Conduit : A Modern Pipeline for the Open Source World

Oliver Staeubli
Blue Sky Studios

Tim Hoff
Blue Sky Studios

Ryan Bland
Blue Sky Studios

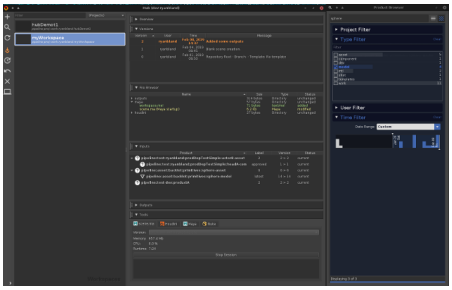
Rebecca Hallac
Blue Sky Studios

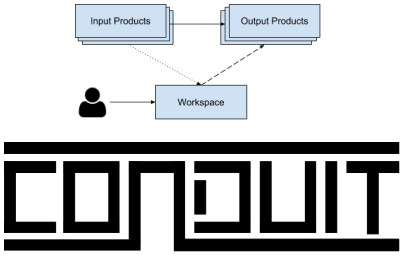
Josh Smeltzer
Blue Sky Studios

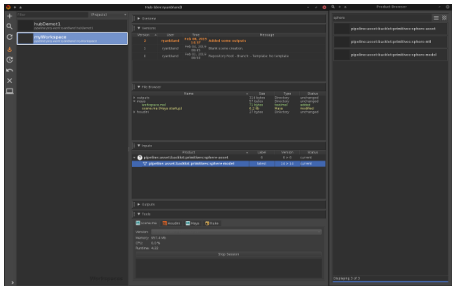
Chris Rydalch
Blue Sky Studios

Karyn Buczek Monschein
Blue Sky Studios

Mark McGuire
Blue Sky Studios







ABSTRACT

We present our modern pipeline, Conduit, developed for Blue Sky's upcoming feature film, Nimona. Conduit refers to a set of tools and web services that allow artists to find, track, version and quality control their work. In addition to describing the system and implementation, we will discuss the challenges and opportunities of developing and deploying a pipeline with the intention of open sourcing the resulting toolset. We found that communicating concepts and progress updates both internally and externally throughout the development process ultimately resulted in a more robust solution.

CCS CONCEPTS

• **General and reference** → **Design**; • **Social and professional topics** → Systems planning; • **Computing methodologies** → Graphics systems and interfaces; • **Software and its engineering** → *Collaboration in software development*.

KEYWORDS

animation, pipeline, microservices, open source

ACM Reference Format:
Oliver Staeubli, Tim Hoff, Ryan Bland, Rebecca Hallac, Josh Smeltzer, Chris Rydalch, Karyn Buczek Monschein, and Mark McGuire. 2019. Conduit : A Modern Pipeline for the Open Source World. In *Proceedings of SIGGRAPH '19 Talks.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3306307.3328175

SIGGRAPH '19 Talks, July 28 - August 01, 2019, Los Angeles, CA, USA

ACM ISBN 978-1-4503-6317-4/19/07.
https://doi.org/10.1145/3306307.3328175

1 INTRODUCTION

For thirty years, much of Blue Sky's proprietary software development was focused around supporting CGI Studio, our Academy Award winning proprietary renderer. As other renderers have matured, we needed a pipeline that was DCC and renderer agnostic. In 2014, Blue Sky began development on the foundation for a new pipeline. Four years later, we have deployed Conduit, a system that standardizes the way we edit, find, and version data generated by artists. In 2018, Blue Sky became a founding member of the Academy Software Foundation which supports and promotes the use of open source software. While the primary goal of the Conduit initiative was to modernize the Blue Sky pipeline, we made specific implementation decisions to ensure the viability of contributing Conduit back to the open source community. Mirroring a strategy employed by some of our colleagues in the tech industry, we regularly updated an external Blue Sky Tech Blog (https://medium.com/blue-sky-tech-blog) to openly discuss the design and deployment challenges of developing a pipeline and seek input from across the industry.

2 CONDUIT

2.1 Products and PRIs

Our first task was to determine how to uniquely identify the production data to be tracked in Conduit. We created a specification called the **Pipeline Resource Identifier (PRI)**. This human readable specification consisting of an entity, category, and version provides

a unique key to the various Conduit microservices and artist facing UIs. PRIs can refer to mutable versions using a label such as "latest" or "published". Or, PRIs can refer to immutable versions with an integer.

The enormous amount of metadata to be tracked required us to develop and deploy a horizontally scalable web services stack. While using an enterprise level commercial database had the advantage of support (for a fee), we chose to use available open source solutions to enable a path to open source. We chose Cassandra as our datastore and Elasticsearch for full-text search. To ensure consistency and shared functionality, we developed both JAVA and Python archetypes from which unique services could be derived. The archetypes provided each service with common interfaces into messaging through RabbitMQ. To ensure scalability, services are stateless. All of the core technologies Conduit is built upon are freely available as open source.

For initial deployment, we chose to track and version media generated by artists. This allowed our team to tune performance and scalability of the stack with minimal impact to production. Additionally, it allowed us to develop core Qt and Web components necessary to build artist friendly UIs. While these initial tools were primarily focused on providing access to media, they were designed with the intent of providing interfaces to all data in the pipeline as we continued deployment.

2.2 Workspaces and the Product File System (ProdFS)

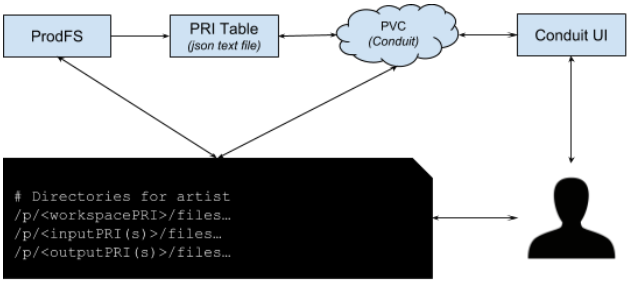


Figure 1: ProdFS communicates with the PRI Table to reduce the load on the datastore.

We define **products** as the unit of work, or set of files, being shared with other artists. For a proper pipeline, we needed a robust versioning system that allowed for product level granularity and dependency tracking. We developed the **Product Version Control (PVC)** system and accompanying Conduit services to meet these needs.

In the Conduit pipeline, an artist has three basic conceptual areas to consider: an artist does their work in a **workspace**, an artist declares dependencies as **input products**, and an artist declares what they want to share as **output products**. Workspaces were designed specifically as different entities to support multiple output products for multi-shot and multi-asset workflows. Within these workspaces and products, PVC manages file versioning, file deduplication, and merge utilities. We also track relationships between workspaces, input products, and output products to support path resolution and version cleanup.

A challenge for any pipeline versioning system is how to resolve paths to versioned files. While there are many approaches including programmatic resolvers such as USD's ArResolver class, we chose to do symlink management through a custom virtual file system we call the **Product Filesystem (ProdFS)**. Built on the open source Filesystem in Userspace (FUSE), ProdFS provides performant file resolution based on a specific **workspace context**. This allows artists both fine and coarse grain control over resolving upstream dependencies. PVC stores with each workspace the desired version for input products. In general, the artist simply chooses latest or published from within Conduit UI. Conduit UI then queries PVC/Conduit to determine the correct versions. PVC/Conduit records the resolved locations in a JSON file called the **PRI table**. ProdFS looks to the PRI table to determine what directories to present to the artist and how to symlink those directories appropriately.

2.3 Conduit UI

Any pipeline is only as useful as the presentation layer to the artists. We developed a Python/QT interface into Conduit, the Conduit UI, as the primary interface for creating and managing workspaces as well as launching applications. The Conduit UI allows artists to manage input and output products. A Conduit UI daemon runs on artist's workstations to handle environment bootstrapping for Conduit UI. Each of the key functional components of the Conduit UI are available as widgets that can be incorporated into various DCCs to provide in-app interaction with the pipeline.

Apart from managing their contributions within the pipeline, a primary goal for the Conduit UI is to encourage artists to version their work. Artists often shy away from version control, which tends to be developer-centric. Conduit UI simplifies the versioning process by managing products rather than multiple files. Debugging issues with assets or shots is simplified as the Conduit UI allows for Technical Directors to replicate an artist's workspace context.

3 CONCLUSION

Writing a new pipeline is far from unique. However, we established an initiative goal to both provide transparency to the industry during the development of Conduit and rely on other open source infrastructure components with the hopes of sharing the resulting toolset with the industry. This had several unforeseen tangible benefits. Regularly updating our industry colleagues ensured feedback and alternative perspectives from other studios with differing workflows. Lastly, encouraging the development team to provide regular updates to the industry throughout the process helped solidify the concepts and documentation for our internal artists.