SONiC Utilities Local Build

Table of Contents

Introduction

This guide will serve as an example of how you can test and validate your changes to sonic-utilities package – in particular the changes related to the CLI. Command-line utilities code is packaged inside a python wheel file that we can deploy in SONiC. The first step is to build the wheel package containing source code changes that you might want to test in SONiC.

Building a wheel package containing your source code, and running it into SONiC is trivial, but this guide will show how to run unit tests related to CLI to validate your changes. Running unit tests require a lot of dependencies to be met, those that are tedious to keep track of manually. A convenient alternative is to let the build process of sonic-buildimage repo take care of that for you. During the build process, the SONiC build process will take care of all the necessary dependencies and install them inside of a container (known as sonic-slave container). The idea is to stay inside the container once the build process starts to make the utilities wheel file and run unit tests inside this environment.

Environment

OS: Ubuntu 20.04

Dependencies

1. Install git, pip3 and jinja in host build machine
   sudo apt install git
   sudo apt install -y python3-pip
   sudo pip3 install j2cli

2. Install Docker and follow post-installation steps to allow running the 'docker' command without root privileges.
   Validate that docker can run without ‘sudo’ through the command line
   sudo gpasswd -a ${USER} docker
   docker run hello-world

Build sonic-utilities wheel package

Build locally

1. Clone the sonic-buildimage repository and navigate to the local repo through the command line

   git clone https://github.com/sonic-net/sonic-buildimage

   cd sonic-buildimage

2. To clone your own fork of sonic-net repository, you can edit the .gitmodules file inside the root directory

   vi .gitmodules

   Edit the “Url” field of the submodule for which you wish to clone your own fork of the repository. Here you will want to change the Url field of the sonic-utilities submodule.

3. Initialize the repository through the command

   make init

   Note that this step may take hours depending on your internet connection, so please be patient.

This step will clone all of the submodules listed in the .gitmodules file inside the src/ directory. After the command completes, head into src/sonic-utilities directory and inspect that you got your own fork of the repository. If the code you wish to build is in another branch, change the branch using

   git checkout <branch_name>

4. Configure the build environment for an ASIC type (any platform will do here for sonic-utilities)

   make configure PLATFORM=generic

5. Make the sonic-utilities wheel file while keeping the Bullseye slave container alive to run unit tests in it.

   make NOSTRETCH=1 NOBUSTER=1 KEEP_SLAVE_ON=yes target/python-wheels/bullseye/sonic_utilities-1.2-py3-none-any.whl

6. When the build finishes, your prompt will change indicating you are inside the sonic-slave container. Navigate to sonic-utilities directory inside the src folder.

cd src/sonic-utilities

7. Now you can make changes inside VS Code or your preferred code editor inside the sonic-buildimage/src/sonic-utilities directory.

8. To test your changes, run the command inside the sonic-slave container

   python3 setup.py test

   The above command will run all the unit tests associated with SONiC CLI. To run an **individual test** you can use the command

   pytest-3 tests/<name_of_test_file>

   Useful options to above command

   -vv: Verbose output. Shows all of the individual functions run within the file.

   -rP: Show standard output while running the test.

   e.g. pytest-3 -rP -vv tests/vlan_test.py

9. Some tests may fail while running tests/building sonic-utilities in a local environment, they are:

- FAILED tests/disk_check_test.py::TestDiskCheck::test_readonly
- FAILED tests/drops_group_test.py::TestDropCounters::test_show_counts
- FAILED tests/drops_group_test.py::TestDropCounters::test_show_counts_with_group
- FAILED tests/drops_group_test.py::TestDropCounters::test_show_counts_with_type

   These tests pass in Azure pipelines under the same piece of code, so you can safely ignore them in a local build if they appear as FAILED.

10. If the rest of the tests pass, you can be sure that the changes are valid and will not cause the SONiC CLI to break. After validating the changes, you can run build the utilities wheel file through the command

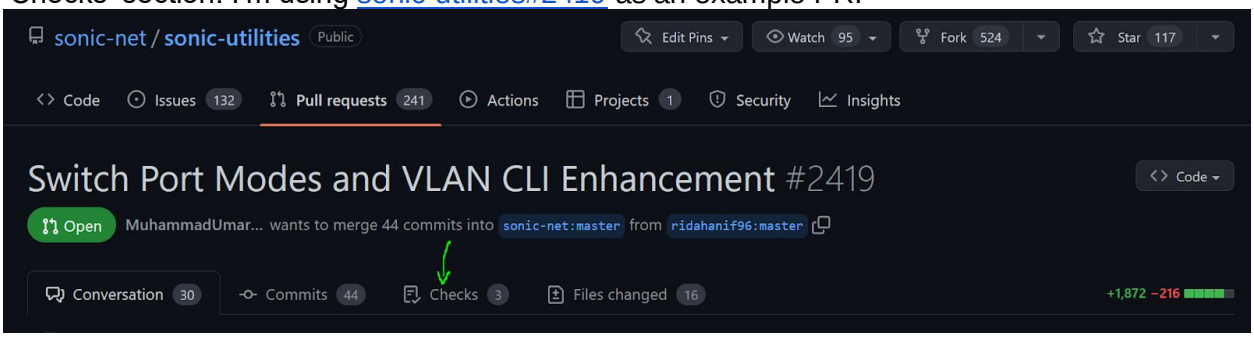
    python3 setup.py bdist_wheel

The wheel file will be created under sonic-utilities/dist directory.

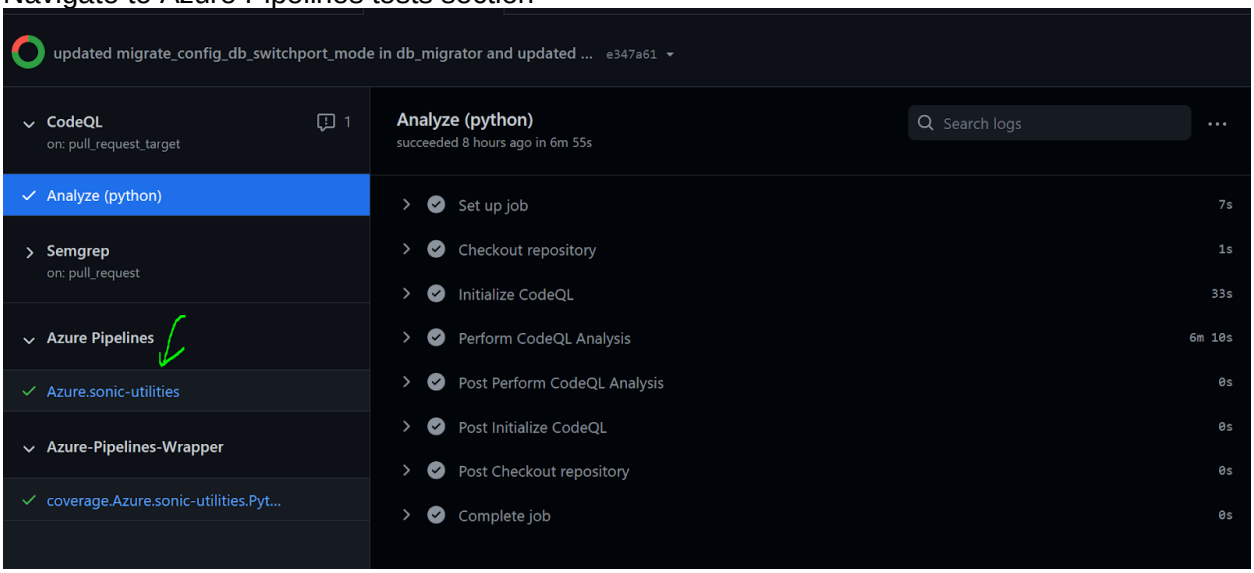
Build through CI/CD Pipelines

You can also create a draft PR on the actual sonic-utilities repository on GitHub. The changes will automatically pass through the Azure CI Pipelines upon pushing a commit.

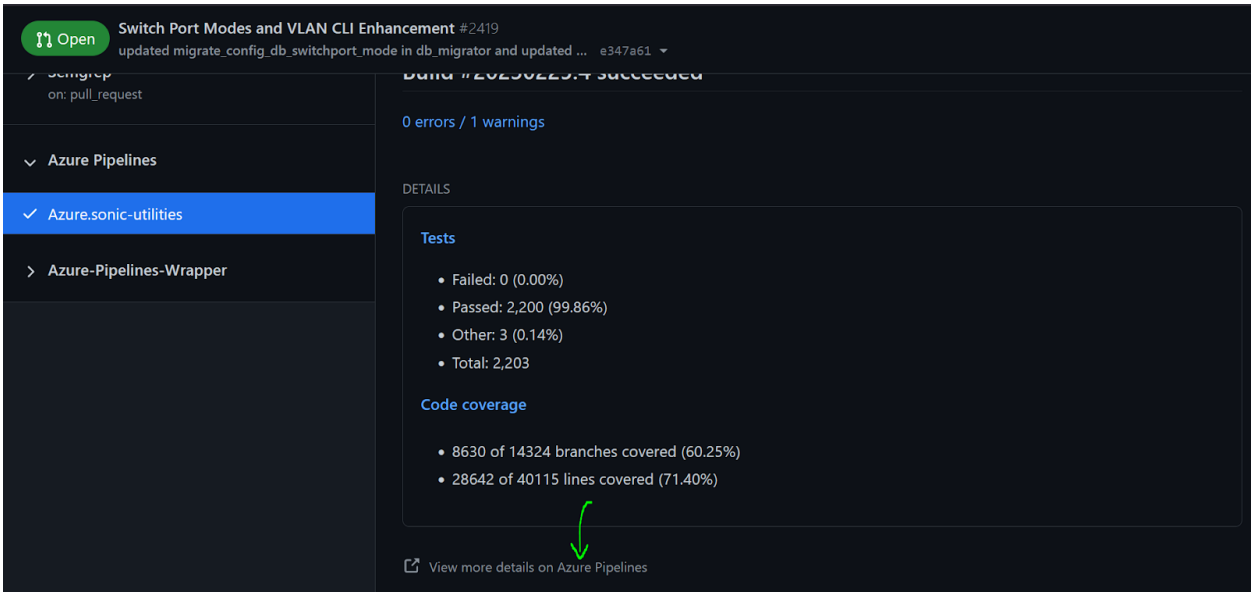
To see the output of the tests or to monitor the tests as they are running, you can navigate to 'Checks' section. I'm using sonic-utilities#2419 as an example PR.



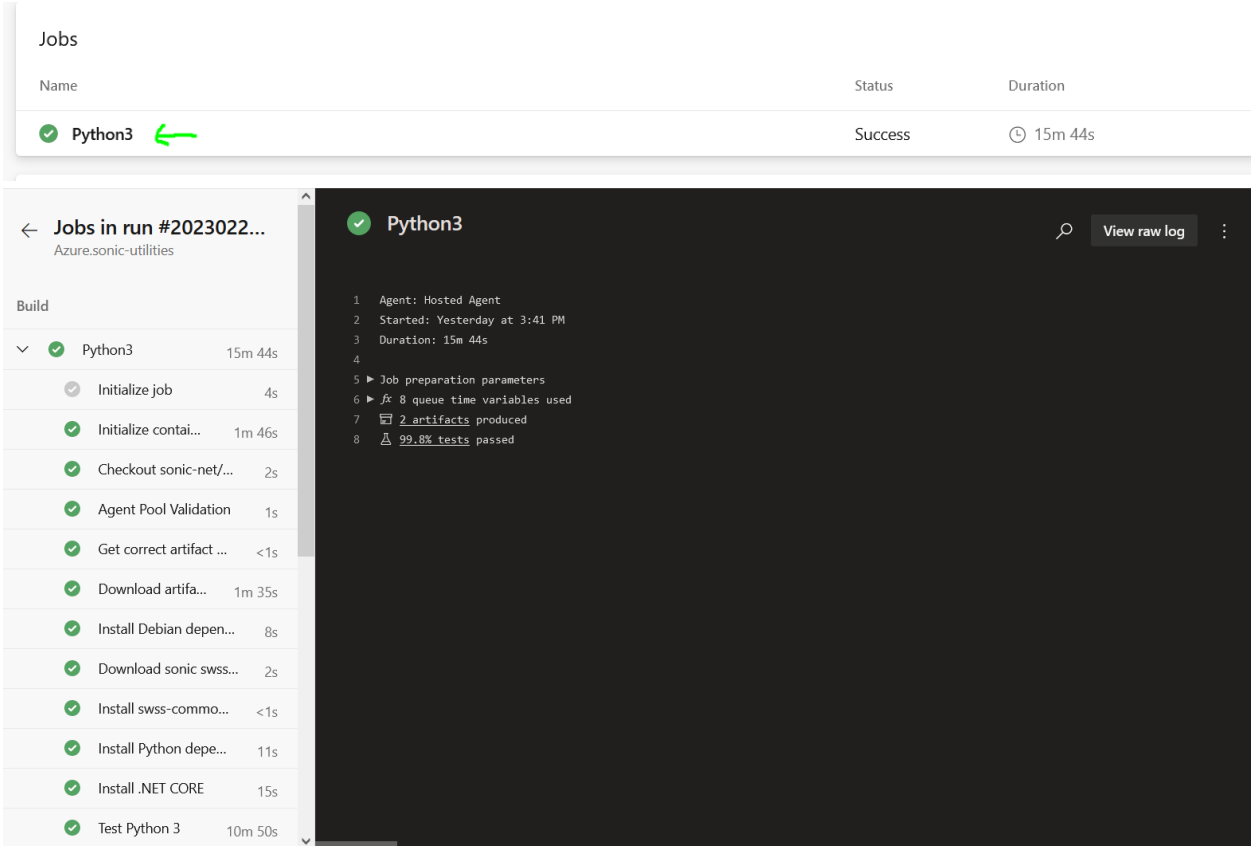
Navigate to Azure Pipelines tests section



You may scroll down now to find the link to the Azure Pipelines



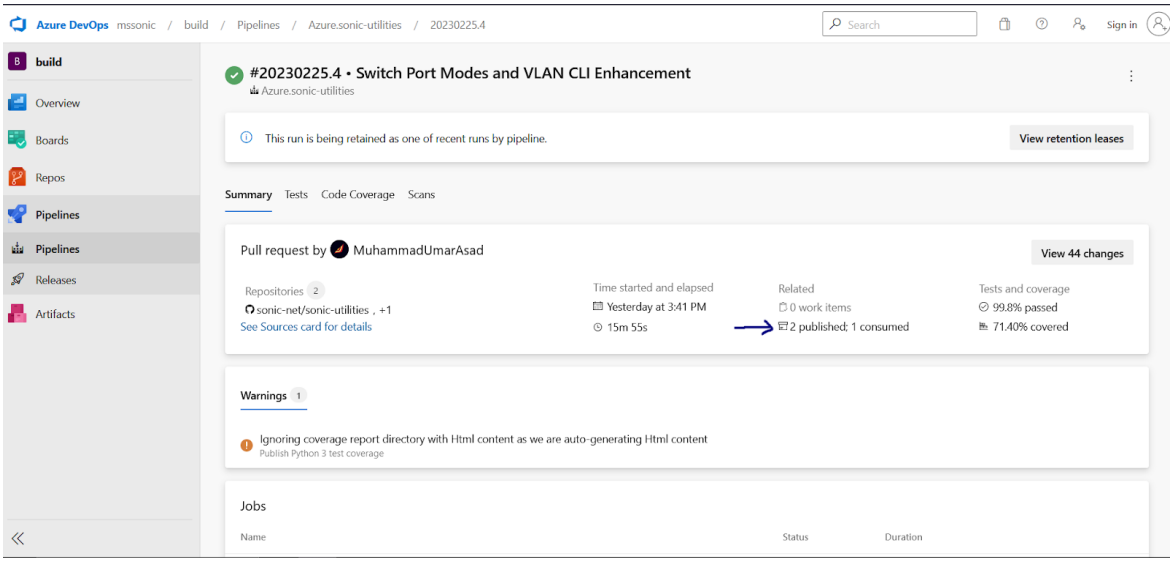
Click on the link that'll redirect you to dev.azure.com pipelines page. You'll find the output on the tests under the Jobs section



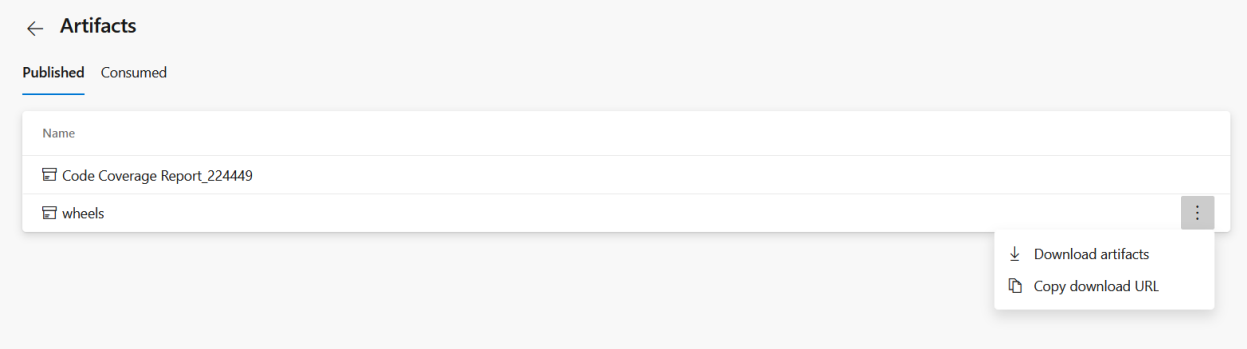
If all the tests pass, you will have a downloadable artifact containing the wheel file to run on SONiC

Download the artifact

Under the summary section, you can find the list of produced artifacts and download them on your local machine.



You can download the **"wheels"** artifacts that would be a zip file containing the sonic-utilities wheel file produced as a result of the build run.



Extract the wheel file into a file into a folder and follow the next instructions to run it on SONiC.

Note: The build run is cleaned after a few days, so the artifacts generated through this method are available for a limited time. They would be re-run if a new commit is pushed thereby generating a fresh artifact.

Run wheel file in SONiC

To incorporate the changes into SONiC, you need to replace the existing sonic-utilities package inside the OS with your newly created package.

1. Start a SONiC instance and log in with your credentials.

2. Delete the current CLI package through pip

   sudo pip uninstall sonic_utilities

3. Navigate to the directory where the newly created wheel file is located. Run a python web server inside the directory to install it into SONiC through the web.

   python3 -m http.server <port> --bind <ip_address>

   Example:

.168.1.64

4. Install the wheel file inside SONiC through the command

   sudo pip install http://<ip_addr>:<port>/sonic_utilities-1.2-py3-none-any.whl
   sudo pip install http://192.168.0.197:8000/sonic_utilities-1.2-py3-none-any.whl

sudo pip3 install --force-reinstall --no-deps http://172.16.7.13:8000/sonic_utilities-1.2-py3-none-any.whl

5. Voila! Your changes are now present inside SONiC.

YourPaSsWoRd