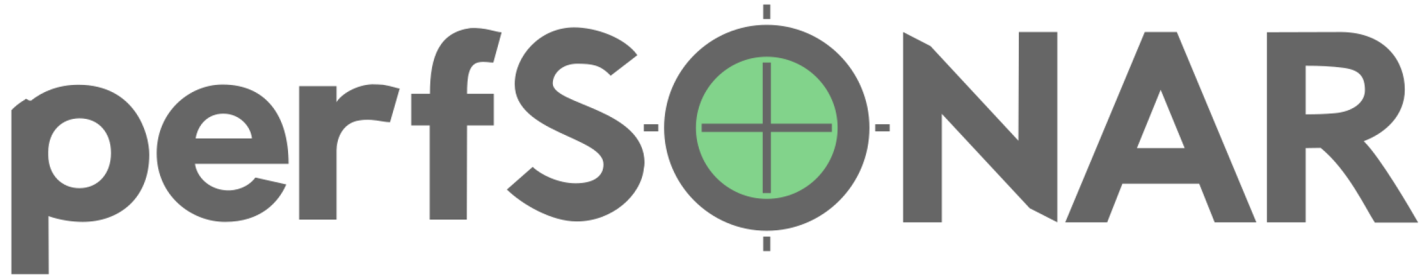

Quality Assurance for perfSONAR

Adriatik Allamani (RASH)

perfSONAR is developed by a partnership of

4th European perfSONAR User Workshop 14th, 15th and 16th May 2024, Trondheim, Norway

Github repository for QA

- Automated/virtualized quality assurance tools; system integration testing, service sanity checks
- This includes these components:

1. Integration test tools - Automated testing of bundle installs, upgrades, etc. - virtualized (docker/vagrant)
2. Sanity Checking - sanity checking for QA testing. Checks that services are running/available, webservices are correctly loading/responding, expected ports are open, etc.
   - Run on demand or automated
   - Includes a dockerised ELK environment for archiving/visualizing results

https://github.com/perfsonar/quality-assurance

- Three main components:
  - integration-testing
  - pscheduler-crawler
  - sanity-checking

sanity-checking

- Building:

  docker build –t sanity .

- Running:

  docker run --privileged --name sanity --network host --rm sanity

- SINGLE

docker build -t single-sanity -f Dockerfile-single .

docker run --privileged --name single-sanity --network host --rm single-sanity

integration-testing

- This project provides ways of performing installation/integration testing on various parts of perfSONAR using virtual environments.

- These modules create fresh OS environments (container or VM), install the relevant components from scratch, making sure that no errors are encountered in the process. Next, optionally, additional sanity checks are run before tearing down the temporary containers.

bundle-deb-install-tests

- Testing perfSONAR installation on Debian and Ubuntu
- OS Support
  - Debian 10 Buster
  - Debian 11 Bullseye
  - Ubuntu 18 Bionic Beaver
  - Ubuntu 20 Focal Fossa
  - Ubuntu 22 Jammy Jellyfish
- Running the script
  - The script attempts to install a perfSONAR bundle and then perform sanity checks using the sanity-checking scripts.
  - Can run the tests by executing

    test_install_instructions.sh -r $REPO -o $OS -b $BUNDLE

bundle-deb-install-tests

$REPO is one of these:

- perfsonar-release (production)
- perfsonar-patch-snapshot
- perfsonar-minor-snapshot
- perfsonar-patch-staging

$OS is one of these (default to test all):

- debian:buster
- debian:bullseye
- ubuntu:bionic
- ubuntu:focal
- ubuntu:jammy

$BUNDLE is one of these (default is tools an testpoint as only those are supported on Debian and Ubuntu at the moment)

- perfsonar-tools
- perfsonar-testpoint
- perfsonar-core
- perfsonar-centralmanagement
- perfsonar-toolkit

Main files

- Dockerfile, docker-compose.yml, docker-bake.hcl
  - for building and running the images
  - copy ps_install_bundle.sh to the containers
- test_install_instructions.sh
  - for running the tests
- ps_install_bundle.sh
  - Run within the container and test pScheduler

test_install_instructions.sh

REPO="perfsonar-release"

declare -a OSimages=("debian:buster" "debian:bullseye" "ubuntu:bionic" "ubuntu:focal" "ubuntu:jammy")

declare -a BUNDLES=("perfsonar-tools" "perfsonar-testpoint" "perfsonar-archive" "perfsonar-core" "perfsonar-toolkit" "maddash")

We build the images from $OSimages

Then test $BUNDLES within $OSimages with ps_install_bundle.sh

enable debug mode (-D) if you want to drop into a shell after the failed install.

Stop the containers and log the results for each test

```
OSimage: debian:buster - REPO: perfsonar-release
perfsonar-tools install SUCCEEDED!
perfsonar-tools service checks RAN OK!
perfsonar-testpoint install SUCCEEDED!
perfsonar-testpoint service checks RAN OK!

OSimage: debian:bullseye - REPO: perfsonar-release
perfsonar-tools install FAILED!
QUITTING ...
perfsonar-testpoint install FAILED!
QUITTING ...

OSimage: ubuntu:bionic - REPO: perfsonar-release
perfsonar-tools install SUCCEEDED!
perfsonar-tools service checks RAN OK!
perfsonar-testpoint install SUCCEEDED!
perfsonar-testpoint service checks RAN OK!

OSimage: ubuntu:focal - REPO: perfsonar-release
perfsonar-tools install SUCCEEDED!
perfsonar-tools service checks RAN OK!
perfsonar-testpoint install SUCCEEDED!
perfsonar-testpoint service checks RAN OK!

OSimage: ubuntu:jammy - REPO: perfsonar-release
perfsonar-tools install FAILED!
QUITTING ...
perfsonar-testpoint install FAILED!
QUITTING ...
```

bundle-rpm-install-tests

- Dockerfile, docker-compose.yml, docker-bake.hcl
  - for building and running the images
  - copy ps_install_bundle.sh to the containers
- test_install_instructions.sh
  - for running the tests
- ps_install_bundle.sh
  - Run within the container and test pScheduler

feature-MNGScriptDevelopment branch

- Developed by Mark Golder
  - Improved docker bake and docker compose setup for 5.0.0 testing.
  - Adding a pscheduler-api debug script to use in testing image when install or test failed.
  - Improved text output and logging to files instead of stdout. Test specific OS and specific bundle through CLI arg.

  - Using getopts to make testing script cleaner.
  - The output and logs of the tests are more organized and fancier.

Merging files to create 1 folder

- docker-compose.yml, docker-bake.hcl
  - Can be merged and added content for deb and rpm
  - copy ps_install_bundle.sh to the containers
- Dockerfile, ps_install_bundle.sh
  - Different set of commands dedicated for deb or rpm packages
- test_install_instructions.sh
  - Same content and instructions, must direct deb to deb instructions and rpm to rpm instruction files

Challenges

- docker-compose.yml, docker-bake.hcl
  - Merged the content of rpm and deb in one file. Still not working, centOS containers not starting
  - Did some test using different files and declaring them in instructions (Dockerfile-rpm and docker-compose.rpm.yml).
- test_install_instructions.sh
  - Tested some options to run deb and rpm with the same file, some issues to be fixed.

Results

- Merge folders bundle-deb-install-tests and bundle-rpm-install-tests and have one folder to run with one command for all the tests
- To have 1 test_install_instructions.sh file and execute all the tests for all given OS and Bundles.
- Add testing for Alma Linux and other OS in the future
  - Customize the files to be compatible for required OS