

Running bluval in CI

This guide describes how to use the tools provided by the validation project in an existing Continuous Integration pipeline. It is assumed that the reader has knowledge of Jenkins, Jenkins Job Builder and CI principles. For general information on CI, check [CI Environment Overview](#) and [Jenkins Guide](#)

The Akraino community has two types of labs:

- Integrated with Linux Foundation Jenkins Master. The jobs that run can be seen at <https://jenkins.akraino.org>. To connect a Jenkins slave to LF master, follow the instructions at [Jenkins Peering Guide](#)
- Private labs, that run individual Jenkins Master

Both labs push the logs to a public repository on Nexus: <https://nexus.akraino.org/content/sites/logs/>. For more details on pushing the logs to Nexus, please check [How to: Push Logs to Nexus](#)



In order to run Bluval in CI, the Jenkins slave must have docker and python3 installed.

The JJB templates

The validation project provides JJB templates that are used to automate running the tests in CI. The templates - *akraino-validation-lab-daily* and *bluval-run-daily-tests* - are located in the Akraino ci-management project, in [jib/akraino-templates/akraino-validation-templates.yaml](#). A Jenkins job is created for each of these templates:

https://gerrit.akraino.org/r/gitweb?p=ci-management.git;a=blob_plain;f=jjb/validation/lab.yaml

```
- project:
  name: lab
  project-name: validation
  project: validation
  build-node: '{build-node}'
  stream:
    - master:
        branch: '{stream}'
        gs-pathname: ''
        disabled: false
  validation_lab:
    - enea
    #- <add_new_lab_here>
  jobs:
    - akraino-validation-lab-daily
```

https://gerrit.akraino.org/r/gitweb?p=ci-management.git;a=blob_plain;f=jjb/validation/bluval.yaml

```
- project:
  name: bluval
  project-name: validation
  project: validation
  build-node: '{build-node}'
  group-id: org.akraino.validation
  refspec: refs/heads/master
  branch: master
  view:
    - project-view
  blueprint:
    - iec
    - rec
    #- <add_new_blueprint_here>
  stream:
    - master:
        branch: '{stream}'
        gs-pathname: ''
        disabled: false
  jobs:
    - bluval-run-daily-tests
```

Labs integrated with LF Jenkins

In order to enable running validation on lab that is integrated with LF, you need to

1. Submit a patch in the validation project that defines the tests that will be ran by your blueprint. The full list of available tests are in the [bluval-base.yaml](#) file. Check [bluval-iec.yaml](#) as an example.
2. Submit a patch in the ci-management project that:
 - a. Add your lab in *validation_lab* list of the *akraino-validation-lab-daily* job, in the [lab.yaml](#) file; this value must be the same as the silo name where you are pushing the results into nexus
 - b. Add your blueprint to the *blueprint* list of the *bluval-run-daily-tests* job, in the [bluval.yaml](#) file; this value must be the same as the one used at step #1
 - c. Call the *akraino-validation-lab-daily* job from your project and set the predefined parameters correspondent with your deployment; make sure that the validation job is called after the installation of the cluster. In the example below, the validation job is called from the [iec installation template](#) as a builder step:

```

- job-template:
  id: akraino-iec-install
  name: 'iec-{iecType}-{installer}-{deploy_type}-{os}-daily-{stream}'
  .....
  builders:
    - trigger-builds:
      # call the job that deploys the k8s cluster
      - project: 'iec-{iecType}-deploy-{installer}-{deploy_type}-{os}-daily-{stream}'
        current-parameters: true
        predefined-parameters:
          DEPLOY_SCENARIO={scenario}
        same-node: true
        block: true
      # call the validation job for the enea lab
      - project: 'validation-enea-daily-{stream}'
        same-node: true
        current-parameters: true
        predefined-parameters: |
          CLUSTER_MASTER_IP=$K8S_MASTER_IP
          CLUSTER_SSH_USER=$K8S_SSH_USER
          CLUSTER_SSH_PASSWORD=$K8S_SSH_PASSWORD
          CLUSTER_SSH_KEY=$K8S_SSH_KEY
          BLUEPRINT={project-name}
          LAYER=k8s
          VERSION=master
          OPTIONAL=false
        block: true
  .....

```

Private labs

Each private lab has their own particularities in terms of how Jenkins master is ran. Here we provide some guidelines on how to take advantage of the LF jobs, but they need to be adapted to the particularities of your lab.

If you use JJB to run the jobs in CI, then you can use the templates as defined for the labs integrated with LF Jenkins at the previous step.

If you don't use JJB, you can generate the xml files from the templates above and inject them into your Jenkins Master. In order to generate the xml files, you need to install the jenkins-jobs tool and follow the steps below:

```

# install Jenkins job builder tool (it can be done in a virtual environment)
user@jenkins_slave:$ pip install jenkins-job-builder
# clone the ci-management repo
user@jenkins_slave:$ git clone "https://gerrit.akraino.org/r/ci-management"
# initialize the LF global jjb submodule
user@jenkins_slave:$ cd ci-management && git submodule update --init
# add your lab, blueprint and job as described in the section for Labs integrated with LF Jenkins
# generate the xml for your lab job
user@jenkins_slave:~/ci-management$ jenkins-jobs test jjb/ validation-<lab_name>-daily-<stream> -o <output-
folder>
# generate the xml for the blueval job
user@jenkins_slave:~/ci-management$ jenkins-jobs test jjb/ validation-<lab_name>-daily-<stream> -o <output-
folder>

```

The *output-folder* will contain files with the XML configurations. Note that since the validation job is a template with variables in its name (e.g. *validation-<lab_name>-daily-<stream>* in the command above) must be manually expanded before use. For example, the commands below will generate the validation jobs for the *iec* blueprint ran in *enea* lab:

```

user@jenkins_slave:~/ci-management$ jenkins-jobs test jjb/ validation-enea-daily-master -o generate_xml
user@jenkins_slave:~/ci-management$ jenkins-jobs test jjb/ bluval-daily-master -o generate_xml
user@jenkins_slave:~/ci-management$ ls -al generate_xml/
total 40
drwxrwxr-x 2 user user 4096 mar  2 17:43 .
drwxrwxr-x 9 user user 4096 mar  2 17:39 ..
-rw-rw-r-- 1 user user  934 mar  2 17:43 '00-Empty View'
-rw-rw-r-- 1 user user 13255 mar  2 17:43 bluval-daily-master
-rw-rw-r-- 1 user user  858 mar  2 17:43 CLM
-rw-rw-r-- 1 user user 4182 mar  2 17:39 validation-enea-daily-master

```

Details about how the tests are ran

In order to run the tests, the validation JJB template calls the [run_bluval.sh](#) script. This script creates the environment needed by Blueprint Validation Framework and it calls blucon tool to run the tests. The script can be used as-is outside the ci-management jobs, and can be customized for your blueprint running in a private lab.

usage: `./jjb/shell/run_bluval.sh -n <blueprint_name>`

The script can be called using parameters detailed below:

`-n <blueprint_name>` The blueprint name is mandatory. The tests are run using the file `validation/bluval/bluval-$blueprint_name.yaml`

`-r <results_dir>` The location to where to store the results. If this parameter is not provided then path `$(pwd)/results` is used

`-b <blueprint_yaml>` The yaml file used to run the tests. This file is copied in `validation/bluval/` folder before running the blucon tool

`-k <k8s_config_dir>` The location where the script copies the kube config files needed for access to the cluster. If not provided the path `$(pwd)/kube` is used

`-j <k8s_master>` The IP address of k8s master node. If this parameter is not given the script will use `$K8S_MASTER_IP`

`-u <ssh_user>` User to connect to k8s master node. If this is not set then `$K8S_SSH_USER` will be used

`-s <ssh_key>` Path to key used to connect to k8s master node. If login is done using password then the `$K8S_SSH_PASSWORD`

`-l <layer>` The blueprint layer to be used. If not set, `$LAYER` will be used.

`-o :` If used, the blucon will run the optional tests also. If not set, then `$OPTIONAL` will be used.

`-v <version>` Version to be used. If not set than `$VERSION` will be used.

The script can be configured by setting shell variables before calling the script or using parameters. Below is an example of shell configuration before calling the script.

```
export K8S_MASTER_IP=172.16.10.36
```

```
export K8S_SSH_USER=ubuntu
```

```
export K8S_SSH_KEY=/var/lib/openssh/mcp.rsa
```

```
export BLUEPRINT=iec
```

```
export LAYER=k8s
```

```
export VERSION=master
```

```
export OPTIONAL=false
```

```
./jjb/shell/run_bluval.sh -n iec
```