Proposal 1: Better WebAssembly support for conda-forge.

Abstract:

Ask: $100,000

WebAssembly is playing an increasingly important role in accessible data science education. Using – for example Jupyterlite or related tools – it is very simple to get started with a computational environment right in the browser, without the explicit need to install any third party libraries from the internet. The benefits are fast startup, decent speed and excellent sandboxing of potentially untrusted code.

We are currently prototyping ways to deliver binary packages compiled to WebAssembly as conda packages. These packages are derived from conda-forge recipes and fully interoperable with each other.

Our vision for conda-forge WebAssembly is that an educator can prepare an environment that contains a given set of packages (such as Python, NumPy and matplotlib) locally, and upload that “prefix” to a static file hosting service from which it can be used by students. Locally, the package resolution will be done using the mamba or conda package managers.

Another important use case is interactive documentation. The QuantStack team has already helped in putting an interactive NumPy experience on the numpy.org website.

We want to make these WebAssembly use cases more accessible to other open source projects which is possible with the packaging infrastructure provided by the conda-forge project.

Proposal 2: A Quetz GUI for powerful distro inspection, and a OCI

Ask: $200,000

We want conda-forge to have its own package server interface based on the open source quetz server and backed by public GitHub packages.

The “anaconda.org” experience that is currently the only way to view the conda-forge packages is outdated and slow. We propose to create a new open-source conda-forge repository interface that unifies the package repository experience with the current conda-forge.org/status website. This will be done as a plugin to the open source quetz package server.

The conda-forge.org website shows information related to the conda-forge project. It was done mainly on volunteer time. The most important feature for package maintainers is the “status” page that shows information about the currently running package migrations and package updates.

Since conda-forge nowadays has thousands (at the time of writing 18k) “feedstocks” – which map to GitHub repositories – keeping an overview has become impossible. To remedy this,

we would like to create a dedicated distribution-management view where we can show things like outdated feedstocks (a newer version of a given package is available), have quick links to the Azure CI builds (to find logs and errors quickly) or show where and why migrations have stalled or failed.

Another area where we would like to invest more work is better search. The only way of searching packages is either anaconda.org which is quite slow or GitHub which is unwieldy. It is impossible to search package contents or recipes on anaconda.org. Package maintainers have workarounds, such as the GitHub search for recipes or searching in specific (gigantic) GitHub indexing repository for package contents. We think that we can improve the experience with a dedicated search UI in Quetz. This new UI will allow users to search in package contents and recipes easily.

OCI registries are a stable, well specified interface to store arbitrary “tagged” blobs of data and are used in many different “packaging” projects such as Docker, Helm Charts, Homebrew, Spack and others. Most importantly, many cloud vendors allow the creation of an OCI registry, such as AWS or Microsoft Azure. That makes it a vendor neutral technology. It is also simple to run one's own OCI registry using the open-source Docker registry. GitHub, which is already a central component of conda-forge (as it hosts all “feedstocks”) has very good support for hosting OCI packages, and storage is free for public packages.
As such we would like to create a full conda-forge mirror on GitHub packages and use it as the storage backend to the Quetz instance.

Proposal 3: Improve conda-forge provisioning infrastructure
Ask: XXXX

Conda-forge depends upon a large amount of infrastructure built on top of a number of Github repositories, external CI services (Azure DevOps, Github Actions, TravisCI, Drone.io, CircleCI), Heroku “dynos” and AWS instances. This situation has resulted in an array of bot accounts, API keys, and bespoke configuration settings for each service. The configuration details and provisioning steps of this infrastructure need to be documented, tested and centralized into a service like Terraform to enable better security, reliability, and recovery from adverse events.