Bluval User Guide

The Blueprint Validation Framework offers a set of tools that can be used to test Akraino deployments on different layers (hardware, os, k8s, openstack, etc).

The framework provides tests at different layers of the stack, like hardware, operating system, cloud infrastructure, security, etc. Since the project is constantly evolving, the full list of available tests can be found in the projects repo, where the tests are located under their respective layer. Each layer has its own container image built by the validation project. The full list of images provided can be found in the project's DockerHub repo.

Getting Start

You can reference how we did bluval testing for the KubeEdge BP in this meeting:

[Akraino TSC] Akraino TSC Meeting (Weekly) - Zoom

Please take a look at the above video starting around 55 minutes.

As a summary, the main reference is:

Bluval User Guide (akraino.org)

There are 2 security related tests: lynis & vuls. And there are 2 k8s related tests: kube-hunter & conformance tests.

The above page shows how to do all the 4 tests in a single framework, i.e Bluval.

I am not sure if you are required to integrate the bluval testing with your Jenkins CI/CD pipeline. I heard from Tina that it's optional. If you do want to integrate, please refer to this page:

Running bluval in CI

Again we have talked about how we integrated Bluval with CI/CD for the KubeEdge BP in the meeting, you can watch the video recording link.

Here are the steps on a high level:

1. Provision a Jenkins server for CI/CD of your BP
2. Provision a jump server, within which to run all the tests.
3. I suggest you directly download lynis and vuls to run them manually for your SUT (system under test).
4. I also suggest you directly download kube-hunter and sonobuoy to run the tests manually for your k8s cluster, if you have any,
5. Follow the procedure on Bluval User Guide (akraino.org)
6. Upload all your logs to nexus, an example of our uploaded logs are here:

Index of /sites/logs/futurewei/kubeedgees/86 (akraino.org)

The gz files are CI/CD logs from the Jenkins server. All the bluval tests logs are under the results folder.

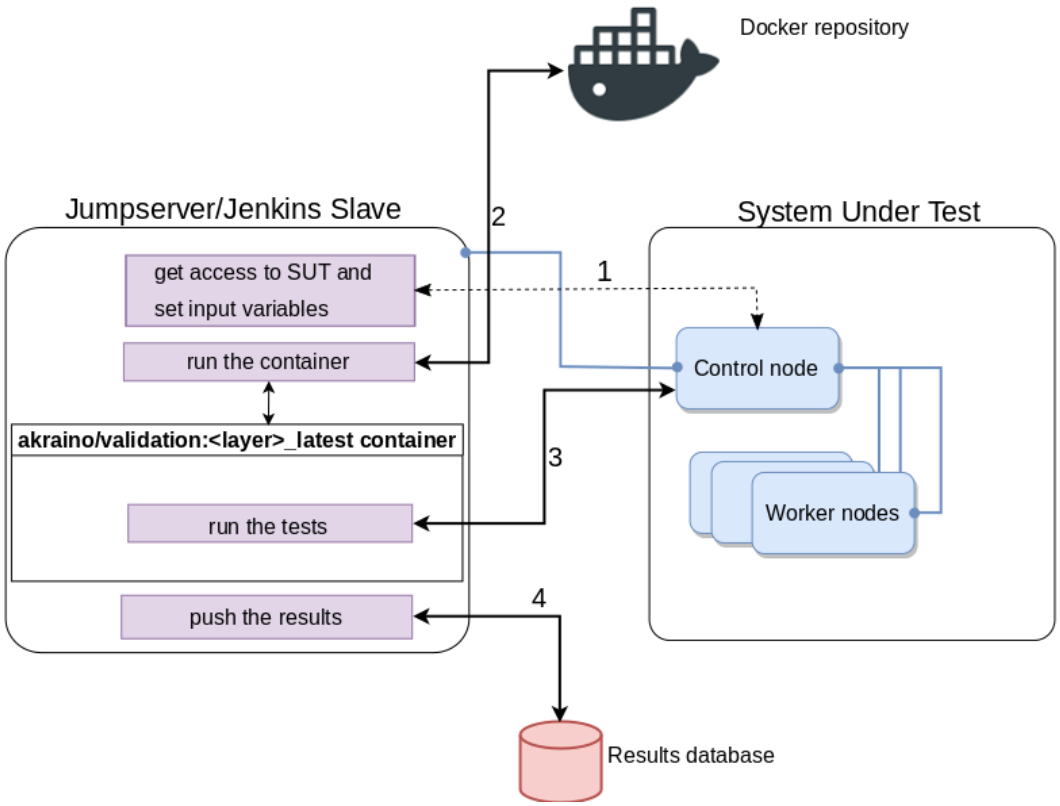
A few Kube-Hunter fixes can reference here:

KubeEdge BP Test Documents - Akraino - Akraino Confluence

Also, if you ever want to run Vuls directly, you can follow this:

https://vuls.io/docs/en/tutorial-docker.html

Topology



General Requirements

The Jumpserver can also act as a Jenkins slave and it needs to have docker installed. All the other tools needed to run the tests are available inside the container.

Some of the tests need to install the testing tools directly on the SUT, so the SUT needs to have access to Internet.

Accessing the cluster

The containers are ran from a Jumpserver with access to the SUT. Some tests use ssh to connect to the cluster, some use other tools (e.g the k8s conformance test uses the kubeclt client). Before running the test, you need to manually retrieve the files needed to access the cluster. The tools needed by each test suite are specified in this guide. Below are some guidelines on how to retrieve these files.

Tool	

kubectl	Copy the folder ~/.kube from your Kubernetes master node to a local folder (e.g. ~/kube).
	`ubuntu@iec01:~$ kubectl get node |grep master`
	`iec01 Ready master 45h v1.13.0`
	`ubuntu@iec01:~$ scp .kube/* ubuntu@jumpserver:/home/ubuntu/kube`

ssh	ssh key file Copy the ssh key to access your cluster to a local folder (e.g. ~/.ssh). The location of where to get the ssh key varies based on how the installation was done. In case you are not sure where to get it, contact your cluster administrator.

Using blucon to run the tests

Each layer has an individual container from which the tests run. The blucon script can be used to both start the container and run the tests. The tests run using the bluval tool. The tool takes as parameters the layer and the blueprint for which to run the tests. A base blueprint is given as an example here.

The steps to do that are:

1. Clone the validation repo

```
ubuntu@jumpserver:~$ git clone http://gerrit.akraino.org/r/validation
```

2. Customize the blueprint - Optional step

   In case your blueprint yaml file is not already in the validation repo inside the bluval folder, or you want to run a subset of the tests, you can create your own bluval yaml file and mount it later in the container. Below is an example of a demo blueprint that will call just the k8s conformance tests.

```
ubuntu@jumpserver:~$ cat validation/bluval/bluval-demo.yaml
blueprint:
    name: demo
    layers:
        - k8s

    k8s: &k8s
        -
            name: conformance
            what: conformance
            optional: "False"
```

3. Fill in volumes.yaml file

   Fill in the file validation/bluval/volume.yaml with the data that applies to your setup. The volumes that don't have a *local* value set will be ignored. I n the example below, only the following volumes are set:

   - Location to the kube config files needed for access to the cluster
   - Location to the customized blueprint file
   - Location to where to store the results

```
ubuntu@jumpserver:~$ cat validation/bluval/volumes.yaml |tail -n +23

volumes:
    # location of the ssh key to access the cluster
    ssh_key_file:
        local: ''
        target: '/root/.ssh'
    # location of the k8s access files (config file, certificates, keys)
    kube_config_dir:
        local: '/home/ubuntu/kube'
        target: '/root/.kube/'
    # location of the customized variables.yaml
    custom_variables_file:
        local: ''
        target: '/opt/akraino/validation/tests/variables.yaml'
    # location of the bluval-<blueprint>.yaml file
    blueprint_dir:
        local: '/home/ubuntu/validation/bluval'
        target: '/opt/akraino/validation/bluval'
    # location on where to store the resulst on the local jumpserver
    results_dir:
        local: '/home/ubuntu/results'
        target: '/opt/akraino/results'
```

   ⚠ Do not modify the *target* part of each volume. These paths will be automatically created inside the container when started, and are fixed paths that the tools inside the container expect to exist as is.

   Depending on which tool is used for connecting to the cluster (ssh, kubectl, etc), the corresponding volume needs to be mounted inside the container. To see which are the specific containers for each layer, check the second section in the same volumes.yaml file

```
ubuntu@jumpserver:~$ cat validation/bluval/volumes.yaml |tail -n +45

# parameters that will be passed to the container at each layer
layers:
    # volumes mounted at all layers; volumes specific for a different layer are below
    common:
        - custom_variables_file
        - blueprint_dir
        - results_dir
    hardware:
        - ssh_key_file
    os:
        - ssh_key_file
    networking:
        - ssh_key_file
    k8s:
        - ssh_key_file
        - kube_config_dir
    k8s_networking:
        - ssh_key_File
        - kube_config_dir
```

4. Update variables.yaml

   Fill in the file with your confidential information like IP address/username/passwords and environment specific information. This file grows along with no of testcases.

```
ubuntu@jumpserver:~$ cat validation/tests/variables.yaml
## ... snippet

### Input variables cluster's master host
host: 172.28.17.206             # cluster's master host address
username: cloudadmin            # login name to connect to cluster
ssh_keyfile: /root/.ssh/id_rsa  # Identity file for authenticatio

...
```

5. Run the tests

```
ubuntu@jumpserver:~$ bash validation/bluval/blucon.sh [-l <LAYER>] [-o] [-n host] <Bluprint Name>
```

Getting the results

The results are stored inside the container in the */opt/akraino/results* folder. If a volume was mounted to this folder (e.g. */home/ubuntu/results*) then they can be viewed from the host where the test was ran. The results can be viewed with a browser, using the *report.html* file that the robot framework provides.

```
ubuntu@jumpserver:~$ ls results/k8s/conformance/
201909110859_sonobuoy_376a4ddc-4498-49fc-af2e-999242c4c245.tar.gz  Conformance.Conformance.log  log.html
output.xml  report.html
```

Development Environment / Trouble Shooting

These following steps helps you to setup development environment if you want to contribute back to community or trouble shoot the issue for yourself.

blucon.sh is building a docker image, mounts required directories and runs a *blucon* container.

blucon container mounts directories as mentioned in volumes.yaml and runs a container to validate each layer. Entire docker command including mount points is printed on the screen. As part of your development process you can modify this docker statement a little and start the container manually and play with it. Couple of modifications needed for every developer are ***-it*** to start the container in the interactive mode and ***-v /home/ubuntu/validation/:/opt/akraino/validation/*** to mount your local code to container. In this way you can modify code in your favorite IDE and just run it in the container. If you are happy with the changes you can commit them to http://gerrit.akraino.org/r/validation

Here is the example to develop/debug OS layer container.

```
... On your host eg. for OS layer
ubuntu@jumpserver:~$ docker run --rm -it \
  -v /home/ubuntu/validation:/opt/akraino/validation \
  -v /home/ubuntu/results:/opt/akraino/results \
  -v /home/ubuntu/.ssh:/root/.ssh \
  akraino/validation:os-latest /bin/sh

... Within in the container
# cd /opt/akraino/validation && python bluval/bluval.py -l os -o rec

... Its running all the testcases mentioned in bluval-<bluprint_name>.yaml OS layer, and prints commands on
screen. One example is here.
Invoking ['robot', '-V', '/opt/akraino/validation/tests/variables_updated.yaml', '-d', '/opt/akraino/results/os
/lynis', '-b', 'debug.log', '/opt/akraino/validation/tests/os/lynis']

... You can convert above example as below and run specific test suites
# robot -V /opt/akraino/validation/tests/variables_updated.yaml \
  -d /opt/akraino/results/os/lynis \
  -b debug.log \
  /opt/akraino/validation/tests/os/lynis

... You can add -t "Your testcase name" and run specific testcase, here is the example.
# robot -V /opt/akraino/validation/tests/variables_updated.yaml \
  -d /opt/akraino/results/os/ltp \
  -b debug.log \
  -t "RunLTP syscalls madvise only" \
  /opt/akraino/validation/tests/os/ltp
```

Tests can be invoked directly using the bluval.py script located at */opt/akraino/validation/bluval/*

Tests are located at */opt/akraino/validation/tests/* and they can be locally modified to print more output.

Common Issues

- FileNotFoundError: [Errno 2] No such file or directory: '/opt/akraino/results/test_info.yaml'

Please take a look at volumes.yaml, results_dir and make sure that entry is correct.

- invalid argument "akraino/validation:blucon-(HEAD" for "-t, --tag" flag: invalid reference format

Please make sure you are not on "detached HEAD". You can use *git checkout -b <new-branch-name>* to name that branch

The OS layer

TBD

The Hardware layer

TBD

The Networking layer

TBD

The Kubernetes layer

Test suites	Access type
k8s confrormance	kubectl tool
ha	ssh key

The OpenStack layer

TBD