PROPOSAL

Automation of Multi-Tenancy Benchmarks

By Anuj

Table of Contents

• Abstract

The Kubernetes Multi-Tenancy Working Group is chartered with exploring and defining multi-tenancy models for Kubernetes. The Multi-Tenancy Benchmarks effort focuses on measuring and validating the desired behaviors for multitenancy. Part of this effort is to automate behavioral and configuration checks on existing clusters, which will be the focus of this project. This automated test suite will help all Kubernetes users validate whether their clusters are set up correctly for multi-tenancy.

• Background

A Kubernetes cluster sometimes needs to be shared among multiple users and/or applications. In an enterprise, this may look like multiple teams deploying applications to the same Kubernetes cluster. For a service provider, this may be multiple different end customers with deployments managed by the same Kubernetes cluster.

As of now, Kubernetes conformance ensures that every vendor's version of Kubernetes supports the required APIs, as do open source community versions.

So similarly, Multi-Tenancy in any cluster should be ensured according to the Multi-Tenancy Profile Levels by passing their respective behavioral and configurational checks as per definitions.

Right now, there are 8 total categories and on the bases of these 3 Multi-Tenancy Profile Levels are defined which can be seen as-

PL1 < PL2 < PL3

In the order from Subset to Superset (Left to Right)

Profile Level 1 checks are all defined, whereas PL2 and PL3 still need some thinking process.

Project Implementation

This project aims to implement all the behavioral and configuration checks that are already defined under PL1 and are not implemented plus proposals as well as implementations of new validations under PL2 and PL3.

After the implementation of all validation in each Profile Level, they can be run manually from the terminal. So the next aim will be the automation of these Benchmarks like Sonobuoy does for the Kubernetes Conformance Test and getting the report of the Benchmark tests, defining which Profile Level Multi-Tenancy it passes or we can do in the opposite way to run just many steps under a specific Profile Level by running selective tests under selected Profile.

Test Definitions and implementation

Profile Level 1

The primary aim is to complete all test definitions and implementation of Profile Level 1 as it was mentioned in the task that they are going to be presented at next KubeCon. So achieving that milestone should be the first goal. Here are some PRs that help in achieving this are:

- Test definition of configure ns object limits

All other test definitions under PL1 are defined, just needs to be implemented that will be completed in the first coding phase of the timeline.

Profile Level 2

- Test definition [MTB-PL2-BC-OPS-3] #553

Automation and report generation of Benchmarks

This part involves the automation of the Multi-Tenancy Benchmarking and report generation is most likely K8s conformance test that is done using sonobuoy.

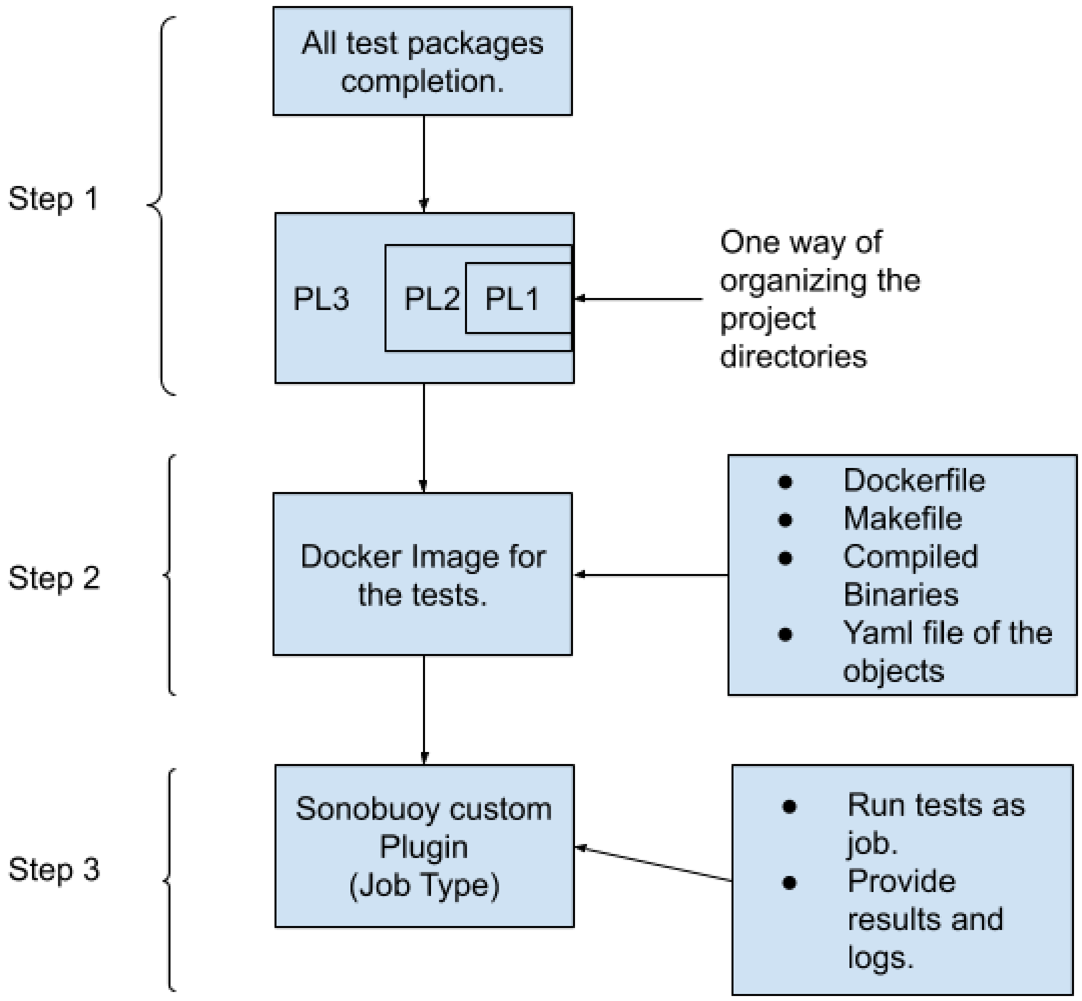
The main function of Sonobuoy is running plugins; each plugin may run tests or gather data in the cluster.

This line is mentioned in the main docs, so a custom plugin can be created for running the benchmarks and hence using the sonobuoy detailed reports it will be easier to specify the multi-tenancy profile level of the given cluster.

The easiest approach can be creating our own custom plugin MTB (Multi-Tenancy Benchmarks) which will create the job in the cluster to run the benchmarks and will create the reports for further analysis as given here.

The extensive part of this is a Go application/tool like Kubebench for CIS Benchmarking.

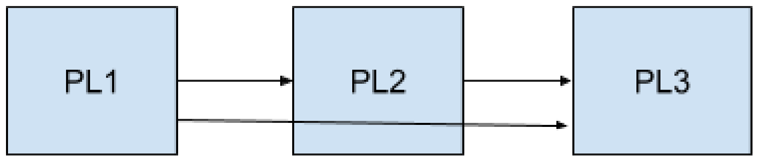
FLOW CHART OF THE AUTOMATION


Step 1
All test packages completion.
PL3
PL2
PL1
One way of organizing the project directories
Step 2
Docker Image for the tests.
Dockerfile
Makefile
Compiled Binaries
Yaml file of the objects
Step 3
Sonobuoy custom Plugin (Job Type)
Run tests as job.
Provide results and logs.

- Step 1

Project Directory Organization

These steps involve the managing and redesigning the files and directories inside the benchmarks for better and fast development as the user requires to run different benchmarks to validate its cluster to different Profile Levels that are in the nested form.
So the one way could be to have all the Profile Level different directories and then import the packages as per the subset of other Profile Levels.



So the user just can pass the Profile which they need to test the cluster, so in this case, the user just needs to pass the level as an argument in the environment variables.
The other way is shown in the flow chart above.

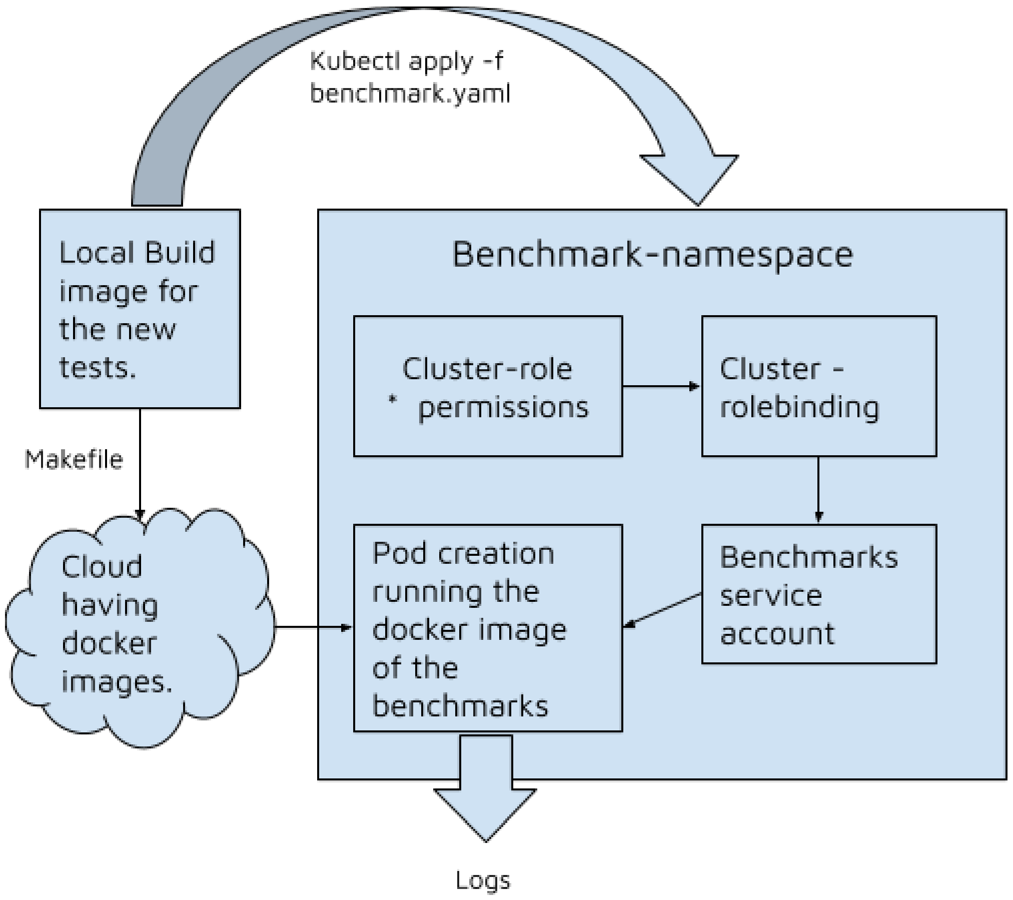
Replacement of Config File

At present compiled binary test file takes input from the static config file or it can be said it is dependent on it, so to replace this file we can have multiple options:

- The static file path can be injected in the command line testing command or directly labels or kubeconfigs.
  --config path/to/file
  --kubeconfig path/to/kubeconfig --labels key=value
- These keys and values can be stored in the environment variables which will make it easier to make benchmarks available as an image.

- ## Step 2

  This step involves delivering the benchmarks as a docker image which will be easier to manage the versions and the targeted platforms. The main effort here is to **imitate** the Kubernetes conformance tests. They have Docker images containing the required file and go binaries that will be discussed below according to our present needs.
  - Go-Binaries: This includes copying the executable files that are in the form of go binaries such as Kubectl, Ginkgo, Benchmarks. Reference
  - Yaml file: This includes the Kubernetes resource objects named after the benchmarks which will be Namespace, Service account, ClusterRole, Clusterrolebinding and the pod definition which will the run the docker image in the cluster and gathering the results for the output as the tar file or in the form of logs. Reference
  - MakeFile: For the two main functions - build and push the image. Reference

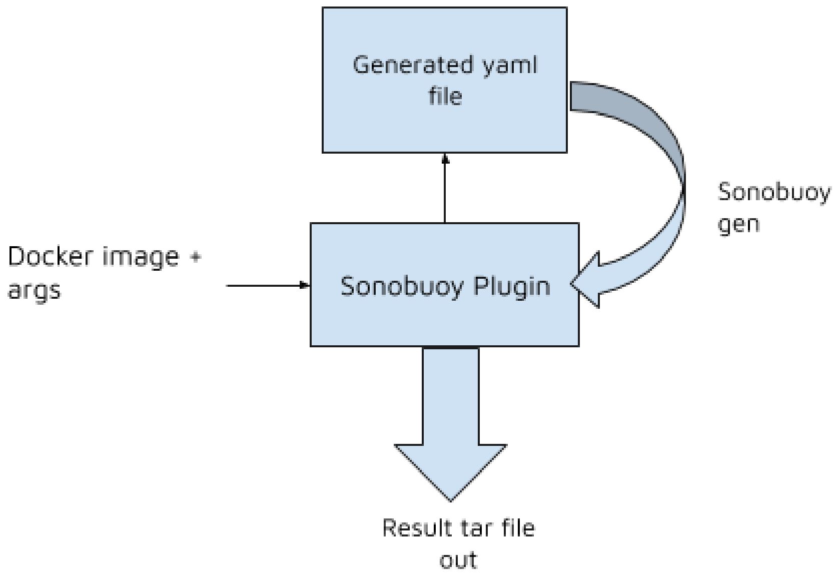

Kubectl apply -f
benchmark.yaml
Local Build image for the new tests.
Makefile
Cloud having docker images.
Benchmark-namespace
Cluster-role
* permissions
Cluster - rolebinding
Benchmarks service account
Pod creation running the docker image of the benchmarks
Logs

Advancements

- At present we are providing KubeConfigs via config file or more than that we can inject it in the command line or event set in the environment variables, but the better solution will be if the tenants' service account tokens are extracted itself so we can get rid of providing them to test file explicitly. Reference

- Step 3

This step includes the integration of benchmarks docker image with the sonobuoy i.e. creating sonobuoy custom plugin, especially for Multi-Tenancy Benchmarking.



Like other major interfaces for launching and running tests like Kubetest or Kube-bench, the main aim of this project would be like that but at primary level, sonobuoy custom plugin can be used to do the same with the help of the Docker images that are built in the second step.

- ## Dashboard Integration

Benefits/Vision

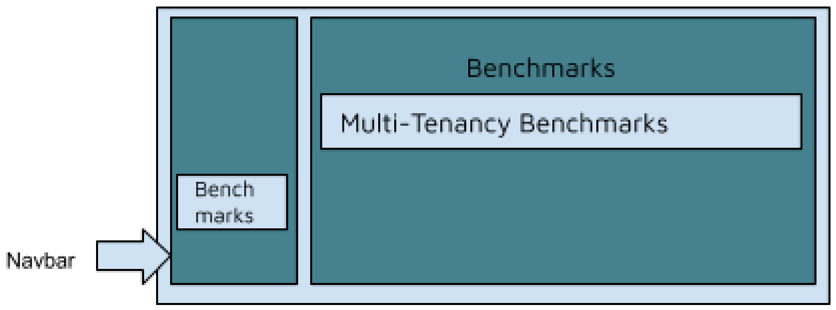
- Kubernetes Dashboard can be very useful for displaying the results of the benchmarks displaying the success/failure of each test run in the cluster.
- Benchmarks can be controlled/managed from the dashboard without any terminal-line command.
- Can display profile levels and their tests included easily.

Difficulties

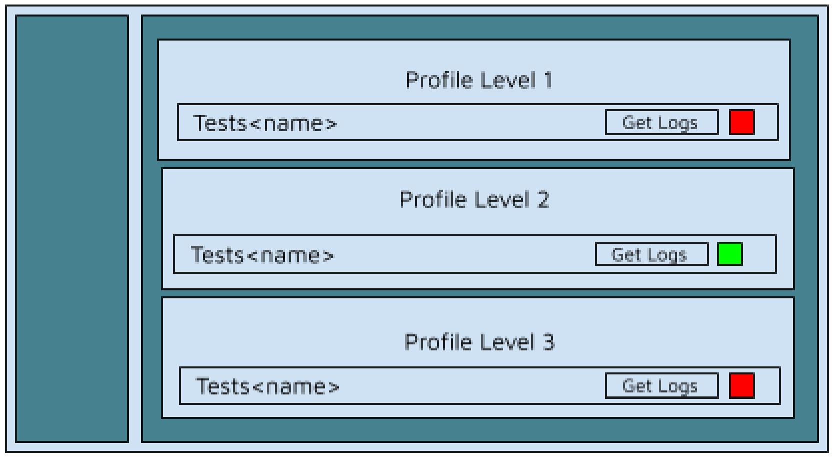
- At present there are no features or extensions of dashboard to run tests via dashboard. It only allows management of API-resources.
- It will lead to an exception to add benchmarks or a discussion with SIG-UI on where to fit this in existing architecture.
- There will be difficulty in writing the whole backend for it in the dashboard repo. Hence this should be the last step of the project, so

that we can call our automation tool/method from the backend to keep the code and time duration for the integration minimal.

Implementation



- The *Benchmarks* item will be added in the the sidebar outside of any category like CRD displaying the list of benchmarks like CIS(just a thought), Multi-Tenancy.



- After this in the main window, Benchmarks result can be shown according to the Profile test that users performed on its cluster telling the success/failure in front of the test name and the *Get logs* option button to know why it failed.
- One Method is to apply the yaml file prepared in the second step along with the benchmarks image which will run it as a job or one time and the result can be shown on the dashboard.
- Second Method could be to upload the tar file containing the logs/results of the tests and dashboard display the result in the presentable manner.

Schedule of Deliverables

The main milestones for completing this project are:

- Definition of all Behavioral and Configurational checks.I
- Implementation of the validations under all PLs.
- Automation of tests as a custom plugin of Sonobuoy.
- Implementation of Benchmarking tool.

In the community bonding period, I am planning to work with the mentor and the working group team to come with a better approach or discuss the alternate ways of automating the benchmarks.

So I will start with the completion of all validations in each PL and submit them for the first evaluation. Then I will work on automation using custom plugins by Sonobuoy/jobs in clusters to enable the user to run the tests successfully in the clusters located on any cloud platform. Up to this, we should be able to get required reports and after this, we can work on the extensive part of the project i.e. to create a go application.

Dates	Tasks
May 4	Community Bonding Period Begins
May 4 - June 1	Work with the community on the - automation and targeted platforms/users of the benchmarks. - Questions and Open Items mentioned in the MTB docs. - Integration with K8s dashboard. - Collaboration with CIS Benchmarks.
June 1	Community Bonding Period Ends
Jun 1 - June 29	- Definition and Implementation of all the tests under different categories. - Reorganizing the project directories and files to get the

	least dependencies at run time.
June 29 - July 3	First Evaluation
June 29 - July 27	- Work on serving the Benchmarks as Image.
June 27 - July 31	Second Evaluation
July 27 - August 24	- Work on integration of image with sonobuoy custom plugin. - Submit the final code and project summaries.
August 24 - August 31	Final Evaluation

General Notes

Communication is very vital to the successful completion of GSoC, as well as coordination with the Kubernetes community. The efforts I will be making in this regard are:

- Participate in all SIG-WG bi-weekly meetings, and any other relevant SIG meetings.
- Keeping my mentor(s) informed about my progress on daily progress.
- Weekly blog posts about project progress, including:
  - Deliverables for the week.
  - Hurdles (if any) encountered while delivering for the week.
  - How the hurdles were surmounted.
- Communicate with the community on Github and Slack.
- Maintain a public Google Doc with daily progress.
- Create a public Github tracker repository with relevant information about project progress.

● About Me

○ Who am I

Name: Anuj

Email: anujjangra25119@gmail.com
Github: https://github.com/phoenixking25
Slack(working-group): phoenix
Linkedin: https://www.linkedin.com/in/phoenixking25/
University: Department of Electronics and Communication, National Institute of Technology, Kurukshetra
Timezone: UTC+05:30 (Indian Standard Time)

○ Why me

I've had over 1 year of experience of working with Golang and Kubernetes open-source contributions as I also tried for the project Add Support for Custom Resource Definitions to the Dashboard in GSOC 19, in which unfortunately I didn't get select but that gave me a lot of experience to learn/work with CRDs and K8s-API server.

I have been fascinated by Kubernetes for almost 2 years, but I started using it in my projects and works about 1 year ago. I have experience of deploying applications on Kubernetes and of CI/CD using Travis with this on cloud providers such as Google Cloud Platform and AWS. My work with Kubernetes prompted my interest in the development process behind the Kubernetes project itself.

I've started contributing to Kubernetes, and to date, I've engaged in issues triage, feature requests, bug fixes and feature implementations in the Multi-Tenancy Benchmarking repository.

These are the issues and pull requests I've made to the repository:

- #397 added block other tenant resources test
- #258 MTB: Test for Block modification of resource quotas (MTB-PL1-CC-TI-1)
- [MTB] corrected relative path of config.yaml
- MTB: Test Script to create tenant CRs · Issue #492
- MTB: panic: runtime error: invalid memory address or nil pointer dereference

- Test definition of configure ns object limits
- added test definition [MTB-PL2-BC-OPS-3] #553
- corrected typo in README #554
- #259 added test for (MTB-PL1-BC-CPI-2)
- added /vendor in gitignore #567

Besides these pull requests, I have been working on implementing all validation tests of all PLs and finding ways to automate plus report generation of benchmarks.