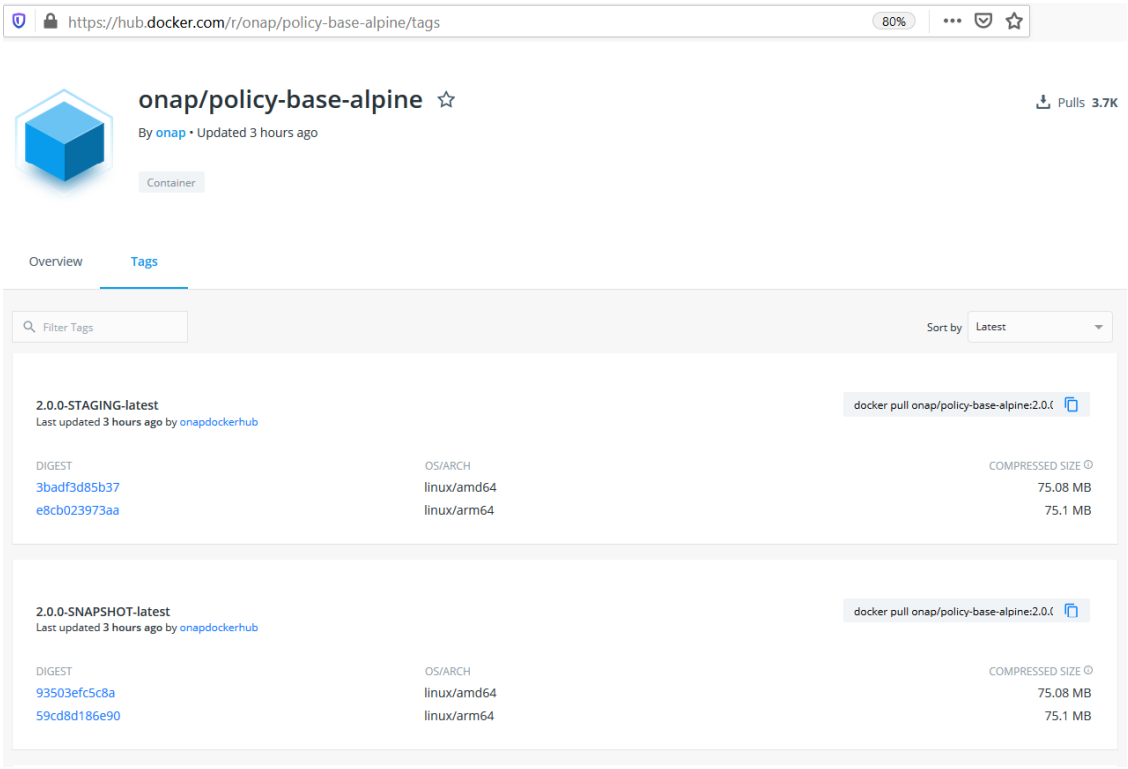
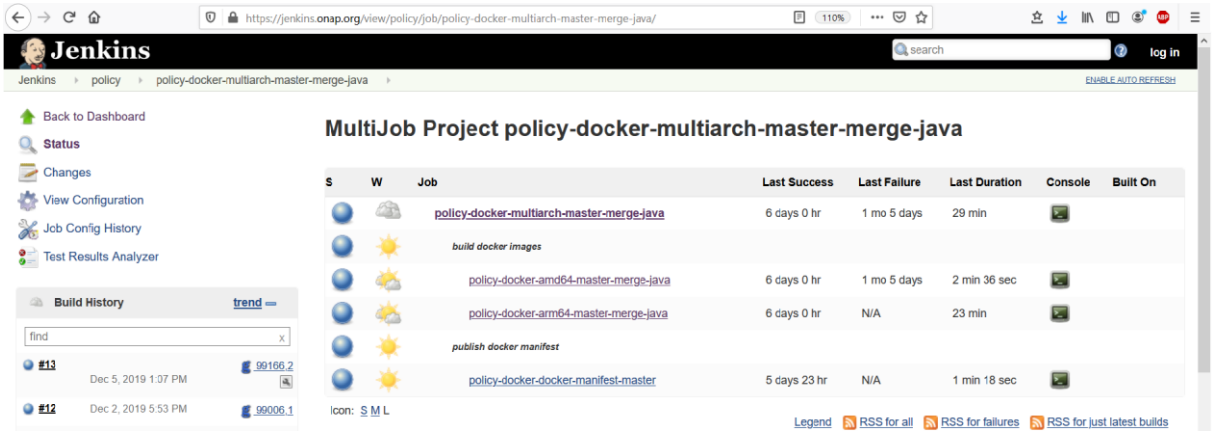
Overview

In order to build multi-arch images in ONAP, the CI infrastructure had to go through the following changes:

1. All images are pushed to the official dockerhub account instead of nexus (https://hub.docker.com/u/onap). The reasoning behind this is that nexus does not support multi-arch images (manifest lists). In the figure below it's an example of dockerhub repo containing multi-arch images for policy base image



2. For each job that builds images on x86 the following multijob is created:
   - One parent job that calls the rest of the jobs
   - Two child jobs that build and pushes the images for each arch (x86 and aarch64); these jobs run in parallel on different hosts
   - One child docker manifest job that will create the manifest list image (which will point to the previously built images)



Above it's an example of multijob created for the policy/docker project

Job breakdown

There are 3 Jenkins Jobs that need to be taken into consideration when building multi-arch images in ONAP

- The merge job, which is ran whenever a patch is merged
- The stage job, which is ran daily
- The release job, which is ran whenever the PTL/team decides to release their images

The merge job

It builds images with the SNAPSHOT tags. Changes on tags build for the policy project repos are summarized in the table below:

Tags built before multiarch (e.g. policy-docker-master-merge-java)	**Tags build for x86 after multiarch** (e.g. policy-docker-amd64-maven-docker-stage-master)	**Tags build for aarch64 after multiarch** (e.g. policy-docker-arm64-maven-docker-stage-master)	**Tags build for manifest list** (e.g. policy-docker-docker-manifest-master)
latest	latest-amd64	latest-arm64	latest
<release_version>-SNAPSHOT	<release_version>-SNAPSHOT-amd64	<release_version>-SNAPSHOT-arm64	<release_version>-SNAPSHOT
<release_version>-SNAPSHOT-<timestamp>	-	-	<release_version>-SNAPSHOT-<timestamp>
<release_version>-SNAPSHOT-latest	<release_version>-SNAPSHOT-latest-amd64	<release_version>-SNAPSHOT-latest-amd64	<release_version>-SNAPSHOT-latest

Note that the timestamp tag is no longer created by the staging jobs (with maven), but it's created by the manifest-list job.

The template for this job is *{project-name}-multiarch-{stream}-merge-java* and it's found at [1]. This template

- calls the *{project-name}-{stream}-merge-java* job from the same file [2] for each architecture; it's doing that by adding the architecture of the image into the project's name (e.g. [3] & [4])
- calls the *{project-name}-docker-manifest-{stream}* job [5] to create the manifest list image with the images previously built

The stage job

It builds images with the STAGING tags. Changes on tags build for the policy project repos are summarized in the table below:

Tags built before multiarch (e.g. policy-docker-maven-docker-stage-master)	**Tags build for x86 after multiarch** (e.g. policy-docker-amd64-maven-docker-stage-master)	**Tags build for aarch64 after multiarch** (e.g. policy-docker-arm64-maven-docker-stage-master)	**Tags build for manifest list** (e.g. policy-docker-docker-manifest-master)
latest	latest-amd64	latest-arm64	latest
<release_version>	<release_version>-amd64	<release_version>-arm64	-
<release_version>-timestamp	-	-	<release_version>-timestamp
<release_version>-STAGING-latest	<release_version>-STAGING-latest-amd64	<release_version>-STAGING-latest-arm64	<release_version>-STAGING-latest

Note that the timestamp tag is no longer created by the staging jobs (with maven), but it's created by the manifest-list job.
Note that the manifest list for the release tag is no longer built by the staging job. This tag will be created by the release job only.

The template for this job is *{project-name}-multiarch-docker-stage-{stream}* and can be found at [6]. This template

- calls the *{project-name}-maven-docker-stage-{stream}* job from the global jjb templates [7] for each architecture; it's doing that by adding the architecture of the image into the project's name (e.g. [8] & [9])
- calls the *{project-name}-docker-manifest-{stream}* job [10] to create the manifest list image with the images previously built

The release job

The self-release job is described in the process at [11] and [12]. This job doesn't build any images, it just re-tags the specified image with the release tag. To make the release job use the new multi-arch images, in the *releases/container-release.yaml* file the *container_pull_registry* and *container_push_registry* need to be set to the dockerhub repo, and the *container name* needs to be the name of the manifest list image. For example:

```
$ cat releases/container-release.yaml
---
distribution_type: container
container_release_tag: 2.0.0
container_pull_registry: hub.docker.com
container_push_registry: hub.docker.com
project: onap
ref: d1b9cd2dd345fbeec0d3e2162e008358b8b663b2
containers:
    - name: policy-base-alpine
      version: 2.0.0-SNAPSHOT-20191211152916
    - name: policy-common-alpine
      version: 2.0.0-SNAPSHOT-20191211152916
```

Note that the configuration shown in this example has not been tested yet so the final version for it may vary.

Using the multiarch templates

The multiarch templates have been added in https://gerrit.onap.org/r/c/ci-management/+/92707
In order to use the templates described above, they need to be called from the project's jjb yaml file. Please check https://gerrit.onap.org/r/c/ci-management/+/95179 as a reference.

The multiarch templates call the existing templates that are used for the x86 jobs *({project-name}-maven-docker-stage-{stream}* and *{project-name}-{stream}-merge-java*). This is implementing by changing the *project-name* variable to append the architecture suffix. For example, instead of calling *{project-name}-{stream}-merge-java* for the *policy-api* project name, it is now called with *policy-api-amd64* and *policy-api-arm64* names.

```
- '{project-name}-{stream}-merge-java':
    docker-pom: 'pom.xml'
    mvn-params: '-P docker'
    build-node: ubuntu1604-docker-8c-8g
```

```
- '{project-name}-{stream}-merge-java':
    project-name: 'policy-api-amd64'
    docker-pom: 'pom.xml'
    mvn-params: '-P docker -Ddocker.pull.registry=docker.io -Ddocker.push.registry=registry-1.docker.io'
    build-node: ubuntu1604-docker-8c-8g
    pattern: 'do_not_match_any_file'
- '{project-name}-{stream}-merge-java':
    project-name: 'policy-api-arm64'
    docker-pom: 'pom.xml'
    mvn-params: '-P docker -Dmaven.test.skip=true -Ddocker.pull.registry=docker.io -Ddocker.push.registry=registry-1.docker.io'
    build-node: ubuntu1604-docker-arm64-4c-2g
    pattern: 'do_not_match_any_file'
- '{project-name}-multiarch-{stream}-merge-java'
```

The configuration for each job needs to be updated given the specifics of each project. In the patch given as example, the changes in configuration are described below.

```
- '{project-name}-{stream}-merge-java':
    project-name: 'policy-api-amd64'
    docker-pom: 'pom.xml'
    mvn-params: '-P docker -Ddocker.pull.registry=docker.io -Ddocker.push.registry=registry-1.docker.io'
    build-node: ubuntu1604-docker-8c-8g
    pattern: 'do_not_match_any_file'
```

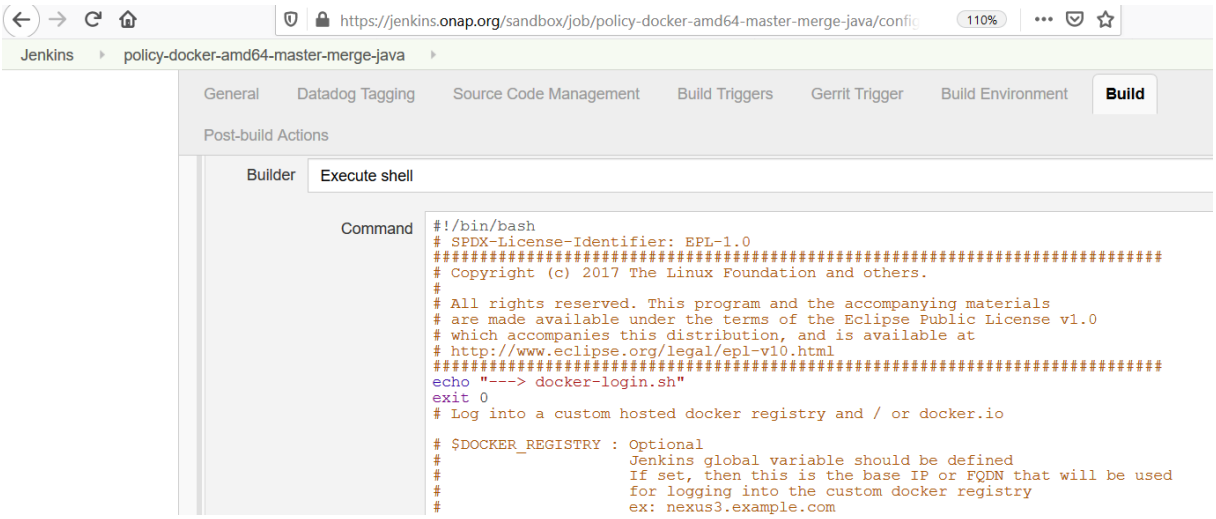
project-name needs to have the architecture in its suffix
mvn-params needs to have the pull and push registry flags set; note that the push registry is registry-1.docker.io as a workaround to this bug
build-node is specific to the architecture (*ubuntu1604-docker-8c-8g for x86* and *ubuntu1604-docker-arm64-4c-2g for aarch64*)
pattern needs to be set to something that will never be mached; the multiarch jobs are triggered by the parent job, but since the child jobs also have their individual triggers (defined by their templates) we need to disable the latter; this is achieved by giving a regex pattern that will never be matched

Testing the jobs in sandbox

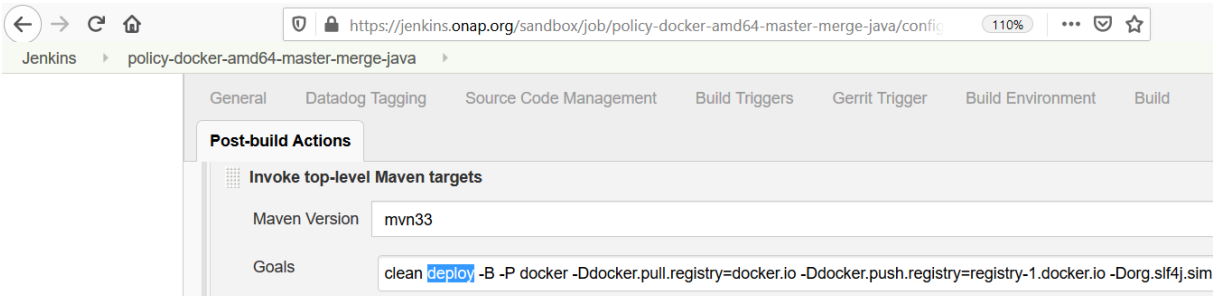
All the jobs can be tested in sandbox, which is available at [13]. To push the jobs in sandbox, follow the instructions at [14].

Since sandbox is a testing environment, it can't push the images in the onap dockerhub account. Therefore, before running a job in sandbox, it needs to be modified make the changes described below, otherwise the jobs will fail.

1. Not try login to dockerhub. This is achieved by entering the configuration of the job in the web browser and skip running the docker-login.sh script (adding an *exit 0* right at the begging of the script is one way to do it). See the picture below for an example on how to disable the login to docker from the job configuration in Sandbox

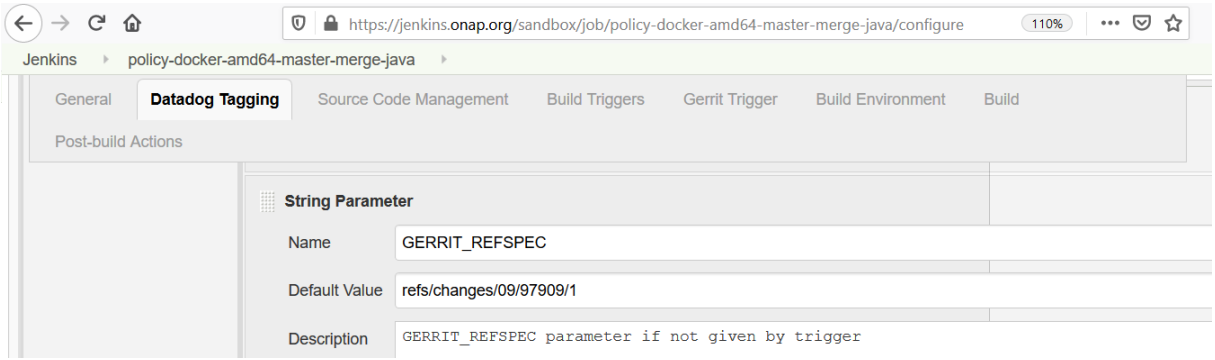


2. Not try to push to dockerhub. This is achieved by entering the configuration of the job in the web browser and changing the maven targets from *deploy* to *install*



Patches made in the onap projects can be directly linked in the jobs verified in sandbox, before they are merged. To do that, the following fields need to be changed in the job configuration, from the web browser. **Note** that when verifying the whole multijob, these changes need to be applied to all the child jobs that are tested in sandbox.

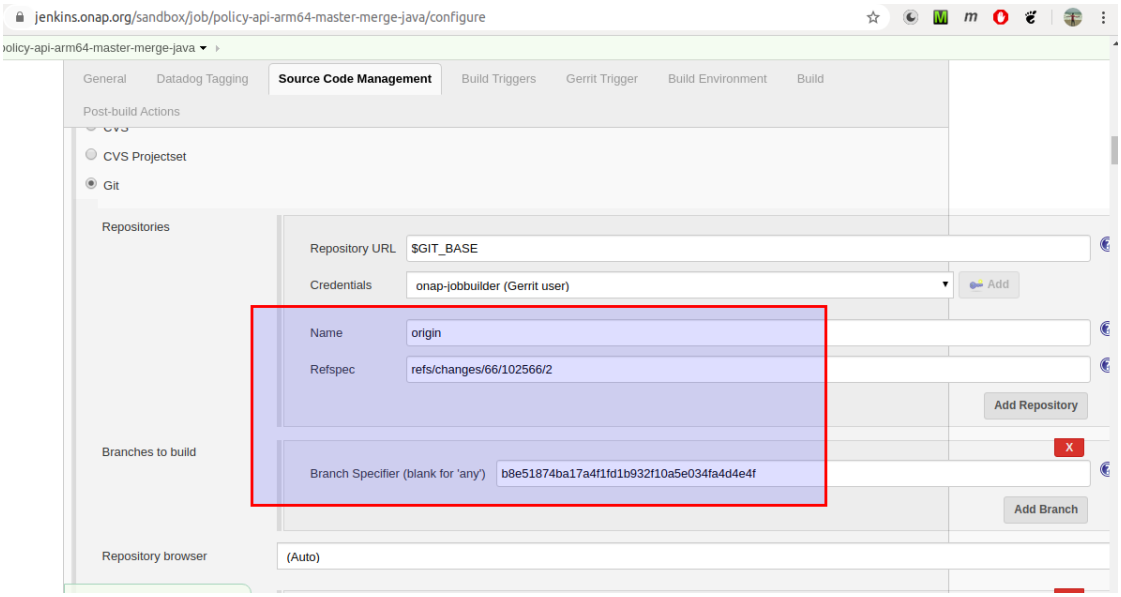
a) The git refspecs need to be changed from *master* to the patch refspecs



b) In Source Code Management section,

In the *Git->Repository* section click on the *Advanced* button then fill in

- refspec has to be updated with refs/changes/xx/xxxxx/x (refspec of the patch)
- In Branches to build, branch specifier section, the SHA nr of the patch (commit-sha) needs to be updated



References

Wiki links

https://wiki.onap.org/display/DW/Migration+to+DockerHub

https://wiki.lfnetworking.org/display/LN/Topic+Proposals%2C+June+%2719?preview=/15630468/15631312/Using%20Docker%20Hub%27s%20public%20registry%20for%20CI_CD%20builds%20-%20An%20ONAP%20use-case.pdf

Patches made so far

https://gerrit.onap.org/r/c/ci-management/+/92707
https://gerrit.onap.org/r/c/ci-management/+/95505
https://gerrit.onap.org/r/c/ci-management/+/95599
https://gerrit.onap.org/r/c/ci-management/+/95179

https://gerrit.onap.org/r/c/policy/docker/+/97909
https://gerrit.onap.org/r/c/policy/docker/+/93953
https://gerrit.onap.org/r/c/policy/api/+/93957
https://gerrit.onap.org/r/c/policy/engine/+/93978
https://gerrit.onap.org/r/c/policy/xacml-pdp/+/93980
https://gerrit.onap.org/r/c/policy/pap/+/93979
https://gerrit.onap.org/r/c/policy/drools-pdp/+/93974
https://gerrit.onap.org/r/c/policy/drools-applications/+/93972
https://gerrit.onap.org/r/c/policy/distribution/+/93971
https://gerrit.onap.org/r/c/policy/apex-pdp/+/93956

https://gerrit.onap.org/r/c/integration/csit/+/94742

https://gerrit.onap.org/r/c/oom/+/94816

Links in this document

[1] https://github.com/onap/ci-management/blob/master/jjb/global-templates-java.yaml

[2] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-java.yaml;h=bcf4bd3bb54b4b761adcb1e94e14ba70fdf5a53f;hb=HEAD#l441

[3] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-java.yaml;h=bcf4bd3bb54b4b761adcb1e94e14ba70fdf5a53f;hb=HEAD#l1157

[4] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/policy/policy-docker-base-common.yaml;h=63daef949ab724ce0c58a556c37841f10d47a70d;hb=HEAD#l36

[5] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-java.yaml;h=bcf4bd3bb54b4b761adcb1e94e14ba70fdf5a53f;hb=HEAD#l1168

[6] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-docker.yaml;h=eecedd464bfaf151748bf059ea1e7ec5beda4e07;hb=HEAD#l826

[7] https://github.com/lfit/releng-global-jjb/blob/da7a332a3179b8b8ddb23f60086884af3dab1365/jjb/lf-maven-jobs.yaml#L920

[8] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-docker.yaml;h=eecedd464bfaf151748bf059ea1e7ec5beda4e07;hb=HEAD#l892

[9] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/policy/policy-docker-base-common.yaml;h=63daef949ab724ce0c58a556c37841f10d47a70d;hb=HEAD#l51

[10] https://gerrit.onap.org/r/gitweb?p=ci-management.git;a=blob;f=jjb/global-templates-docker.yaml;h=eecedd464bfaf151748bf059ea1e7ec5beda4e07;hb=HEAD#l903

[11] https://github.com/lfit/releng-global-jjb/blob/master/docs/jjb/lf-release-jobs.rst

[12] https://wiki.onap.org/display/DW/Self+Release+Workflow

[13] https://jenkins.onap.org/sandbox/

[14] https://docs.releng.linuxfoundation.org/en/latest/jenkins-sandbox.html