11falsenonelisttrue

Compute Resources

Have questions or need help with compute, including activation or issues? Follow this link.

User Agreement

Docker Usage

The information on this page assumes that you have a knowledge base of using Docker to create images and push them to a repository for use. If you need to review that information, please see the links below.

https://washu.atlassian.net/wiki/spaces/RUD/pages/1705115761/Docker+and+the+RIS+Compute1+Platform?atlOrigin=eyJpIjoiNzc4YTZjNjIxYmQwNGI3OTk4M2Q0MV

https://washu.atlassian.net/wiki/spaces/RUD/pages/1864892726/Docker+Basics+Building+Tagging+Pushing+A+Custom+Docker+Image?atlOrigin=eyJpIjoiMTVjMjNlM

storageN

The use of `storageN` within these documents indicates that any storage platform can be used.

Current available storage platforms:

storage1

storage2

Overview

The purpose of this tutorial is to demonstrate the steps required to build and use a custom conda-based environment in RIS using an existing Docker image. The conda environment will be located in the RIS Data Storage Platform so that any changes will persist between jobs.

As an example, a sequence analysis conda environment will be created containing the following tools:

bwa (http://bio-bwa.sourceforge.net/)

citipy (https://pypi.org/project/citipy/)

fastp (https://github.com/OpenGene/fastp)

fastqc (https://github.com/s-andrews/FastQC)

multiqc (https://multiqc.info/)

samtools (http://www.htslib.org/)

spades (https://cab.spbu.ru/software/spades/)

Interactive Job Submission

Defining Environment Variables

Begin by defining the environment variables that will be used during creation of the conda environment. The environment variables can be defined in your `.bashrc` or `.condarc` file to avoid having to enter them for each job.

The environment variables are:

`CONDA_ENVS_DIRS`: the path to the directory where conda environments

will be created.

`CONDA_PKGS_DIRS`: the path to the directory where conda packages

will be downloaded to.

See the conda documentation for more information.

In the example below, the `CONDA_ENVS_DIRS` and `CONDA_PKGS_DIRS` are set to a folder in the RIS Data Storage Platform. Make sure the folders exist and are writable.

Please also define the following environment variables to mount the RIS Data Storage Platform and add conda binaries to the `PATH`:

Job Submission

Once the appropriate environment variables are defined, the next step is to submit the interactive job using the `continuumio/anaconda3:2021.11` Docker image.

After the interactive job lands on an execution host, the command-line prompt will begin with `(base)`. For example:

If this is not the case, conda needs to be initialized. Please see our documentation for instructions on how to resolve this issue.

Creating the conda Environment File

The conda environment will be created using an environment `YAML` file. The interactive session has the `nano` text editor installed to create the file with. The file will be saved in the home folder and named `environment.yml`. Please see the conda documentation for more information.

Note

For users wanting to create multiple environments, naming each `YAML` file the same name as the environment is recommended to avoid confusion. If using more than one `YAML` file, be sure to replace the occurrences of `environment.yml` in the tutorial with the name of each `YAML` file.

Open a New File in nano named `environment.yml`:

Copy and paste following into the file:

=1.7 - samtools<=1.11 - spades=3.9.1 - pip - pip: - citipy]]>

The `pip` dependency is required to install the `citipy` package using the `pip` package manager. The `pip` dependencies are added to the end of the environment file to reduce conda/pip installation issues from being used in the same environment. This is the ideal order of dependencies and should be followed to reduce possible installation issues.

The `environment.yml` file provides the following instructions for creating the conda environment:

- `name`: the name of the conda environment. This name will be used to reference the environment in the `conda activate` command.
- `channels`: the channels to be used when creating the conda environment. Channels can be thought of as additional repositories that contain packages.
- `dependencies`: the packages that will be installed in the conda environment.
- `pip`: the `pip` package manager is used to install additional packages. In the example, the `citipy` package is installed using `pip`.
- Specific package versions can be specified using the < , > and = operators.
  - For example, `samtools<=1.11` will install the latest version of `samtools` less than or equal to 1.11.

Please see the conda environment file documentation for more information.

Installing the conda Environment From `environment.yml`

Run the following command to install the conda environment using the `environment.yml` file:

The conda environment creation may take several minutes to complete. Please be patient as the environment is created. Once the environment is created, the following output will be displayed:

Activating the conda Environment

Once the example `sequencing` conda environment is created, it can be activated using the following command:

All of the packages listed in the `environment.yml` file are available for use. To view a list of the installed packages in an environment, run the following command after activating the environment:

For example, the `sequencing` environment will list the following packages:

Using the above command, we can ensure that the version requirements for the conda environment were met. As a reminder, the environment file had the following version requirements:

=1.7 samtools<=1.11 spades=3.9.1]]>

From the package list, we can validate that the version requirements were met.

A list of the currently installed environments can be viewed with the command:

Sharing conda Environments

To share a conda environment with others, you can create an environment file. First activate the environment you wish to share. Then, run the following command:

~/environment.yml]]>

This will create a file in your home folder named `environment.yml`. It is recommended to name the environment file after the environment you wish to share. Once the environment file is shared, another user can create the environment using this tutorial.

Compatible Docker Images

The following Docker images have been tested with this tutorial to create a custom conda environment:

- Jupyter Notebook Data Science Stack (https://hub.docker.com/r/jupyter/datascience-notebook/).
  - Tested with `jupyter/datascience-notebook:ubuntu-20.04`
- mambaforge (https://hub.docker.com/r/condaforge/mambaforge).
  - Tested with `condaforge/mambaforge:4.11.0-0`
  - Replace `conda` commands with `mamba`. See the mamba documentation for more information.

Creating IPython Kernel for Usage with Jupyter Notebooks/Labs

In order to create an IPython Kernel you will need to first activate your conda environment that you created (using the above examples) and then follow the below commands:

Then to install IPython Kernel you would need run the below commands to install the needed IPython packages and then register your kernel for this particular conda environment:

Make sure to replace “env_name”, “Some_Name” and “User_Friendly_Name” with respected Values.

Then once you have a Jupyter Notebook/Lab up and running on RIS platform (through OnDemand Portal or Command Line). Select the Kernel using either the top right corner button or the Kernel menu in the Menu bar.

You will need to repeat these steps for each conda environment for which you wish to register a kernel and use with a Jupyter Notebook/Lab.