ci-management Documentation

Release 0.0.1

Ramesh Babu Thoomu

Aug 26, 2018

CI Process

CHAPTER 1

Summary

Welcome to the Hyperledger Fabric community! The Hyperledger Fabric (and associated) projects use various tools and workflows for the continuous project development.This documentation will assist you in using these tools and understanding the workflow(s) for our contributors while working with the Fabric CI infrastructure.

CHAPTER 2

Finding Help on Hyperledger CI

We are excited that you want to contribute to the Continuous Integration/Release Engineering efforts. You're in the right place to get started.In this event that you need additional assistance, we encourage you to engage with our CI contributors via the following channels:

- #ci-pipeline channel on Rocket.Chat (For general continuous integration discussions) https://chat.hyperledger.org/channel/ci-pipeline
- #fabric-ci channel on Rocket.Chat (For fabric continuous integration discussions) https://chat.hyperledger.org/channel/fabric-ci
- #infra-support channel on Rocket.Chat (For general infrastructure discussions) https://chat.hyperledger.org/channel/infra-support
- Send an email to the fabric@lists.hyperledger.org mailing list
- Contact helpdesk@hyperledger.org for any infrastructure support

CHAPTER 3

Common Job Types

There are several Jenkins job types that are common across most Hyperledger Fabric projects. In some cases, you may or may not see all the common job types in every project.

Let's have a look at the common job types.

CHAPTER 4

Verify Jobs

Verify jobs get triggers when a “patchset-created-event” is triggered. This usually happens when a patchset is submitted to Gerrit repo. All verify jobs will depend on the patchset’s parent commit (not the latest commit of the repo) and patchset commit. Developers have to rebase their patchset to build the patchset on latest commit of the repo. A verify job can also be triggered from Gerrit by posting a comment in the patchset. Every Verify job has a unique trigger comment for that job.

CHAPTER 5

Merge Jobs

Merge jobs get triggers when a “change-merged-event” event is triggered. This usually happens when a patchset is merged in Gerrit repo. In all the merge jobs, Jenkins clones the latest code and perform the tests unlike verify jobs. Just like Verify jobs, Merge jobs can also be retriggered from Gerrit by posting a comment to trigger a specific Merge job.

CHAPTER 6

Release Jobs

Release jobs get triggered when a "ref-updated-event" is triggered. This usally happens when a release tag is created in the repository. Release jobs are intended to create to publish docker images, binaries and npm modules. Release jobs can be triggered from Gerrit by posting the specific comment for the job, just like Verify and Merge jobs.

For more information regarding the release process, you can refer the Release Process Document. # TODO.

CHAPTER 7

Supported Architectures

Most of the job types, the jobs are broken down further to build, test, and release the Hyperledger Fabric projects with support for varying CPU architectures. They include:

- x86_x64 (Open stack minions)
- s390x

CHAPTER 8

Supported test types

Additionally with most job types, you will notice the Jenkins jobs are further isolated to include a number of test types. Those include:

- End-to-End tests (e2e_cli, java sdk e2e, node sdk e2e)
- BYFN tests (fabca-samples, byfn, eyfn with default, custom channels, couchdb, node lang chaincode)
- Unit tests (linter, spelling, license etc..)
- Smoke/Functional/Performance/Release tests (More functional tests from fabric-test repository)
- Multihost tests

CHAPTER 9

Writing Jenkins Job Definitions using Jenkins Job Builder

Most of the CI work involves the creation and modification of Jenkins job definitions. To get a better understanding of how to write Jenkins job definitions, start with reading through the JJB Job definitions documentation. ## Sandbox_Setup

CHAPTER 10

Contributing to Fabric CI

Contributing to the Fabric CI process starts with identifying tasks to work on, and bugs to be fixed. The JIRA site provided by the Hyperledger Community is where to find these items. To narrow down tasks and bugs that are directly related to Fabric CI, use the following URLs:

https://jira.hyperledger.org/issues/?filter=11500

Note: If you have questions not addressed by this documentation, or run into issues with any of build jobs, please post your question in https://chat.hyperledger.org/channel/ci-pipeline or https://chat.hyperledger.org/channel/fabric-ci channels.

10.1 Jenkins Sandbox Process

Hyperledger Jenkins Sandbox provides Jenkins Job testing/experimentation environment that can be used before pushing job templates to the production Jenkins. It is configured similar to the Hyperledger ci-management production instance; however, it cannot publish artifacts or vote in Gerrit. Be aware that this is a test environment, and as such there are a limited allotment of minions to test on before pushing code to the Hyperledger repos.

Keep the following points in mind prior to beginning work on Hyperledger Jenkins Sandbox environment:

- Jobs are automatically deleted from Jenkins Sandbox every SATURDAY
- Committers can login and configure Jenkins jobs in the SANDBOX directly
- Jenkins Sandbox jobs CANNOT upload build images to Docker Hub
- Jenkins Sandbox jobs CANNOT vote on Gerrit
- Jenkins nodes are configured using Hyperledger openstack VMs and you can not access these VMs directly.

Before you proceed further, ensure you have a Linux Foundation ID (LFID), which is required to access Gerrit & Jenkins. The documentation on requesting a Linux Foundation account is available here lf-account. Also, to get an access to Sandbox environment please send an email to helpdesk@hyperledger.org (LF helpdesk team)

10.1.1 Follow below steps to work on the Jenkins Sandbox:

Step 1:

To download **ci-management**, execute the following command to clone the **ci-managment** repository.

```
git clone ssh://<LFID>@gerrit.hyperledger.org:29418/ci-management && scp -p -P
29418 <LFID>@gerrit.hyperledger.org:hooks/commit-msg ci-management/.git/hooks/
```

Step 2: After cloning we need to make sure the submodules are also synced:

Make sure to sync global-jjb submodule using: *git submodule update –init*

Step 3:

Once you successfully clone the repository, the next step is to install JJB (Jenkins Job Builder) in order to do experiment with the Jenkins jobs.

10.1.2 Execute the following commands to install JJB on your machine:

```
cd ci-management
sudo apt-get install python-virtualenv
virtualenv hyp
source hyp/bin/activate
pip install jenkins-job-builder
jenkins-jobs --version
jenkins-jobs test --recursive jjb/
```

Step 3:

10.1.3 Make a copy of the example JJB config file (in the builder/ directory)

Backup the jenkins.ini.example to jenkins.ini

```
cp jenkins.ini.example jenkins.ini
```

After copying the jenkins.ini.example, modify `jenkins.ini` with your **Jenkins LFID username**, **API token** and **Hyperledger JENKINS SANDBOX URL**

```
[job_builder]
ignore_cache=True
keep_descriptions=False
include_path=.:scripts:~/git/
recursive=True

[jenkins]
user=rameshthoomu <Provide your Jenkins Sandbox username>
password= <Refer below steps to get API token>
url=https://jenkins.hyperledger.org/sandbox
This is deprecated, use job_builder section instead
ignore_cache=True
query_plugins_info=false
```

10.1.4 How to retrieve API token?

Login to the Jenkins Sandbox, go to your user page by clicking on your username. Click **Configure** and then click **Show API Token**.

To work on existing jobs or create new jobs, navigate to the `/jjb` directory where you will find all job templates for the project. Follow the below commands to test, update or delete jobs in your SANDBOX environment.

Step 4:

10.1.5 To Test a Job

After you modify or create jobs in the above environment, it is good practice to test the job in sandbox environment before you submit this job to production CI environment.

`jenkins-jobs --conf jenkins.ini test jjb/ <job-name>`

Example: `jenkins-jobs --conf jenkins.ini test jjb/ fabric-verify-x86_64`

If the job you'd like to test is a template with variables in its name, it must be manually expanded before use. For example, the commonly used template `fabric-verify-{arch}` might expand to `fabric-verify-x86-64`.

A successful test will output the XML description of the Jenkins job described by the specified JJB job name.

Execute the following command to pipe-out to a directory:

`jenkins-jobs --conf jenkins.ini test jjb/ <job-name> -o <directoryname>`

The output directory will contain files with the XML configurations.

Step 5:

10.1.6 To Update a job

Ensure you've configured your `jenkins.ini` and verified it by outputting valid XML descriptions of Jenkins jobs. Upon successful verification, execute the following command to update the job to the Jenkins SANDBOX.

`jenkins-jobs --conf jenkins.ini update jjb/ <job-name>`

Example: `jenkins-jobs --conf jenkins.ini update jjb/ fabric-verify-x86_64`

Step 6:

10.1.7 Trigger jobs from Jenkins Sandbox:

Once you push the Jenkins job configuration to the Hyperledger Sandbox environment, run the job from Jenkins Sandbox webUI. Follow the below process to trigger the build:

Step 1: Login into the Jenkins Sandbox WebUI

Step 2: Click on the **job** which you want to trigger, then click **Build with Parameters**, and finally click **Build**.

Step 3: Verify the **Build Executor Status** bar and make sure that the build is triggered on the available executor. In Sandbox you may not see all platforms build executors and you don't find many like in production CI environment.

Once the job is triggered, click on the build number to view the job details and the console output.

Step 7:

10.1.8 Modify an Existing Job

In the Hyperledger Jenkins Sandbox, you can directly edit or modify the job configuration by selecting the job name and clicking on the **Configure** button. Then, click the **Apply** and **Save** buttons to save the job. However, it is recommended to simply modify the job in your terminal and then follow the previously described steps in **To Test a Job** and **To Update a Job** to perform your modifications.

Step 8:

10.1.9 To Delete a Job:

Execute the following command to Delete a job from Sandbox:

`jenkins-jobs --conf jenkins.ini delete jjb/ <job-name>`

Example: `jenkins-jobs --conf jenkins.ini delete jjb/ fabric-verify-x86_64`

The above command would delete the `fabric-verify-x86-64` job.

10.2 Fabric

This document explains the CI process for the **Fabric** repository. The below steps explains what CI follows or executes when a patch set is submited to the Fabric repository.

Whenever a patchset is submitted to the Fabric repository, Jenkins triggers the CI build process to test and validate the patchset. Fabric CI **verify and merge** jobs are configured to test the patchset in the below environment.

The Hyperledger Fabric (and associated) projects utilize various tools and workflows for continuous project development. The Fabric CI is currently utilizing the following versions in the **Master** and **Release-1.1**, **Release-1.0** branches.

Master:

- GO version:(e.g. v1.10) https://github.com/hyperledger/fabric/blob/master/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version:(e.g. 0.4.6) https://github.com/hyperledger/fabric/blob/196c0de7c1618952a8f342e406a1021203845eba/Makef

Release-1.0:

- GO version:(e.g. v1.7.5) https://github.com/hyperledger/fabric/blob/release-1.0/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version:(e.g. 0.4.6)

Release-1.1:

- GO version:(e.g. v1.9.2) https://github.com/hyperledger/fabric/blob/release-1.1/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version:(e.g. 0.4.6) https://github.com/hyperledger/fabric/blob/da14b6bae4a843dfb3fcece5a08ae0ea18488a7a/Makefil

If you would like to know more details on the tool versions, you can refer from any of the Fabric jobs listed over here Fabric, select one of the jobs, Click on any build number in the bottom left and view the output for details.

There are several job types that are common across Hyperledger Fabric projects. In some cases, you may or may not see all of the common job types in every project. This depends on the specific needs of that Hyperledger Fabric project. The CI configuration is prepared in Jenkins Job Builder to create, update and modify the Jenkins Jobs.

As part of the CI process, we create JJBs (Jenkins Job Builder) in YAML format to configure Jenkins jobs. JJB has a flexible template system, so creating multiple jobs with a common configuration which is easy. More details about Jenkins Job Builder are available in the JJB webpage.

The following steps explains, what happens when a developer submits a patchset to the **Fabric** repository.

When a patchset is submitted to the Fabric repository, the Hyperledger Community CI server (Jenkins) triggers **Verify** jobs on **x86_64** platform using the patchset's parent commit which may or may not be the latest commit on **Fabric**.

10.2.1 Build Process

The Fabric **verify** build process is split up into multiple jobs. The initial job (fabric-verify-build-checks-x86_64) is to build and publish docker images and binaries to Nexus3 and Nexus2. These images are later pulled/downloaded in the downstream jobs, when the triggered conditions meets in `fabric-verify-build-checks-x86_64` CI job.

Below are the conditions to trigger relevant jobs based on the patchset:

- `fabric-verify-build-checks-x86_64` job triggers when a `patchset` is created and it validates the patchsets git commit message.
  - If the commit message has a WIP, the above build job **ignores** to build the patchset and will not post a voting back to Fabric patchset. That means, this job skips the build process. You can see "WIP - No build" in the patchset's result.
  - If the patchset has a **non WIP** in the commit message or if it is a **documentation change** with these file extensions (.rst, .md, .py, .png,.css,.html and .ini), the above job posts `Run DocBuild` comment and sends Fabric voting as `F1-VerifyBuild=+1 F2-SmokeTest=+1 F3-UnitTest=+1` against the patchset.

```
* Run DocBuild
    - This comment triggers `fabric-docs-build-x86_64` CI job. Once the doc␣
↪build is
      successfully executed, Jenkins sends Fabric vote as `F2-DocsBuild=+1`␣
↪otherwise as
      `F2-DocsBuild=-1`
```

  - If the patchset has **non WIP** in the commit message or a **code and documentation changes** (see the above file extensions), fabric-verify-build-checks-x86_64 executes the below flow. The below flow also applies to the **code only** patchset excluding documentation build process.

```
* Executes `make basic-checks`, `make docker` (builds, re-tag and publish␣
↪images to nexus3),
  `make dist` (builds binaries) and publishes to nexus2. If any of these make␣
↪targets
  fails, fabric-verify-build-checks-x86_64 sends `F1-VerifyBuild=-1` to the␣
↪Fabric
  patchset otherwise it sends `F1-VerifyBuild=+1` and triggers **DocsBuild**␣
↪and
  **SmokeTest** jobs parallely by posting below comments to the patchset.

* Run DocsBuild
    - This comment triggers `fabric-docs-build-x86_64` job and posts `F2-
↪DocsBuild=+1`
      if successful, otherwise `F2-DocsBuild=-1`. See the doc RTD output in␣
↪the nexus
       log server.

       What happens in **fabric-docs-build-x86_64** job
```

(continues on next page)

(continued from previous page)

```
      Step1: Builds the documentation changes:
         - Extracts the documentation files(.md, .rst, .txt, .py,
          .png, .css, .html & .ini) from the patchset submitted and builds␣
↪the
           documentation after verification checks like syntax, and tox␣
↪verification.
           This job is triggered only when a patchset contains␣
↪documentation files.

      Step2: Documented output is published to Nexus:
         - Once the documentation build is successful, it is archived, and␣
↪the archives
           built are published to Nexus.

* Run SmokeTest
    - This comment triggers `fabric-smoke-tests-x86_64` job and posts `F2-
↪SmokeTest=+1`
      to the patchset and triggers Unit-Test job by posting `Run UnitTest`␣
↪comment if
      successful, otherwise posts `F2-SmokeTest=-1` which doesn't trigger␣
↪Unit-Test job.

* Run UnitTest
    - This comment triggers `fabric-verify-unit-tests-x86_64` job and posts
      `F3-UnitTest=+1` vote against the patchset if successful, otherwise `F3-
↪UnitTest=-1`.
```

10.2.2 Conditions to merge the patch set

Maintainers have to look for +1 on all the labels before they merge the patchsets. The votes on the patchset should look like below.

```
F1-VerifyBuild +1 Hyperledger Jobbuilder
F2-DocBuild    +1 Hyperledger Jobbuilder
F2-SmokeTest   +1 Hyperledger Jobbuilder
F3-UnitTest    +1 Hyperledger Jobbuilder
```

patchset is not elible to merge, if it even gets one -1.

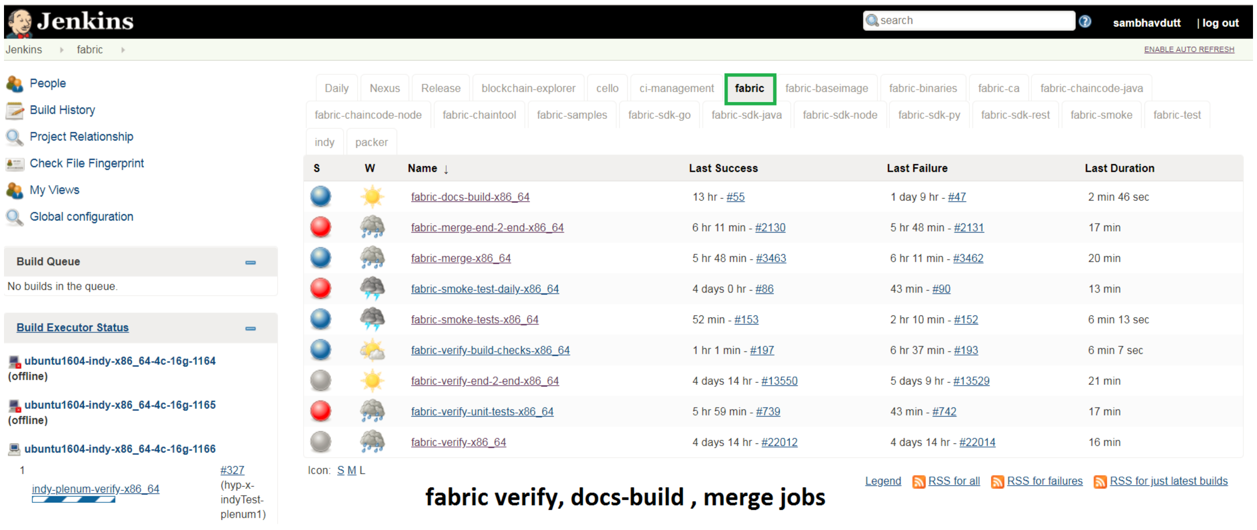
10.2.3 Merge process for Fabric

Once the patchset is approved by CI and the maintainers, they will merge the patchset which triggers below **Merge** jobs on the latest Fabric commit (doesn't use the patchset's parent commit).

fabric-merge-end-2-end-x86_64: https://jenkins.hyperledger.org/view/fabric/job/fabric-merge-end-2-end-x86_64/

Step1: Clones the fabric-ca repository:

- Clones the latest commit from the Fabric fabric-ca repository and then checksout to the Branch. If the patchset is triggered on fabric-ca release-1.1 branch, script will checkout to release-1.1 branch.
- After the fabric-ca repository is cloned in the above step, CI script executes to build docker images to kick off the e2e tests.



S	W	Name ↓	Last Success	Last Failure	Last Duration
🔵	☀	fabric-docs-build-x86_64	13 hr - #55	1 day 9 hr - #47	2 min 46 sec
🔴	🌧	fabric-merge-end-2-end-x86_64	6 hr 11 min - #2130	5 hr 48 min - #2131	17 min
🔵	🌧	fabric-merge-x86_64	5 hr 48 min - #3463	6 hr 11 min - #3462	20 min
🔴	⛈	fabric-smoke-test-daily-x86_64	4 days 0 hr - #86	43 min - #90	13 min
🔵	⛈	fabric-smoke-tests-x86_64	52 min - #153	2 hr 10 min - #152	6 min 13 sec
🔵	⛅	fabric-verify-build-checks-x86_64	1 hr 1 min - #197	6 hr 37 min - #193	6 min 7 sec
⚪	☀	fabric-verify-end-2-end-x86_64	4 days 14 hr - #13550	5 days 9 hr - #13529	21 min
🔴	🌧	fabric-verify-unit-tests-x86_64	5 hr 59 min - #739	43 min - #742	17 min
⚪	🌧	fabric-verify-x86_64	4 days 14 hr - #22012	4 days 14 hr - #22014	16 min

Fig. 1: Views

Step 2: Executes the e2e tests:

Below are the tests triggers in Fabric e2e job:

- 1. e2e-cli - Runs fabric/examples/e2e_cli tests.
  - Executes the network_setup.sh that spins up the network with docker-compose file from fabric/examples/e2e_cli folder.
- 2. e2e-node - Runs the sdk-node e2e tests (Executes **gulp test** command).
  - Clones fabric-sdk-node repository and will checkout to Branch
  - Spins up network using the docker-compose file from test/fixtures folder
  - Install nodejs 8.9.4 version
  - RUN `istanbul cover --report cobertura test/integration/e2e.js`
- 3. e2e-java - Runs e2e java integration tests.
  - If the patchset is on release-1.0 branch, we ignore java e2e tests for now.
  - If not, run the java e2e tests by executing `source cirun.sh`
- 4. byfn and efyn - Runs byfn and eyfn tests with default, custom channel with couchdb and nodejs chaincode and fabric-ca sample tests
- 5. After the above tests have worked as expected, merge job publishes images and binaries to nexus repository with "stable" image tag.

  TODO# Script will be pushed to fabric repository to download these images and binaries.

fabric-merge-x86_64: https://jenkins.hyperledger.org/view/fabric/job/fabric-merge-x86_64

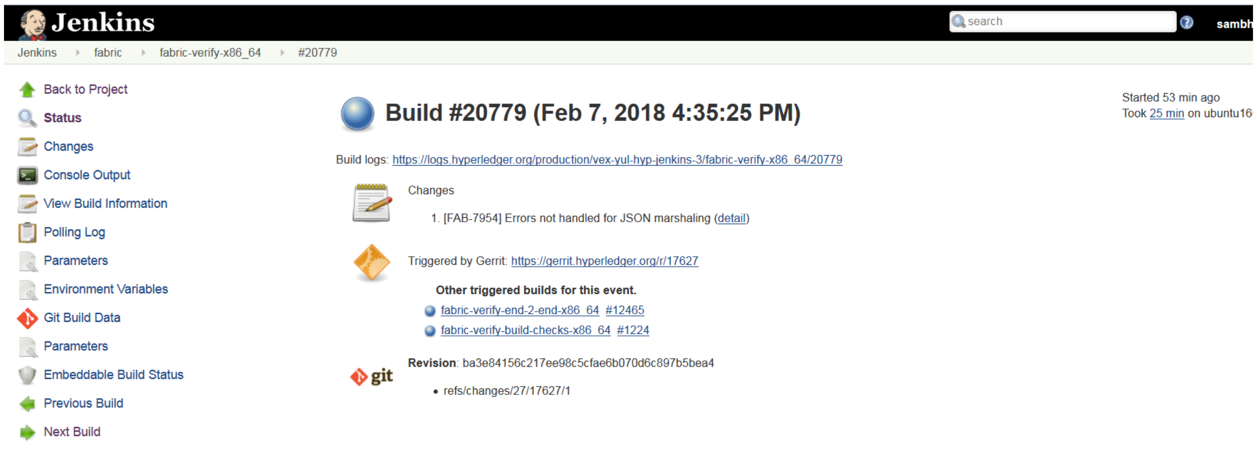
Step1: Pulls the third party docker images:

- Pulls the fabric baseimage version third party docker images(kafka, zookeeper, couchdb). The image name is appended with 'hyperledger' and tagged with the latest tag.

Step2: Executes Fabric tests using below two commands:

```
make linter make unit-test
```

After the verify or merge tests are executed, It is time to archive the logs (artifacts). CI publishes the logs(artifacts) on Jenkins console.



Build #20779 (Feb 7, 2018 4:35:25 PM)

Started 53 min ago
Took 25 min on ubuntu16

Build logs: https://logs.hyperledger.org/production/vex-yul-hyp-jenkins-3/fabric-verify-x86_64/20779

Changes

1. [FAB-7954] Errors not handled for JSON marshaling (detail)

Triggered by Gerrit: https://gerrit.hyperledger.org/r/17627

Other triggered builds for this event.

- fabric-verify-end-2-end-x86_64 #12465
- fabric-verify-build-checks-x86_64 #1224

Revision: ba3e84156c217ee98c5cfae6b070d6c897b5bea4

- refs/changes/27/17627/1

Fig. 2: ConsoleOutPut

10.2.4 Build Notifications

The build results can be viewed on the Jenkins console, where depending on the result it displays with a colored bubble (green for success, red for failure).

10.2.5 Trigger failed jobs through Gerrit comments

Re-trigger of builds is possible in Jenkins by entering a comment to the Gerrit change that re-triggers a specific verify job. To do so, follow the below process:

Step 1: Open the Gerrit patchset for which you want to reverify the build

Step 2: Click on **Reply**, then type one of the below comments and click **Post**

> `VerifyBuild` – Triggers fabric-verify-build-checks-x86_64 CI job, developers have to check the result of this job before posting the below comments on the patchset. As mentioned above, this job publishes images and binaries to nexus which further downloaded by SmokeTest and UnitTest jobs. Please make sure, images and binaries are published for that sepecific commit.
>
> `Run SmokeTest` – Triggers fabric-smoke-tests-x86_64.
>
> `Run UnitTest` – Triggers fabric-verify-unit-tests-x86_64.
>
> `Run DocsBuild` – Triggers fabric-docs-build-x86_64

This kicks off the specified Fabric verify jobs. Once the build is triggered, verify the Jenkins console output and go through the log messages if you are interested to know how the build is making progress.

10.2.6 Questions

Please reach out to us in https://chat.hyperledger.org/channel/ci-pipeline or https://chat.hyperledger.org/channel/fabric-ci RC channels for any questions.

10.3 Fabric-CA

This document explains the Fabric-ca CI process. The below steps explains what CI follows or executes when a patchset is submited to the fabric-ca repository.

Whenever a patchset is submitted to the fabric-ca repository, Jenkins triggers the CI build process to test and validate the patchset. Fabric-ca CI **verify and merge** jobs are configured to test the patchset in the below environment.

The Hyperledger Fabric (and associated) projects utilize various tools and workflows for continuous project development. The FABRIC-CA is currently utilizing the following versions in the **Master** , **Release-1.0** and **Release-1.1** branches.

Master:

- GO version(e.g. v1.9.2): https://github.com/hyperledger/fabric-ca/blob/master/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version(e.g. 0.4.6): https://github.com/hyperledger/fabric-ca/blob/be7180447ce9b47a4d3ae33210b5cf00d67ff6d9/Makefile#L55

Release1.0:

- GO version(e.g. 1.7.5): https://github.com/hyperledger/fabric-ca/blob/release-1.0/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version(e.g. 0.3.1): https://github.com/hyperledger/fabric-ca/blob/d5aa9afd6044201acd225494ee8c7537cd5a6673/Makefile#L47

Release1.1:

- GO version(e.g. 1.9.2): https://github.com/hyperledger/fabric-ca/blob/release-1.1/ci.properties
- DOCKER version: 17.12.0-ce
- baseimage version(e.g. 0.4.6): https://github.com/hyperledger/fabric-ca/blob/d536f5a4b9dbfe057af16dd5ae2ab87841b80f9c/Makefile#L62

If you would like to know more details on the tool versions, you can refer from any FABRIC-CA jobs listed here fabric-ca, select one of the jobs, click on any build number in the bottom left and view the output for details.

10.3.1 Build Process

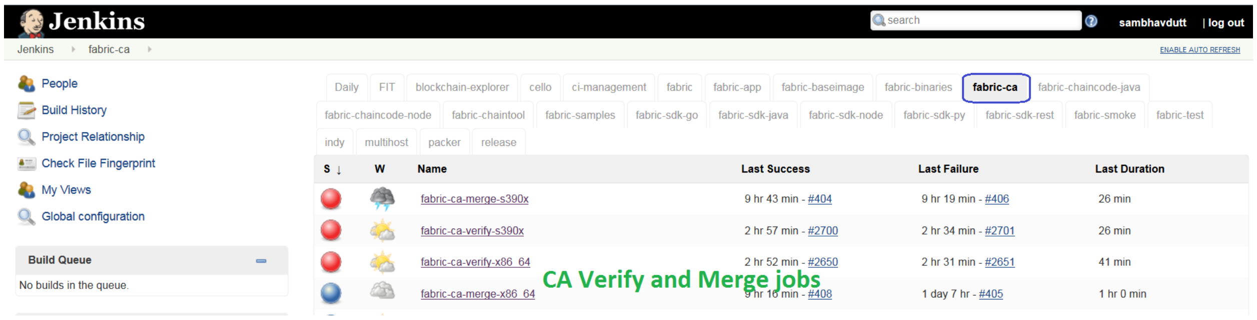
There are several Jenkins job types that are common across Hyperledger Fabric projects. In some cases, you may or may not see all of the common job types in every project. This depends on the specific needs of that Hyperledger Fabric project. The CI configuration is prepared in Jenkins Job Builder to create, update and modify the Jenkins Jobs.

As part of the CI process, we create JJBs (Jenkins Job Builder) in YAML format to configure Jenkins jobs. JJB has a flexible template system, so creating multiple jobs with a common configuration which is easy. More details about Jenkins Job Builder are available in the JJB webpage.

The following steps explains, what happens when we submit a patch to the **fabric-ca** repository.

When a patchset is submitted to the fabric-ca repository, the Hyperledger Community CI server (Jenkins) triggers **Verify** jobs on **x86_64** and **s390x** platforms using the patchset's parent commit which may or may not be the latest commit on **fabric-ca**. The following verify jobs are triggered.

- fabric-ca-verify-x86_64
- fabric-ca-verify-s390x
- fabric-ca-verify-end-2-end-x86_64



S ↓	W	Name	Last Success	Last Failure	Last Duration
🔴		fabric-ca-merge-s390x	9 hr 43 min - #404	9 hr 19 min - #406	26 min
🔴		fabric-ca-verify-s390x	2 hr 57 min - #2700	2 hr 34 min - #2701	26 min
🔴		fabric-ca-verify-x86_64	2 hr 52 min - #2650	2 hr 31 min - #2651	41 min
🔵		fabric-ca-merge-x86_64	9 hr 16 min - #408	1 day 7 hr - #405	1 hr 0 min

Fig. 3: Views

Below are steps CI executes on **Verify** and **Merge** jobs:

fabric-ca-verify-end-2-end-x86_64

Step1: Clones the Fabric repository:

- Clones the latest commit from the Gerrit Fabric repository and then checkout to the Branch. If the patchset is triggered on Fabric release-1.0 branch, script will checkout to release-1.0 branch.
- After the Fabric and Fabric-ca repositories afre cloned in the above step, CI script executes to build DOCKER images to kick off the e2e tests.

Step 2: Executes the e2e tests:

Below are the tests triggers in Fabric-ca e2e job:

- 1. e2e-cli - Runs fabric/examples/e2e_cli tests.
    - Executes the network_setup.sh that spins up the network with docker-compose file from fabric/examples/e2e_cli folder.
- 2. e2e-node - Runs the sdk-node e2e tests (Executes **gulp test** command).
    - Clones fabric-sdk-node repository and will checkout to Branch
    - Spins up network using the docker-compose file from test/fixtures folder
    - Install nodejs 8.9.4 version
    - run `istanbul cover --report cobertura test/integration/e2e.js`
- 3. e2e-java - Runs e2e java integration tests.
    - If the patchset is on release-1.0 branch, we ignore java e2e tests for now.
    - If not, run the java e2e tests by executing `source cirun.sh`

fabric-ca-verify-x86_64 & **fabric-ca-verify-s390x**

Step1: Clones the Fabric repository:

- Clones the latest commit from the Gerrit fabric-ca and then checkout to the Branch. If the patchset is triggered on fabric-ca release-1.0 branch, script will checkout to the release-1.0 branch.

Step2: Executes fabric-ca tests using below two commands:

```
make ci-tests
```

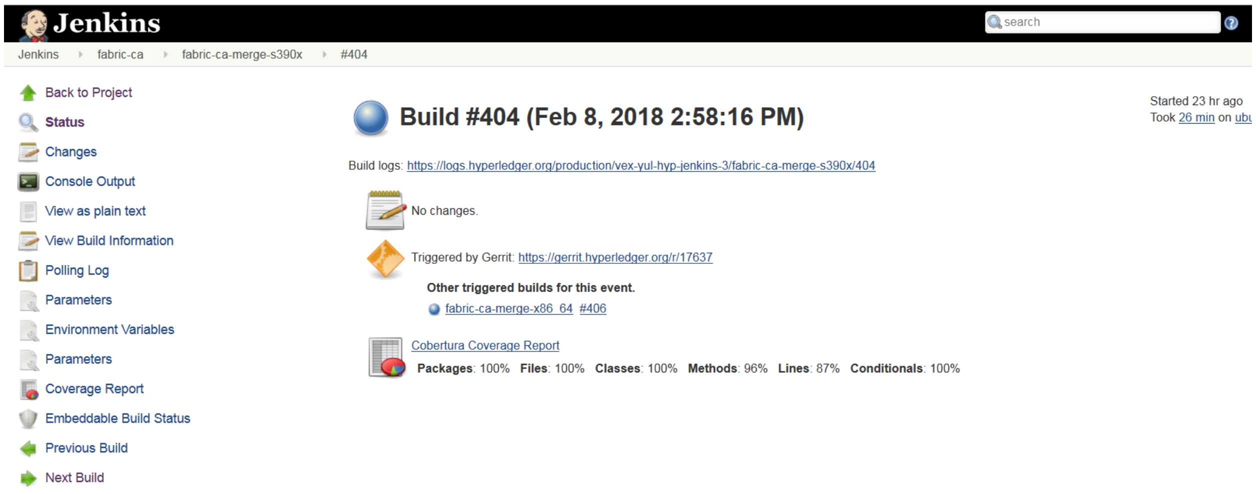
```
make docker-fvt
```

Once the tests are completed, Jenkins posts +1 vote to the patchset **+1 –> Hyperledger Jobbuilder** upon successful completion and -1 **-1 –> Hyperledger Jobbuilder** in case of failure.

Once the patchset is approved by CI and the maintainers, they will merge the patchset which triggers below **Merge** jobs and runs the above mentioned tests on the latest fabric-ca commit (doesn't use the patchset's parent commit).

- fabric-ca-merge-x86_64
- fabric-ca-merge-s390x
- fabric-ca-merge-end-2-end-x86_64

After the tests are executed, It is time to archive the logs (artifacts) and publish the code coverage. CI publishes the logs(artifacts) and the Code Coverage report(Cobertura Coverage Report)on Jenkins console.



Build #404 (Feb 8, 2018 2:58:16 PM)

Started 23 hr ago
Took 26 min on ubu

Build logs: https://logs.hyperledger.org/production/vex-yul-hyp-jenkins-3/fabric-ca-merge-s390x/404

No changes.

Triggered by Gerrit: https://gerrit.hyperledger.org/r/17637

Other triggered builds for this event.

fabric-ca-merge-x86_64 #406

Cobertura Coverage Report

Packages: 100% **Files**: 100% **Classes**: 100% **Methods**: 96% **Lines**: 87% **Conditionals**: 100%

Fig. 4: ConsoleOutPut

10.3.2 Build Notifications

The build results can be viewed on the Jenkins console, where depending on the result it displays with a colored bubble (green for success, red for failure) and a vote from the CI (+1 or -1) on the Gerrit commit/change.

Also, it sends out an email notification to all the Fabric-ca maintainers in case of merge job failure.

10.3.3 Trigger failed jobs through Gerrit comments

Re-trigger of builds is possible in Jenkins by entering **reverify** in a comment to the Gerrit change that retriggers all the verify jobs. To do so, follow the below process:

Step 1: Open the Gerrit patchset for which you want to reverify the build

Step 2: Click on **Reply**, then type `reverify` and click **Post**

This kicks off all the Fabric-ca verify jobs. Once the build is triggered, verify the Jenkins console output and go through the log messages if you are interested to know how the build is making progress.

In some cases, Jenkins may fail only in one or two CI jobs due to which network issues. In such cases, restarting all the fabric-ca jobs through `reverify` comment is not necessary. Instead, the developer can post below comment to trigger the particular failed build:

`reverify-e2e` - re-triggers fabric-ca-merge-end-2-end-x86_64 CI job.

`reverify-x` - re-triggers fabric-ca-verify-x86_64 on x86_64 platform.

`reverify-z` - re-triggers fabric-ca-verify-s390x on s390x platform.

10.3.4 Questions

Please reach out to us in https://chat.hyperledger.org/channel/ci-pipeline or https://chat.hyperledger.org/channel/fabric-ci RC channels for Questions or concerns related to fabric-ca CI process.

10.4 Fabric-SDK-Java

This document explains about the fabric-sdk-java CI process. The below steps explain what CI follows or executes when a patch set is submit to the fabric-sdk-java repository.

Whenever a patchset is submitted to the fabric-sdk-java repository, Jenkins triggers the CI build process to test and validate the patchset. Fabric-sdk-java CI **verify and merge** jobs are configured to test the patchset in the below environment.

The Hyperledger Fabric (and associated) projects utilize various tools and workflows for continuous project development. Thefabric-sdk-java is currently utilizing the following versions in the **Master** and **Release-1.0** branches.

Master

- go version: v1.9.2
- docker version: 17.12.0-ce

Release-1.0

- go version: v1.9.2
- docker version: 17.12.0-ce

If you would like to know more details on the tool versions, you can refer from any fabric-sdk-java jobs listed here fabric-sdk-java, Select one of the jobs, Click on any build number in the bottom left and view the output for details.

10.4.1 Build Process

There are several Jenkins job types that are common across Hyperledger Fabric projects. In some cases, you may or may not see all of the common job types in every project. This depends on the specific needs of that Hyperledger Fabric project. The CI configuration is prepared in Jenkins Job Builder to create, update and modify the Jenkins Jobs.

As part of the CI process, we create JJB's (Jenkins Job Builder) in YAML format to configure Jenkins jobs. JJB has a flexible template system, so creating many similar jobs with a common configuration is easy. More about Jenkins Job Builder is available on the JJB webpage.

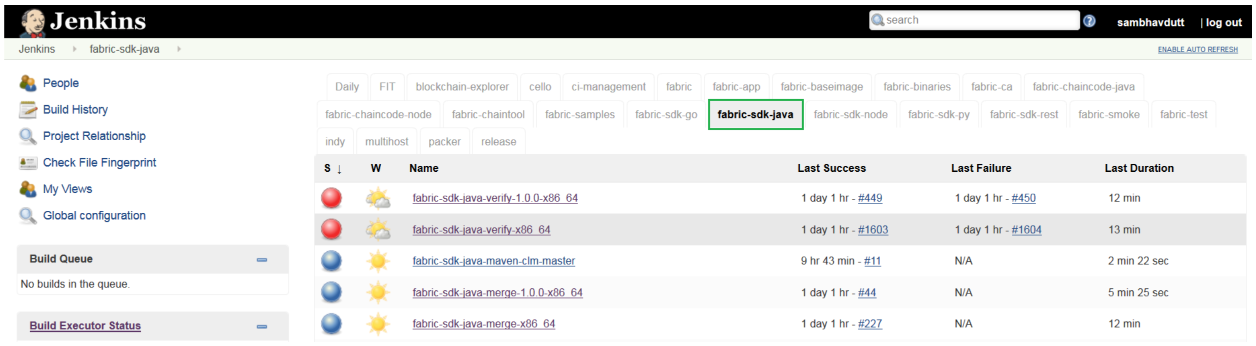
The following explains what happens when we submit a patch to the **fabric-sdk-java** repository.

When a patchset is submitted to fabric-sdk-java repository, the Hyperledger Community CI server (Jenkins) triggers **Verify** and jobs on **x86_64** platform using the patchset's parent commit which may or may not be the latest commit on **fabric-sdk-java**.

fabric-sdk-java-verify-x86_64

fabric-sdk-java-verify-1.0.0-x86_64

As part of the CI process on **fabric-sdk-java** repository, the following tests are executed on **x86_64**(x) platform, see the arch value at the end of the job name to know on which platform we run this job.



No builds in the queue.

S ↓	W	Name	Last Success	Last Failure	Last Duration
🔴	⛅	fabric-sdk-java-verify-1.0.0-x86_64	1 day 1 hr - #449	1 day 1 hr - #450	12 min
🔴	⛅	fabric-sdk-java-verify-x86_64	1 day 1 hr - #1603	1 day 1 hr - #1604	13 min
🔵	☀	fabric-sdk-java-maven-clm-master	9 hr 43 min - #11	N/A	2 min 22 sec
🔵	☀	fabric-sdk-java-merge-1.0.0-x86_64	1 day 1 hr - #44	N/A	5 min 25 sec
🔵	☀	fabric-sdk-java-merge-x86_64	1 day 1 hr - #227	N/A	12 min

Fig. 5: Views

Below is the process we execute in CI on fabric-sdk-java verify and merge jobs:

Step 1: - Clone Fabric & Fabric-ca repositories:

- Clone the latest commit from the Gerrit for Fabric and Fabric-ca repositories.
- Check if the repositories are on the latest commit and the specified branch the tests must execute on. Checkout to the branch that is specified. Build the images with `make docker` command to build the latest docker images for Fabric & Fabric-ca.

Step 2: - When the images are ready, execute the cirun.sh script in the /src/test directory of fabric-sdk-java

- Export environment settings, wait time for SDK test integration.
- With the latest JSDK version, now that we have the latest images for Fabric and Fabric-ca, run the java integration tests.

  For the **fabric-sdk-java-verify-1.0.0-x86_64** or **fabric-sdk-java-merge-1.0.0-x86_64** jobs,

- This job is to run the integration tests using the latest JSDK version with Fabric & Fabric-ca 1.0.0 version. A check is made to verify if the tests have to execute on Fabric & Fabric-ca 1.0.0 docker images.
- The version for Fabric and Fabric-ca is set to 1.0.0. The 1.0.0 version docker images are fetched for Fabric and Fabric-ca. The Fabric generated configuration version is set to v1.0

Step 3: - Execute the docker compose file from the /src/test/fixture/sdkintegration/fabric.sh

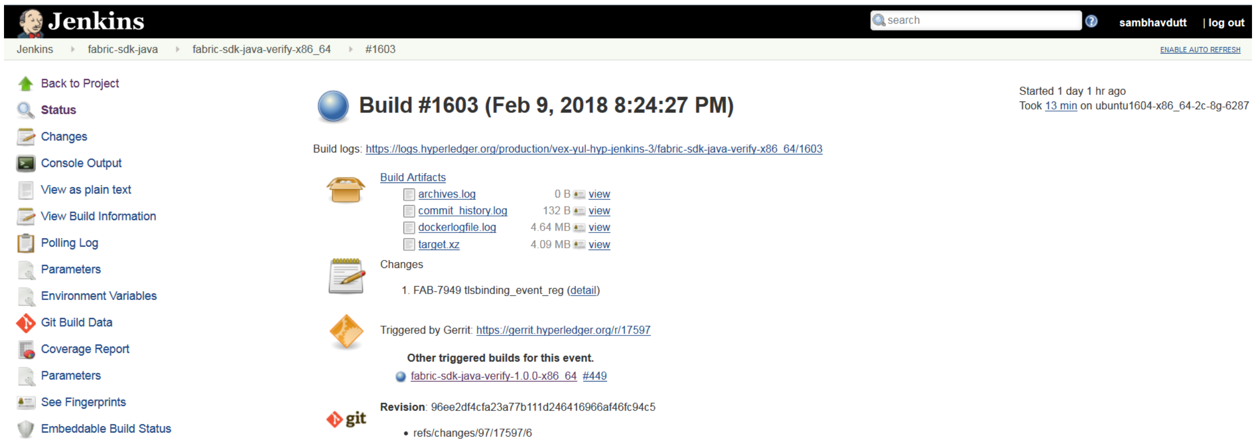
- Clean up the unnecessary containers/images if any with the *clean()* function.
- Bring the network up with function *up()* , this executes the docker-compose up and creates the docker containers.
- Bring down the network and start it again, wait till the containers are started, and execute the java integration tests.

Above process is applicable to both **verify** and **merge** jobs.

After the builds are executed successfully, it sends a voting to Gerrit patch set with a +1, or -1 if the build fails.

Once the patchset is approved by CI and the maintainers, they will merge the patchset which triggers the Merge jobs and runs the above mentioned tests on the latest fabric-sdk-java commit (doesn't use the patchset's parent commit).

After the tests are executed, It is time to archive the logs (artifacts) and publish the code coverage. CI publishes the logs(artifacts) and the Code Coverage report(JaCoCo Coverage Report)on Jenkins console.



Jenkins | search | sambhavdutt | log out

Jenkins › fabric-sdk-java › fabric-sdk-java-verify-x86_64 › #1603 — ENABLE AUTO REFRESH

- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- View Build Information
- Polling Log
- Parameters
- Environment Variables
- Git Build Data
- Coverage Report
- Parameters
- See Fingerprints
- Embeddable Build Status

Build #1603 (Feb 9, 2018 8:24:27 PM)

Started 1 day 1 hr ago
Took 13 min on ubuntu1604-x86_64-2c-8g-6287

Build logs: https://logs.hyperledger.org/production/vex-yul-hyp-jenkins-3/fabric-sdk-java-verify-x86_64/1603

Build Artifacts

archives.log	0 B	view
commit_history.log	132 B	view
dockerlogfile.log	4.64 MB	view
target.xz	4.09 MB	view

Changes

1. FAB-7949 tlsbinding_event_reg (detail)

Triggered by Gerrit: https://gerrit.hyperledger.org/r/17597

Other triggered builds for this event.

fabric-sdk-java-verify-1.0.0-x86_64 #449

Revision: 96ee2df4cfa23a77b111d246416966af46fc94c5

- refs/changes/97/17597/6

Fig. 6: ConsoleOutPut

10.4.2 Build Notifications

The build results can be viewed on the Jenkins console, where depending on the result it displays with a colored bubble (green for success, red for failure) and a vote from the CI (+1 or -1) on the Gerrit commit/change.

Also, it sends out an email notification to all the fabric-sdk-java maintainers in case of merge job failure.

10.4.3 Trigger failed jobs through Gerrit comments

Re-trigger of builds is possible in Jenkins by entering **reverify** in a comment to the Gerrit change that re-triggers all the verify jobs. To do so, follow the below process:

Step 1: Open the Gerrit patchset for which you want to reverify the build

Step 2: Click on **Reply**, then type **reverify** and click **Post**

This kicks off all the fabric-sdk-java verify jobs. Once the build is triggered, verify the Jenkins console output and go through the log messages if you are interested to know how the build is making progress.

In some cases, Jenkins may fail only in one or two CI jobs due to which network issues. In such cases, restarting all the fabric-sdk-java jobs through `reverify` comment is not necessary. Instead, the developer can post below comment to trigger the particular failed build:

- reverify-x - to retrigger the build on fabric-sdk-java-verify-x86_64.
- reverify-1.1.0 - to retrigger the build on fabric-sdk-java-verify-1.0.0-x86_64.

10.4.4 Questions

Please reach out to us in https://chat.hyperledger.org/channel/ci-pipeline or https://chat.hyperledger.org/channel/fabric-ci RC channels for Questions or concerns related to fabric-sdk-java CI process.

10.5 Fabric-SDK-Node

This document explains about the fabric-sdk-node CI process. The below steps explain what CI follows or executes when a patchset is submits to the fabric-sdk-node repository.

Whenever a patchset is submitted to the fabric-sdk-node repository, Jenkins triggers the CI build process to test and validate the patchset. Fabric-sdk-node CI **verify and merge** jobs are configured to test the patchset in the below environment.

The Hyperledger Fabric (and associated) projects utilize various tools and workflows for continuous project development. The fabric-sdk-node is currently utilizing the following versions in the **Master** and **Release-1.0** and **Release-1.1** branches.

Master:

- go version: v1.9.2
- docker version: 17.12.0-ce
- npm version: 8.9.4

Release-1.0:

- go version: v1.9
- docker version: 17.12.0-ce
- npm version: 6.9.5

Release-1.1:

- go version: v1.9.2
- docker version: 17.12.0-ce
- npm version: 8.9.4

If you would like to know more details on the tool versions, you can refer from any fabric-sdk-node jobs listed here fabric-sdk-node. Select one of the jobs, Click on any build number in the bottom left and view the output for details.

10.5.1 Build Process

There are several Jenkins job types that are common across Hyperledger Fabric projects. In some cases, you may or may not see all of the common job types in every project. This depends on the specific needs of that Hyperledger Fabric project. The CI configuration is prepared in Jenkins Job Builder to create, update and modify the Jenkins Jobs.

As part of the CI process, we create JJB's (Jenkins Job Builder) in YAML format to configure Jenkins jobs. JJB has a flexible template system, so creating many similar jobs with a common configuration is easy. More about Jenkins Job Builder is available on the JJB webpage.

The following explains what happens when we submit a patch to the **fabric-sdk-node** repository.

When a patchset is submitted to fabric-sdk-node repository, the Hyperledger Community CI server (Jenkins) triggers **Verify** and jobs on **x86_64** ans **s390x** platforms using the patchset's parent commit which may or may not be the latest commit on **fabric-sdk-node**.

The following verify jobs are triggered.

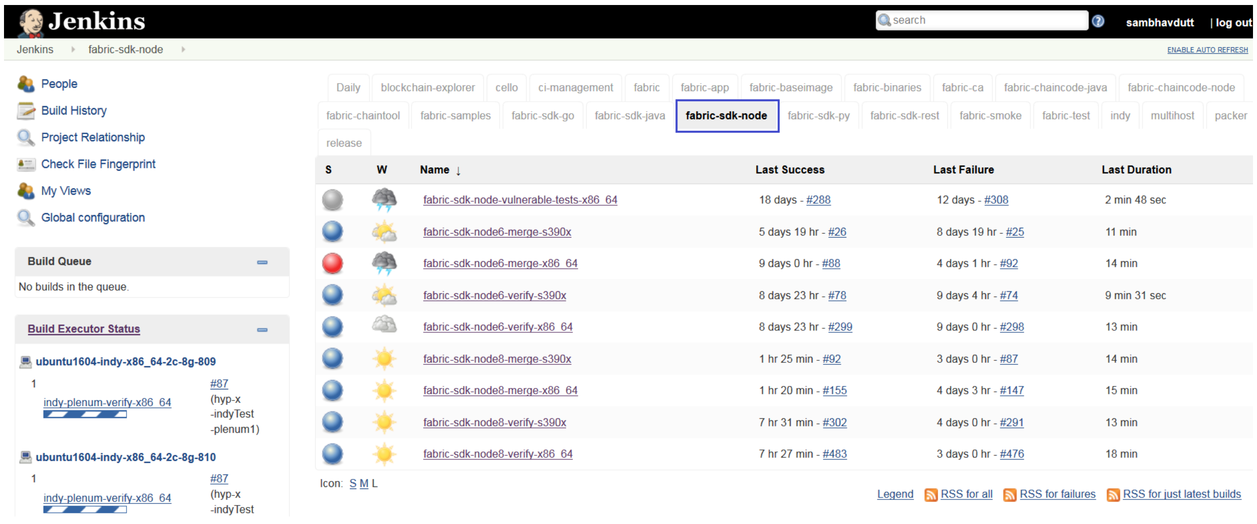
fabric-sdk-node6-verify-x86_64

fabric-sdk-node8-verify-x86_64

fabric-sdk-node6-verify-s390x

fabric-sdk-node8-verify-s390x

As part of the CI process on **fabric-sdk-node** repository, the following tests are executed on **x86_64**(x) and **s390x**(z) platforms, see the arch value at the end of the job name to know on which platform we run this job.



Jenkins

search | sambhavdutt | log out

Jenkins ▸ fabric-sdk-node ▸

ENABLE AUTO REFRESH

- People
- Build History
- Project Relationship
- Check File Fingerprint
- My Views
- Global configuration

Build Queue

No builds in the queue.

Build Executor Status

ubuntu1604-indy-x86_64-2c-8g-809

1 indy-plenum-verify-x86_64 #87 (hyp-x -indyTest -plenum1)

ubuntu1604-indy-x86_64-2c-8g-810

1 indy-plenum-verify-x86_64 #87 (hyp-x -indyTest

Daily | blockchain-explorer | cello | ci-management | fabric | fabric-app | fabric-baseimage | fabric-binaries | fabric-ca | fabric-chaincode-java | fabric-chaincode-node | fabric-chaintool | fabric-samples | fabric-sdk-go | fabric-sdk-java | **fabric-sdk-node** | fabric-sdk-py | fabric-sdk-rest | fabric-smoke | fabric-test | indy | multihost | packer | release

S	W	Name ↓	Last Success	Last Failure	Last Duration
⚪	⛈	fabric-sdk-node-vulnerable-tests-x86_64	18 days - #288	12 days - #308	2 min 48 sec
🔵	⛅	fabric-sdk-node6-merge-s390x	5 days 19 hr - #26	8 days 19 hr - #25	11 min
🔴	⛈	fabric-sdk-node6-merge-x86_64	9 days 0 hr - #88	4 days 1 hr - #92	14 min
🔵	⛅	fabric-sdk-node6-verify-s390x	8 days 23 hr - #78	9 days 4 hr - #74	9 min 31 sec
🔵	☁	fabric-sdk-node6-verify-x86_64	8 days 23 hr - #299	9 days 0 hr - #298	13 min
🔵	☀	fabric-sdk-node8-merge-s390x	1 hr 25 min - #92	3 days 0 hr - #87	14 min
🔵	☀	fabric-sdk-node8-merge-x86_64	1 hr 20 min - #155	4 days 3 hr - #147	15 min
🔵	☀	fabric-sdk-node8-verify-s390x	7 hr 31 min - #302	4 days 0 hr - #291	13 min
🔵	☀	fabric-sdk-node8-verify-x86_64	7 hr 27 min - #483	3 days 0 hr - #476	18 min

Icon: S M L

Legend | RSS for all | RSS for failures | RSS for just latest builds

Fig. 7: Views

Below is the process we execute in CI on fabric-sdk-node verify and merge jobs: Step 1: - Clone fabric & fabric-ca repositories:

- Clone the latest commit from the gerrit fabric repository and then check for the Branch Name, if the patch is triggered on fabric-sdk-node release-1.0 branch, we checkout to fabric release-1.0 branch.
- Build Docker Images:
  - fabric-sdk-node makes use of just the peer and orderer images, we build only peer-docker and order-docker docker images using `make peer-docker` & `make orderer-docker` to reduce the build time.

The same proceess applies to fabric-ca repository too.

Step 2: - Once the images are ready, CI script execute docker-compose file to spinup the network from /test/fixures directory. `docker-compose up >> dockerlogfile.log`

Step 3: - After the network is up, install the nodejs version based on the Job we are running. If the job name says `fabric-sdk-node8-verify-x86_64`, script installs nodejs version `8.9.4` in x86_64 build machine

Step 4: - After the nodjs version installed, CI script executes `npm install`, `gulp` & `gulp ca` commands to download all the dependent packages.

Step 5: - Once the environment is ready, CI script executes `gulp test` command which executes ['clean-up', 'lint', 'pre-test', 'docker-ready', 'ca'] build tasks.

Above process is applicable to both ** verify ** and ** merge ** jobs.

After the builds are executed successfully, it sends a voting to gerrit patch set with a +1, or -1 if the build fails.

Next, on a successful code review(+1) and merge by the maintainers, Jenkins triggers the **Merge** jobs. The merge jobs for **fabric-sdk-java** perform all steps detailed above and also publish node modules to NPM Open Source Registry.

- An initial validation is made to check the version of the npm modules created. The version of the created npm modules is compared with the version specified in **package.json** file. The package.json file holds the current npm details that include the version number too.
- When the npm version matches with the current specified version in **package.json** file, this fabric client/fabric-ca client npm version is not published in Merge jobs.
- If the npm module version does not match the current version in the package.json and it has a 'snapshot' in it's version tag, it is published as the next unstable version of npm.

If the npm module matches the current existing npm version in the package.json file and it has a `snapshot` in it's version tag, it is incremented and published as the next unstable version for the existing npm version. For example, if the existing unstable npm version with the `snapshot` tag ends with number 84, the next unstable version is incremented by +1 and is stored with the `snapshot` tag ending with 85. The folowing are two unstable npm versions.

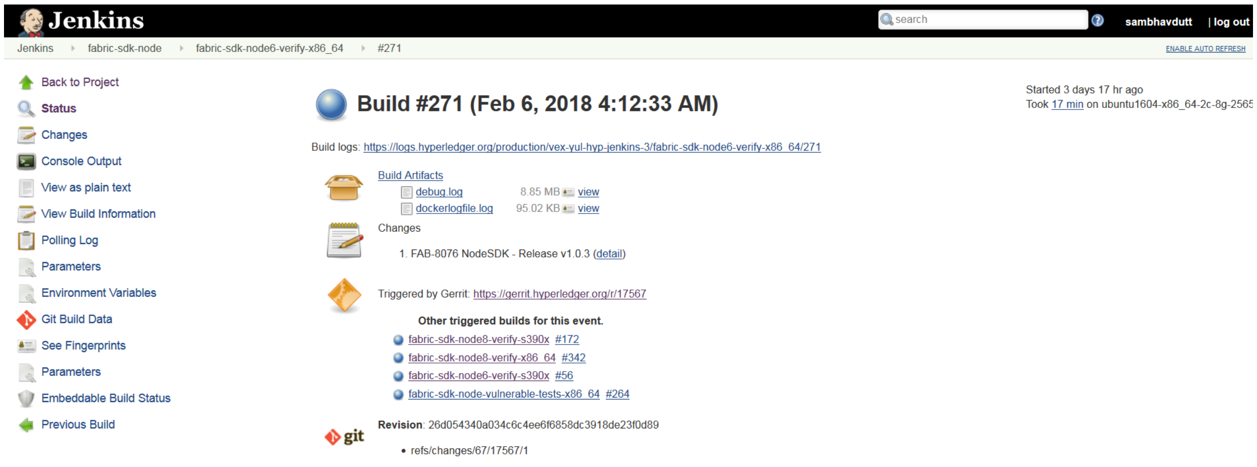
```
fabric-client@1.1.0-snapshot.85
fabric-client@1.1.0-snapshot.84
```

The same process is followed in fabric-ca Merge jobs.If you wish to look at npm packages for **fabric-client** or **fabric-ca-client**, you can select the following links.

- fabric-client npm
- fabric-ca-client npm

Once the tests are executed, Jenkins performs some pre-defined tasks to project the progress of each of the tests from beginning to end, also known as *Post Build* actions, In this case for the **fabric-sdk-node**.

- Jenkins publishes and displays the code coverage report on console output.
- The CI team configured one of Jenkins feature/plugin, the Cobertura code coverage report to publish the code coverage in a well presented format.
- Archive the build artifacts and display these build logs on the Jenkins console.



Build #271 (Feb 6, 2018 4:12:33 AM)

Started 3 days 17 hr ago
Took 17 min on ubuntu1604-x86_64-2c-8g-2565

Build logs: https://logs.hyperledger.org/production/vex-yul-hyp-jenkins-3/fabric-sdk-node6-verify-x86_64/271

Build Artifacts
debug.log 8.85 MB view
dockerlogfile.log 95.02 KB view

Changes
1. FAB-8076 NodeSDK - Release v1.0.3 (detail)

Triggered by Gerrit: https://gerrit.hyperledger.org/r/17567

Other triggered builds for this event.
- fabric-sdk-node8-verify-s390x #172
- fabric-sdk-node8-verify-x86_64 #342
- fabric-sdk-node6-verify-s390x #56
- fabric-sdk-node-vulnerable-tests-x86_64 #264

Revision: 26d054340a034c6c4ee6f6858dc3918de23f0d89
- refs/changes/67/17567/1

Fig. 8: ConsoleOutPut

10.5.2 Build Notifications

The build results can be viewed on the Jenkins console, where depending on the result it displays with a colored bubble (green for success, red for failure) and a vote from the CI (+1 or -1) on the gerrit commit/change.

10.5.3 Trigger failed jobs through gerrit comments

Re-trigger of builds is possible in Jenkins by entering **reverify** in a comment to the gerrit change that retriggers all the verify jobs. To do so, follow the below process:

Step 1: Open the gerrit patchset for which you want to reverify the build

Step 2: Click on **Reply**, then type `reverify` and click **Post**

This kicks off all the fabric-sdk-node verify jobs. Once the build is triggered, verify the Jenkins console output and go through the log messages if you are interested in knowing how the build is making progress.

In somecases, Jenkins may fail only in one or two CI jobs due to which network issues. In such cases, restarting all the fabric-sdk-node jobs through `reverify` comment is not necessary. Instead, the developer can post below comment to trigger the particular failed build:

`reverify-node8z` - to restart the build on sdk-node8-verify s390x platform.

`reverify-node8x` - to restart the build on sdk-node8-verify x86_64 platform.

`reverify-node6z` - to restart the build on sdk-node6-verify s390x platform.

`reverify-node6x` - to restart the build on sdk-node6-verify x86_64 platform.

10.5.4 Questions

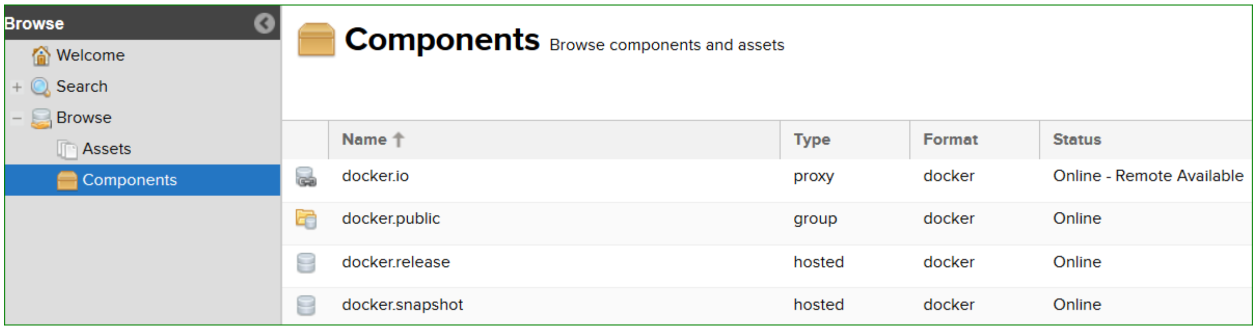
Please reachout to us in #fabric-ci or #ci-pipeline RC channels for Questions or concerns related to fabric-sdk-node CI process.

10.6 Publish Docker Images to Nexus3

We publish v1.0 docker images to Nexus repository for quicker and better usage of docker images and this also reduces time to build images manually. Once the images are stable, we publish them to **docker.release** component based on the release plan. Below is the process we follow in CI and instructions are provided to pull these images.

We have a script to pull all these images, re-tag it to Hyperledger images then delete Nexus Docker images from machine.

Jenkins CI publishes below listed Docker images to [Daily Snapshots] (https://nexus3.hyperledger.org) after every successful end-to-end tests of CLI, NODE, JAVA. Example:



Name	Type	Format	Status
docker.io	proxy	docker	Online - Remote Available
docker.public	group	docker	Online
docker.release	hosted	docker	Online
docker.snapshot	hosted	docker	Online

Fig. 9: nexus

Daily snapshots are pushed to Nexus3 from port 10003 into docker.snapshot.

nexus-docker CI job executes below steps:

10.6.1 Build & Push Docker images

- Clone **latest** commit from Fabric repository `git clone ssh://hyperledger-jobbuilder@gerrit.hyperledger.org:29418/fabric`

- Run `make docker` to build v1.0 Docker images
- Tag `hyperledger/fabric-$IMAGE_NAME` to Nexus as mentioned below

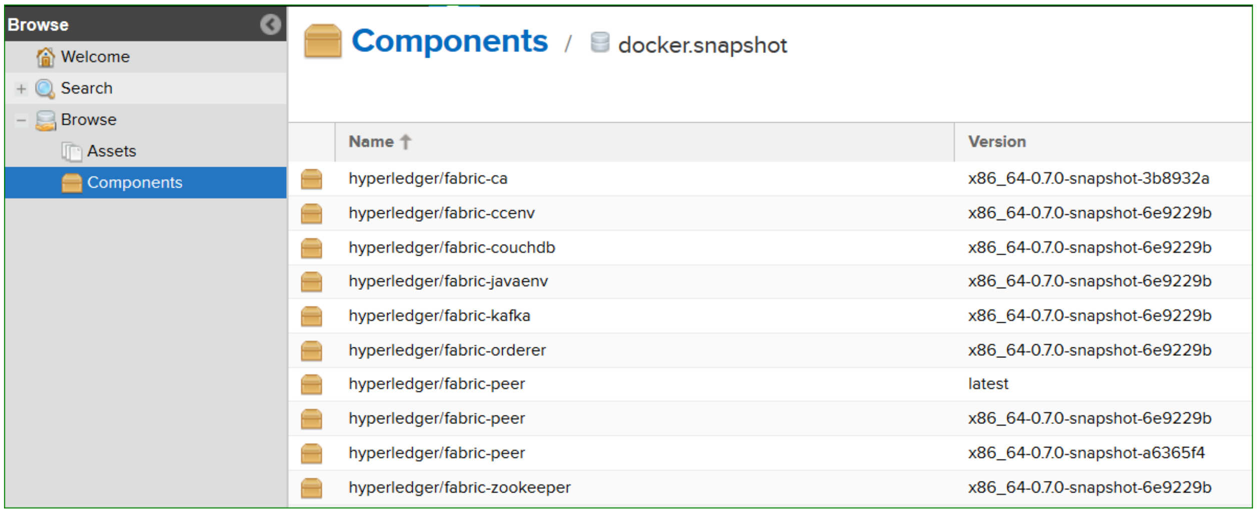
```
docker tag $ORG_NAME-$IMAGE_NAME:latest $NEXUS_URL/$ORG_NAME-$IMAGE_NAME:$FABRIC_
↪SNAPSHOT_TAG
```

- Push Docker images to Nexus as mentioned below

```
docker push $NEXUS_URL/$ORG_NAME-$IMAGE_NAME:$FABRIC_SNAPSHOT_TAG
```

After images are published to docker.snapshot component, developers has to pull Docker images from nexus repository. Follow the below steps to download Docker images from NEXUS repository.

You can see image references here



Name	Version
hyperledger/fabric-ca	x86_64-0.7.0-snapshot-3b8932a
hyperledger/fabric-ccenv	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-couchdb	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-javaenv	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-kafka	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-orderer	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-peer	latest
hyperledger/fabric-peer	x86_64-0.7.0-snapshot-6e9229b
hyperledger/fabric-peer	x86_64-0.7.0-snapshot-a6365f4
hyperledger/fabric-zookeeper	x86_64-0.7.0-snapshot-6e9229b

Fig. 10: Docker Images

10.6.2 Pull Docker Images

To pull Docker images from Nexus repository, follow below steps

- Login as docker user
  - *docker login -u docker -p docker nexus3.hyperledger.org:10001*
  - *docker pull nexus3.hyperledger.org:10001/:math:'ORG_NAME-'IMAGE_NAME:$FABRIC_SNAPSHOT_TAG*

Example:

```
docker pull nexus3.hyperledger.org:10001/hyperledger/fabric-peer:x86_64-0.7.
0-snapshot-6e9229b
```

Use 10001 for any read/pull requests. Port 10002 and 10003 are used strictly for pushing images and not pulling.

10.6.3 Re-Tag Docker Images

After pulling the Docker images, follow below steps to re-tag Nexus tag to hyperledger

- Re-Tag

  - `docker tag $NEXUS_URL/$ORG_NAME-$IMAGE_NAME:$FABRIC_SNAPSHOT_TAG hyperledger/fabric-$IMAGE_NAME:$SNAPSHOT_TAG`
  - `docker tag $NEXUS_URL/$ORG_NAME-$IMAGE_NAME:latest hyperledger/fabric-$IMAGE_NAME:latest`

Example:

```
docker tag nexus3.hyperledger.org:10001/hyperledger/fabric-peer:x86_64-0.7.0-snapshot-
↪6e9229b
hyperledger/fabric-peer:latest
```

Above process applies to **fabric-ca** Docker images as well.

10.7 Release Process

Below is the detailed plan on the Hyperledger Fabric release process, namely the steps taken to publish docker images, fabric binaries and publish npm modules.

On the release day a maintainer submits a patch set to the hyperledger/fabric repository to trigger the release process by changing the following variables in the Makefile - `IS_RELEASE = TRUE`, `BASE_VERSION` and `PREVIOUS_VERSION` - to the appropriate versions.

CI triggers the **fabric-verify-x86_64** and **fabric-verify-z** jobs and returns **SUCCESS (+1)** or **FAILURE (-1)** results (Gerrit voting) back to the Gerrit patch set commit. After approving the successful patch, maintainers merge the patch which in turn triggers the CI merge jobs.

Upon notification of the successful merge, the Release Engineer creates a "Release Tag" in the **hyperledger/fabric** repository which in turn kicks off the Build Process and CI (Jenkins) triggers release jobs (listed below) based on the newly created "Release Tag".

The above process also applies to **hyperledger/fabric-ca** and **hyperledger/fabric-baseimage**.

As part of the release process, CI automatically triggers the following release jobs after a "Release Tag" is created in each repository:

- **Publish Fabric Docker images**:

CI triggers release jobs on all three platforms (**x86_64, s390x and ppc64le**) and upon a successful run, publishes Docker images (*peer, orderer, javaenv, ccenv, zookeeper, couchdb, kafka, tools*) to the (https://hub.docker.com/u/hyperledger/) account.

- fabric-app-image-release-docker-s390x
- fabric-app-image-release-docker-x86_64
- fabric-app-image-release-docker-ppc64le

What happens?:

- Release job executes `make docker` and builds docker images. The job then calls a docker push script to publish the docker images to the Hyperledger Docker Hub account.
- Each image is then tagged as follows:

  ARCH_Name-BASE_VERSION ex: (hyperledger/fabric-peer:s390x-1.0.0-alpha2)

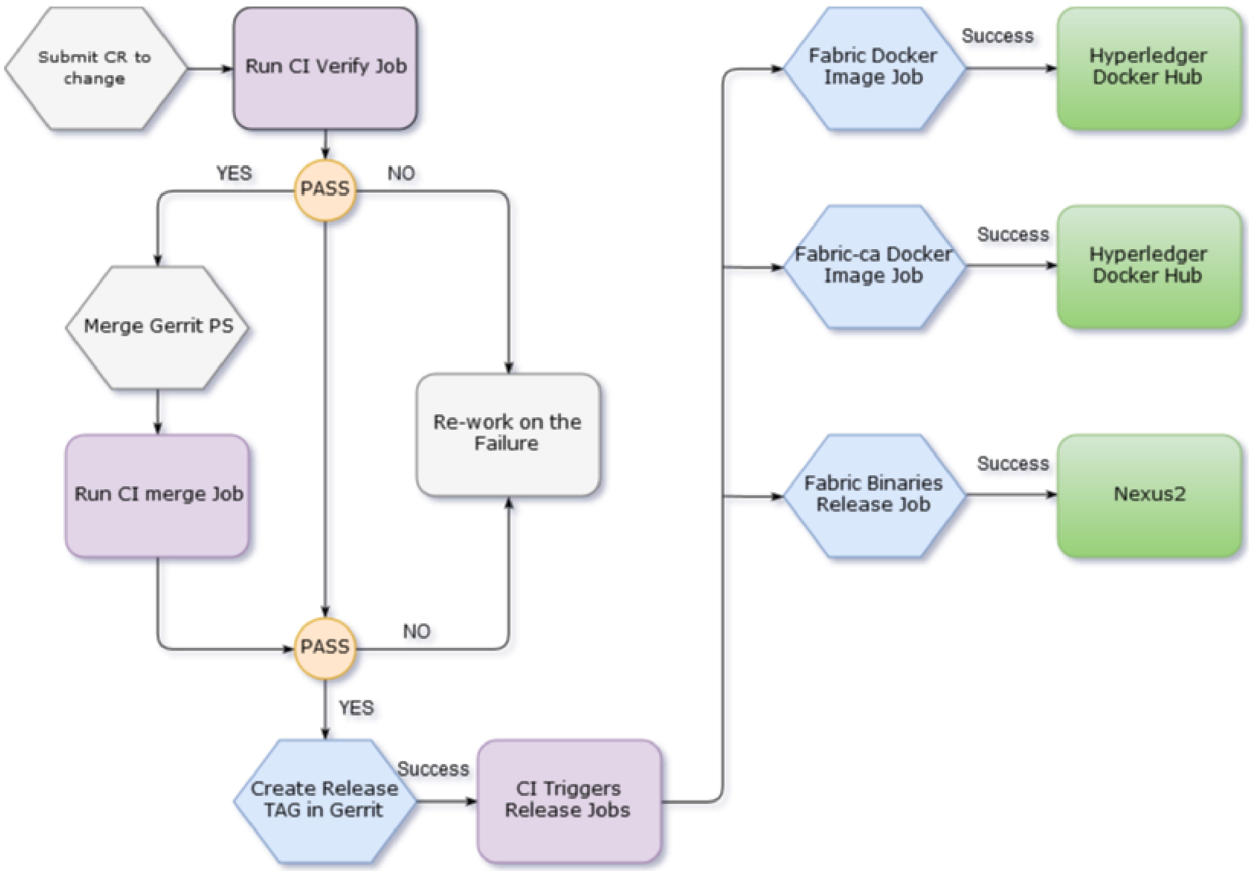


Fig. 11: Release_CI

10.7.1 Publish fabric binaries

CI triggers the below release job on x86_64 platform to publish fabric binaries to NEXUS release URL - [(https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric/fabric-binary/)] ((https://nexus.hyperledger.org/content/repositories/releases/org/hyperledger/fabric/fabric-binary/) "fabric binary")

```
fabric-binaries-release-x86_64
```

The CI does the following when a release job is triggered:

- Builds fabric binaries (cryptogen & configtxgen) for all platforms (windows-amd64, linux-amd64, linux-s390x, linux-ppc64le and darwin-amd64) using `make release-all`
- Copies the contents of the e2e_cli folder and places into release folder
- tar the complete release folder and push the tar.gz folder to NEXUS releases URL using maven-deploy-plugin:deploy-file plugin from CI.

10.7.2 Publish fabric-CA Docker images

As part of the hyperledger/fabric-ca release process, we trigger the following release job after successfully creating the "Release Tag" in the hyperledger/fabric-ca repository, along with release notes.

CI triggers release jobs on all three platforms (**x86_64, s390x and ppc64le**) and upon a successful run, publishes fabric-ca DOCKER image (**ca**) to the (https://hub.docker.com/u/hyperledger/) account.

- fabric-ca-release-x86_64
- fabric-ca-release-s390x
- fabric-ca-release-ppc64le

What happens?:

- Release job executes `make docker` and builds DOCKER images. The job then calls a DOCKER push script to publish the DOCKER images to the Hyperledger Docker Hub account.
- Each image is then tagged as follows:

  componentname-Platform ARCH Name-BASE_VERSION ex: (hyperledger/fabric-ca:s390x-1.0.0-alpha2)

If the release jobs are not triggered automatically, the CI team triggers the above release jobs manually by providing the release version in the `GERRIT_REFSPEC` variable. Below is the process to do that:

1. Login to jenkins.hyperledger.org
2. Go to **fabric-app** view
3. Click on each platform specific release job

   (fabric-app-image-release-docker-x86_64)
4. Click on **Build with Parameters**
5. Provide `release tag` in `GERRIT_REFSPEC`

   ```
   (+refs/tags/*:refs/remotes/origin/tags/*)
   ```

   ex: +refs/tags/v1.0.0-alpha2:refs/remotes/origin/tags/v1.0.0-alpha2
6. Click on Build

10.7.3 Publish npm modules

As part of the hyperledger/fabric-sdk-node npm release process, CI triggers the following job after successfully update the **version number** in **package.json** in fabric-client and fabric-ca-client directories in hyperledger/fabric-sdk-node repository by a maintainer.

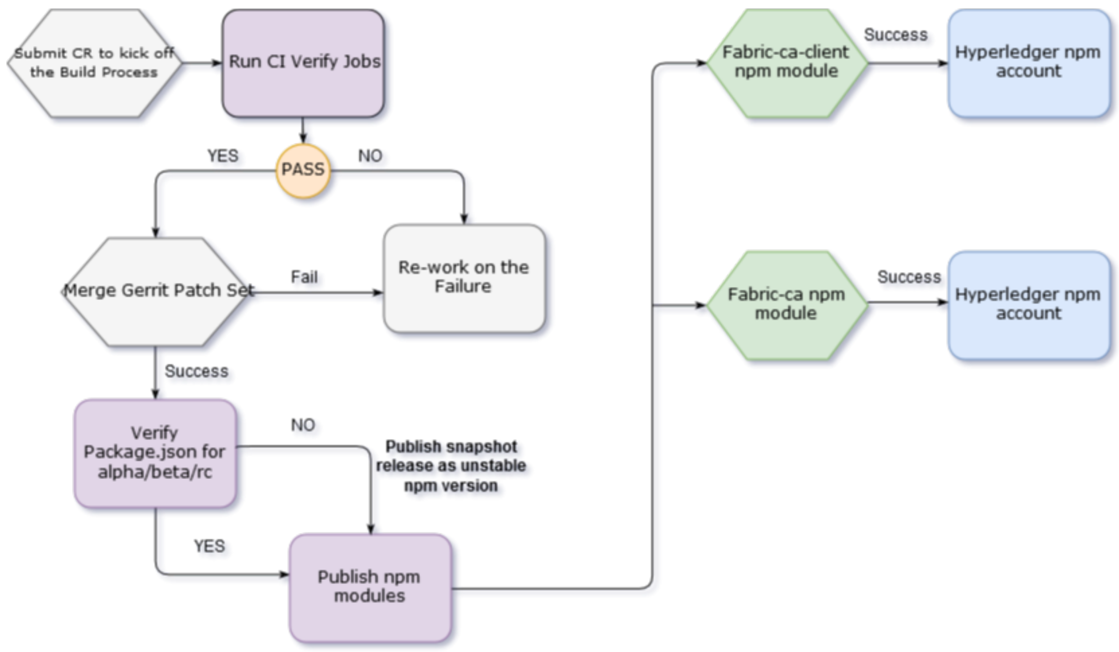


Fig. 12: npm publish

CI triggers release process on x86_64 platform and upon successful run, job publishes npm versions of fabric-client and fabric-ca-client to hyperledger npm repository (https://www.npmjs.com/package/fabric-client) and (https://www.npmjs.com/package/fabric-ca-client).

`Fabric-sdk-node-merge-x86_64` job triggers on every commit merge and look for the version number in package.json. If the version number matches **snapshot** then it releases a npm version as **unstable** like mentioned below

```
fabric-ca-client@1.0.0-snapshot.xx,
fabric-client@1.0.0-snapshot.xx
```

Otherwise it publishes npm version as a **stable** version like mentioned below

```
fabric-ca-client@1.0.0-alpha2,
fabric-client@1.0.0-alpha2
```