Building Conda Packages And Releases

LCLS1

List of commands to build psana1 release on s3df

Login and set environment

> s3dflogin

ssh psana -l psreldev

source /sdf/group/lcls/ds/ana/sw/conda1-v3/manage/bin/psconda.sh # conda1-v3 is a latest version of conda 23.10.0
conda deactivate
conda activate conda_build

Generate .tar.gz file with source code

cd /sdf/group/lcls/ds/ana/sw/conda1-v3/manage/ # if needed git clone git@github.com:slaclab/anarel-manage.git manage

bin/ana-rel-admin --force --cmd psana-conda-src --name 4.0.58 --basedir `pwd`

Build release

cd ~psreldev/git/psana1-feedstock/   # if needed git clone git@github.com:slac-lcls/psana1-feedstock.git

Update in recipe/meta.yaml fields for set version and sha256

conda build -c lcls-i -c conda-forge recipe

Debugging

In case of problem with tests look at log file like

/sdf/group/lcls/ds/ana/sw/conda1/inst/envs/conda_build2/conda-bld/psana_<13-digit-build number>/test_tmp/work/<log-file-name>

Upload .tar.bz2 file with release to anaconda lcls-i channel

The name of the .tar.bz2 file can be found at the end of conda build response. Then use command like

anaconda upload -u lcls-i /sdf/group/lcls/ds/ana/sw/conda1/inst/envs/conda_build2/conda-bld/linux-64/psana-4.0.57-py39hed0727e_1.tar.bz2

response on anaconda upload

```
(conda_build2) [psreldev@sdfiana001 psana1-feedstock]$ pwd
/sdf/home/p/psreldev/git/psana1-feedstock
(conda_build2) [psreldev@sdfiana001 psana1-feedstock]$ anaconda upload -u lcls-i /sdf/group/lcls/ds/ana/sw
/conda1/inst/envs/conda_build2/conda-bld/linux-64/psana-4.0.57-py39hed0727e_1.tar.bz2
Using Anaconda API: https://api.anaconda.org
Using "lcls-i" as upload username
Processing "/sdf/group/lcls/ds/ana/sw/conda1/inst/envs/conda_build2/conda-bld/linux-64/psana-4.0.57-
py39hed0727e_1.tar.bz2"
Detecting file type...
File type is "Conda"
Extracting conda attributes for upload
Creating package "psana"
Creating release "4.0.57"
The action you are performing requires authentication, please sign in:
Using Anaconda API: https://api.anaconda.org
Username: dubrovin
dubrovin's Password:
login successful
Using Anaconda API: https://api.anaconda.org
Using "lcls-i" as upload username
Processing "/sdf/group/lcls/ds/ana/sw/conda1/inst/envs/conda_build2/conda-bld/linux-64/psana-4.0.57-
py39hed0727e_1.tar.bz2"
Detecting file type...
File type is "Conda"
Extracting conda attributes for upload
Creating package "psana"
Creating release "4.0.57"
Uploading file "lcls-i/psana/4.0.57/linux-64/psana-4.0.57-py39hed0727e_1.tar.bz2"
15.4MB [00:01, 12.6MB
/s]

Upload complete

conda located at:
  https://anaconda.org/lcls-i/psana

(conda_build2) [psreldev@sdfiana001 psana1-feedstock]$
```

Create new environment

conda create --name ana-4.0.58-py3 --clone ana-4.0.57-py3

response on conda create

```
(base) [psreldev@sdfiana002 psana1-feedstock]$ conda create --name ana-4.0.58-py3 --clone ana-4.0.57-py3
Retrieving notices: ...working... done
Source:      /sdf/group/lcls/ds/ana/sw/conda1/inst/envs/ana-4.0.57-py3
Destination: /sdf/group/lcls/ds/ana/sw/conda1/inst/envs/ana-4.0.58-py3
Packages: 488
Files: 11

Downloading and Extracting Packages:

Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: /
For Linux 64, Open MPI is built with CUDA awareness but this support is disabled by default.
To enable it, please set the environment variable OMPI_MCA_opal_cuda_support=true before
launching your MPI processes. Equivalently, you can set the MCA parameter in the command line:
mpiexec --mca opal_cuda_support 1 ...

In addition, the UCX support is also built but disabled by default.
To enable it, first install UCX (conda install -c conda-forge ucx). Then, set the environment
variables OMPI_MCA_pml="ucx" OMPI_MCA_osc="ucx" before launching your MPI processes.
Equivalently, you can set the MCA parameters in the command line:
mpiexec --mca pml ucx --mca osc ucx ...
Note that you might also need to set UCX_MEMTYPE_CACHE=n for CUDA awareness via UCX.
Please consult UCX's documentation for detail.

done
#
# To activate this environment, use
#
#     $ conda activate ana-4.0.58-py3
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) [psreldev@sdfiana002 psana1-feedstock]$
```

conda deactivate
conda activate ana-4.0.58-py3
conda install -c lcls-i -c conda-forge psana=4.0.58

response on conda install

```
(ana-4.0.58-py3) [psreldev@sdfiana002 psana1-feedstock]$ conda install -c lcls-i -c conda-forge psana=4.0.58
Channels:
 - lcls-i
 - conda-forge
 - defaults
 - lcls-ii
 - cogsci
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /sdf/group/lcls/ds/ana/sw/conda1/inst/envs/ana-4.0.58-py3

  added / updated specs:
    - psana=4.0.58


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    psana-4.0.58               |   py39hed0727e_1        15.4 MB  lcls-i
    ------------------------------------------------------------
                                           Total:        15.4 MB

The following packages will be UPDATED:

  psana                              4.0.57-py39hed0727e_1 --> 4.0.58-py39hed0727e_1


Proceed ([y]/n)? y


Downloading and Extracting Packages:


Preparing transaction: done
Verifying transaction: done
Executing transaction: done
(ana-4.0.58-py3) [psreldev@sdfiana002 psana1-feedstock]$
```

Deprecated New Pinned-Approach

NOTE: (Oct. 10, 2023) we are renaming the .condarc to .condarc_dontuse files in home-directories and directories like /cds/sw/ds/ana/conda2-v2/inst/. condarc to not implicitly depend on those. So channels must be specified explicitly in the conda commands.

To create a source-code .tar.gz file:

ssh psbuild-rhel7-01 -l psreldev

Current as of Oct, 9, 2023

Valerio writes: It's not different from the psana2 approach at all. First you need to package the source in a tar.gz like we always did, as shown in the "Checklist for building psana1 environments from feedstocks" section below:

- source /cds/sw/ds/ana/conda1/inst/etc/profile.d/conda.sh
- conda activate conda_build
- cd /cds/sw/ds/ana/conda1/manage/
- bin/ana-rel-admin --force --cmd psana-conda-src --name 4.0.53 --basedir `pwd` (this command assembles all the source code from the tags into a .tar.gz file)

(no longer necessary since we don't build with GitHub-actions at the moment): ~~cp .tar.gz (from manage/downloads/anarel/) to where GitHub can see it: /reg/g/psdm/web/swdoc/tutorials/.~~ Instead, put this sort of line in recipe/meta.yaml: "url: file:///cds/sw/ds/ana/conda1/manage/downloads/anarel/psana-conda-4.0.56.tar.gz"

All of the next steps to build the binary .tar.bz1 conda package should be done as user "cpo" or "valmar" or "dubrovin" (make sure you have enough disk space: I believe the build will go in ~/.conda and ~/conda-bld).

checkout github feedstock repository:
many feedstocks (all in slac-lcls) e.g.
https://github.com/slac-lcls/psana1-feedstock

update version and checksum in

https://github.com/slac-lcls/psana1-feedstock/blob/main/recipe/meta.yaml

generate new checksum with: sha256sum .tar.gz like this:

sha256sum /reg/g/psdm/web/swdoc/tutorials/psana-conda-4.0.10.tar.gz

the 3 important files in the psana1-feedstock git repo are: conda_build_config.yaml (specifies the versions of all package dependencies as recommended by condo) recipe/meta.yaml (specifies location of source code and psana build and run-time dependencies) recipe/build.sh (the actual scons build instructions).

NOTE: cpo thinks the psana1 recipe needs to be built with the python3 conda_build env from **LCLS2** (source /cds/sw/ds/ana/conda2/manage/bin/psconda.sh). If you don't do this I think we get this error from "conda build recipe/" since conda_build env in LCLS1 is py2. built the recipes under your own username. Use these commands to get the build environment (it is LCLS2!):

source /cds/sw/ds/ana/conda2/manage/bin/psconda.sh
conda deactivate
conda activate conda_build

The py2 and py3 psana recipes have now the pinning file (conda_build_config.yaml) in the top level directories. You just need to update the meta.yaml file, run "`conda build recipe/". **NOTE: need to do this above the recipe/ directory, to pick up the conda_build_config.yaml` pinnings.` (did this as user "cpo", but could also do as psreldev, although psreldev might need "anaconda login" as another user so "anaconda upload" will work) upload and build the environment from the "pinned" YAML file.

The py3 recipe is in https://github.com/slac-lcls/psana1-feedstock while the py2 recipe is in https://github.com/slac-lcls/psana1-py2-feedstock.

Build the source code in the psana1-feedstock directory with the command "conda build recipe/" (effectively executes recipe/build.sh which has scons commands in it)

```
TypeError: apply_pin_expressions() argument after ** must be a mapping, not unicode
```

Upload the file to anaconda lcls-i channel with:

anaconda upload -u lcls-i /cds/home/c/cpo/conda-bld/linux-64/psana-4.0.53-py39hb869b97_2.tar.bz2

The yaml files for creating the environment are in /cds/sw/ds/ana/conda1/manage/jenkins/.

To create the env's without changing the pinnings, clone the previous environment and install the newly upload psana version like this:

```
conda install --experimental-solver=libmamba -c lcls-i -c conda-forge psana=4.0.53
```

To create new pinnings (which may require also changing conda_build_config.yaml in the py2/py3 recipe repos) you can create two env's like this:

```
source /reg/g/psdm/etc/psconda.sh -v2 (v2 to get experimental mamba solver)
conda env create --name ana-4.0.44-py3 --experimental-solver=libmamba --file=/cds/sw/ds/ana/conda1/manage
/jenkins/ana-env-py3.yaml
conda env create --name ana-4.0.44 --experimental-solver=libmamba --file=/cds/sw/ds/ana/conda1/manage/jenkins
/ana-env-py2.yaml
```

Deprecated Instructions

Information from Valerio on Jan. 27, 2021.

NOTE: the feedstock approach is currently deprecated because conda forge moved too fast for us and there were often problems in the release. See "Additions for Pinned Approach" section below for modifications for LCLS1 "pinned approach".

(see also checklist below for which env to activate and how to access ana-rel-admin)

cd /cds/sw/ds/ana/conda1/manage/
bin/ana-rel-admin --force --cmd psana-conda-src --name 4.0.52 --basedir `pwd`

cp .tar.gz (from manage/downloads/anarel/) to where GitHub can see it:
/reg/g/psdm/web/swdoc/tutorials/

psreldev can't write to the above, do as user "cpo" or "valmar"

checkout github feedstock repository:
many feedstocks (all in slac-lcls) e.g.
https://github.com/slac-lcls/psana1-feedstock

update version and checksum in

https://github.com/slac-lcls/psana1-feedstock/blob/main/recipe/meta.yaml

generate new checksum with: sha256sum .tar.gz

sha256sum /reg/g/psdm/web/swdoc/tutorials/psana-conda-4.0.10.tar.gz

git push launches the build on GitHub (using GitHub Actions) when it finishes it uploads it
to anaconda.org/lcls-i

this is our real control of the build process (e.g. the GitHub Actions Workflow yaml file gets
generated automatically)

https://github.com/slac-lcls/psana1-feedstock/blob/main/conda-forge.yml

if the above file is modified have to run "conda smithy rerender -c
auto" "-c auto" means automatically make a git commit. but still have
to push to git.

There are a few other cases in which the feedstock needs to be rerendered.
See the conda-forge documentation:

https://conda-forge.org/docs/maintainer/updating_pkgs.html#when-to-rerender

If no relevant changes have taken place, the command will simply not create a
commit. so maybe it is a good practice to run it every time a feedstock is
updated.

"targets" first entry: the conda channel to upload to, second entry is
called a "label" (e.g. a devel and a stable). conda forge always looks
for "main" label.

then "conda create" as usual (as seen in the jenkins CI).

for testing the feedstock, can run the run_docker_build.sh script locally
but requires docker to be installed. The file "build_locally.py" included in
the feedstock takes care of running run_docker_build.sh

Valerio uses this to test the feedstock on his laptop.

If any dependency is updated (e.g. boost) then feedstocks are rebuilt
automatically, but ONLY if our packages are in conda-forge, which they
are not. To emulate this, Valerio bumps all the feedstock build
numbers for each release (even ndarray because it depends on boost) so
they will get built against the latest versions in conda-forge.

separate feedstock for py27 since everything is pinned in the recipe:
https://github.com/slac-lcls/psana1-py2-feedstock
recipe/conda_build_config.yaml overrides the equivalent in the conda forge
pinning file (most recent version pulled automatically from conda-forge)

in recipe/conda_build_config.yaml (ONLY for psana1-py2-feedstock),
zip_keys: override specification for the 3 special packages python,
python_impl, numpy. conda-forge people helped specify these, in
particular Billy Poon.

feedstocks:
libpressio-feedstock psana1-py2-feedstock sz-feedstock
ndarray-psana-feedstock psgeom-feedstock ztcav-py2-feedstock xtcav2-feedstock
ndarray-psana-py2-feedstock psocake-py2-feedstock psocake-feedstock
psana1-feedstock stdcompat-feedstock

libpressio,sz,stdcompat are all needed by chuck/sz.

update build numbers for all the above for every psana1 release.

Deprecated: Checklist for building psana1 environments from feedstocks

As psreldev on psbuld-rhel7:

- source /cds/sw/ds/ana/conda1/inst/etc/profile.d/conda.sh
- conda activate conda_build
- cd /cds/sw/ds/ana/conda1/manage/
- bin/ana-rel-admin --force --cmd psana-conda-src --name 4.0.11 --basedir `pwd`

As normal user:

- cd /cds/sw/ds/ana/conda1/manage/downloads/anarel
- cp psana-conda-4.0.11.tar.gz /reg/g/psdm/web/swdoc/tutorials
- sha256sum psana-conda-4.0.11.tar.gz <copy the checksum>

Activate an environment that contains the 'conda smithy' package. For example:

```
conda activate /cds/home/v/valmar/.conda/envs/conda_forge_build3.9
```

Then for each of these feedstock (more or less in this order):

github.com/slac-lcls/sz-feedstock
github.com/slac-lcls/stdcompat-feedstock
github.com/slac-lcls/ndarray-psana-feedstock
github.com/slac-lcls/ndarray-psana-py2-feedstock
github.com/slac-lcls/libpressio-feedstock
github.com/slac-lcls/xtcav2-feedstock
github.com/slac-lcls/psgeom-feedstock
github.com/slac-lcls/psocake-feedstock
github.com/slac-lcls/psana1-py2-feedstock
github.com/slac-lcls/psana1-feedstock

As normal user:

- Modify recipe/meta.yaml
  - Check if new version of packages has been released
  - If yes, bump up version, update sha256 checksum and reset build number
  - Otherwise, bump up build number
- Commit
- conda smithy rerender -c auto
- Push

After all packages have finished building on GitHub (Unfortunately, GitHub Actions do not provide a "dashboard" to check if the packages have been built or when they are finished. There are only two ways to check this: 1) Manually check every feedstock repository (Under the "Actions" tab) 2) Wait for either the package to appear on anaconda.org/lcls-i or for an email notifying a failure to arrive):

As psreldev:

- source /cds/sw/ds/ana/conda1/inst/etc/profile.d/conda.sh
- cd /cds/sw/ds/ana/conda1/manage/jenkins/
- conda env create -n ana-4.0.11 --file ana-env-py2.yaml
- conda env create -n ana-4.0.11-py3 --file ana-env-py3.yaml

LCLS2 Github Actions Approach (Deprecated)

Currently deprecated since conda forge changes versions more rapidly than we would like, so we use the "Pinned Approach" described below.

First, tag lcls2 and ami repos.

checkout the 20 feedstock packages (in github.com/slac-lcls/):

```
python /cds/sw/ds/ana/conda2/manage/bin/feedstock.py --clone
```

After these repos are cloned, you can use this to do commands in all repos:

```
python /cds/sw/ds/ana/conda2/manage/bin/feedstock.py --cmd "pwd; git pull --rebase"
```

Either update version (with new git tag) or update the build number in each meta.yaml. conda-forge enforces manual maintenance of version numbers and sha256 hashes (no automatic determination from latest GitHub tag). If version has changed, compute new sha256 from .tar.gz from GitHub with:

sha256sum 1.1.7.tar.gz

For a new version, remember to reset build number to zero. Commit the changes. This command allows you to modify build numbers. It will prompt you for each package, and you can enter "y" (increment build number) "n" (don't increment, default) or "0" (set to 0):

```
python /cds/sw/ds/ana/conda2/manage/bin/feedstock.py --incbuildnum
```

The next step requires an environment where "conda-smithy" is installed (we have put this in psrel's conda_build env). Then rerender with "conda smithy rerender -c auto" (again using feedstock.py --cmd option). This last command does its own git-commit for you with the changes. When must this be run? Valerio writes: "In general any change in conda forge config (for example, when we want to upload to a different channel, or to switch to a different CI). However, it is also needed to pick up the newest version of the conda-forge pinning file, so I run it every time. In the worst case, it tells me everything is up to date and does not create a commit".

The final "git push" of the above changes must be done carefully because it triggers the GitHub build, and order of the GitHub builds matters because of conda's "build:, host:" section dependencies (run-time dependencies do not affect this order). We believe pure-python packages can go in the first wave, since they have no complex dependencies in "build:, host:" sections of meta.yaml. These are the waves of builds that can be launched in parallel:

- libnl, libnl3, roentdek, amityping, prometheus-cpp, libusdusb4, psmon, networkfox, rogue, epix, lcls2_timetool, cameralink-gateway, lcls2-pgp-pcie-apps, xtcdata, ami
- rmda-core (depends on libnl), psalg (depends on xtcdata)
- fabric (depends rmda-core), psana (depends on psalg)
- psdaq (depend on psalg, fabric)

The 4 build waves above can be launched with a command like this.

```
python /cds/sw/ds/ana/conda2/manage/bin/feedstock.py --cmd "git push" --wave 1
```

If a package is built for python build matrix tells it to build for the officially supported conda-forge versions (3.6,3.7,3.8).

If we were in conda-forge officially, they take care of these build-order dependencies (but not the versions/sha256/buildnumber). They may have a bot that tells you that the version was updated, but we still have to update the version number by hand. We haven't gone the conda-forge route because (1) we don't know how much work it is, and (2) cannot upload source .tar.gz to conda-forge, usually get it from GitHub, but the rogue packages are not public.

Unfortunately, GitHub Actions do not provide a "dashboard" to check if the packages have been built or when they are finished. There are only two ways to check this: 1) Manually check every feedstock repository (Under the "Actions" tab) 2) Wait for either the package to appear on anaconda.org/lcls-ii or for an email notifying a failure to arrive.

To create the environment as psrel, use these commands:

conda env create --name ps-N.Nodd.N --file env_create.yaml (devel env)

conda env create --name ps-N.Neven.N --file prod_create.yaml (prod env)

Automated feedstock environment building

1. ### GitHub Access Token

   A new set of python scripts has been developed to automate the creation of LCLS-II conda environment. They can be found here:

   `/cds/sw/ds/ana/conda2/manage/buildenv`

   The scripts need a valid personal GitHub token to be used (unfortunately, GitHub caps the number of API requests that can be done without a token). The token can be created on GitHub, after logging in:

   https://github.com/settings/tokens

   The token must be exported and made available as an environment variable called GITHUB_ACCESS_TOKEN. Technically it is only needed by step (4) below, but the way the scripts are written it should be set for all steps.

   For an example, please see ~valmar's bashrc file (also in psrel bashrc).

2. ### Package Version File

   (this file-editing should be run as user "psrel")

   In order to generate packages, a YAML file listing the required version of each package must be created. For a file with the package versions in the latest environments, see:
   /cds/sw/ds/ana/conda2/manage/buildenv/table.yaml

   A few lines as an example:

```
ami: 2.4.7
amityping: 1.1.7
cameralink-gateway: 7.6.2
epix: 0.0.3
lcls2-pgp-pcie-apps: 2.2.0
lcls2_timetool: 3.3.0
epix-hr-single-10k: 3.1.1
lcls2-epix-hr-pcie: 1.2.0
....
```

3. **Preparing source tarballs**

   (this step should be run as a normal user like valmar, cpo,...)

   Before building the feedstocks, source tarballs must be created for the rogue-related packages (which are private repos, so need source .tgz files generated) by running the prepare_source.py script.

   The script must be run using the conda_build environment (conda activate conda_build) It must also be run as a normal user because psrel cannot write to the /reg/g/psdm/web/swdoc/tutorials/.
   The script takes the package version file as an argument:

   ```
   python /cds/sw/ds/ana/conda2/manage/buildenv/prepare_source.py --package-version-file=/cds/sw/ds/ana/conda2/manage/buildenv/table.yaml
   ```

   The source tarballs are automatically generated and copied to /reg/g/psdm/web/swdoc/tutorials/

   PS: The script will clone the required repostiories in the current working directory!!!! Working in a temporary directory that can be later deleted is strongly advised

4. **Generating packages**

   (this step should be run as a normal user like valmar, cpo,...)

   The packages can now be built using the build environment script, again in the conda_build environment (conda activate conda_build):

   ```
   python /cds/sw/ds/ana/conda2/manage/buildenv/build_environment.py --generate-packages --package-version-file=/cds/sw/ds/ana/conda2/manage/buildenv/table.yaml
   ```

   The script will build the packages wave-by wave. For each wave, the script will clone the feedstocks one by one, automatically make the necessary changes to the recipes, run "conda smithy" and push the changes to the git repositories, triggering the building of the packages. The individual builds can be seen at URLs like https://github.com/slac-lcls/libnl3-feedstock/actions

   The script will then check with GitHub every thirty seconds, and report the status of the build process ("Not started yet", "running", "success" or "failed"). It will waint until all builds have finished and are either in a "success" or "failed" state. If no build has failed, the script will then proceed to the next package "wave". Otherwise it will exit.

   Instead of going through all the waves, one can start from a certain wave (using the --start-from-wave option) and/or stop at a certain wave (using the --stop-at-wave option)

   PS: The script will clone the required repostiories in the current working directory!!!! Working in a temporary directory that can be later deleted is strongly advised

5. **Building environments**

   (this step should be run as user psrel)

   The production and development environments can be created using the normal "conda env create" commands (see above), or using the build_environment script:

   ```
   (from /cds/sw/ds/ana/conda2/manage directory)
   (NOTE: this command produced the env in a non-standard location, cpo used "conda env create --name ps-4.4.1 -f prod_create.yaml" instead)
   python buildenv/build_environment.py --build-environment --environment-name=ps-4.4.0 --environment-file=prod_create.yaml
   ```

   The environments must be created using the psrel user, because only the psrel user can write to the environment directories

   This step and the previous one can be combined in a single command for convenience:

   ```
   python build_environment.py --generate-packages --package-version-file=/cds/sw/ds/ana/conda2/manage/buildenv/table.yaml --build-environment --environment-name=ps-4.4.0 --environment-file=prod_create.yaml
   ```

LCLS2 Pinned Packages Approach (Current as of 10/9/23)

Building A Package

NOTE: (Oct. 10, 2023) we are renaming the .condarc to .condarc_dontuse files in home-directories and directories like /cds/sw/ds/ana/conda2-v2/inst/. condarc to not implicitly depend on those. So channels must be specified explicitly in the conda commands. **Addition on Jan. 5, 2024 by cpo**: I found that when I was building cameralink-gateway (as user cpo) that it was trying to write into /cds/sw/ds/ana/conda2/inst/envs/conda_build/pkgs/python-3.9.18-h0755675_1_cpython. So I restored my .condarc to get it to work. I guess I could have tried to run as “psrel” instead. Valerio writes: "I think this is another of the bug in conda. I think it is tied to the environment being group writable. I don't think that having a condarc file is bad per se. I also have one with the paths pointing to non-home directories with more space. I think what we should avoid is having channels in the condarc, because when we run a conda build, conda also takes those channels into consideration. To have better control on what channels I am using, I removed all the channels from my condarc file and I specify the channels manually, but I still use the condarc for everything else!".

Need to create a feedstock for every new package. LCLS2 feedstock packages are in (for example) https://github.com/slac-lcls/epix-feedstock.

For rogue packages only, need to create a .tar.gz since their git repos are not public:

- Checkout the package (remove the ".git" subdirectory because it is large).
- Create .tar.gz file with "tar cvfz epix-quad-1.2.0.tar.gz epix-quad/"
- Copy the .tar.gz to the directory where it can be seen: "cp epix-quad-1.2.0.tar.gz /reg/g/psdm/web/swdoc/tutorials/"
- Compute the sha256sum with "sha256sum epix-quad-1.2.0.tar.gz"
- Put this sha256 in the epix-quad-feedstock/recipe/meta.yaml

NOTE: To generate a small .tar.gz file:

- use the following command to ignore the large git-lfs files when cloning: "GIT_LFS_SKIP_SMUDGE=1 git clone git@github.com:slaclab /cameralink-gateway cameralink-gateway-8.2.2 --recursive"
- remove the ".git" subdirectory because it is large

The pinning is in conda_build_config.yaml (our packages are towards the end of the file, the conda-forge pinnings are earlier). The conda forge pinnings are obtained from files like: https://github.com/conda-forge/conda-forge-pinning-feedstock/blob/main/recipe/conda_build_config.yaml. Valerio wrote a small throwaway python script to take the packages from our environment and them to the conda_build_config.yaml. This script also pins the low-level underlying packages that conda-forge does not explicitly pin and adds them to conda_build_config.yaml. This conda_build_config.yaml cannot trivially be used by GitHub actions (because it will pull the latest version from conda-forge) so we are using this file and building locally on psbuild-rhel7 (with infinite time we could write a custom action).

Then: "conda deactivate", "conda activate conda_build". Need a .condarc that looks like this (otherwise goes to conda "defaults" instead of conda-forge for packages). Need the "pkgs_dirs" variable in .condarc, otherwise conda-build tries to write to /cds/sw/ds/ana/conda2/inst and has permissions issues.

```
(conda_build) psbuild-rhel7-01:epix-quad-feedstock$ more ~/.condarc
channels:
  - lcls-ii
  - lcls-i
  - conda-forge
  - defaults
  - tidair-tag
  - tidair-packages
pkgs_dirs:
  - ~/.conda/pkgs
(conda_build) psbuild-rhel7-01:epix-quad-feedstock$
```

Then build package with "conda build recipe/". Important to launch from this directory since it has the conda_build_config.yaml.

For the setup.py (e.g. epix-quad/setup.py) the list of python modules can be determined by finding all the directories with __init__.py in them, e.g. with "find . -name __init__.py". Then the "package_dir" variable in setup.py is filled in with the directories where all the modules are.

After creating the package then upload, e.g. with "anaconda upload -u lcls-ii yaml-cpp-0.5.3-h1d15853_72.tar.bz2". May need to do "anaconda login" (once per year?).

Rogue Package Recipe Creation

Since we have moved the rogue package creation to be done with pip it is necessary to explicitly list all the packages in setup.py similar to this: https://github.com/slac-lcls/epix-hr-m-320k-feedstock/blob/main/recipe/setup.patch. This patch file gets run when the conda builds the recipe. All the relevant packages to be included in this patch file can be found by looking for directories containing __init__.py.

Devel Env Creation

Export the current environment to yml (with includes pinnings). Then we can do "small tweaks" to versions. But big version changes will break the solver. If we have a big change, we have to start from scratch: use /cds/sw/ds/ana/conda2/manage/env_create.yaml in the usual fashion (but use libmamba), e.g.:

```
conda env create --name ps-4.5.16 --experimental-solver=libmamba --file=/cds/sw/ds/ana/conda2/manage/env_create.yaml
```

Prod Env Creation

As for the devel env, but with a file like "/cds/sw/ds/ana/conda2/manage/prod_create.yml" and when we do "conda create" it will pick up the versions that were specified in the feedstock package conda_build_config.yml.

If the pinnings have not changed: clone the current environment into a new one, and install the new versions (psana, psdaq, etc.) with "conda install" (remembering to use the experimental libmamba).

```
conda install -c lcls-ii xtcdata=3.3.36 psalg=3.3.36 psana=3.3.36 ami=2.4.16 --experimental-solver=libmamba
```

If the pinnings have changed: To make a conda_build_config.yaml, make a devel env, export the pinnings to a yml file, then run a short python script from Valerio to make the new conda_build_config.yml, then regenerate all feedstock packages:

```
with open("ana-4.0.42.yml", "r") as fh:
    lines = fh.readlines()
for l in lines:
    items = l.split("=")
    name=items[0].split('- ')[1].replace("-","_")
    print(f"{name}:\n  - {items[1]}={items[2]}")
```

To run this script, this file must only be a list of packages: early lines the file (list of channels, etc.) must be removed by hand. In the generated conda_build_config.yml there are a few compiler-related lines like cxx_compiler* that have to be removed, as well as backports.* lines (two or three).

Disappearing Packages

Sometimes our precise pinned package versions disappear from conda-forge. In the past when this happened Valerio was able to manually modify the above pinned yaml files to choose a nearby version. This so far has not broken the fast conda-solver behavior.

New Pinnings

From time to time, an important package with big ramifications in the environment needs to be updated (e.g.: mypy). Or a general refresh of the packages in the environment. is needed. In these cases, new pinnings and new pinning files need to be created. This requires performing the following steps:

- In all feedstocks, replace conda_build_config.yaml with the version from: conda-forge-pinning-feedstock/conda_build_config.yaml at main · conda-forge/conda-forge-pinning-feedstock (github.com)
- Rebuild all feedstocks in the following order:
    - libnl
    - libnl3
    - rmda-core
    - libfabric
    - roentdek
    - amityping
    - cameralink-gateway
    - epix-100a-gen2
    - epix
    - lcls2-ephi-hr-pcie
    - lcls2-pgp-pcie-apps
    - lcls2_timetool
    - networkfox
    - prometheus-cpp
    - xtcdata
    - psalg
    - psana
    - psdaq
    - ami
    - psmon
- Create new production and dev environment (see above)
- Create new conda_build_config.yaml using the script in the section above and copy it in every feedstock to create packages with pinned dependecies in the future

LCLS I + AMI2 Environment

This environment is based on the LCLS I but contains all the dependencies needed to run AMI 2. To create it, we follow the strategy of trying to keep all main packages of the LCLS-I environment at the right version, and all the AMI2 dependencies at the version we have in the LCLS-II environment. We let the second level dependencies fluctuate freely, especially the compilers. Currently we follow this procedure (I use the ana-4.0.44-py3 environment as an example):

- Export the base lcls-i environment to a YAML file:

```
conda env export > ana-4.0.44-py3.yaml
```

- Run a script similar to the following to pin the main lcls-i packages from at the right version:

```
import yaml

with open("ana-4.0.44-py3.yaml", "r") as fh:
      curr_env = yaml.safe_load(fh)

dep_dictionary = {}
for entry in curr_env['dependencies']:
    items = entry.split("=")
    dep_dictionary[items[0]] = items[1]

with open("/cds/sw/ds/ana/conda1/manage/jenkins/ana-env-py3.yaml", "r") as fh:
    base_env = yaml.safe_load(fh)

new_dependencies = []
for entry in base_env['dependencies']:
    items = entry.split("=")
    new_dependencies.append(f"{items[0]}={dep_dictionary[items[0]]}")

base_env["dependencies"] = new_dependencies

with open("ana-4.0.44-py3-ami2.yaml", "w") as fh:
    yaml.dump(base_env, fh)
```

- Add to the ana-4.0.44-py3-ami2.yaml file the following dependencies, at the version that the are in the current lcls-II version:
    - networkfox
    - asyncqt
    - amityping
    - mypy
    - setproctitle.
    - jedi
    - sympy
    - p4p
    - pyqode.python1
    - pint
    - pyfftw

      For example, in the case of 4.0.44, the following lines were added to the file (the version numbers come from the current lcls-ii development environment ps-4.5.16):

```
- networkfox=0.0.8
- asyncqt=0.8.0
- amityping=1.1.11
- mypy=0.961
- setproctitle=1.2.3
- jedi=0.17.2
- sympy=1.10.1
- p4p=3.5.5
- pyqode.python=2.12.1
- pint=0.18
- pyfftw=0.13.0
```

- Remove the version from the compiler dependency, so that it looks just like the following, without any version number:

```
- compilers
```

- Use the resulting yaml file to generate the environment, using the libmamba solver:

```
conda env create --experimental-solver=libmamba -n ana-4.0.44-py3-ami2 --file ana-4.0.44-py3-ami2.yaml
```

Environments on S3DF

When creating an LCLS-II release on S3DF, the following lines

in the setup_env.sh file need to be specifically adapted to S3DF

```
source /sdf/group/lcls/ds/ana/sw/conda2/inst/etc/profile.d/conda.sh
export CONDA_ENVS_DIRS=/sdf/group/lcls/ds/ana/sw/conda2/inst/envs
export DIR_PSDM=/sdf/group/lcls/ds/ana/
```

NOTE / TODO: DIR_PSDM should point to the directory containing the "detector" directory. We should make sure that the content of the detector directories on psana (/cds/group/psdm/detector) and sdf (/sdf/group/lcls/ana/detector) match. Currently, their content seem different: the size of the directory on psana is 1.3Tb on psana and 2.3TB on S3DF

Utility Scripts

The following two python scripts can be used to explore the content of a repodata.json file, which can be downloaded from a conda online package repository and lists in detail the content of the repository

The fist script, dependency.py, lists all the dependencies of a specific package

The second, reverse_dependency.py, lists all the package that depend on the specified package

For both scipts, the syntax is: <script name> repodata.josn <package name>