are a hub for software containers. Currently, Arcade offers a single type of session container: a NoVNC\textsuperscript{2} graphical display container. Users connect to these containers with a browser and have the experience of a windowing desktop environment. No other software is available in the NoVNC session container. So, for users to perform tasks, they must launch software containers in the session container.

Software containers must be associated with a specific session container. For NoVNC sessions, the display of the software containers is shown on the NoVNC windowing environment. This makes the software containers appear as though they are running on the session container even though they are not. Any number of software containers can be associated with a session container, limited by the number of resources available in the entire Arcade platform.

Software containers are specialized to perform specific tasks. They are currently being built by the operators of Arcade ‘on demand’, but we envision supporting user contributed containers in the future. Some examples of software containers that have been built are:

- Containers for various versions of CASA (Common Astronomy Software Applications, National Radio Astronomical Observatory McMullin et al. 2007)
- SAOImage DS9 (Smithsonian Astrophysical Observatory\textsuperscript{3})
- Python software for interacting with CANFAR (VOSpace (Graham et al. 2018) client, TAP (Dowler et al. 2010) client, astropy (Price-Whelan et al. 2018))

2.1. Shared Home, Shared Scratch

Each session container and software container that is launched has some common characteristics, one of which is shared storage. The $\textit{HOME}$ directory is the same (per user) which allows containers to share application configuration information and small amounts of data. Also, a larger scratch directory is shared between containers. This directory is intended to be used for staging any temporary files required for processing. Whereas the $\textit{HOME}$ directory will be preserved indefinitely, the scratch directory may be cleared by the system at arbitrary times. These directories are not intended to store processing inputs or outputs. Those products should be stored in one of the CANFAR VOSpace implementations where data sharing capabilities exist.

2.2. Shared Credentials

Upon startup, the users’ credentials, in the form of X.509 (Boeyen et al. 2008) proxy certificates, are injected into the container. Any CADC or CANFAR command line tools will make use of these credentials when calling their respective services, thus giving users automatic and individualized authenticated access to those services. Proxy certificates are obtained through the private API of the IVOA Credential Delegation Protocol (Plante et al. 2010).

\textsuperscript{2}https://novnc.com/

\textsuperscript{3}http://ds9.si.edu/
3. The Future for Arcade

3.1. Scalability

Arcade is currently a prototype running on a single virtual machine. For it to be useful in a production sense (multiple users performing intensive data analysis) it needs to scale appropriately.

The first step in the scalability plan is to use Kubernetes\(^4\) as a platform for running all three levels of Arcade containers. This transition gives Arcade the ability to scale to the capacities defined in the Kubernetes installation and configuration.

The software containers have different resource utilization requirements. Some may run best with graphical processing units, some may require lots of memory. To satisfy these differences, Arcade can offer a variety of execution environments for the software containers with different resource profiles. When a software container is launched, Arcade can try to match the container to the most appropriate execution environment based on the container’s requirements and the current state of the Arcade system. Eventually, Arcade may detect when certain resource profiles are scarce and add more of them to the pool. In this way, users can have a heterogeneous set of software containers running with optimal resource profiles in a single session.

3.2. Custom Containers, Batch Execution

The provided set of software containers in Arcade could be thought of as a starting point for further development. Scientists would launch one (or more) of the software containers that most closely fitted the type of processing being performed. Changes or additions could be made to the container(s) to customize the work until the scientist is satisfied with the results. This is a step towards reproducible science as this container could be referenced in a publication. Thus, we would like to provide a way for users to save this customized version of the container.

If the work is meant to be parameterized and parallelized, the scientist would then like to launch many instances of the container over a set of input parameters. How is this batch execution invoked and defined? These questions will be explored through further prototyping Arcade to meet a core set of batch execution use cases.

References


\(^4\)http://kubernetes.io
Daniel Durand grabbing some extra ADASS spices after some extra encouragement from Elizabeth Warner (Photo: Peter Teuben)