

École Polytechnique de Montréal
Département de génie informatique et de génie logiciel

Log 1000
Ingénierie logicielle

TP3
Processus de développement logiciel et d'assurance qualité d'un logiciel open source

Soumis par:
William Harvey (1851388) et
Mathieu Bélanger (1850591)
Section 4

20 mars 2017

[1] <https://code.wireshark.org/review/Documentation/intro-quick.html>

E1 Révision technique de code source [/ 13]

a. Expliquer à quoi sert le dépôt Gerrit exactement. [/1]

Le dépôt Gerrit est un outil servant à la révision de code par une communauté de développeur. Cela permet aux membres de l'équipe de développement d'accéder au code source développé et de donner leur approbation à toute modification. [1] Gerrit prend particulièrement son sens lors du développement de projets open-source, puisqu'il s'agit d'un outil de révision de code entièrement en ligne.

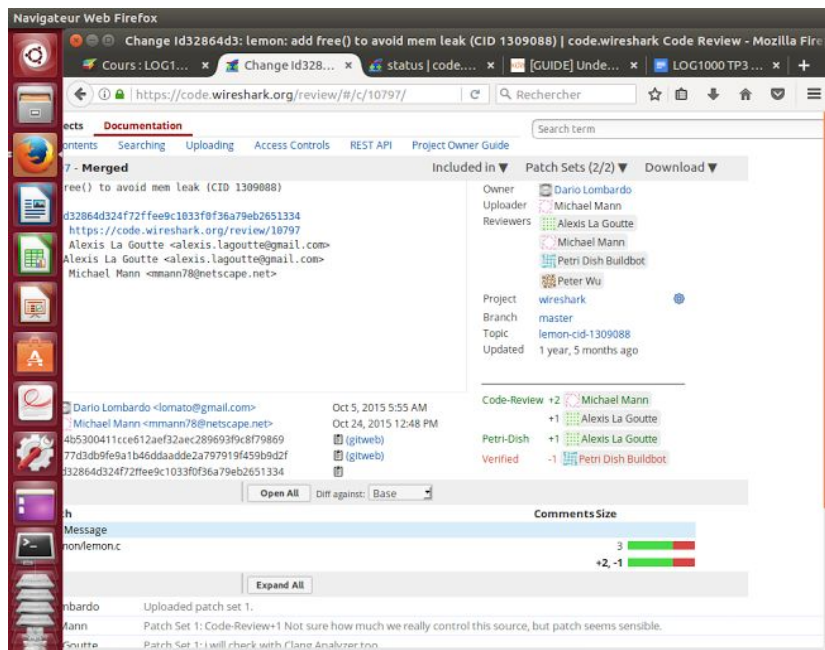
b. Que signifient "open", "merged" et "abandoned"? [/1]

Ces termes représentent le statut d'une proposition (commit) de modification au projet. "Open" signifie que la proposition est toujours sujette au jugement des membres, "Merged" représente une proposition qui a été approuvée et fusionnée à la branche principale du projet, puis, finalement, "abandoned" représente une proposition qui a été abandonnée par le reste de la communauté.

c. Qu'est-ce qu'un "patch"? [/1]

Un patch représente simplement toute proposition de modification apportée au dépôt Gerrit. Il s'agit d'une proposition dont le statut peut être divers.

d. Cherchez la révision technique Id32864d324f72ffee9c1033f0f36a79eb2651334.



e. Identifiez les fichiers qui ont été modifiés et décrivez brièvement ce que fait ce patch. [/1]

Dans la section "Files", on peut voir que les fichiers concernés par ce patch sont Commit Message et lemon.c. Puisqu'on sait que Commit Message n'est qu'une description des modifications effectuées, on peut déduire que c'est le fichier lemon.c qui a été retravaillé. D'ailleurs, le Commit Message nous explique que ce patch a ajouté la fonction free() dans lemon.c, afin de prévenir d'éventuelles fuites de mémoire.

f. Est-ce que le patch a été accepté? Comment est-ce que l'on peut voir ça? [/1]

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>

Le statut du patch est «Merged». On peut voir dans l'historique que la dernière entrée mentionnée est le “cherry-pick” du changement par Michael Mann (le réviseur du code).

g. Identifiez les développeurs qui critiquent ce patch. [/1]

Les développeurs qui ont offert une critique sont Alexis La Goutte, Michael Mann et Peter Wu.

h. Identifiez les valeurs de la section “Code-Review” à droite. Que signifient ces valeurs? [/1]

Les valeurs de la section Code-Review représentent l’approbation des paires quant à la révision technique de ce patch. On peut voir que Michael Mann approuve ce patch, puisqu’il donne la mention +2, tandis qu’Alexis La Goutte ne fait que donner une opinion positive. Au contraire, si un membre avait donné la mention -2 la modification aurait été refusée, car il doit y avoir consensus. [2] Finalement, -1 signifierait qu’un membre offre une opinion négative du patch.

i. Naviguez la partie “History” en bas, qu’est-ce qui s’est passé pour cette révision? Pourquoi cette partie est importante? [/1]

La partie “History” montre tout l’historique de la révision technique du patch. Tout d’abord, cette proposition de modification a tout d’abord été ajouté au dépôt gerrit par Dario Lombardo. Ensuite,

Michael Mann donne une opinion positive de la proposition après avoir fait une révision technique, puis Alexis La Goutte émet un commentaire. Peter Wu donne aussi une opinion positive, mais se désiste plus tard lorsqu’il remarque des problèmes de fuites de mémoire provenant de lemon.c. Alexis La Goutte donne ensuite une opinion positive. La proposition est “build”, puis Michael Mann donne son approbation, puisqu’il considère que la révision technique est terminée et que la proposition est bonne. Il ajoute donc la proposition à Wireshark.

j. Faites le même travail pour la révision technique lc34c9fd45740b834ce826d2ce083c615191bdef3 du développeur “Michael Oed” [/5]

1. Ce patch remplace le message lancé par Wireshark lorsqu’un des logiciels de la suite est en cours d’exécution et qu’une tentative de mise à jour est lancée. Au lieu de mentionner précisément le logiciel en question, le message indique seulement de fermer Wireshark pour procéder à la mise à jour.
2. Le patch a été classé comme abandonné par son auteur. On peut le voir dans la dernière entrée de l’historique : «Michael Oed Abandoned Mar 4, 2014»
3. Evan Huus et Pascal Quantin
4. Deux développeurs ont mentionné leur désaccord face aux modifications proposées. (-1 * 2 Dans code-review)
5. Après que Michael ait publié sa proposition de modification (patch set 1), Pascal Quantin répond par un commentaire mentionnant le problème dans le patch proposé. Finalement puisque le patch n’avançait pas, un autre reviewer Evan, indique à Michael que s’il ne compte pas continuer à améliorer son patch pour satisfaire au code-Review, il devrait «abandonner» le changement. Finalement Michael abandonne effectivement le patch.

E2 Le répertoire de gestion des versions

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>

- a. **Selon vous, est-ce que cet auteur contribue beaucoup au développement du logiciel (selon l'ensemble de ses "commits") en comparaison avec le committer de ce commit? Comment est-ce que vous avez trouvé la réponse?**

Dario Lombardo contribue un peu au projet Wireshark, puisqu'il a fait 78 commits en moins d'un an et demi. Toutefois, Michael Mann a contribué plus au projet que Dario Lombardo et ce, en une plus courte période de temps. En effet, on peut répertorier 100 commits effectués par Michael et ce, en seulement 5 mois. Pour trouver cette réponse, j'ai recherché ces participants dans la barre de recherche faisant partie de l'interface de Wireshark.git. Ainsi, il a été possible de retrouver l'ensemble des commits effectués par ces membres du projet et toutes les dates associées à ces commits.

- b. **Copier le message complet du commit**

lemon: add free() to avoid mem leak (CID 1309088)

Change-Id: Id32864d324f72ffee9c1033f0f36a79eb2651334

Reviewed-on: <https://code.wireshark.org/review/10797>

Reviewed-by: Alexis La Goutte <alexis.lagoutte@gmail.com>

Petri-Dish: Alexis La Goutte <alexis.lagoutte@gmail.com>

Reviewed-by: Michael Mann <mmann78@netscape.net>

- c. **Pourquoi cette section est utile?**

Cette section est utile, car elle permet de constater toutes les différences entre la version précédente du fichier et la version courante. Ainsi, on peut voir toutes les modifications effectuées dans ce commit.

- d. **Copiez le lignes de code qui ont été modifiées par l'auteur. (Cela inclut les lignes enlevées et ajoutées)**

```
- if( filesize>100000000 || filebuf==0 ){  
+ if( filesize>100000000 || filebuf==NULL ){  
  
+ free(filebuf);
```

- e. **Est-ce que ce commit ajouté une nouvelle fonctionnalité ou résout un bogue? Comment avez-vous déterminé cela?**

Le commit empêche une fuite de mémoire éventuelle, en ce sens il résout une erreur passée du code et n'ajoute pas de fonctionnalité. C'est donc qu'il résout un bogue.

E3 Gestion des bogues

- a. **Trouvez le bug #9887. Quel est le nom de la personne qui a trouvé le bogue? Est-ce que ce rapporteur est un être humain?**

C'est «Buildbot Builder» qui a initialement rapporté le bogue. *Reported: 2014-03-15 03:20 UTC by Buildbot Builder* . Comme son nom l'indique c'est un logiciel automatisé qui teste les

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>

composants du logiciel en effectuant des *builds*. Lorsque celui-ci rencontre un *crash* ou une erreur quelconque, il soumet un rapport de bogue.

b. Quel est le niveau d'importance de ce bogue? Comment est déterminé le niveau d'importance d'un bogue? Est-ce que ce bogue est critique à un tel point qu'il compromet le fonctionnement du logiciel? (dites pourquoi)

Le niveau d'importance du bogue est indiqué dans le résumé du bogue. *Importance: High Major*. Comme l'indique la notice, ce bogue est donc de priorité *Haute* et de sévérité *Majeure*. Le bot soumet probablement un niveau d'importance initial selon son rapport, mais il est possible pour les développeurs de voter par la suite pour faire évoluer le niveau. Malgré le fait qu'il soit *sévère*, les développeurs semblent indiquer qu'ils n'arrivent pas à recréer la situation menant au *crash* rapporté par le Bot on peut donc conclure que le bogue ne compromet pas (dans la pratique) le fonctionnement du logiciel.

c. Combien de personnes ont commenté ce problème? (à part la description originale)

5 individus ont commenté le problème, en plus des commentaires automatiques effectués par les revues de code associées au bogue.

d. Est-ce que la communauté a confirmé la validité du bogue? Comment?

Non, puisqu'après plusieurs tentatives Peter Wu signale : «Cannot reproduce the crash with tshark or Wireshark (Qt) [...]» il classe ensuite le bogue comme RESOLVED WORKSFORME conformément à la notice «All attempts at reproducing this bug were futile, and reading the code produces no clues as to why the described behavior would occur.» [3]

e. Est-ce que le bogue a été résolu? Comment est-ce que vous avez déterminé ça?

Comme mentionné en d) le bogue n'a pas réellement été validé donc il a été de facto déclaré comme RESOLVED WORKSFORME. Peter Wu signale dans le dernier log: «I'll assume it to be fixed unless you can observe otherwise.» C'est donc qu'un bug est considéré comme *fixed* s'il ne peut être reproduit.

E4 Intégration continue

a. Expliquez l'utilité de "Buildbot" dans le contexte d'un projet "open source"

Un buildbot est un outil qui permet d'automatiser les build et d'effectuer certains tests de façon automatique. Cela permet d'informer directement le développeur sur la capacité de son travail à s'intégrer au sein du projet. Ainsi, cela facilite le travail en équipe en permettant de vérifier si le travail effectué correspond à certains requis du projet. Le buildbot cadre donc parfaitement avec le concept d'intégration continue. [4]

b. Identifiez les différents "builders" disponibles pour ce projet cliquant sur "Builders" en haut de la page.

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>

<u>Clang Code Analysis</u>	#3955 failed check-abi	building 17 pending
<u>OSX 10.6 x64</u>	#13454 build successful	idle
<u>Ubuntu 16.04 x64</u>	#1617 build successful	idle
<u>Visual Studio Code Analysis</u>	#18346 build successful	building ETA in ~ 40 mins at 04:48
<u>Windows 8.1 x86</u>	#8824 build successful	idle
<u>Windows Server 2012 R2 x64</u>	#3076 failed compile 1	idle

a. Que signifient les couleurs vert rouge et orange?

Les couleurs indiquent l'état courant du *build* en question. Vert signifie que le *build* a été complété sans accroc. Jaune, que le build est en attente d'être effectué. Finalement rouge annonce qu'il s'est soldé par un échec

b. Que signifient les différentes colonnes

Chacune des colonnes représente un builder différent. Ces builders permettent de vérifier si le code s'intègre bien sur différents OS. Ainsi, on retrouve sur une colonne un ensemble de tentatives de build effectués par le buildbot.

c. Identifiez un build échoué ou avec des avertissements, faites un screenshot et expliquez la cause des problèmes en cliquant sur le lien pour voir des messages du build.

<https://buildbot.wireshark.org/wireshark-master/builders/Windows%20Server%202012%20R2%20x64/builds/3076>

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>

Home - Waterfall Grid T-Grid Console Builders Recent Builds Buildslaves Changesources - ISON API - About

Builder Windows Server 2012 R2 x64 Build #3076

Results:

Failed compile 1

SourceStamp:

Project: wireshark
 Repository: ssh://wireshark-buildbot@code.wireshark.org:29418/wireshark
 Branch: refs/heads/master
 Revision: 9e82cffeea10fce62cdfc49cec3b9e3794ac04bf
 Got Revision: 9e82cffeea10fce62cdfc49cec3b9e3794ac04bf
 Changes: 1 change

BuildSlave:

windows-2012r2-x64

Reason:

The SingleBranchScheduler scheduler named 'Gerrit' triggered this build

Steps and Logfiles:

1. **git update** (20 secs)
 1. **git update**
2. **make version.pl set release information** (2 secs)
 1. **git update**
3. **shutil created CMake build directory** (0 secs)
 1. **git update**
4. **compile ran CMake** (6 mins, 25 secs)
 1. **git update**
5. **compile_1 compiled with MSBuild failed** (32 mins, 23 secs)
 1. **git update**
 2. **warnings (30)**
6. **compile_2**
 1. **- no logs -**
7. **compile_3**

Build Properties:

Name	Value	Source
branch	refs/heads/master	Build
builddir	C:\buildbot\wireshark\wireshark-master-64\windows-2012r2-x64	slave
buildname	Windows Server 2012 R2 x64	Builder
buildnumber	3076	Build
codebase		Build
commit-description	v2.3.0rc0-2700-g9e82cf	Git
event.eventCreatedOn	1489374392	Change
event.refUpdate.newRev	9e82cffeea10fce62cdfc49cec3b9e3794ac04bf	Change
event.refUpdate.oldRev	c70927cde11a251a6c14bc3bb391cd18cacba6db	Change
event.refUpdate.project	wireshark	Change
event.refUpdate.refName	refs/heads/master	Change
event.submitter.email	mmann78@netscape.net	Change
event.submitter.name	Michael Mann	Change
event.submitter.username	mmann	Change
event.type	ref-updated	Change
got_revision	9e82cffeea10fce62cdfc49cec3b9e3794ac04bf	Git
project	wireshark	Build
repository	ssh://wireshark-buildbot@code.wireshark.org:29418/wireshark	Build
revision	9e82cffeea10fce62cdfc49cec3b9e3794ac04bf	Build
scheduler	Gerrit	Scheduler
slavename	windows-2012r2-x64	BuildSlave
warnings-count	30	WarningCountingShellCommand
workdir	C:\buildbot\wireshark\wireshark-master-64\windows-2012r2-x64	slave (deprecated)

Forced Build Properties:

Name	Label	Value

Responsible Users:

1. Michael Mann

Timing:

Start: Mon Mar 13 03:11:34 2017

L'erreur dans l'exécution de ce build provient du builder MSBuild qui n'a pu exécuter un build sur Windows Server 2012

d. Quel est le numéro de la révision technique de ce build échoué, et qui est le développeur?

9e82cffeea10fce62cdfc49cec3b9e3794ac04bf, effectuée par Michael Mann.

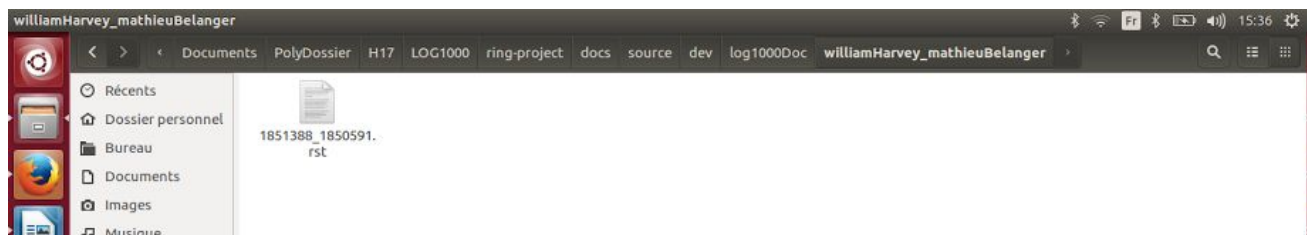
E5 Contribuer au projet Ring: [/ 25]

Étape 1 : Préparer l'environnement :

```
william@SlavickKalinkaMaster:~/Documents/PolyDossier/H17/LOG1000$ git clone https://billytherat@gerrit-ring.savoirfairelinux.com/ring-project
Clonage dans 'ring-project'...
remote: Total 931 (delta 0), reused 931 (delta 0)
Réception d'objets: 100% (931/931), 246.43 KiB | 0 bytes/s, fait.
Résolution des deltas: 100% (481/481), fait.
Vérification de la connectivité... fait.
william@SlavickKalinkaMaster:~/Documents/PolyDossier/H17/LOG1000$
```

Etape 2 : Faire un «Hello Word» de la documentation de Ring:

a.



c.

[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status


[4] <http://buildbot.net/>


```

ch any documents
/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/index.rst:50: WARNING: toctree glob pattern u'dev/lrc/*' didn't match
any documents
/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/index.rst:57: WARNING: toctree glob pattern u'dev/gnome-client/*' didn
't match any documents
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [100%] index
generating indices... genindex
writing additional pages... search
copying static files... WARNING: html_static_path entry u'/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/_static' doe
s not exist
done
copying extra files... done
dumping search index in English (code: en) ... done
dumping object inventory... done
build succeeded, 4 warnings.
env/bin/sphinx-build -b texinfo docs/source docs/build/texinfo
Running Sphinx v1.4.1
loading pickled environment... done
building [mo]: targets for 0 po files that are out of date
building [texinfo]: all documents
updating environment: 0 added, 1 changed, 0 removed
reading sources... [100%] index
/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/index.rst:43: WARNING: toctree glob pattern u'dev/daemon/*' didn't mat
ch any documents
/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/index.rst:50: WARNING: toctree glob pattern u'dev/lrc/*' didn't match
any documents
/home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/source/index.rst:57: WARNING: toctree glob pattern u'dev/gnome-client/*' didn
't match any documents
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
processing Ring.texi... index dev/log1000Doc/williamHarvey_mathieuBelanger/1851388_1850591 intro/getting_started users/downloading_and_installi
ng users/connecting_to_ldap users/setting_up_turn_stun dev/contributing dev/compiling_and_installing/index dev/compiling_and_installing/ring-pr
ject dev/compiling_and_installing/daemon dev/compiling_and_installing/lrc dev/compiling_and_installing/gnome_client dev/releasing
resolving references...
writing... done
copying Texinfo support files... /home/william/Documents/PolyDossier/H17/LOG1000/ring-project/docs/build/texinfo/Makefile done
build succeeded, 3 warnings.
william@slavickkalinkaMaster:~/Documents/PolyDossier/H17/LOG1000/ring-project$

```

d.



Ring

1.0

HELLO WORD

Documentation des étudiants "1851388" et "1850591"

INTRODUCTION

Getting Started

USER DOCUMENTATION

Downloading and installing

Connecting Ring contacts to LDAP

Setting up a TURN or STUN server

DEVELOPERS DOCUMENTATION

Contributing

Compiling and installing

Ring release process

RING DAEMON

LIBRINGCLIENT

GNOME CLIENT

Docs » Welcome to Ring's documentation! [View page source](#)

Welcome to Ring's documentation!

As a new user, [getting started](#) is a good place to start.

As a developer looking to get started with a contribution, see [contributing](#) and [how to build](#).

Hello Word

- [Documentation des étudiants "1851388" et "1850591"](#)

Introduction

- [Getting Started](#)

User Documentation

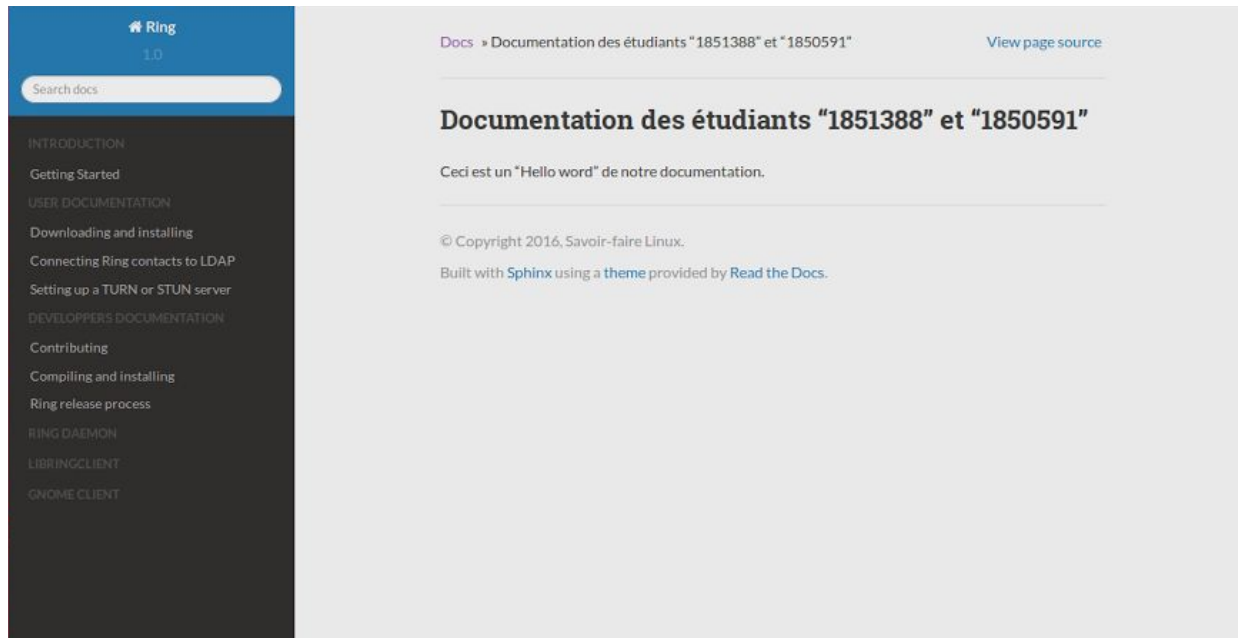
- [Downloading and installing](#)
- [Connecting Ring contacts to LDAP](#)
- [Setting up a TURN or STUN server](#)

Developers Documentation

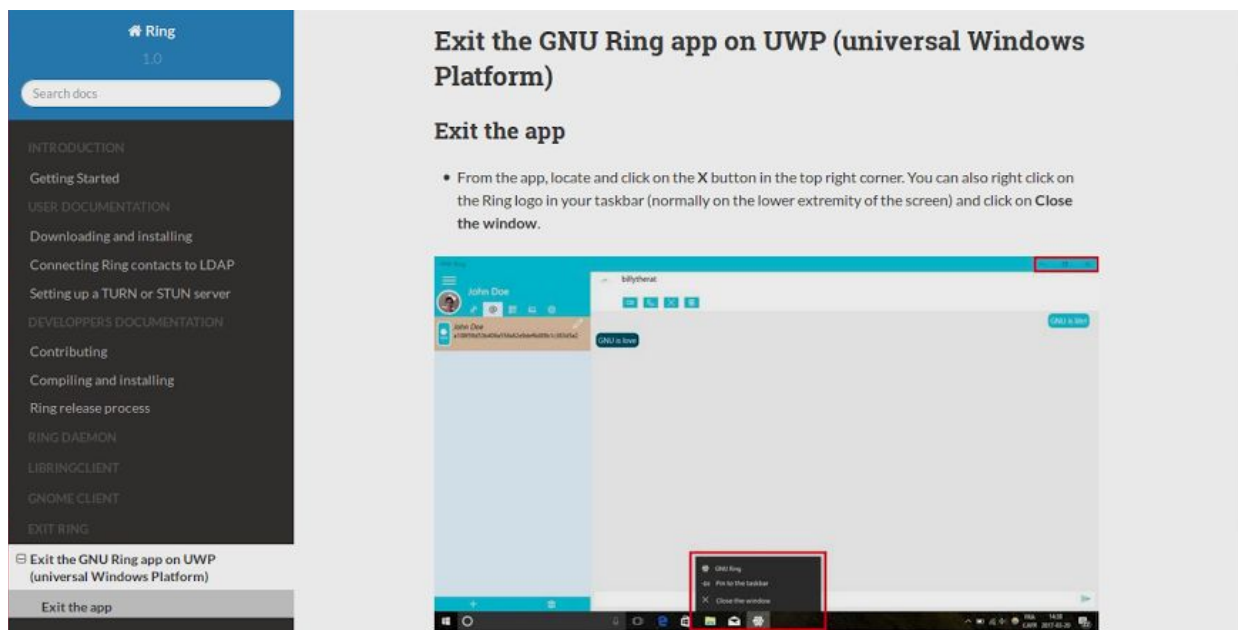
- [Contributing](#)

e.

- [2] <https://review.openstack.org/Documentation/intro-quick.html>
- [3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status
- [4] <http://buildbot.net/>



Étape 3 : Documenter Ring :



[2] <https://review.openstack.org/Documentation/intro-quick.html>

[3] https://bugs.wireshark.org/bugzilla/page.cgi?id=fields.html#bug_status

[4] <http://buildbot.net/>