Syndesis Developer Handbook

Version 1.1.3-4713-g59fd16359, 2018-01-25

Syndesis Developer Handbook

Chapter 1. Introduction

This handbook contain all information required for installing, running and developing Syndesis from a developer's point of view.

Chapter 2. Buildtool "syndesis"

Syndesis uses a single tool for controlling various aspects of the build and developer related tasks. This script can be found in `$SYNDESIS_DIR/tools/bin` and is called `syndesis`. It requires bash and can be used on any Unix or macOS.

To have this script handy all the time it is recommended to either but this `bin/` directory into the path or add a symlink from `syndesis` into a directory which is already on your execution path.

The script can be used for various tasks which are selected by a so-called command_ which is the first argument to the script.

Just type `syndesis -h` to get an overview of the commands available:

Usage message

```
Syndesis Developer Tool

Usage: syndesis <command> [... options ...]

with the following commands

   build            Build Syndesis
   dev              Developer tools
   doc              Generate Syndesis Developer Handbook (SDH)
   minishift        Initialize and manage a Minishift developer environment
   system-test      Run system tests

"build" is the default command if no command is specified.
```

There are a handful of global options which can be used:

`--help`	`-h`	Print usage information. If a command is given print out commands specific options
`--rebase`	`-r`	Rebase your project directory before building to integrate upstream changes to your local repository. See below for details how this works.
`--verbose`		Set verbose mode, which is useful mostly only for debugging the script itself.

Rebase to upstream

To easily rebase on changes which have been merged upstream to master, you can use the option `--rebase` (short: `-r`). This command assumes that you have forked the Syndesis GitHub repositories and you have "origin" and "upstream" remotes like

```
$ git remote -v
origin    git@github.com:rhuss/syndesis.git (fetch)
origin    git@github.com:rhuss/syndesis.git (push)
upstream  https://github.com/syndesisio/syndesis (fetch)
upstream  https://github.com/syndesisio/syndesis (push)
```

With this in place, a `--rebase` performs the following steps:

- Refresh upstream remote: `git fetch upstream master`
- Try a `git rebase upstream/master` which rebases your current local working branch.
- If this fails because you have uncommitted work:
  - A `git stash` is performed
  - The rebase is retried and should succeed
  - `git stash pop` brings back your changes. Stashing can fail with conflicts which you would have to resolve on your own.

Development Modes

The Syndesis application consists of a set of Docker images OpenShift resources descriptors for installing Syndesis. For development, minishift is used, and most of the commands assume that you have minishift installed locally and executable directly from your path. Minishift can be downloaded and installed from https://github.com/minishift/minishift/releases

You can choose from two different modes how Docker images are installed and updated:

Docker mode

The Docker mode avoids OpenShift specific image stream and uses plain Docker image references in the deployment descriptors. Without an extra Docker registry push, this only works smoothly on single node OpenShift installation like with Minishift or `oc cluster up`. For Minishift, the Docker daemon exposed with `minishift docker-env` is used. The advantage of this mode is that building is a bit faster and you avoid the overhead of image streams. On the other side, you have to kill pods when images are updated so that the new images are picked up.

OpenShift mode

The OpenShift mode uses OpenShift S2I builds and image streams. This mode works also when running with a real OpenShift cluster and is not restricted to Minishift usage. Another advantage is that a build automatically triggers a redeployment, so you don't have to kill any pods manually (or with this script). On the con side is a slightly slower build.

Both modes are currently supported, but the OpenShift mode is recommended.

Commands

All other options are specific to each command. You get a list of those options with `syndesis <cmd> -h`. These options are described in detail in the next sections.

The following commands are available:

Command	Description
build	Used for building Syndesis and its various modules.
ui	Start the UI for local development *(not implemented yet)*
minishift	Start and install Syndesis on Minishift
deploy	Deploy Syndesis to a running cluster (other than Minishift) *(not implemented yet)*
system-test	Run System test against an OpenShift cluster
dev	Utility commands useful during development
doc	Generating and publish this documentation
release	Release Syndesis' Maven and NPM artefacts *(not implemented yet)*

If no command is given, `build` is the default. Remember a command must be the first argument, but there are additional possibilities to specify commands:

- You can use the form `--command` anywhere as an option, too. E.g. using `--minishift` is the same as specifying "minishift" as the first argument.
- A `--mode command` can be used, too (e.g. `--mode sytem-test`)

The next sections describe the commands in detail. To add a new command, just drop a script file into

2.1. syndesis build

The primary goal of `syndesis` is to build Syndesis. It is mainly a wrapper around Maven but adds some convenience for common developer workflows.

2.1.1. Usage

```
Usage: syndesis build [... options ...]

Options for build:
-b  --backend                Build only backend modules (core, extension,
integration, connectors, rest, verifier)
    --images                 Build only modules with Docker images (ui, rest,
verifier, s2i)
-m  --module <m1>,<m2>, ..   Build modules
                             Modules: ui, rest, connectors, s2i, verifier,
integration, extension, core
-d  --dependencies           Build also all project the specified module depends on
    --init                   Install top-level parent pom, too. Only needed when used
with -m
    --skip-tests             Skip unit and system test execution
    --skip-checks            Disable all checks
-f  --flash                  Skip checks and tests execution (fastest mode)
-i  --image-mode  <mode>     <mode> can be
                             - "none"      : No images are build (default)
                             - "openshift" : Build for OpenShift image streams
                             - "docker"    : Build against a plain Docker daemon
                             - "auto"      : Automatically detect whether to
                                             use "openshift" or "docker"
    --docker                 == --image-mode docker
    --openshift              == --image-mode openshift
-p  --project <project>      Specifies the project to create images in when using '--
images s2i'
-k  --kill-pods              Kill pods after the image has been created.
                             Useful when building with image-mode docker
-c  --clean                  Run clean builds (mvn clean)
    --batch-mode             Run mvn in batch mode
```

2.1.2. Modules

A plain `build` command without any options performs a plain `mvn install` for all modules. This compiles all Java and Javascript artefacts and also runs all tests and code checks.

You also can compile only specific modules by using the `--module` (short: `-m`) option with a comma-separated list of modules.

The following modules are available:

rest	Main backend providing a REST API for the user interface
ui	The SPA user interface application
verifier	Verifier used for verifying connections and providing connector metadata
connectors	All connectors used by Syndesis out of the box
integration	Support libraries for running integrations
extension	Tools for developing and running Syndesis custom extensions

core	Core module containing common code
s2i	S2I base image used for building the runtime images

Also, to specify each module individually, you can also use some module groups:

- `--backend` (short: `-b`) for the modules `rest`, `integration`, `verifier`, `connectors`, `extension`, `core`
- `--images` build all modules which result in Docker images, and also builds the Docker images in addition to the Maven artefacts.

When the option `--dependencies` (short: `-d`) is given in addition to `--modules`, also all modules which the specified modules depend on are build, too.

2.1.3. Tuning

By default, all checks like license or code quality checks are performed. Also, all unit and local integration tests are run. A full build eats up quite some time, but you should always run at full blast before submitting a pull request.

However, to speed up the turnaround, several speed-up options are available. The following table shows these options, and also how long a full clean build over all modules takes: (but without building images)

Option	Description	Time
none	Default mode with all checks and tests	
`--skip-tests`	Skip all unit and local integration tests	
`--skip-checks`	Skip sanity checks like for correct license headers and	
`--flash`	Fastest mode with skipping all checks and tests and with even some other aggressive optimizations	

2.1.4. Images

As described in Development Modes there are two ways how Syndesis Docker images can be created and deployed.

You can decide which mode to use with the option `--image-mode` (short: `-i`) and the following options are available:

Option	Shortcut	Description
`-i auto`		Auto-select the mode based whether you are connected to OpenShift (e.g via Minishift) or not.
`-i docker`	`--docker`	Enable Docker build mode. Access to a Docker daemon is required.
`-i openshift`	`--openshift`	Use OpenShift S2I for building the images. An active login to OpenShift is a precondition for this. You can use `--project` (`-p`) for specifying the target namespace where to create the image. By default, it is created in the current namespace
`-i s2i`	`--s2i`	Shortcut for `--openshift`

Option	Shortcut	Description
`-i none`		Don't build Docker images even when implicitly enabled (e.g. when using `--images`)

Specifying the mode also enables building the Docker images which otherwise would not be built.

If the direct Docker mode is used it, you have to restart the Pods, so the newly created images are picked up. For this you can use `--kill-pods` (short: `-k`) which will kill the corresponding pod after the image has been created. This only works when you are connected to an OpenShift cluster (like for minishift)

Example

```
# Build all image with openshift mode
syndesis --images --openshift

# Build to the minishift Docker daemon
eval $(minishift docker-env)
syndesis -i docker -k

# Or:
# syndesis --image-mode docker

# Or:
# syndesis --docker
```

2.2. syndesis ui

"syndesis ui" is not implemented yet

This is just a placeholder, still in the planning phased. Nevertheless we already start to document the feature for an "UX first" approach.

2.3. syndesis minishift

With `syndesis minishift` you can adequately manage a minishift installation for hosting Syndesis. This command is especially useful for a simple and self-contained development workflow.

`syndesis minishift` requires that you have a current minishift in your path. You can download it directly from GitHub.

2.3.1. Usage

```
Usage: syndesis minishift [... options ...]

Options for minishift:
    --install                 Install templates to a running Minishift.
-p  --project                 Install into this project. Delete this project if
    --reset                   Reset the minishift installation by
                              'minishift delete && minishift start'.
    --full-reset              Full reset by
                              'minishift stop && rm -rf ~/.minishift && minishift
start'
    --memory <mem>            How much memory to use when doing a reset. Default: 4GB
    --cpus <nr cpus>          How many CPUs to use when doing a reset. Default: 2
    --disk-size <size>        How many disk space to use when doing a reset. Default:
20GB
    --show-logs               Show minishift logs during startup
    --openshift-version <ver> Set OpenShift version to use when reseting (default:
v3.6.0)
                              already existing for --install.
                              Default project: "syndesis"
-i  --image-mode <mode>       Which templates to install: "docker" for plain images,
                              "openshift" for image streams (default: "openshift")
-o  --open                    Open Syndesis in the browser
```

2.3.2. Installing Syndesis

You can easily install Syndesis with the option `--install`. This option triggers the creation of all relevant OpenShift resources objects in the currently connected OpenShift project.

If you want to use a different project, then use `--project` (short: `-p`) to specify this project.

Any existing project will be deleted first when specified with `--project`. This option is also an easy and quick way to recreate a Syndesis installation.

As explained in Development Modes there are two workflows for developing Syndesis: Either with plain Docker images and killing pods (`--image-mode docker`) or with OpenShift S2I builds and image streams (`--image-mode openshift`). You have to specify the mode already when installing Syndesis as this choice also determines the OpenShift templates to use. By default the `openshift` mode is selected, so you have to specify `--image-mode docker` only when using the plain Docker image mode.

2.3.3. Resetting Minishift

The quickest way to get a fresh Syndesis setup is to use `--project` which will install Syndesis into a clean, new project.

However, you can also recreate the whole Minishift installation with `--reset`. This will delete the Minishift VM (`minishift delete`) and create a new one (`minishift start`). It doesn't harm if the Minishift VM does not exist so that you can use `--reset` also on a fresh Minishift installation.

If you want to get a real clean installation use `--full-reset` which deletes the `~/.minishift` directory

which holds downloaded artefacts like the ISO image for the Minishift VM. Using `--full-reset` forces Minishift to re-download all those files.

There are several options which influence the re-creation of the VM:

Option	Description	Default
`--memory`	Memory to use for the Minishift VM.	4 GB
`--cpus`	Number of CPUs used for the Minishift VM.	2
`--disk-size`	Disk space used for Minishift.	20 GB
`--show-logs`	Whether to show OpenShift logs during startup.	false
`--openshift-version`	OpenShift version to use	3.6.0

2.3.4. Example

This short example performs the following actions:

- Stops and deletes a running Minishift VM (if existent)
- Removes `~/.minishift` (if existent)
- Install Syndesis in OpenShift modes (S2I builds & image streams) in project `syndesis`
- Open Syndesis UI in the default browser

```
# Complete fresh installation in project "syndesis"
syndesis minishift --full-reset --install --project syndesis

# Open Syndesis in default browser
syndesis minishift -o
```

2.4. syndesis system-test

The `system-test` command is for running a full blown system test of Syndesis by installing it in one of multiple projects which are managed in a pool.

2.4.1. Usage

```
Usage: syndesis system-test [... options ...]

Options for system-test:
    --project <project>       The test project to use
    --token <token>           Token for connecting to the server
    --server <url>            OpenShift aerver url to use for the tests.
                              If not given, use the currently connected server
    --pool <project>          If no project is given use, a pooling mechanism.
                              This pool has to be created before with --create-pool
    --test-id <id>            Id to identify the test run
    --create-pool <prefix>    Create project pool for system-tests with
                              all projects with the given prefix
    --list-pool               Show all locks for the pool
    --release-project <t-id>  Release project for given test id (or all if no test id
is given)
```

2.4.2. How it works

This section needs a more detailed explanation how the systems tests are working in detail.

2.5. syndesis dev

Dev commands are useful helpers for developing Syndesis

2.5.1. Usage

```
Usage: syndesis dev [... options ...]

Options for dev:
    --debug <name>            Setup a port forwarding to <name> pod (default: rest)
```

This command enable port-forwarding of port 5005 from a specific pod (by default: "rest") to port 5005 on the localhost. You then can point your Java IDE to port 5005 on localhost for connecting for remote debugging. As argument to `--debug` "rest", "verifier" and "atlasmap" can be used, which are our Java based services.

2.6. syndesis doc

This command is used to generated and manage this documentation which you are currently reading.

2.6.1. Usage

```
Usage: syndesis doc [... options ...]

Options for doc:
-d  --directory <dir>         Top-level dir holding doc source and output directory.
Default: "doc/sdh"
-i  --input <file>            Input file to use. Default: "index.adoc"
-o  --out <dir>               Directory to generate files (default: "output")
    --html                    Generate HTML pages
    --pdf <out>               Generate PDF and write it to <out> (default: "sdh.pdf")
    --epub <out>              Generate Epub and write it to <out> (default:
"sdh.epub")
    --gh-pages <msg>          Create everything into the gh-pages branch and commit
with <msg>
-l  --local                   Use locally installed commands instead of Docker image
```

`syndesis doc` uses by default a Docker container to create the documentation. Therefore a Docker daemon must be accessible, and it must allow bind volume mounts to the local `doc/sdh` directory. Bind mounts are possible for Linux Docker daemons and for "Docker for Mac". But it is not the case for the Minishift exposed Docker daemon, as this daemon is running isolated in a VM.

You can always run `asciidoctor` locally with the `--local` (short: `-l`) options. See the Asciidoctor manual for more information how to install Asciidoctor.

2.6.2. Output

Impit and output has sane defaults but can be changed with the following options

`--input <file>`	`-i <file>`	Input document in Asciidoc format which is by default `$SYNDESIS_DIR/doc/sdh/index.adoc`
`--output <dir>`	`-o <dir>`)	Output director, default id `$SYNDESIS_DIR/doc/sdh/output`

By default `syndesis doc` creates the documentation in HTML format, but more formats are supported:

`--html`	Create HTML documentation in the output directory
`--pdf <name>`	Create PDF documentation. `<name>` is optional, but when given, then this specifies the file name generated in the output directory. By default `sdh.pdf` is used.
`--epub <name>`	Create the documentation in Epub format. `<name>` is optional, but when given, then this specifies the file name generated in the output directory. By default `sdh.epub` is used.

2.7. syndesis release

"syndesis release" is not implemented yet

This is just a placeholder, still in the planning phased. Nevertheless we already start to document the feature for an "UX first" approach.

2.8. syndesis deploy

With `syndesis deploy` you can deploy Syndesis to an arbitrary OpenShift cluster. You have to be connected to the cluster with `oc login` for this command to work. It's an alternative to `syndesis minishift` but does not support a Development workflow.

You have to provide the route name with `--route`, how the project in the given cluster can be reached. E.g. `--route app-proj186023.6a63.fuse-ignite.openshiftapps.com`. By default this commands installs in the currently connected project, but you can specify an alternative project with `--project <project>`. If this project already exists, it gets deleted unconditionally.

2.8.1. Usage

```
Usage: syndesis deploy [... options ...]

Options for deploy:
-p  --project                Install into this project. Delete this project if it
                             already exists. If not given, install into the current
project
    --route                  Route to use (mandatory)
-w  --watch                  Wait until cluster is up
    --man                    Open HTML documentation in the Syndesis Developer
Handbook
```

Chapter 3. Development Process

3.1. Issue and Pull-Request Labels

We use GitHub labels to categorize epics, issues and tasks. They are the foundation of our process, so please use labels for issues.

Labels are living entities. This document describes the current status and might be slightly outdated. Please send a PR to adopt this section if the label structure changes. Also feel free to discuss the label structure anytime. It's essential that labels describe our process, not that we have to adapt our process for these labels.

Labels are grouped. Each label consists of two part: A **Group** and a **Name** which are separated by a slash (`/`). For example, the label `module/ui` is used to mark issue which is relevant to the Syndesis UI module.

The following label groups are available. There must be only at most one label from the "Exclusive" groups.

Group	Description	Excl.
cat/	Misc categories which are can be added freely	
prio/	Priority of the issue. Only one `prio/` label must be added per issue. `prio/p0` is of highest priority, `prio/p4` the lowest one.	
ext/	Reference to external projects	
d/	Day labels for tagging an Epic to talk about at specific day in the daily meeting	
module/	Internal Syndesis modules	
notif/	Notification label which can be added and removed to ping certain subteams	
size/	Tee shirt size for issues. Sizing is a subjective assessment and should be done relative to other issues.	
status/	Status of an issue or PR.	
target/	Target milestone for an Epic. These are the Technical Preview (TP) targets and GA.	

Each label group serves a particular purpose, and for each issue and PR, it should be considered whether a label from a group applies.

3.1.1. Modules

Labels from this group reference our application components like "rest", "ui", "uxd" or "connectors". Each sub-team is responsible for one or more modules, and every module has an 'owning' team. That does not mean that members of other teams are allowed to work on such modules. Contrary, this is even encouraged. But its just there so that teams can filter on issues and PRs which are relevant to them.

An issue can carry many module labels. Especially Epics will carry more than such label as they touch more than one module (otherwise it wouldn't be an epic).

Module	Descriptions
module/connectors	Supported camel connectors
module/deploy	OpenShift templates
module/rest	REST backend for managing integrations
module/runtime	Library used in the the integration runtimes
module/s2i	S2I base image for building integrations
module/tests	System tests for testing the whole applications
module/ui	User interface SPA, talking to the REST backend
module/uxd	User experience (UX) designs
module/verifier	Service for connector meta-data and verification of connections
module/extension	Tools for developing Syndesis extensions
module/core	Syndesis core module

3.1.2. Categories

Labels from the `cat/` group are labels which can always be applied and which does not fit in another category. Currently we have these categories:

Category	Description
cat/bug	A bug which needs fixing.
cat/blocker	A blocker, which is a bug which needs to be fixed as soon as possible.
cat/enhancement	PR label for an enhancement of an existing feature.
cat/feature	PR label for a new feature
cat/discussion	This issues requires a discussion.
cat/question	For issues holding a question.
cat/build	For issues which have relevance for the build system.
cat/design	A concrete UX design. Use this for PRs containing UX designs.
cat/process	Development process related issues carry this label.
cat/research	Label used for issues which describe some research work
cat/retro	Label for action items which are the result of a retrospective.
cat/starter	An issue which is easy to solve and can be used for ramping up new developers.
cat/techdebt	Label for issues identifying technical debt.

Category	Description
cat/techdoc	Technical developer information (likes this handbook ;-) related issues.
cat/tooling	Issues used for tooling around the development process (note: could be merged with "build")

3.1.3. Notification

Notification labels from the `notif/` group serve a particular purpose. They are used when one team wants to notify another group that a specific issue might have them relevance to them.

Currently we have two notifcation labels:

Notification	Description
notif/doc	The issue needs some attention from the docs team. This might because a new feature has been introduced or, more important, an existing feature has changed for which a documentation already exists.
notif/uxd	This label should be used for issues which needs some attention from the UX team. This might because a new feature has been introduced or, more important, an existing feature has changed for which a UX design already exists.

It is important to note that these labels also be removed when the notification has been received.

For example, when a UI feature like an input form changes. Then the UI team attaches a `notif/uxd` label to the PR which introduces this change. The UX team, detects with a filter search on this label, that there is a new notification. It then decides, whether UX design needs to be updated or not. In any case, they are removing the `notif/uxd` label and add a `module/uxd` label if this PR indeed requires a UX design update. If no update is required, then the label is removed without replacement.

3.1.4. External references

This label group should be used if an external system is referenced, which is not part of the Syndesis mono repo.

External Project	Description
ext/atlasmap	atlasmap data mapper
ext/camel	Camel
ext/qe	syndesis-qe suite
ext/docs	syndesis-documentation End user documentation

For the future, we plan to add more of these external repos into the Syndesis mono repo (like documentation or QE). If this happens, then labels should be converted to `module/` kind of labels.

3.1.5. Daily Meeting Labels

This category holds five labels: `d/mon`, `d/tue`, `d/wed`, `d/thu`, `d/fri`, one for each working day. They are

used to mark an Epic so that it is talked about the daily meeting on that day. The reason for this selection is, that we want to keep the daily meetings still for 15 mins but don't yet want to split up. More than one of such label can and actually should be added to one Epic. At least two-day labels must be added to an epic.

Status	Daily Meeting
d/mon	Monday
d/tue	Tuesday
d/wed	Wednesday
d/thu	Thursday
d/fri	Friday

3.1.6. Status

Status labels are unique since they may trigger some automatic actions.

The current status labels are:

Status	Description
status/blocked	The current issue is blocked by another issue. Refer to the issue itself to see what is blocking this issued. This label is purely informal.
approved	This label will be automatically applied to a PR as soon as the PR has been approved at the end of a review. It is an indicator for our PR bot to automatically merge the pull request if it passes all required tests. (Note: Should probably be renamed to `status/approved`)
status/wip	This is a PR request label which should be used for "Work-in-Progress" kind of PRs which has been submitted for early review. If this label is present on a PR, the PR is not merged, even when it is "Approved"

Status	Description
status/2s2f	Use this label to mark issues which should be self-merged without requiring a PR review, because of its "too small to fail". Be very careful with this label, and remember a review is a service to you to help in your code quality. It is alone your responsibility when you chose this label. It's useful for minor doc updates or one line where you are 100% sure that it doesn't break the system. Please use it sparingly and responsibly. (*Need still to be implemented*) # Development Workflows Local Development If you'd like to get a local development environment up and running, for both the UI and REST API, this is how you'd do it. **Tips** - Build on branch, not master. - Callback URL Example: https://syndesis.192.168.64.29.nip.io/api/v1/credentials/callback --- # Requirements You can follow these steps if it's your first time setting up Syndesis, or if you want a fresh local installation to replace an existing one. Some environment-specific instructions may be available below as well. 1. Make sure you have installed node version >= 6.x.x and Yarn version >= 0.18.1. 2. Get a developer deployment of Syndesis running in a Minishift environment as described in the Syndesis Quickstart. Most are specific to your environment, so follow the sections below for a quick setup. The general instructions are: - Install a hypervisor for Minishift. - Install Minishift. - Install the OpenShift CLI. - Make sure it's in your `$PATH` **macOS** If you'll be using the Homebrew method, you'll obviously need to have Homebrew installed. Then, to install the hypervisor for Minishift and Minishift itself: `` ` $ brew install docker-machine-driver-xhyve $ brew cask install minishift ` `` Finally, to install the OpenShift CLI, we recommend using Homebrew: `brew install openshift-cli` **Linux & Windows** - Install a hypervisor for Minishift. For macOS, we recommend using the Docker xhyve plugin here, which can be installed using Homebrew. - Install Minishift. For macOS, we recommend you use the Homebrew method.