U.S. DEPARTMENT OF ENERGY

BROOKHAVEN NATIONAL LABORATORY

Automation of the nsls2forge conda infrastructure

Thomas Hopkins, Undergraduate, Rensselaer Polytechnic Institute, Troy, NY 12180
Mentor: Maksim Rakitin, NSLS-II, Brookhaven National Laboratory, Upton, NY 11973

Abstract

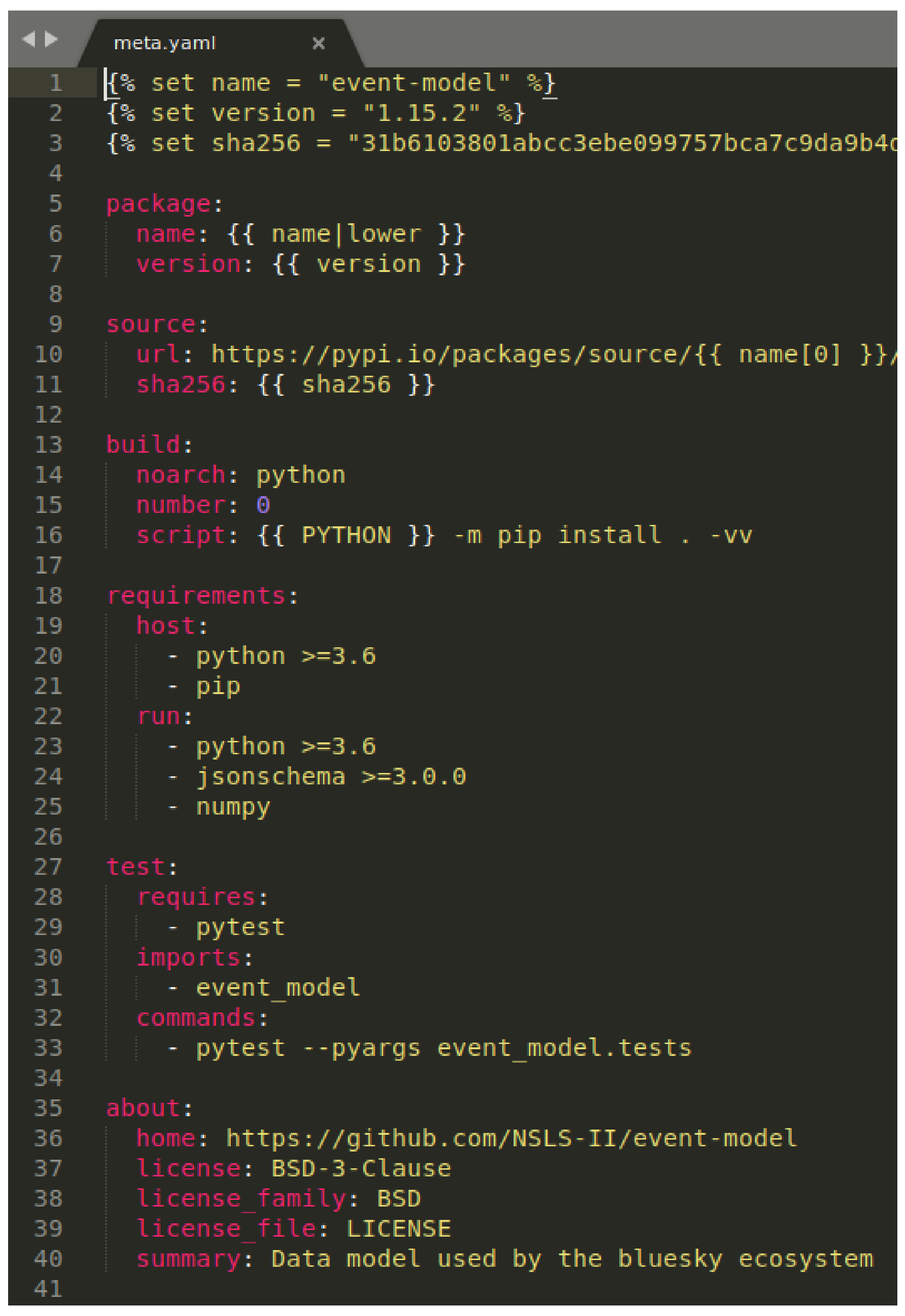
The nsls2forge conda infrastructure at the National Synchrotron Light Source II (NSLS-II) at Brookhaven National Laboratory requires large amounts of automation to be feasible in the long term. Through the use of emerging continuous integration technologies, we have created a bot that will automatically search for and update software packages as they release. This makes newly created or updated software packages created at NSLS-II readily available to install at any beamline workstation or personal computer. The approach we took includes the development of a large dependency graph, the automation of file migrations, and the employment of cloud computing for easy access. A dependency graph is a directed acyclic graph that contains a node for each software package with links highlighting the relationship between packages. This is essential for maintaining a hierarchy of packages which is used by the bot for correct and efficient updating. Migrations are used to change the metadata of a specific package by updating its configuration files. We make use of Microsoft Azure for hosting our bot that executes its job a few times a day. Conda is well known in the scientific Python community for providing software packages and environments. Existing open source software, by the conda-forge team, provided many key utilities and ideas for the implementation of our bot. By automating the processes of our conda infrastructure at the NSLS-II, we are providing up-to-date, tested, and packaged software at a significantly better rate and volume than before.

Introduction

Conda is an open-source, cross-platform, language-agnostic package manager and environment management system. NSLS-II uses Conda to deploy read-only production environments at the beamlines. Software packages that are developed and built at NSLS-II may be needed at many different beamlines and analysis servers. The Anaconda Cloud allows us to publish software packages so we can share our software not only with others at NSLS-II but also around the world.

Most packages that are uploaded to the Anaconda Cloud have a separate "feedstock" repository. Feedstock repositories allow for automatic testing of new versions of software packages on many different platforms prior to being published. This is done using continuous integration/continuous deployment (CI/CD) technologies such as Microsoft Azure and TravisCI. We publish our packages from feedstock repositories to the nsls2forge conda channel on the Anaconda Cloud.

Managing feedstock repositories at the nsls2forge requires a lot of manual work. This includes manually editing configuration files (Figure 1), rendering the feedstock using conda-smithy, submitting pull requests, and monitoring build statuses. All these steps are required for each of the 200+ feedstock repositories belonging to nsls2forge.



```
meta.yaml
{% set name = "event-model" %}
{% set version = "1.15.2" %}
{% set sha256 = "31b6103801abcc3ebe099757bca7c9da9b4c

package:
  name: {{ name|lower }}
  version: {{ version }}

source:
  url: https://pypi.io/packages/source/{{ name[0] }}/
  sha256: {{ sha256 }}

build:
  noarch: python
  number: 0
  script: {{ PYTHON }} -m pip install . -vv

requirements:
  host:
    - python >=3.6
    - pip
  run:
    - python >=3.6
    - jsonschema >=3.0.0
    - numpy

test:
  requires:
    - pytest
  imports:
    - event_model
  commands:
    - pytest --pyargs event_model.tests

about:
  home: https://github.com/NSLS-II/event-model
  license: BSD-3-Clause
  license_family: BSD
  license_file: LICENSE
  summary: Data model used by the bluesky ecosystem

```

Figure 1: `meta.yaml` configuration file for the `event-model-feedstock` repository. Found here https://github.com/nsls-ii-forge/event-model-feedstock/blob/master/recipe/meta.yaml

Software that we write at NSLS-II often *depends* on other software to build and function correctly. This can be seen most notably when importing different software modules when writing code. For example, writing some software in Python that imports functionality from the `numpy` package means that that software *depends* on `numpy`. This adds an interesting level of complexity to our problem of automation.

Methods

There are two main ways we chose to automate the nsls2forge conda infrastructure. The first is that we created a library of utilities that allows the user to gather important information and make changes to items in feedstock repositories. The second is a bot that automatically updates configuration files, re-renders, and submits pull requests for feedstocks at the nsls2forge.

We decided to create a suite of utilities that would not only help the end-user perform operations on feedstock repositories, but also be used by the bot to perform the entire process automatically. Other utilities, such as the dashboard (Figure 2) help maintainers see an overview of every feedstock's attributes. Figure 3 highlights the four main utilities we built.

When automatically updating software packages with new versions, a problem arises when considering the relationships between packages. To alleviate this problem, we implemented a directed acyclic graph (Figure 4) to represent every feedstock package at nsls2forge along with their dependencies. This method of representing the relationship between packages was developed and implemented by the conda-forge team for similar purposes [1, 2]. We will refer to this graph as the *dependency* graph.

Each node in the graph can be represented by a piece of software. This definition can range from programming languages to software applications and software packages. We store each feedstock along with its raw configurations at each node in the graph. Edges between nodes are defined as an "is a dependency of" relationship. For example, the package `event-model` imports some functionality from `numpy`, therefore, `numpy` is a *dependency* of `event-model`. A subgraph of the full dependency graph is shown in Figure 5, displaying only `event-model` and its related pieces of software. Sorting the nodes topologically gives us a correct order to update software packages so no dependency issues come up at build time.

Utility	Purpose
all-feedstocks	Performs operations on every feedstock belonging to a GitHub organization
dashboard	Creates a dashboard with build statuses, downloads, versions, and code health for every feedstock
graph-utils	Performs operations on the dependency graph (build, query, and update)
meta-utils	Performs operations on individual feedstock configuration files

Figure 3: Table of utilities built to automate operations at the nsls2forge. These are available via the command line interface. The code can be found here https://github.com/nsls-ii-forge/nsls2forge-utils.

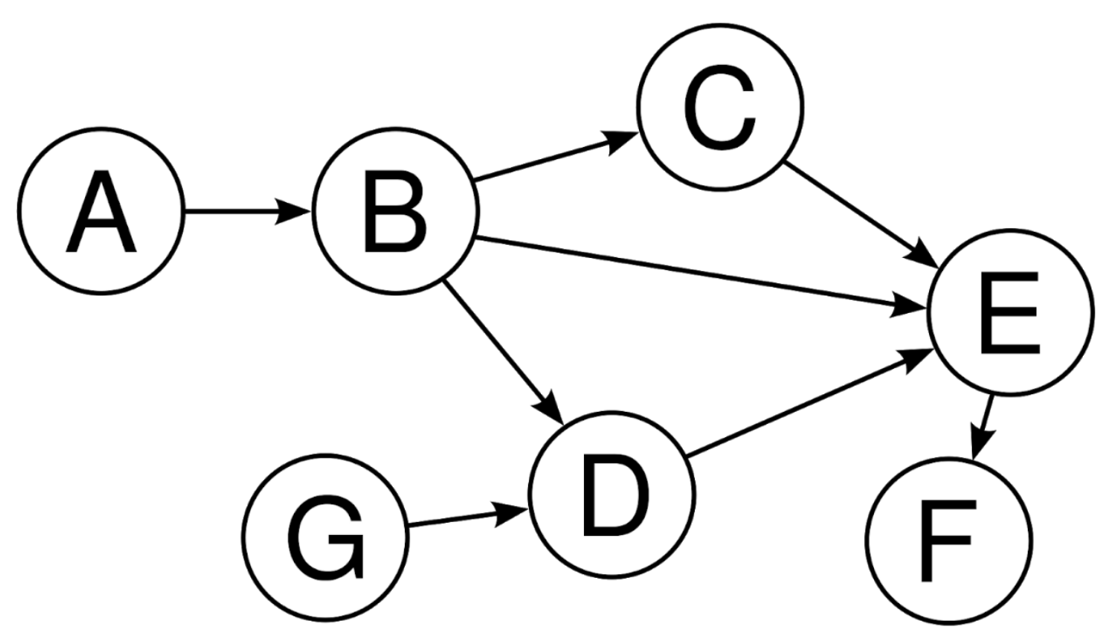
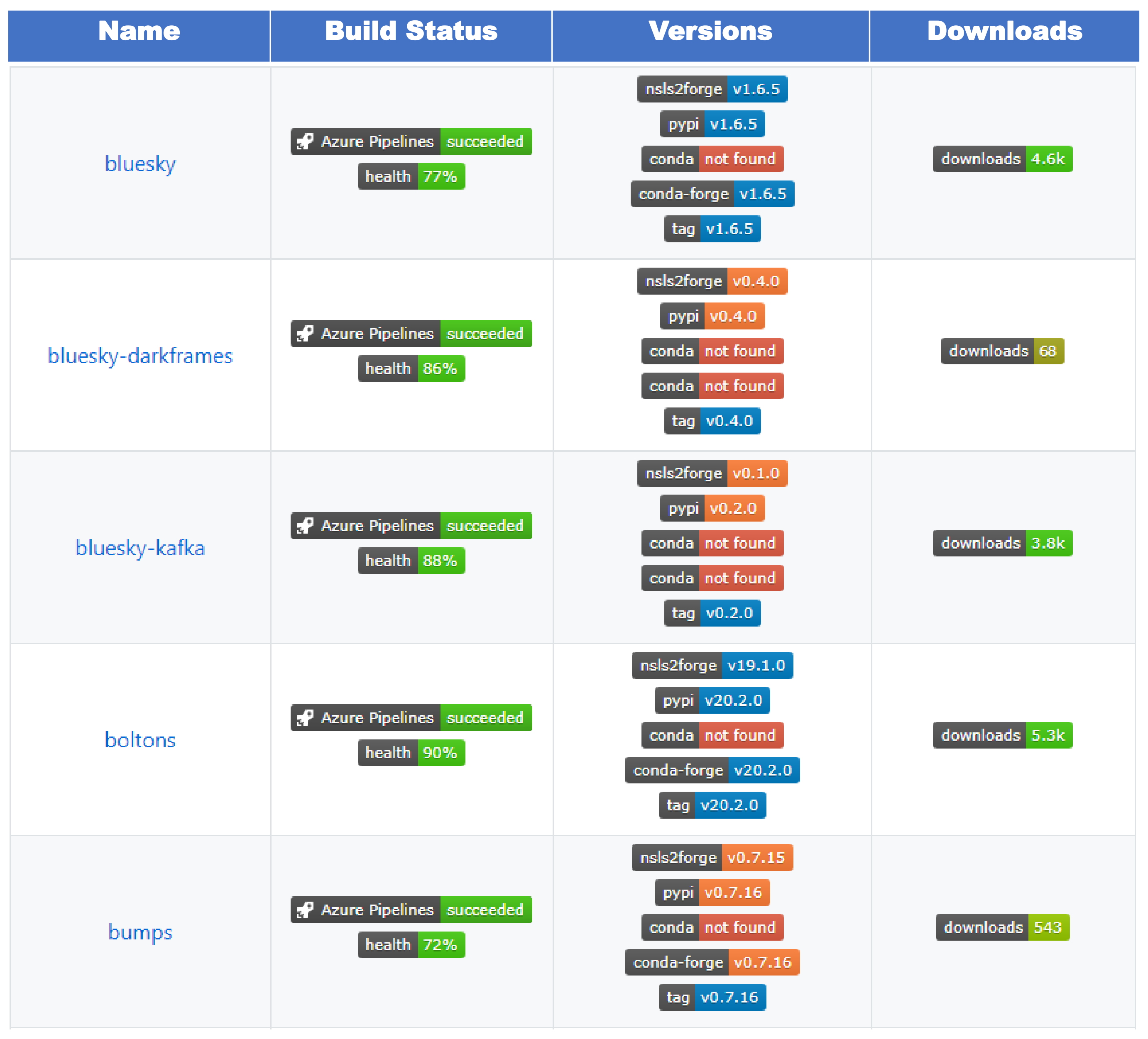


Figure 4: Directed Acyclic Graph (DAG). Useful for representing a complex set of relationships.



Name	Build Status	Versions	Downloads
bluesky	Azure Pipelines succeeded health 77%	nsls2forge v1.6.5 pypi v1.6.5 conda not found conda-forge v1.6.5 tag v1.6.5	downloads 4.6k
bluesky-darkframes	Azure Pipelines succeeded health 86%	nsls2forge v0.4.0 pypi v0.4.0 conda not found conda not found tag v0.4.0	downloads 68
bluesky-kafka	Azure Pipelines succeeded health 88%	nsls2forge v0.1.0 pypi v0.2.0 conda not found conda not found tag v0.2.0	downloads 3.8k
boltons	Azure Pipelines succeeded health 90%	nsls2forge v19.1.0 pypi v20.2.0 conda not found conda-forge v20.2.0 tag v20.2.0	downloads 5.3k
bumps	Azure Pipelines succeeded health 72%	nsls2forge v0.7.15 pypi v0.7.16 conda not found conda-forge v0.7.16 tag v0.7.16	downloads 543

Figure 2: Part of the dashboard for feedstock repositories at nsls2forge. Lists build statuses, code health, versions, and number of downloads for every feedstock. Full dashboard can be found here https://github.com/nsls-ii-forge/project-management.

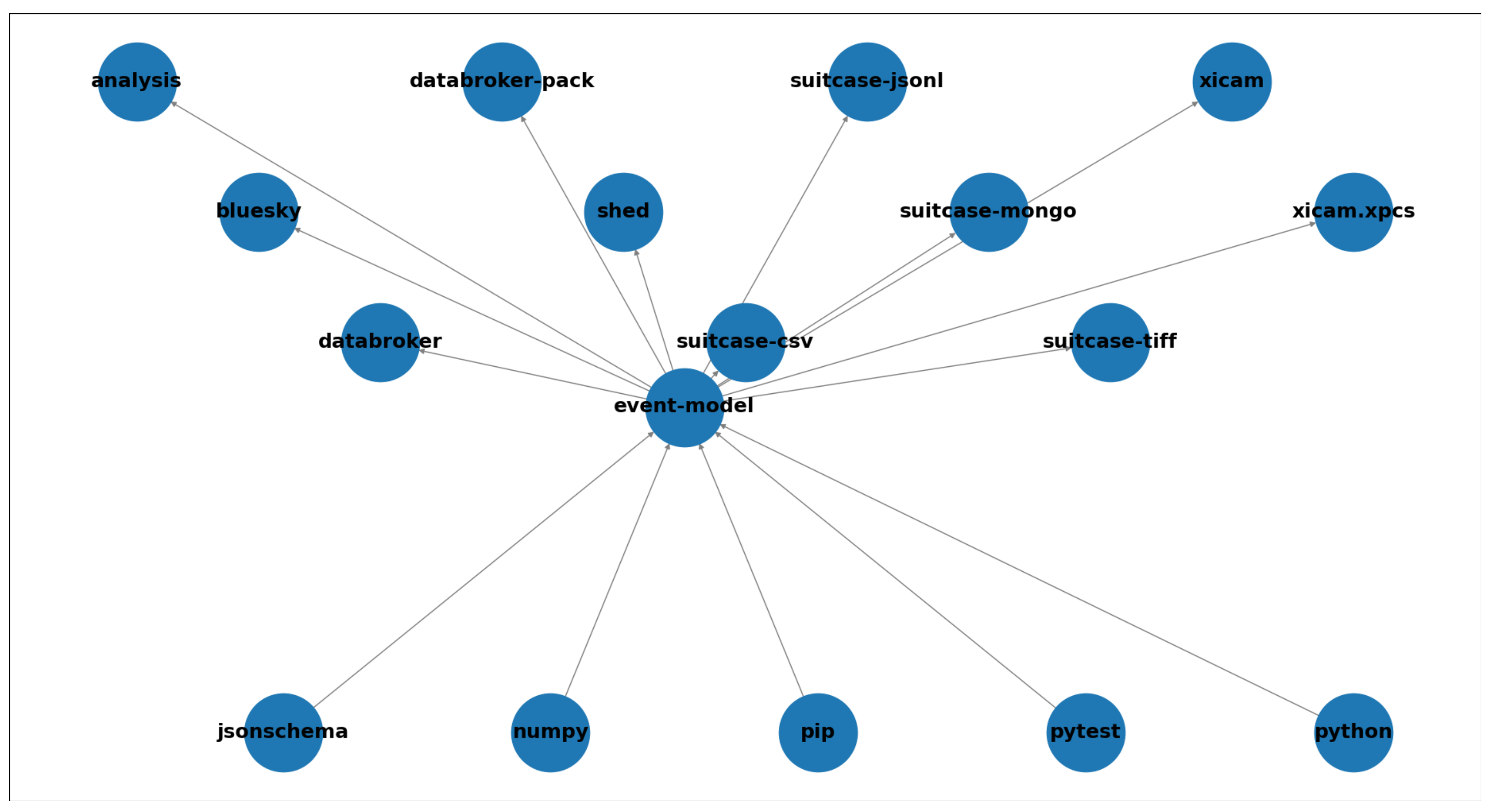
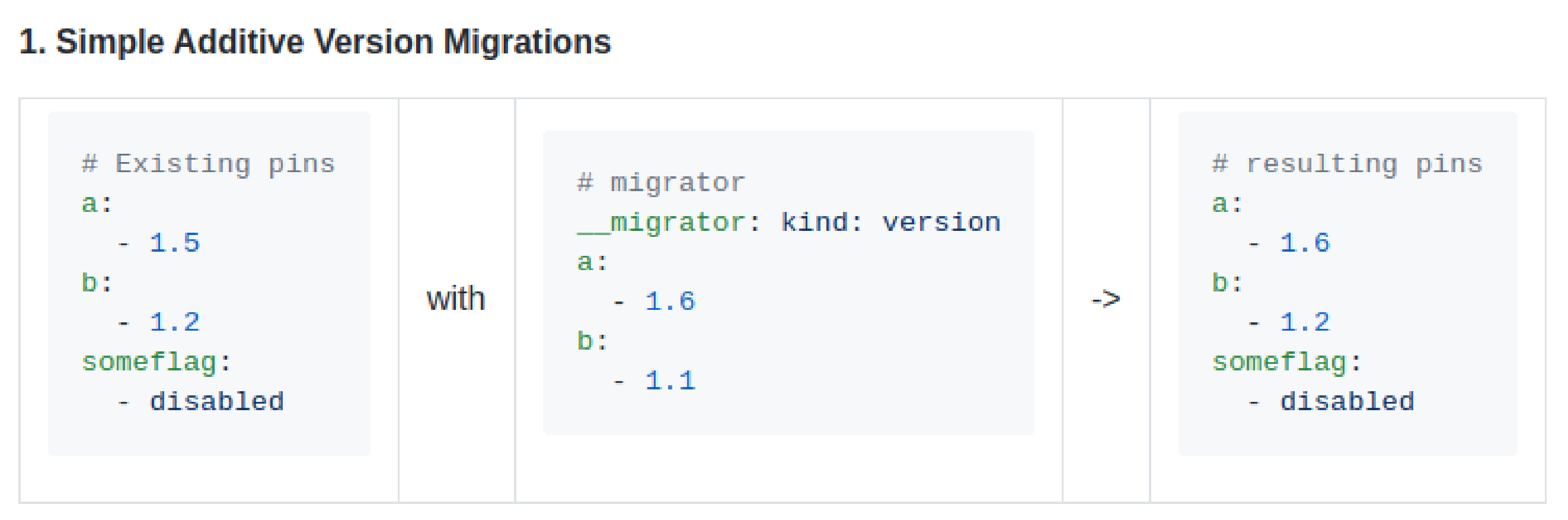


Figure 5: A subgraph of our much larger dependency graph. This subgraph shows only the software associated with the `event-model` Python package built at nsls2forge. Nodes (in blue) along the bottom are packages that `event-model` depends on to run/build. Nodes along the top are packages that require `event-model` to run/build. This structure for representing package dependencies was developed by the conda-forge team [1, 2].

The bot itself will run in the cloud and use the various prebuilt utilities to automatically update software package feedstock with new versions as they release. It will go through a cycle of retrieving all feedstock names from nsls2forge, creating/updating the dependency graph, retrieving new versions of packages directly from their sources, editing configuration files, and finally submitting pull requests to feedstock repositories on GitHub.

To edit configuration files with new versions, we use file migrations (Figure 6). These migrations were first developed by the conda-forge team for their bot [1, 2]. They allow us to keep a record of all changes that are made by the bot and update specific portions of configuration files.



1. Simple Additive Version Migrations

```
# Existing pins
a:
  - 1.5
b:
  - 1.2
someflag:
  - disabled
```

with

```
# migrator
__migrator: kind: version
a:
  - 1.6
b:
  - 1.1
```

->

```
# resulting pins
a:
  - 1.6
b:
  - 1.2
someflag:
  - disabled
```

Figure 6: Simple version migrator used to update configuration YAML files. Original file on the left, migrator in the middle, and resulting file on the right. Tags are only updated if the version number is greater than the original version number.

Conclusion

We have shown that the current process for maintaining feedstock packages at the nsls2forge is unfeasible in the long term. Automation of the nsls2forge's operation helps alleviate most of the manual work in the maintenance process. The library of utilities helps not only the bot but also any user or maintainer work more efficiently in diagnosing issues and performing updates. The dependency graph is a useful representation for recognizing how packages relate to each other and for making decisions when updating packages. The bot provides an end-to-end tool that will update versions of packages automatically. While the changes made by the bot will still need to be reviewed manually, it improves the speed and volume at which packages can be updated and tested at nsls2forge.

Some future improvements to the utilities and bot are to extend the functionality to update more than just version numbers of packages, retrieve more complex information from the dependency graph, perform file migrations across all feedstock repositories at once, and create an interactive version of the bot that responds to requests for changes in feedstock repositories.

Acknowledgements

I wish to thank my mentor, Dr. Maksim Rakitin, for his professionalism and generosity during the SULI program. Further, I wish to express my gratitude to the conda-forge and regro teams for providing many key utilities and ideas for the implementation of our bot. This project was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program (SULI).

References

[1] Wright, C. 2020. The Automation of Conda-Forge. AnacondaCON 2020. https://gateway.on24.com/wcc/eh/2336732/lp/2357492/the-automation-of-conda-forge/?campaign=web

[2] regro/cf-scripts GitHub repository. Updated 2020. https://github.com/regro/cf-scripts

Hopkins, Thomas