Continuous delivery from Travis CI

Summary	We are going to use Travis CI to deliver a new snap version to your early adopters every time you make a change in your master branch.
URL	continuous-delivery-travis
Category	snapcraft
Status	Draft
Feedback Link	https://github.com/ubuntu/codelabs/issues
Tags	snapcraft, travis, ci, cd
Difficulty	3

Overview

Duration: 1:00

`Travis CI` is a continuous integration service that can be used to build, test and deliver software. It is free for free software projects, with Travis Pro also available for private Github repositories. We will use Travis CI to build your snap and push it automatically to the edge channel of the store every time you make a change to your branch in Github.

What you'll learn

- How to build your snap in Travis CI.
- How to push your snap to the store automatically from Travis CI.
- How to use snapcraft to enable all this, with a simple command.

What you'll need

- Ubuntu 16.04 or later.
- A little knowledge of Linux command line and git.

How will you use this tutorial?

- Only read through it
- Read it and complete the exercises

What is your current level of experience working with snap?

- Novice
- Intermediate
- Proficient

What is your preferred continuous integration system?

- Travis CI
- Circle CI
- Jenkins CI
- GitLab CI
- Other
- None

Setting up GitHub

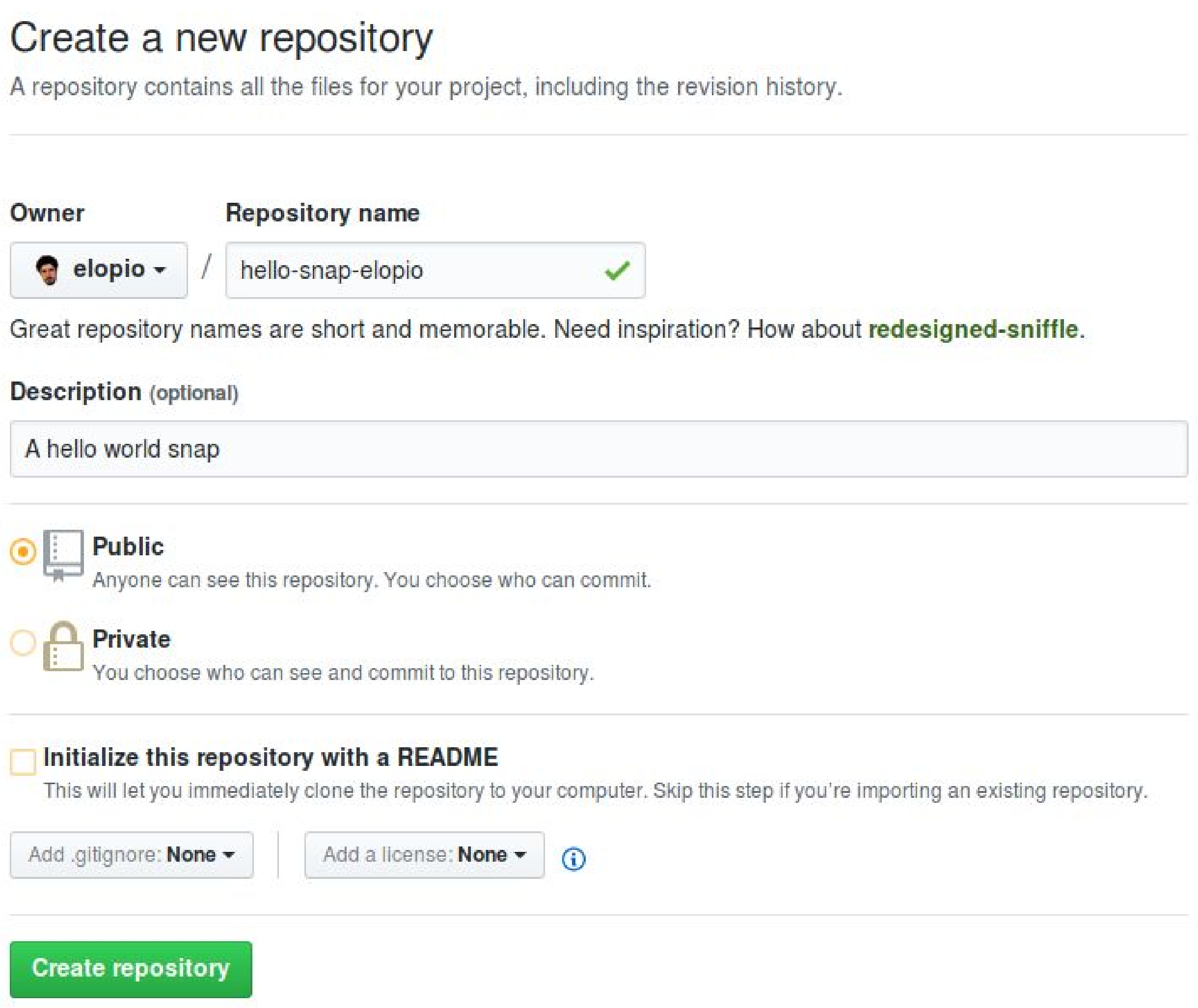
Duration: 7:00

Before we can start the continuous delivery of snaps to the store, of course we will need a project with snap packaging. It has to be hosted in GitHub to be able to execute the Travis CI job.

If you already have your free software project in GitHub packaged as a snap, you can skip this step and jump straight to the next one. If you have your project in GitHub but haven't packaged it yet, we have many tutorials that can help you making your snap. The "Create your first snap" tutorial is a good place to start.

Let's set up one GitHub project here, in case you don't have one ready.

First, go to https://github.com/ and sign up. Then, go to https://github.com/new where we will fill the details of our new repository. The only required field is the repository name, and you can enter there something like `hello-snap-yourusername`. For example, this is how the form looks like for me:


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: elopio / Repository name: hello-snap-elopio

Great repository names are short and memorable. Need inspiration? How about **redesigned-sniffle**.

Description (optional): A hello world snap

[✓ Public] Anyone can see this repository. You choose who can commit.

Private — You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Next, click the `Create repository` button, and we get back an empty repository in GitHub. Let's put our snap project in there!

Open a terminal, and run the following commands. The first one is to install git:

```
$ sudo apt install git
```

Now, clone your repository to your local workspace, using the address that's displayed in GitHub. Something like this, but with your username instead of mine:

```
$ mkdir -p ~/workspace
$ cd workspace
$ git clone https://github.com/elopio/hello-snap-username.git
$ cd hello-snap-username
```

Open a text editor, and write this:

```
name: hello-yourname
version: '2.10'
summary: GNU Hello, the "hello world" snap
description: GNU hello prints a friendly greeting.
grade: stable
confinement: strict

apps:
  hello:
    command: bin/hello

parts:
  gnu-hello:
    source: http://ftp.gnu.org/gnu/hello/hello-2.10.tar.gz
    plugin: autotools
```

Save it to `~/workspace/hello-snap-username/snap/snapcraft.yaml` (remember to replace `username` with yours!)

This is the packaging metadata of a very simple snap, that, as the description says, will just print a greeting. The "Create your first snap" tutorial mentioned before includes explanations of all the lines of this file, so please refer to it to know more about the packaging. In here we will be focusing just on the continuous delivery, so let's go on and push this snap to our GitHub repository:

```
$ git add snap/snapcraft.yaml
$ git commit -m 'add the hello snap'
$ git push
```

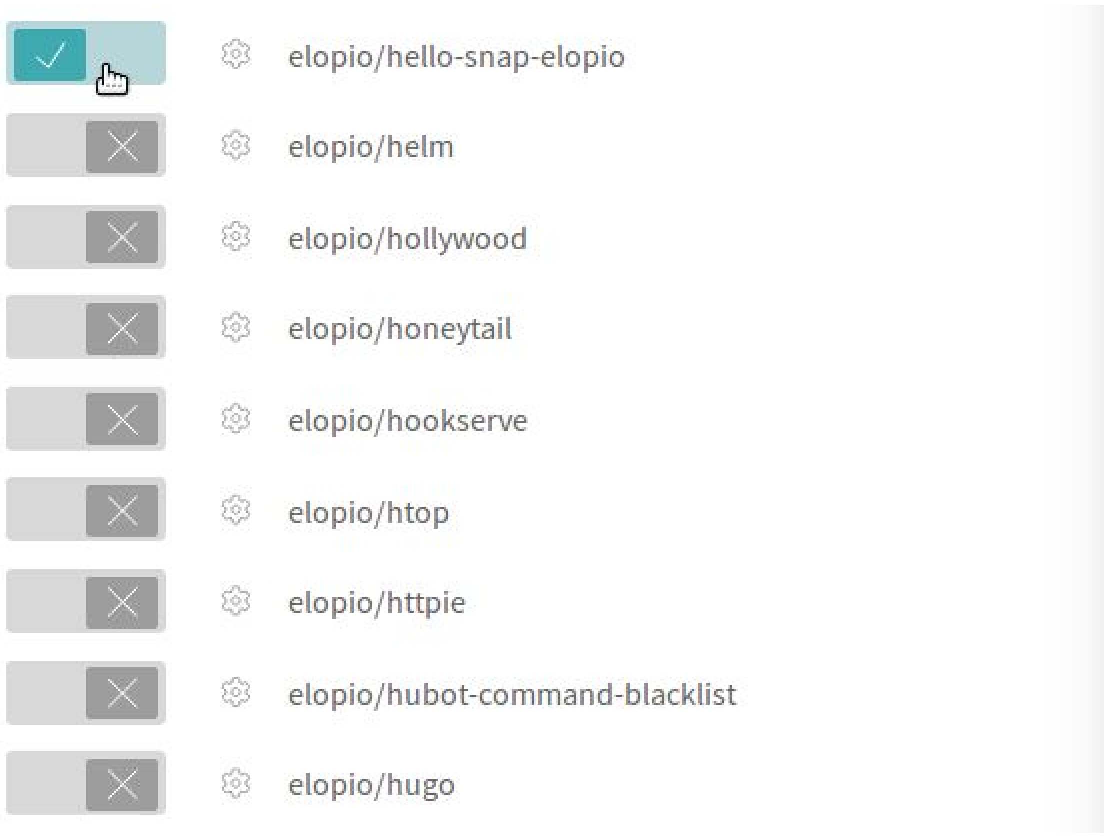
And with all the preliminaries ready, it's time to look at Travis CI...

Setting up Travis CI

Duration: 2:00

To use Travis CI, first we will have to sign up there. Let's go to https://travis-ci.org/auth and Sign in with GitHub. Then, click Authorize application to allow travis to access your GitHub account.

Now, enable the GitHub repository of your snap by flipping the switch next to it at https://travis-ci.org/profile/



elopio/hello-snap-elopio
elopio/helm
elopio/hollywood
elopio/honeytail
elopio/hookserve
elopio/htop
elopio/httpie
elopio/hubot-command-blacklist
elopio/hugo

We will also need the `.travis.yml` file for your project. Travis CI will run the steps specified in this file on every pull request and every time a change is pushed to the master branch.
During this tutorial we will configure travis to just run a few bash commands and scripts, so create a `.travis.yml` file in your locally cloned git master branch, with just the following content: `language: bash`.

```
$ cat .travis.yml
```

```
language: bash
```

Later you can use Travis CI to run your tests. For that you will have to adjust this file depending on the language of your project, and add the `install` and `script` steps. But for now, we don’t need any of that as we’ll just make continuous delivery to the store.

In the next step we’ll make sure that your snap can build successfully in Travis CI, so we can push the generated snap to the store.

Building the snap in a clean environment

Duration: 5:00

The Travis CI executors are using Ubuntu 14.04, missing some kernel changes to be able to run snapcraft from a snap. They support docker, so we will use the snapcraft docker container to build the snap for your project.

But before we build in Travis CI's executors, it would be better to make sure that the snap builds correctly in the container. When you build a snap in your machine, it could be using some dependencies that are already installed but you forgot to declare as `build-packages` in the `snapcraft.yaml`. This means that when somebody tries to build the same snap in a clean machine where those packages are missing, it will fail.

Let's install docker, to get the same environment that Travis CI will use to build the snap:

```
$ snap install docker
```

Go to the project directory, and run the `snapcraft` command in the container:

```
$ sudo docker run -v $(pwd):/my-snap snapcore/snapcraft sh -c "apt
update && cd /my-snap && snapcraft"
```

This will take some time while docker downloads the image, gets all the packages required by your snap, and builds it.

In the meantime, let's take a look at the command we just ran. We are calling the `docker run` command as the superuser, that's clear. This will run a command in a docker container. The `-v` argument mounts a directory in the container and the `pwd` command is used to get the current directory, so this is basically giving access to the current directory to the container, and mounting it in the `/my-snap` path. `snapcore/snapcraft` is the name of the snapcraft docker image. What follows next is the command that we want to run in that container, so we first cd to the mounted project directory and then run `snapcraft` in there.

If everything goes well, a message like this one will be printed at the end, with the name of your snap:

```
Snapped hello-snap-elopio_2.10_amd64.snap
```

If you can build your snap locally, but it fails when building it in the container, it’s probable that you are missing some `build-packages` in the `snapcraft.yaml` file. Use the error message to identify the missing packages, and try again until you are able to build the snap in a clean environment, without depending on your local machine.

Once you get the snap building in the docker container, you can be confident that it will build without problems in Travis CI. We are ready to enable continuous delivery!

Building and pushing the snap in Travis CI

Duration: 6:00

The final step to enable continuous delivery from Travis CI to all your early adopters is to add some commands to the `.travis.yml` file. But we don't really have to do much, `snapcraft` will take care of it for us.

We will need install the travis command-line utility, which will be used by snapcraft in a moment. This ruby gem is not available in the Ubuntu repositories, so installing it nicely requires a few commands:

```
$ sudo apt install ruby-dev
$ export GEM_HOME=$HOME/gems
$ export PATH=$PATH:$GEM_HOME/bin
$ gem install travis
```

To make the `travis` command permanently available in your path, you can add those two `export` lines to your `~/.bashrc` file.

Use the command we have just installed to login to your account:

```
$ travis login
We need your GitHub login to identify you.

[...]

Username: elopio
Password for elopio: ****
Two-factor authentication code for elopio: ####
Successfully logged in as elopio!
```

And again from the project directory, run:

```
$ snapcraft enable-ci travis
```

It will print a detailed explanation of what's happening. In the previous steps we made sure to have the project configured in Travis CI, the `travis` command installed and the `.travis.yml` file initialized. We have all the needed dependencies mentioned in the first paragraphs.

```
Snapcraft integration for Travis (CI).

This is an *EXPERIMENTAL* feature and subject to incompatible changes in
the future, please use with caution.

This command currently depends on a working `travis` CLI environment and
a previously initialized Travis project (`.travis.yml`).

Make sure your Travis project is also configured to "Build pushes", this
way every new push to `master` will result in a new snap revision in the
Store.

This operation will acquire properly attenuated Store credentials and
encrypt them for use in your testbed (`.snapcraft/travis_snapcraft.cfg`),
only Travis has the private key to decrypt it and will be only available
to branches of the same repository, not forks.

Then it will adjust Travis configuration ('.travis.yml') with the commands
to decrypt credentials during 'after_success' phase and install latest
`snapcraft` to build and release your snap (inside a snapcore:snapcraft
docker container) during the 'deploy' phase.

See the example below::

      sudo: required
      services:
      - docker
      after_success:
      - openssl aes-256-cbc -K <travis-key> -iv <travis-iv>
        -in .snapcraft/travis_snapcraft.cfg
        -out .snapcraft/snapcraft.cfg -d
      deploy:
        skip_cleanup: true
        provider: script
        script: docker run -v $(pwd):$(pwd) -t snapcore:snapcraft sh -c
          "apt update -qq && cd $(pwd) &&
          snapcraft && snapcraft push *.snap --release edge"
        on:
          branch: master

The dedicated credentials will be functional for exactly 1 year or until
a password change on the related Ubuntu One SSO account. In order to
refresh the project credentials, please run the following command::

      $ snapcraft enable-ci travis --refresh
```

A few interesting details of the rest of the message:

- You will be sharing a credential with Travis CI, so you are trusting this service to properly handle the decryption key.
- The store has [four channels](): stable, candidate, beta and edge. Each representing an increasing level of risk. This credential works only to release your snap to `edge;` so there's never risk to land a version built from a broken master in the other channels.
- The `deploy` step will run only after the rest of the steps declared in the `.travis.yml` file succeed. At the moment we have no other steps, but this will be very handy after you add steps to run your unit tests. In the case that a test fails, the snap will not be deployed.
- The `deploy` step will run only in the master branch, so the snap will not be deployed when a branch or a pull request is created. It will only be deployed when something new lands in master.
- After one year, your credential will be revoked and your travis execution will start reporting an error. You will have to refresh the credential with the command: `snapcraft enable-ci travis --refresh`

Whenever you are ready, we can continue.

```
Continue (y/N): y
Enabling Travis testbeds to push and release 'hello-snap-elopio' snaps
to edge channel in series '16'
Acquiring specific authorization information ...
Enter your Ubuntu One SSO credentials.
Email: leo.arias@canonical.com
Password: ****
Second-factor auth: ####

Login successful.
Encrypting authorization for Travis and adjusting project to
automatically decrypt and use it during "after_success".
Configuring "deploy" phase to build and release the snap in the Store.
Done. Now you just have to review and commit changes in your Travis
project (`.travis.yml`).
Also make sure you add the new `.snapcraft/travis_snapcraft.cfg` file.
```

Snapcraft will ask for your account information, update the `.travis.yml` file, encrypt the credential in `.snapcraft/travis_snapcraft.cfg` and prepare the Travis CI service to be able to decrypt that file so it can push to the store.

That's all it took! To finish, add the two files to the git repository, commit and push.

```
$ git add .travis.yml .snapcraft/travis_snapcraft.cfg
$ git commit -m 'Enable continuous delivery of the snap from Travis CI'
$ git push
```

If you go to https://travis-ci.org/ you will see that the execution for your repository is waiting for a free slot to run, or it has already started. It will show the execution log, and if everything goes as planned, that log should end with a message like this one:

```
Snapped hello-snap-elopio_2.10_amd64.snap
Using local configuration (`.snapcraft/snapcraft.cfg`), changes will not
be persisted.
Pushing 'hello-snap-elopio_2.10_amd64.snap' to the store.
Using local configuration (`.snapcraft/snapcraft.cfg`), changes will not
be persisted.
Uploading hello-snap-elopio_2.10_amd64.snap
[=============================] 100%
Ready to release!|
Revision 1 of 'hello-snap-elopio' created.
Using local configuration (`.snapcraft/snapcraft.cfg`), changes will not
be persisted.
Track Arch Channel Version Revision
latest amd64 stable - -
candidate - -
beta - -
edge 2.10 1
The 'edge' channel is now open.
Done. Your build exited with 0.
```

You should have also received an email message from Travis CI notifying you about the successful execution, and another from the store notifying you about the new snap.

Congratulations! Now spread the word and tell everybody to help you testing your app, with a single command:

```
$ snap install hello-snap-elopio --edge
```

Now let’s do some crowd testing!

Duration: 1:00

From now on, every single change in your project will be made available to early adopters almost immediately. Your community of testers will be very happy because their edge snap will be updated automatically; they won’t have to do anything complex to help you testing your most recent changes. It’s very likely that your community will grow, because they can be more involved with what’s happening in your project, they can report any issues as soon as they appear, and contribute making a better release. After letting this crowd testing happen for a while, you can then push to lower risk channels until you are confident enough to make it available to the world in the stable channel.

Final code

Your final code directory should now have a `.travis.yml` and a `.snapcraft/travis_snapcraft.cfg` like the ones in this repository.

You should successfully have built your snap in a clean environment, configured your project to build the snap continuously on Travis CI, and deliver a new version to the edge channel for every change on your master branch. You can relax because your delivery pipeline is all automated. Now let your community know about this, encourage them to try the edge snap, and to tell their friends how cool it is to get even more testers.

Next steps

- Travis CI is very nice, but it can only build snaps for the amd64 architecture. If you need to deliver your snaps for users in other architectures like armhf and arm64, you can use the snapcraft build service.
- Learn some more advanced techniques on how to use your snap system looking for our others tutorials!
- Join the snapcraft.io community by subscribing to our mailing list.

Further readingshttps://snapcraft.io/docs/build-snaps/ci-integration#using-launchpad

- Travis CI documentation has a lot of information about adding tests and other things to your CI executions.
- Snapcraft CI integration documentation, with information about Travis and Launchpad integrations.
- Check how you can contact us and the broader community.