Floday

Documentation française

Date de compilation 8 décembre 2017
Version logicielle de référence 1.1.0

Kevin Hagner <jsaipakoimetr@spyzone.fr>

Floday est un gestionnaire d'amorçage qui utilise actuellement *LXC*. Cette solution a pour objectif de rendre plus aisée et robuste l'administration d'ordinateurs sur lesquels beaucoup de services différents peuvent être amenés à fonctionner, sans pour autant avoir de contrôle assidu sur l'infrastructure. Le cas typique d'utilisation est celui d'un amateur propriétaire de quelques machines pour l'autohébergement de certains services, comme un serveur email, web, *XMPP* ou *Mumble*, et qui malgré le très faible temps qu'il consacre chaque mois à ces activités, peut avoir une bonne confiance dans la résilience de son infrastructure.

Table des matières

1 Ressources . . . 2
1.1 Ce document . . . 2
1.2 La documentation du code . . . 2
1.3 Le bug tracker . . . 2
1.4 Le dépôt *Git* . . . 2
2 Utilisation de *Floday* . . . 3
2.1 Licence de distribution . . . 3
2.2 Comprendre les principes de base . . . 3
2.2.1 Une mise en situation . . . 3
2.2.2 À quoi *Floday* répond . . . 4
2.3 Les éléments centraux . . . 4
2.3.1 Le fichier de configuration . . . 4
2.3.2 Le *runfile* . . . 5
2.3.3 La définition de conteneurs . . . 6
2.3.4 Écrire les scripts d'exécution . . . 8
2.3.5 Déboguer un script . . . 9
2.4 L'exécution . . . 9
2.4.1 Les différentes étapes . . . 9
2.4.2 La gestion des logs . . . 12
2.5 Installation de *Floday* . . . 13
2.5.1 Quelle version choisir ? . . . 13
2.5.2 Comment l'installer ? . . . 14
2.6 Écrire son jeu de conteneurs . . . 14
3 Comment contribuer ? . . . 15
3.1 Le bug tracker . . . 15
3.1.1 Les types de tickets . . . 15
3.1.2 La rédaction d'un ticket . . . 15
3.1.3 Le workflow du bug tracker . . . 16
3.2 Démarrer une instance de développement . . . 16
3.3 Apporter des modifications au code source . . . 17
3.3.1 Convention de code . . . 17
3.3.2 Workflow *Git* . . . 18
3.4 Participer à la documentation . . . 21
3.4.1 Comment compiler soi-même la documentation . . . 21
3.4.2 Quoi documenter ? . . . 21
3.4.3 Traductions de la documentation . . . 21
Glossaire . . . 22
A The GNU General Public Licence . . . 24

Ressources

Avant toute chose, il est de bon ton de prendre un petit moment pour lister les différents éléments qui gravitent au tour de *Floday*. Ils peuvent se montrer appréciables pour prolonger la compréhension que vous voulez avoir de l'outil, ou pour participer de façon plus active au projet.

1.1 Ce document

1ère apparition : v1.0.0

Vous êtes en train de lire la documentation du logiciel *Floday*. Il peut vous sembler étrange que celle-ci soit délivrée sous la forme d'un fichier *PDF*, plutôt que d'être présenté sous une forme plus interactive permise par les technologies du Web. Ce choix me parait néanmoins pertinent, car il impose la création d'une ressource clairement délimitée censée contenir l'intégralité des informations existantes. Dans la même philosophie, une version de cette documentation existera pour chaque version de *Floday* délivrée. Il n'y a donc pas de risque à ce que celle-ci commence à diverger avec le logiciel que vous êtes effectivement en train d'utiliser. Enfin, il me semble plus pratique de n'avoir qu'un seul document à conserver plutôt que de devoir aller cliquer partout le jour ou une information viendra à nous manquer. Notez aussi que par manque de temps, cette documentation n'existe aujourd'hui qu'en français.

1.2 La documentation du code

1ère apparition : v1.0.0

Le code source du projet en lui-même possède une documentation utile aux développeurs. Celle-ci est accessible soit en le lisant directement, ou alors à travers l'outil *perldoc*, car *Floday* est intégralement écrit en *Perl 5*.

1.3 Le bug tracker

1ère apparition : v1.0.0

Un bug tracker est disponible à l'adresse `https://dev.spyzone.fr/floday/query`. Comme son nom l'indique, il liste l'intégralité des demandes d'évolutions et des rapports de bogues constatés sur le projet. C'est aussi un point d'entrée privilégié si cette présente documentation ne répond pas correctement à l'une de vos questions. En effet, ce document est considéré comme faisant partie intégrante du projet.

Avant de poster un nouveau ticket, il vous est néanmoins demandé de vérifier si aucun doublon ne sera introduit par votre action. Limiter le bruit et les redondances est un élément simple et qui permet de faire gagner du temps précieux aux contributeurs qui l'allouerons à meilleur escient ailleurs ☺.

Ce document présente une partie entière réservée à l'utilisation du bug tracker. La lire vous est également vivement conseillé si vous comptez participer.

1.4 Le dépôt *Git*

1ère apparition : v1.0.0

Certes, le projet est censé être libre, mais le dépôt *Git* n'est pour l'instant pas totalement public… Les raisons derrière cette limitation sont en fait techniques : actuellement, *Gitolite* est utilisé pour faire la gestion des permissions sur le dépôt. Or, ce dernier nécessite une connexion via *SSH*, qui, elle-même impose au serveur de posséder votre clef publique. Sans cette clef, il est donc impossible de récupérer la moindre source. Bien sûr, je vous donnerais les accès en lecture si vous m'envoyez la vôtre par email. Ce problème devrait être résolu un jour, car le projet devrait migrer sous *Gitlab*.

Utilisation de *Floday*

Nous verrons dans cette partie tout ce qui est relatif à l'utilisation de *Floday* :

- **Dans un premier temps, on fera un point juridique rapide.**
- **Puis, les principes de bases seront exposés afin de voir avec précision les tenants et aboutissants du logiciel.**
- **Une fois au clair sur ses objectifs, les différents éléments techniques seront présentés afin de permettre la prise en main.**
- **Il ne nous restera plus qu'à montrer une vision d'ensemble pour comprendre les processus de fonctionnement.**
- **Finalement, il sera peut-être intéressant de l'installer.**

2.1 Licence de distribution

1ère apparition : v1.0.0

Le code source de *Floday*, ainsi que cette documentation sont délivrés sous la licence `GNU GPL v3`. Un exemplaire de celle-ci est disponible en anglais en annexe, ou sur la page : `https://www.gnu.org/licenses/gpl.html`

De façon résumée, cela signifie qu'une fois en possession du logiciel, vous avez automatiquement accès à une copie du code source que vous êtes libre de modifier et de redistribuer. Vous avez donc droit d'employer *Floday* et cette documentation comme bon vous semble, vous pouvez également revendre quelque chose qui l'utilise. Les seules restrictions résident dans l'obligation que vous avez à partager le code et vos modifications éventuelles à vos utilisateurs, qui eux aussi disposeront des mêmes libertés et contraintes.

2.2 Comprendre les principes de base

2.2.1 Une mise en situation

1ère apparition : v1.0.0

Prenons la problématique suivante : Kevin (le nom du personnage n'a pas été choisi par hasard), un passionné d'informatique s'adonne souvent aux tests de beaucoup de logiciels. La plupart auront été déployés avec une méthode proche de la *RACHE*[1], et ce dans des temps reculés dont toutes traces du mode opératoire auront étés oubliées le jour malheureux ou l'auteur réalisera qu'ils ne fonctionnent plus comme il s'y attendait.

De plus, cette perte de repères est amplifiée par la grosse hétérogénéité des applications : certaines ont leurs fichiers de configuration dans le répertoire `/etc` et leurs données dans `/home/$USER`, d'autres seront plutôt `/home/$USER/<service>/.config` pour la configuration et `/var/lib/` pour les données, etc. Sans oublier les scripts aux fonctionnements variés qui peuvent être présents n'importe où (des règles *iptables* dans `/etc/init.d/80-custom`, des programmes de backup aux quatre coins de l'arborescence, etc.).

Quant au matériel, il est en général du même acabit : un Raspberry Pi dans une boite en carton[2], un dédié moisi qui doit tourner dans une cave d'un pays indéterminé ou alors un bout de cyberespace qu'on squatte sans réelle garantie qu'on y aura encore accès dans cinq minutes. Bref... L'infrastructure physique n'étant pas pérenne, on sera un jour ou l'autre amené à tout redéployer ailleurs ce qui, on s'en doute, ne sera probablement pas effectué de façon optimale à la vue des remarques précédentes.

Toutes ces difficultés peuvent être acceptables (bien que peu réjouissantes) si l'on conserve le postulat de base, celui qui dit que le seul but de cette démarche est de tester. En réalité, on se retrouve vite à employer ces instances en production. Bien sûr, l'échelle reste en général très petite quant aux personnes impactées en cas d'éventuelle avarie, car elles se limitent souvent à l'utilisateur voir son entourage proche, mais elle ne rend pas cette perte négligeable pour autant. On peut parler ici de serveurs d'emails ou de clavardage, des forums, des blogs, etc.

Pouvons-nous réellement prendre le risque de perdre tout ces « petits » services ? Non. Certes, ça ne serait en général pas non plus la mort, mais ça ferait tout de même bien chier. Du coup, on rajoute encore plus de complexité au désordre ambiant pour essayer de garantir une certaine résilience : mise en place d'un système de backup, du monitoring, une gestion avancée de la sécurité (utilisation d'*AppArmor* par exemple), etc. Puis on fait des incantations aux dieux que l'on vénère pour espérer ne jamais avoir à affronter ce genre d'accident, car souvent nous n'avons pas nous-mêmes confiance en notre propre infrastructure (il serait prétentieux de parler de stratégie de

1. La *RACHE* est une méthodologie de génie logiciel assez controversée. Plus d'information sur ce site : `http://www.la-rache.com/`
2. C'est une autopromotion concernant un de mes anciens serveurs de backup : `http://blog.spyzone.fr/2013/11/utiliser-un-raspberry-pi-comme-serveur-de-backup/`

restauration), tellement celle-ci est bancale. C'est triste, n'est-ce pas ? En tout cas, c'est contre ça que *Floday* essaye de lutter.

2.2.2 À quoi *Floday* répond

1ère apparition : v1.0.0

De façon un peu moins romancée que lors du précédent paragraphe, on peut résumer *Floday* aux points suivants :

Regrouper la configuration logicielle Le système a été pensé pour contenir toute la configuration et tous les scripts d'installation et de fonctionnement qu'un logiciel requière, cantonné à un même endroit. De cette façon, indépendamment de la complexité de celui-ci, la zone de recherche dans laquelle investiguer ou agir lors d'évolutions sur le service en question reste très délimitée et à l'écart du bruit introduit par l'extérieur. Il s'agit ici de la notion de conteneurs.

Clarifier par confinement Chaque application doit pouvoir fonctionner indépendamment des autres, dans une portée bien définie. La communication interapplication n'est prévue pour se faire que via sockets fichiers ou réseaux explicites. C'est pour cela que la conteneurisation via *LXC* semble être un bon choix.

Uniformiser l'infrastructure Un mécanisme d'héritage est également présent au niveau des conteneurs, permettant de factoriser certaines tâches et du coup, de favoriser l'uniformité. Ainsi, nous pouvons imaginer le fait que plusieurs conteneurs aient besoin du réseau, ou d'un accès *ssh*. Ces tâches pourront donc être héritées d'un conteneur parent rendant leur fonctionnement identique, ou au moins similaire.

Faciliter la configuration Trouver le juste milieu entre une uniformité excessive et un désordre entropique est souvent délicat. C'est avec cette remarque en contexte que le système de configuration, complexe au premier abord, mais semblant répondre aux nécessités d'adaptations, a été pensé. En effet, chaque conteneur est défini en fonction de nombreux attributs. Tous peuvent se voir réécrits dans des sous-conteneurs. Les paramètres applicatifs peuvent même l'être au niveau du runfile.

Clarifier l'interaction logicielle Chaque conteneur peut être gestionnaire et sous-conteneur. L'interaction entre eux est forte, c'est-à-dire qu'un sous-conteneur n'est pas prévu pour pouvoir fonctionner sans son gestionnaire. Il est également possible d'avoir une contrainte sur plusieurs niveaux : c'est-à-dire, de permettre au sous-conteneur d'être lui aussi gestionnaire. Nous pouvons prendre l'exemple d'un conteneur *Wordpress*, chargé de déployer le *CMS* éponyme, dont le gestionnaire serait *Web*, un conteneur ayant rôle de proxy *HTTP*. Cette hiérarchisation est explicite depuis le runfile et depuis le jeu de conteneurs.

Faciliter la gestion multihôte Un seul runfile devrait être utilisé par infrastructure, même si celle-ci s'appuie sur plusieurs hôtes réels. *Floday* permet de définir celui que l'on s'apprête à déployer. Encore une fois, le fait d'avoir toute la configuration dans un seul fichier facilite la prise de conscience de l'intégralité des acteurs. Imaginons le cas d'un serveur de backup : il aura directement accès aux autres application à backuper.

Permettre une bonne résilience Le déploiement avec *Floday* n'est pas universel, dans le sens ou il pourrait se faire partout, indépendamment du socle technologique ou physique sur lequel nous nous appuyons. Par contre, il se veut entièrement automatisé, sous réserve que le nouvel hôte respecte des postulats de base propres à chaque jeu de conteneurs. De cette façon, si une panne physique survient sur une machine, les services pourront facilement être redéployés ailleurs et remis en route.

2.3 Les éléments centraux

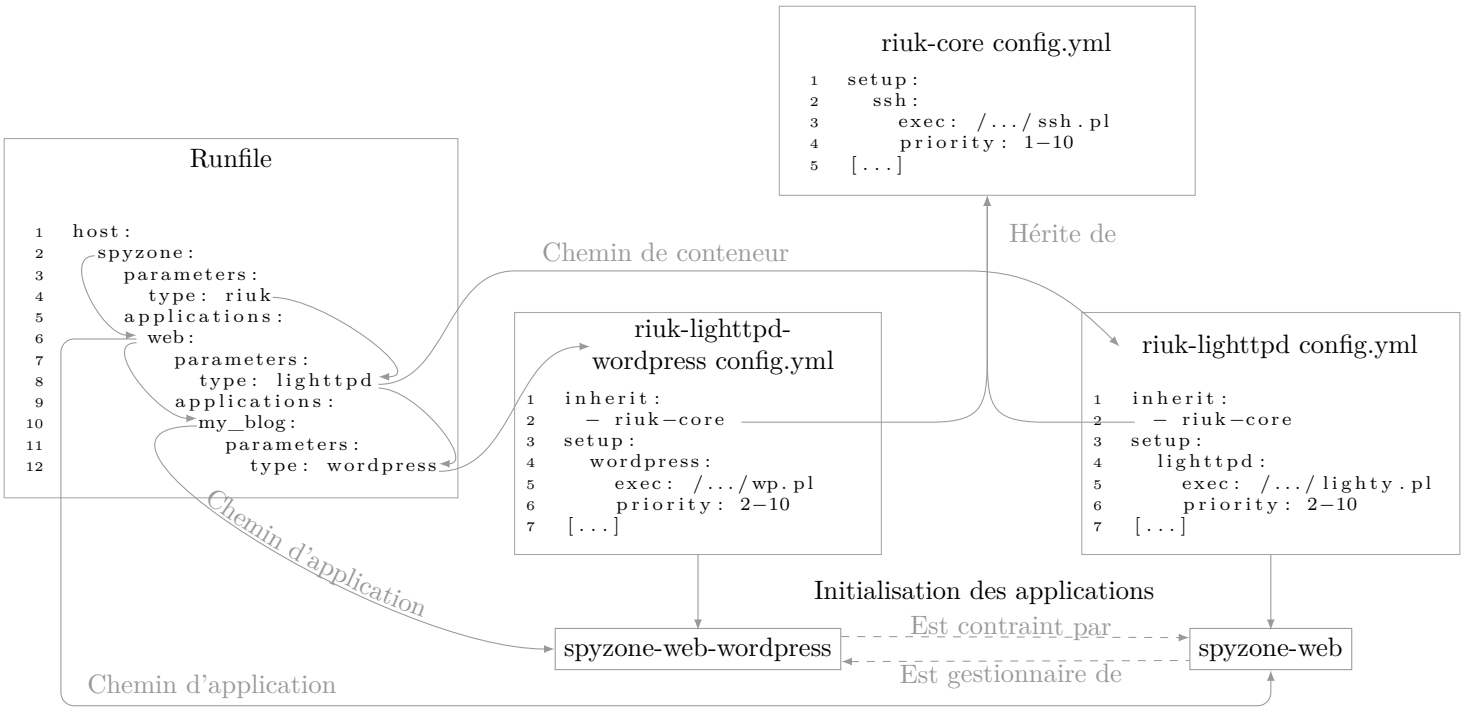
2.3.1 Le fichier de configuration

1ère apparition : v1.0.0

Un petit fichier de configuration existe à l'emplacement `/etc/floday/config.cfg`. Celui-ci est au format *INI*[3]. Il s'agit du seul fichier dont l'emplacement soit imposé. Voici un descriptif de ce

3. Courte définition du format *INI* sur le site de *Wikipedia* : https://fr.wikipedia.org/wiki/Fichier_INI

FIGURE 1: Illustrations des relations inter-conteneurs et des notions de chemins.



qu'il contient :

Séction	Paramètre	Valeur par défaut	Description
containers	path	/etc/floday/containers	Emplacement du jeu de conteneurs.
floday	runfile	/etc/floday/runfile.yml	Emplacement du runfile.
logging	metadata_folder	/tmp/floday/logging	Dossier utilisé de façon interne par *Floday* pour gérer l'indentation des messages de logs.

Pour le moment, il est aussi possible d'avoir des valeurs de configuration utilisées par le jeu de conteneurs, voir par l'initialisation. Je ne suis pas certain qu'il s'agisse de la meilleure façon de faire, du coup il faut s'attendre à ce que cela change peut-être un jour.

Voici les paramètres de configuration existants actuellement pour *flodayalpine*, l'initialisation livrée par défaut avec *floday* :

Séction	Paramètre	Valeur par défaut	Description
LXC	cache_folder	/tmp/floday/lxc-flodayalpine	Dossier de cache utilisé pour stocker les logiciels déjà téléchargés.
LXC	id_groups	1000	Nombre de groupes d'uid et de gid disponibles pour l'application en cours d'instanciation. Un groupe est choisi au hasard parmi ceux encore libres.
LXC	id_range	100000	Nombre de uid et de gid présents au sein d'un même groupe.
LXC	repo	dl-4.alpinelinux.org	Dépôt sur lequel récupérer les packages à installer.

2.3.2 Le *runfile*

1ère apparition : v1.0.0

Il s'agit du cœur de l'application : le fichier qui indique avec précision ce qui doit être déployé. Le code 1 est un modèle que l'on utilisera comme exemple.

On constate tout d'abord que ce fichier est en *YAML*. On y voit ensuite la définition de deux hôtes : *websites* et *backup*. Dans le premier, trois applications y sont définies :*web*, qui est ges-

Listing 1: Exemple de runfile.yml

```yaml
---
hosts:
  websites:
    parameters:
      type:           riuk
      external_ipv4: 192.168.15.151
    applications:
      web_application:
        parameters:
          ipv4:     10.0.3.5
          gateway: 10.0.3.1
          type:     web
        applications:
          my_blog:
            parameters:
              ipv4:       10.0.3.6
              gateway:    10.0.3.1
              type:       wordpress
              data:       /var/www/my_blog
              hostname:   blog.spyzone.fr
          mum_blog:
            parameters:
              ipv4:       10.0.3.7
              gateway:    10.0.3.1
              type:       pluxml
              data:       /var/www/mum_blog
              hostname:   mum.spyzone.fr
  backup:
    parameters:
      type: jaxe
      external_ipv4: 192.168.15.141
    applications:
      backup_application:
        parameters:
          type: backup_web
```

tionnaire des deux autres : *my_blog* et *mum_blog*. Cela signifie que *web* peut se baser sur ses sous-conteneurs pour se configurer correctement. La figure 1 essaye de représenter visuellement ces notions.

L'instanciation elle aussi est hiérarchique : dans un premier temps, l'hôte sera déployé, puis viendra le tour de *web_application* et enfin celui de *my_blog* et de *mum_blog*. Notez d'ailleurs que l'ordre de ces deux derniers est aléatoire. Ce n'est donc pas parce que *my_blog* a été écrit avant, qu'il sera forcément déployé en premier.

Nous pouvons voir que pour chaque hôte ou application deux clefs principales y sont définies :

applications indique toutes les sous-applications qui seront gérées par celle en cours de définition.

parameters surcharge la valeur des paramètres applicatifs créés au niveau du conteneur. Le seul paramètre obligatoire est *name* car il permet d'identifier le nom de l'application et de former son chemin d'application. Tous les autres dépendent du conteneur en cours de définition.

2.3.3 La définition de conteneurs

1ère apparition : v1.0.0

Dans le paragraphe précédent, on a vu comment indiquer les applications que l'on voudrait déployer sur un hôte. Il nous reste à présent à définir en quoi elles consistent concrètement.

Pour cela, la première notion à introduire est celle de chemin de conteneur. Comme le glossaire nous l'apprend, il ne s'agit ni plus ni moins qu'une agrégation des types de l'imbrication courante. Ce chemin permet de trouver facilement le fichier `config.yml` du conteneur en question.

En prenant à nouveau l'exemple du code 1, on y trouve un conteneur de chemin *riuk-web-wordpress*, un autre de chemin *riuk-web-pluxml*, le gestionnaire *riuk-web*, et enfin, un hôte *riuk*. Avec le jeu de conteneurs à l'emplacement par défaut, on peut trouver la définition de notre premier conteneur dans le fichier `/etc/floday/containers/jaxe/children/web/children/wordpress/config.yml`. Le code 2 sera utilisé pour illustrer ce qui est attendu de ce fichier.

Listing 2: config.yml

```yaml
inherit:
  -jaxe-core
setups:
  deploy:
    avoidable: true
    exec:      riuk/children/web/children/wordpress/setups/deploy.pl
    priority:  10
  import:
    avoidable: false
    exec:      riuk/children/web/children/wordpress/setups/import.pl
    priority:  30
parameters:
  public:
    mandatory: false
    pattern: ^(true|false)$
  hostname:
    mandatory: true
    pattern:   ^[\w.-]$
  template:
    mandatory: true
    value:     flodayalpine -- version 3.4
hooks:
  lxc_deploy_before:
    open_firewall:
      exec:      riuk/children/[...]/wordpress/hooks/lxc_deploy_before/of.pl
      priority: 10
  lxc_deploy_after:
    close_firewall:
      exec:      riuk/children/[...]/wordpress/hooks/lxc_deploy_after/cf.pl
      priority: 10
  lxc_destroy_before:
    clear_filesystem:
      exec:      riuk/children/[...]/wordpress/hooks/lxc_destroy_before/cf.pl
      priority: 10
  lxc_destroy_after:
    update_fstab:
      exec:      riuk/children/[...]/wordpress/hooks/lxc_destroy_after/uf.pl
      priority: 10
avoidance:
  new_data:
    exec: riuk/children/[...]/wordpress/avoidance/new_data.pl
    priority: 10
```

Le nœud *hooks* Les hooks nous permettent de greffer des scripts durant le processus d'*initialisation*. Ils ne doivent être utilisés que dans les cas ou des actions doivent être faites durant cette phase. Elles devraient rester le plus possible cantonnées au nœud *setups* et *end_setups* car il devient vite complexe de comprendre correctement l'étape de déploiement d'un conteneur si beaucoup de hooks y sont greffés. Notez qu'ils sont aussi implémentés au niveau de la méthode d'initialisation, et donc dépendent de celle-ci.

Les attributs qu'ils comportent sont les mêmes que ceux des nœuds *setups* et *end_setups*. Voici la liste des hooks existant pour l'initialisation via le template *LXC flodayalpine*, utilisé par défaut :

lxc_deploy_before Les scripts seront exécutés avant `lxc-deploy`.

lxc_deploy_after Les scripts seront exécutés après `lxc-deploy`.

lxc_destroy_before Les scripts seront exécutés avant `lxc-destroy`.

lxc_destroy_after Les scripts seront exécutés après `lxc-destroy`.

Le nœud *inherit* Cette liste comprend les chemins de conteneurs de l'ensemble des parents du conteneur courant. Attention, l'ordre dans lequel ils sont écrits est ignoré. Deux parents ne doivent donc pas réécrire un même attribut !

Dernière MàJ : v1.1.0

Le nœud *parameters* Nous y définissons les différents paramètres accessibles dans les scripts d'installation. Leurs valeurs peuvent être surchargées au niveau du runfile. Chaque paramètre applicatif peut avoir ces attributs :

mandatory Peut valoir *true* ou *false*[4]. Si l'attribut est obligatoire, mais non défini lors du déploiement, une erreur sera émise par *Floday*.

pattern Cet attribut peut contenir une expression rationnelle *PCRE* à laquelle la valeur sera soumise. Si le test échoue, le déploiement sera là aussi annulé.

value Définit une valeur par défaut. Si celui-ci ne se retrouve pas surchargé au niveau du runfile, c'est cette valeur qui sera utilisée.

avoidable Indique si le script peut-être contourné, ou si son execution se révèle indispensable dans tous les cas. Comme les autres pseudo-booléen, la valeur de ce champ peut être de *true* ou *false*. Cette notion de contournement est expliquée en détail dans un autre paragraphe.

Dernière MàJ : v1.0.0

Le nœud *setups* Une fois l'initialisation complète, les scripts présents dans cette partie sont exécutés les uns après les autres pour finir le déploiement. Chaque nœud peut avoir ces attributs :

exec Chemin du script à exécuter. Celui-ci doit être exécutable par l'utilisateur effectuant le déploiement. Il peut se situer n'importe où du moment qu'il est accessible. Une convention veut cependant qu'il soit présent dans le dossier *setups* au même emplacement que l'est le fichier *config.yml* en cours d'écriture.

priority Gère l'ordre d'exécution. Attention à bien prendre en compte les éventuels conteneurs parents ! Les scripts sont exécutés par ordre croissant.

Le nœud *end_setup* . Ils sont similaires en tout point avec les scripts *setup*. La différence est qu'ils sont exécutés une fois l'ensemble des sous-applications de déployées et non avant. Les attributs sont les mêmes que pour le nœud *setup* et *hooks*.

1ère apparition : v1.1.0

Le nœud *avoidance* Il contient un ensemble de scripts a exécuter pour savoir si l'application en cours de déploiement doit être marquée comme pouvant être évitée (évitement). Cette notion permet d'accélérer le déploiment de l'hôte en ne traitant effectivement que les applications qui ont besoin de l'être. Chaque script présent dans ce nœud devra retourner quelque chose. Si la valeur de retour est différente de *0*, le test d'évitement sera considéré comme échoué, rendant l'application non-évitable. Il suffit d'un unique échec pour que l'application soit redéployée, même si d'autres scripts d'évitement sont considérés comme réussi.

Malgré tout, certains scripts (principalement de *setups* et de *end_setups*) peuvent nécessiter une exécution à tous les coups (par exemple la reconfiguration d'*iptables* qui serait réinitialisé à chaque déploiement). Il est donc possible de forcer l'exécution d'un script en renseignant le paramètre *avoidable* à false au niveau du script en question, dans la définition du conteneur. À noté que ça sera le comportement par défaut pour tous scripts n'ayant pas le paramètre de renseigné.

Si ce nœud n'est pas présent, l'application sera toujours considérée comme non-évitable. Chaque nœud peut avoir les attributs *exec* et *priority* qui ont les mêmes rôles que ceux présents au sein du nœud *setups*.

2.3.4 Écrire les scripts d'exécution

1ère apparition : v1.0.0

Le code 3 est un exemple de script permettant de configurer *lighttpd* de façon à ce qu'il puisse communiquer avec l'extérieur et qu'il puisse faire le proxy entre ses différentes sous-application.

Tout d'abord, on constate que ce script est écrit en *Perl 5*. Ce n'est absolument pas une obligation, la seule contrainte étant que ce fichier soit exécutable. Par contre, il s'agit de l'unique langage possédant le module *Floday : :Setup*, bien pratique pour travailler sur l'application en cours d'instanciation. Il vous est conseillé de lire sa page de documentation pour plus de détails à son sujet : `perldoc Floday::Setup`.

La première partie intéressante débute à la ligne 15, en montrant comment faire pour manipuler l'application depuis le script. On voit que l'on passe par la variable `$APP` automatiquement déclarée par *Floday : :Setup*. Dès la ligne 16, on l'utilise pour rapatrier un objet *Linux : :LXC* qui nous permet de toucher directement le conteneur *LXC* qui aura à charge notre application, ce qui est illustré les trois lignes suivantes.

4. Attention, le support du type booléan en *YAML* étant assez hasardeux en Perl 5, nous utilisons dans *Floday* des chaines de caractères égales à « true » ou « false ». On ne peut donc pas employer les autres formes officielles de valeurs existantes.

Listing 3: Exemple de script de setup

```
1  #!/usr/bin/env perl
2
3  use strict;
4  use warnings;
5  use v5.20;
6
7  use Backticks;
8  use Floday::Setup;
9
10 $Backticks::autodie = 1;
11
12 ##########################
13 # Lighttpd installation #
14 ##########################
15 my $lxc = $APP->get_lxc_instance();
16 $lxc->start() if $lxc->is_stopped();
17 $lxc->exec('apk add lighttpd');
18 $lxc->exec('rc-update add lighttpd');
19 $lxc->exec('/etc/init.d/lighttpd start');
20
21 ######################################
22 # Lighttpd configuration management #
23 ######################################
24 $APP->generate_file(
25   'jaxe/children/www/setups/lighttpd/lighttpd.conf',
26   undef,
27   '/etc/lighttpd/lighttpd.conf'
28 );
29 for ($APP->get_sub_applications()) {
30   $APP->generate_file(
31     $_->getParameter('lighttpd_config'),
32     {$_->getParameters()},
33     '/etc/lighttpd/conf.d/'.$_->get_application_path().'.conf'
34   );
35 }
36
37 ###################################
38 # Lighttpd routing instructions #
39 ###################################
40 my $ipv4 = $APP->get_parameter('networking_ipv4');
41 my ($ipv6) = $APP->get_parameter('networking_ipv6') =~ /^(.*)\//;
42 `iptables -t nat -A PREROUTING ! -i lxcbr0 -p tcp --dport 80 -j DNAT --to-dest $ipv4`;
43 `iptables -t filter -A FORWARD ! -i lxcbr0 -p tcp --dport 80 -j ACCEPT`;
44 `ip6tables -t nat -A PREROUTING ! -i lxcbr0 -p tcp --dport 80 -j DNAT --to-dest $ipv6`;
45 `ip6tables -t filter -A FORWARD ! -i lxcbr0 -p tcp --dport 80 -j ACCEPT`;
```

La ligne 24 présente une autre fonctionnalité du module *Floday : :Setup*, celle de faciliter la gestion des fichiers de configuration. Mais là encore, il est conseillé de se référer à *perldoc* pour avoir des détails quant à l'utilisation qui devrait en être faite.

La ligne 29 montre comment nous devons procéder pour agir non pas sur l'application en cours de déploiement, mais sur une autre (ici, les sous-applications qu'il gère).

2.3.5 Déboguer un script

1ère apparition : v1.0.0

Chaque script `setups`, `end-setup` ou autres peuvent facilement être testés sans avoir à redéployer l'ensemble des applications. Nous pouvons lancer un script de façon unitaire en lui renseignant un paramètre *application* ayant la valeur du chemin d'application sur laquelle nous voulons voir le script travailler. Par exemple : `/etc/floday/containers/jaxe/children/backup_client/end_setup/volume_mounter.pl --application spyzone-backup`

2.4 L'exécution

2.4.1 Les différentes étapes

1ère apparition : v1.0.0

Voici les différentes étapes par lesquelles on procède lors du déploiement d'un hôte : La figure 3 illustre à peu près le fonctionnement via un genre d'*UML*. Ce schéma pourrait être plus représentatif, mais je me suis promis de ne pas passer plus de 7h dessus.

FIGURE 2: Illustration du processus de définition de configuration et de déploiement d'applications.

1 Le runfile est parcouru, et la définition de chaque application est faite. Ici, on illustre celle de *spyzone-web*.

2 Pour chaque nœud d'application, nous récupérons la configuration du conteneur en question.

3 La définition du conteneur est ainsi générée. Celle-ci prend en compte toutes les notions d'héritages.

4 La définition de l'application est ensuite obtenue, en se contentant de fusionner la définition du conteneur avec les paramètres surchargés du runfile.

5 À ce moment, la validité des paramètres applicatifs est testée. Il s'agit notamment de vérifier si tous les attributs obligatoires (ceux avec le nœud *mandatory : true* dans leur définition sont renseignés, et si leur forme est correcte (le nœud *pattern*).

6 Si tout est bon, la runlist est générée. Il ne s'agit en fait que d'un hash interne à *Floday* représentant la définition de toutes les applications à déployer sous une forme simplifiée.

7 Les applications sont déployées.

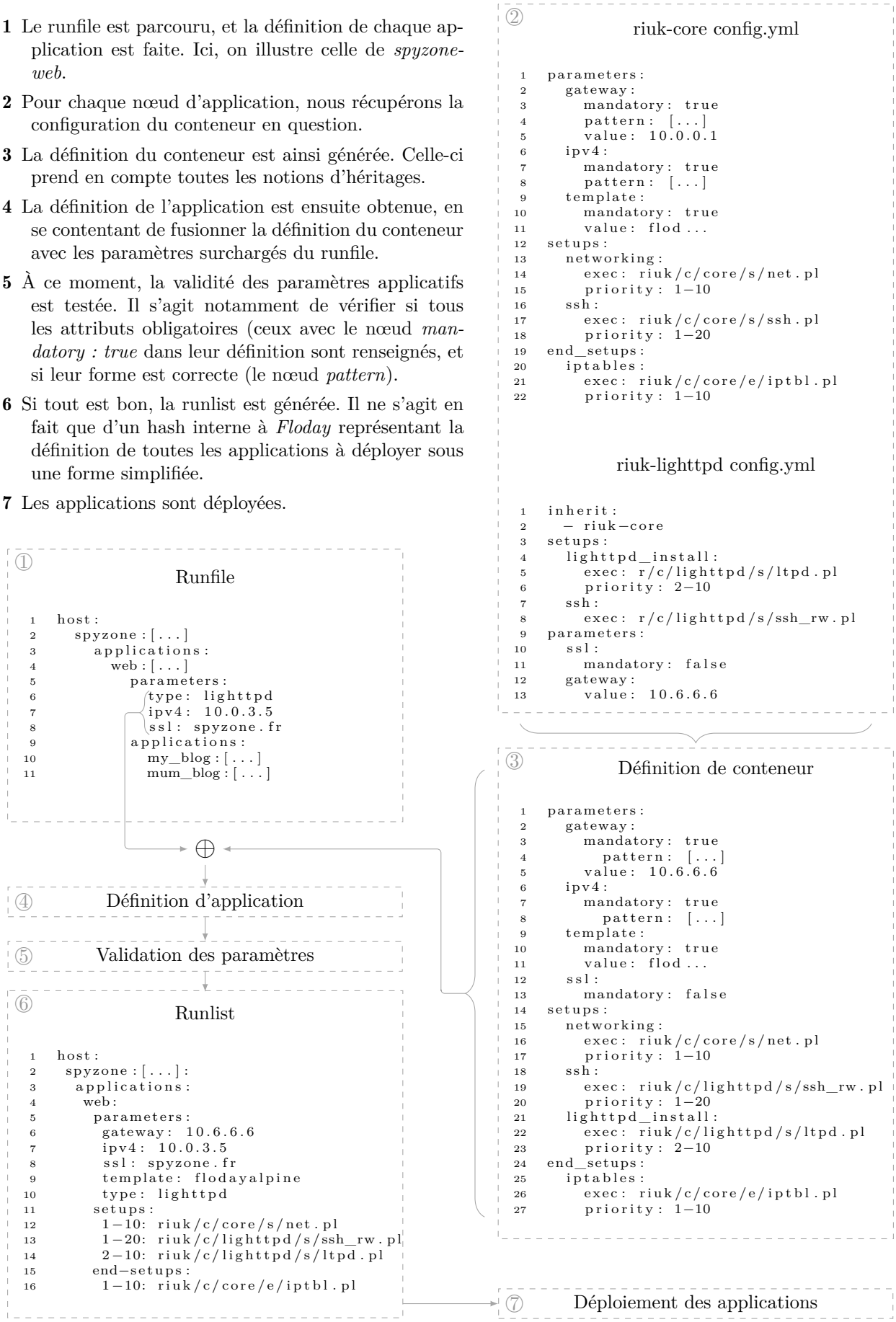
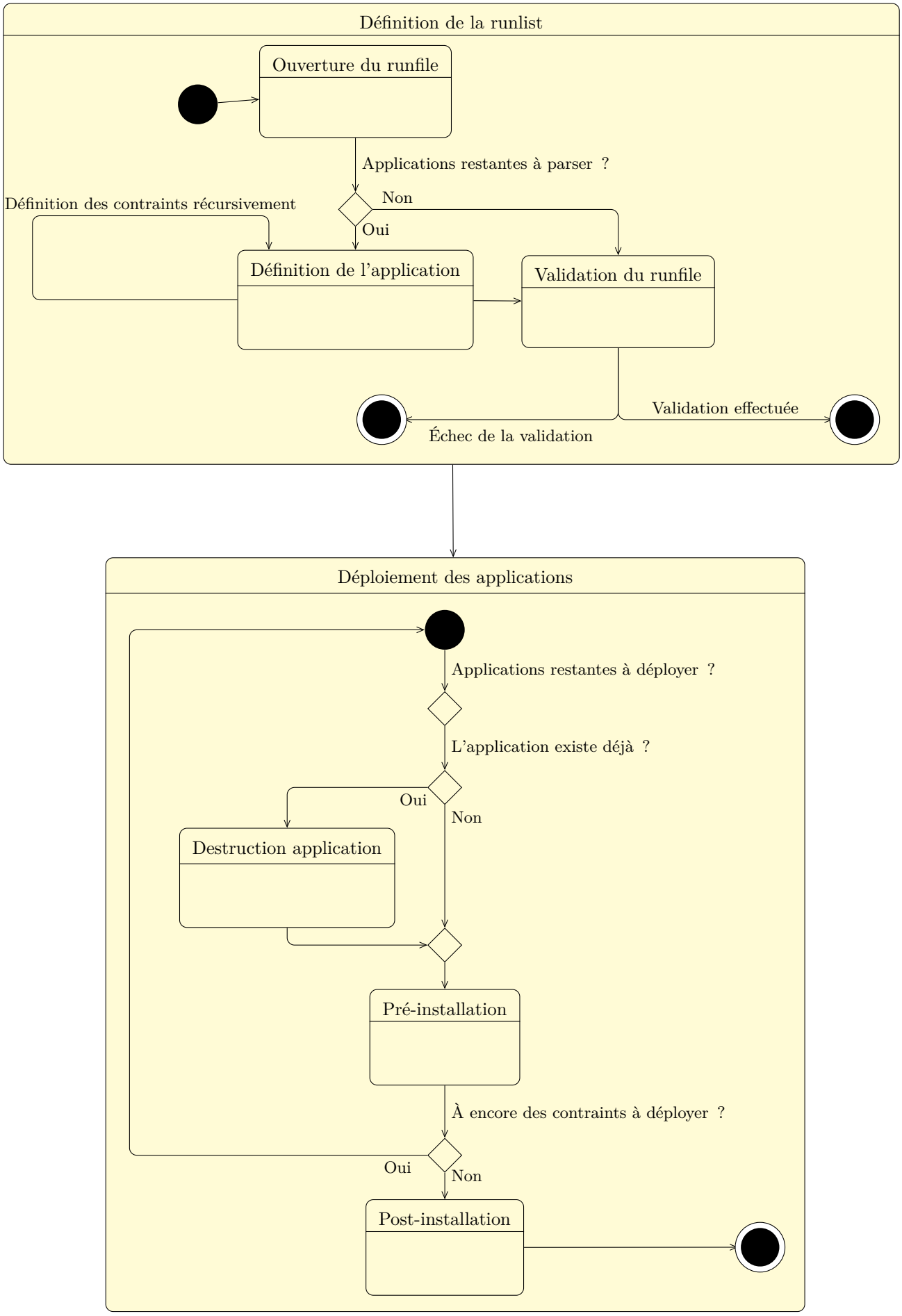


FIGURE 3: Algorithme complet d'un déploiement



Définition de la runlist Cette étape permet d'obtenir un tableau multidimensionnel clair de tous les éléments structurants nécessaires au déploiement. Il peut lui-même se distinguer en trois parties :

Ouverture du runfile Nous récupérons le *runfile* depuis le fichier de configuration. Il est ensuite parcouru pour trouver les chemins de conteneurs à déployer.

Définition des applications En fonction des chemins de conteneurs obtenus, on cherche à présent les fichiers définition des conteneurs utilisés. Les éventuels héritages sont gérés à ce moment. Ensuite sont mergées les valeurs propres à l'application courante à déployer. Il s'agit des paramètres applicatifs présents au niveau du runfile.

Validation de la runlist Nous pouvons à présent valider la conformité de la runlist. Si quelque chose ne correspond pas, le déploiement est annulé avant de faire réellement quoi que ce soit.

La figure 2 illustre ces principes de façon plus visuelle.

Déploiement des applications Pour chaque application présente dans l'hôte en cours de déploiement au niveau du runfile, on effectue les actions suivantes :

Destruction de l'application S'il s'agit d'un redéploiement, c'est-à-dire qu'une application avec un chemin similaire est déjà existante, la première étape consistera en la destruction de celui-ci. C'est à cette étape que nous exécuterons les hooks `lxc-destroy-before` et `lxc-destroy-after`. Bien entendu, entre ces deux hooks, le précédent conteneur servant à accueillir l'ancienne version de l'application sera supprimé.

Pré-installation On commencera par exécuter les scripts présents au niveau du hook `lxc-deploy-b` Ensuite, le conteneur *LXC* sera déployé via le template défini (par défaut *lxc-flodayalpine*), puis les hooks `lxc-deploy-after` seront lancés. Finalement, c'est l'ensemble des *setup* de l'application qui seront exécutés.

Déploiement des sous-applications Les sous-applications sont ensuite intégralement déployées (elles passent donc elles aussi par les étapes de la partie *Déploiement des applications* de cette liste).

Post-installation Pour finir, nous exécutons les scripts présents dans le nœud *end_setup*.

Et voilà, on se retrouve à terme avec un hôte correctement déployé.

2.4.2 La gestion des logs

1ère apparition : v1.0.0

Il ne reste plus qu'à amorcer le tout et permettre à *Floday* d'effectuer sa principale raison d'être ! Pour cela, nous n'avons qu'à lancer la commande `floday` avec l'option `--host`, définissant quel hôte du runfile nous voulons déployer. Des logs d'exécution seront ensuite affichés sur *STDOUT* ainsi que via *syslog*.

Dernière MàJ : v1.1.0

Pour modifier le niveau des logs lors d'un déploiement, l'option `--loglevel` peut être renseignée. Par défaut, c'est le niveau *info* qui sera utilisé.

Dernière MàJ : v1.0.0

Les niveaux disponibles sont ceux habituellement en vigueur sur n'importe quel système *UNIX*. Pour plus de détails, voir le hash *%SYSLOG_PRIORITY_MAPPER* dans le fichier *Floday/Helper/Logging.pm*.

Le listing 4 présente une sortie représentative d'un déploiement. Les messages sont découpés en plusieurs colonnes dont voici les descriptions :

— La première annonce la sévérité du log.
— La seconde, le module d'émission.
— La dernière, le message. Notez que l'indentation dans l'alignement indique le degré d'imbrication. Grâce à cela, nous pouvons rapidement voir, par exemple, que le script *networking.pl* du log ligne 53 aura été exécuté dans le cadre des setups sur *website-web_application_galek*, sous-application de *website-web_application* sur l'hôte *websites*.

Pour le moment, cette sortie est également disponible via syslog, bien que le format sera peut-être amené à évoluer sur celle-ci.

Attention, notez qu'il y a actuellement un bug générant une erreur dans le fichier de log lors du déploiement de l'hôte (présente à la ligne 10 sur le listing 4) celui-ci devrait être résolu un jour [5].

5. Lien vers le ticket présentant l'anomalie : https://dev.spyzone.fr/floday/ticket/31

Listing 4: Sortie d'une exécution de *Floday*

```
# ./floday.pl --host integration
[WARN]                  Floday::Deploy:   Deploying integration host
[WARN]                  Floday::Deploy:    Start running setups scripts.
[WARN]                  Floday::Deploy:    End running setups scripts.
[WARN]                  Floday::Deploy:    Start deployment of integration applications.
[WARN]                  Floday::Deploy:     Launching integration-web application.
[INFO]                  Floday::Deploy:      Start avoidance checks.
[INFO]                  Floday::Deploy:       Running avoidance check: riuk/children/core/avoidance/parameters.pl
[INFO]                  Floday::Deploy:       This script flag application as non-avoidable.
[INFO]                  Floday::Deploy:      End avoidance checks.
[WARN]          Floday::Lib::Linux::LXC:     Start pre destruction hooks.
[INFO]          Floday::Lib::Linux::LXC:      Running script: riuk/children/core/hooks/lxc_destroy_before/clear_filesystem.pl
[WARN]          Floday::Lib::Linux::LXC:     End pre destruction hooks.
[WARN]          Floday::Lib::Linux::LXC:     Start LXC container integration-web destruction.
[WARN]          Floday::Lib::Linux::LXC:     End LXC container destruction.
[WARN]          Floday::Lib::Linux::LXC:     Start post destroy hook.
[INFO]          Floday::Lib::Linux::LXC:      Running script: riuk/children/core/hooks/lxc_destroy_after/update_fstab.pl
[WARN]          Floday::Lib::Linux::LXC:     End post destroy hook.
[WARN]          Floday::Lib::Linux::LXC:     Start pre deployment hooks.
[INFO]          Floday::Lib::Linux::LXC:      Running script: riuk/children/core/hooks/lxc_deploy_before/open_firewall.pl
[WARN]          Floday::Lib::Linux::LXC:     End pre deployment hook.
[WARN]          Floday::Lib::Linux::LXC:     Start deploying LXC integration-web container.
[WARN]          Floday::Lib::Linux::LXC:      End deploying LXC container.
[WARN]          Floday::Lib::Linux::LXC:     Start post deployment hook.
[INFO]          Floday::Lib::Linux::LXC:      Running script: riuk/children/core/hooks/lxc_deploy_after/close_firewall.pl
[WARN]          Floday::Lib::Linux::LXC:     End post deployment hook.
[WARN]                  Floday::Deploy:     Start running setups scripts.
[INFO]                  Floday::Deploy:      Running script: /etc/floday/containers/riuk/children/core/setups/network.pl
[INFO]                  Floday::Deploy:      Running script: /etc/floday/containers/riuk/children/web/setups/lighttpd.pl
[INFO]                  Floday::Deploy:      Running script: /etc/floday/containers/riuk/children/core/setups/data.pl
[WARN]                  Floday::Deploy:     End running setups scripts.
[WARN]                  Floday::Deploy:     Launching integration-web-secondtest application.
[INFO]                  Floday::Deploy:      Start avoidance checks.
[INFO]                  Floday::Deploy:       Running avoidance check: riuk/children/web/children/php/avoidance/parameters.pl
[INFO]                  Floday::Deploy:       Running avoidance check: riuk/children/core/avoidance/importer.pl
[INFO]                  Floday::Deploy:       Application was flagged as avoidable.
[INFO]                  Floday::Deploy:      End avoidance checks.
[WARN]                  Floday::Deploy:      Start running setups scripts.
[INFO]                  Floday::Deploy:       Running script: /etc/floday/containers/riuk/children/core/setups/network.pl
[INFO]                  Floday::Deploy:       Running script: /etc/floday/containers/riuk/children/web/children/php/setups/php.pl
[INFO]                  Floday::Deploy:       Avoided script: /etc/floday/containers/riuk/children/core/setups/data.pl
[WARN]                  Floday::Deploy:      End running setups scripts.
[WARN]                  Floday::Deploy:      Start running end_setups scripts.
[WARN]                  Floday::Deploy:      End running end_setups scripts.
[WARN]                  Floday::Deploy:     Launching integration-web-test application.
[INFO]                  Floday::Deploy:      Start avoidance checks.
[INFO]                  Floday::Deploy:       Running avoidance check: riuk/children/web/children/php/avoidance/parameters.pl
[INFO]                  Floday::Deploy:       Running avoidance check: riuk/children/core/avoidance/importer.pl
[INFO]                  Floday::Deploy:       Application was flagged as avoidable.
[INFO]                  Floday::Deploy:      End avoidance checks.
[WARN]                  Floday::Deploy:      Start running setups scripts.
[INFO]                  Floday::Deploy:       Running script: /etc/floday/containers/riuk/children/core/setups/network.pl
[INFO]                  Floday::Deploy:       Running script: /etc/floday/containers/riuk/children/web/children/php/setups/php.pl
[INFO]                  Floday::Deploy:       Avoided script: /etc/floday/containers/riuk/children/core/setups/data.pl
[WARN]                  Floday::Deploy:      End running setups scripts.
[WARN]                  Floday::Deploy:      Start running end_setups scripts.
[WARN]                  Floday::Deploy:      End running end_setups scripts.
[WARN]                  Floday::Deploy:     Start running end_setups scripts.
[INFO]                  Floday::Deploy:      Running script: /etc/floday/containers/riuk/children/web/end_setup/iptables_save.pl
[WARN]                  Floday::Deploy:     End running end_setups scripts.
[WARN]                  Floday::Deploy:    End deployment of integration applications.
[WARN]                  Floday::Deploy:    Start running end_setups scripts.
[WARN]                  Floday::Deploy:    End running end_setups scripts.
[WARN]                  Floday::Deploy:   integration deployed.
```

2.5 Installation de *Floday*

2.5.1 Quelle version choisir ?

1ère apparition : v1.0.0

Floday utilise le système de normalisation des numéros de version sémantique[6]. Cela signifie que la version du logiciel sera toujours donnée sous la forme de trois entiers : `x.y.z`. Voici leurs représentations :

x correspond à la version majeure. Deux versions majeures différentes ne sont pas interopérables. Il faudra donc s'attendre à ce qu'il y ait de la casse lors d'une mise à jour incrémentant ce numéro.

y correspond à la version mineure, soit à l'ajout de nouvelles fonctionnalités, ou à la réalisation de refactoring mais qui ne modifie pas la façon dont le logiciel s'emploie.

z correspond aux correctifs. Ils sont toujours recommandés, car ils ne sont utilisés que pour rectifier un fonctionnement sensé déjà être opérationnel, les bogues quoi.

Avec ces informations en tête, libre à vous de choisir le cycle d'évolution que vous préférez. Une branche *git* existe pour chacun d'entre eux :

master qui correspondra toujours à la dernière version publiée du logiciel.

v1 qui qui représente la version mineure la plus élevée, mais de la version majeure 1. Votre configuration ne devrait donc pas se voir incompatible après ce genre de mise à jour.

v1.0 qui ne prendra en compte que les correctifs de sécurité ou de bogues.

6. En savoir plus sur le versionnage sémantique : http://semver.org/

Quand une version 1.1 de *Floday* sera propulsée, nous aurons une nouvelle branche *v1.1* de publiée, et il en sera de même pour les autres. Notez que vous pouvez directement récupérer une version en question (par exemple `git checkout 1.0.1`) car un tag sera aussi associé à chacune d'entre elles.

En tant qu'utilisateur, vous serez probablement intéressé par l'évolution de la version majeure que vous employez. Si vous choisissez de mettre à jour directement le logiciel via le dépôt Git, vous pouvez puller régulièrement la branche portant le nom de la version (par exemple v1). Si par contre vous maintenez ce logiciel pour le compte d'une distribution quelconque, vous pouvez continuer à suivre les corrections d'anomalies après le freeze des évolutions en vérifiant régulièrement les nouveaux commits de la branche ayant le nom de la version mineure présente sur la distribution (par exemple v1.1).

2.5.2 Comment l'installer ?

1ère apparition : v1.0.0

Pour le moment, je ne vous cache pas que c'est bien la merde. Mais là aussi, des efforts seront peut-être (probablement) faits un jour.

Dépendances nécessaires pour *Floday*

- Perl 5 (version 5.20 minimum).
- `apt-get install -y -no-install-recommends bridge-utils cgroup-tools cgroupfs-mount curl apparmor apparmor-utils lxc`

Processus d'installation

- Il est conseillé de cloner le dépôt de *Floday*, par exemple dans `/opt` et de le déployer sur le commit correspondant à la version du logiciel que vous désirez utiliser.
- À présent, il faut exécuter le fichier `install.pl` présent à la racine du dépôt. Celui-ci « installera » les modules propres de *Floday* dans un endroit accessible par Perl (`/etc/perl`) et téléchargera tous les modules nécessaires via le *CPAN*.
- Faire un lien symbolique entre `/opt/floday/lxc-template/lxc-flodayalpine` et `/usr/lib/lxc/template/lxc-flodayalpine` pour que le template soit directement accessible depuis `LXC`.

À présent, tout devrait être bon. Pour être sûr, vous pouvez configurer votre machine comme une interface de développement (voir le paragraphe en question) et jouer les tests d'intégration. Mais c'est probablement un peu lourd et foireux de faire ça sur votre future production…

2.6 Écrire son jeu de conteneurs

1ère apparition : v1.0.0

Floday ne prévoit pas de distribuer en plus de cette mécanique, des conteneurs concrets que vous pourriez utiliser directement. C'est à vous d'écrire les vôtres, de façon à ce qu'ils conviennent parfaitement à votre architecture, et que vous sachiez précisément ce qu'ils font ! Vous pouvez tout de même vous inspirer du jeu de conteneurs `riuk`, présent dans le dossier `floday/containers/riuk` pour avoir des indications sur la façon de procéder.

Globalement, le seul fichier imposé est le *config.yml* définissant le conteneur. Les scripts d'exécutions peuvent être localisés où vous le souhaitez, bien que conserver une structure cohérente (comme celle présente dans *riuk*) peut être conseillé. De la même façon, les scripts peuvent être écrits dans le langage que vous voulez, la seule contrainte est que ceux-ci aient les droits d'exécutions par l'utilisateur exécutant *Floday*.

Utiliser Perl pour les scripts peut néanmoins être un choix à privilégier, car le module `Floday::Setup` permet de facilement accéder à l'application en cours de déploiement, à ses paramètres applicatifs ainsi qu'à quelques autres outils comme la génération de fichiers. La documentation du module étant complète, je vous invite à faire un `perldoc Floday::Setup` pour avoir une définition exhaustive de ce qu'il propose.

Par défaut, le jeu de conteneurs devra être présent dans le dossier `/etc/floday/containers`. Une fois vos conteneurs d'écrits, il ne vous restera plus qu'à faire le runfile qui se chargera de les transformer en applications !

Comment contribuer ?

Utiliser une application, c'est bien. Mais contribuer à son amélioration, c'est mieux ! Vous pouvez apporter votre pierre à l'édifice via plusieurs façons. La première consiste simplement à expliquer via le bug tracker les difficultés que vous rencontrez, les bogues découverts ou encore les évolutions que vous aimeriez voir. La seconde serait de directement apporter vous-même ces modifications au code source du projet. Et enfin, une dernière possibilité serait de contribuer à la rédaction de cette documentation, en français, certes, mais aussi dans d'autres langues !
Cette section évoquera ces parties via les axes suivants :

- **Comment utiliser le bug tracker ?**
- **Démarrer une instance de développement de *Floday*.**
- **Contribuer au code source.**
- **Contribuer à la documentation.**

3.1 Le bug tracker

1ère apparition : v1.0.0

Le bug tracker est disponible à l'adresse : https://dev.spyzone.fr/floday/query. C'est l'endroit privilégié d'échange entre utilisateurs et développeurs de l'application. En effet, on y trouve toutes les demandes d'évolutions, les constats d'anomalies, ou des questions dont la réponse n'est pas encore dans ce guide. Ces tickets peuvent donc être de trois types :

3.1.1 Les types de tickets

1ère apparition : v1.0.0

Les tickets d'évolution Ils représentent les nouvelles fonctionnalités à implémenter dans le programme. Ces développements seront à merger dans la prochaine version majeure ou mineure publiée, et doivent généralement être documentés dans ce manuel.

Les tickets d'anomalies Ils représentent les bogues repérés dans le logiciel. Quand un correctif est disponible, il est livré sur chaque branche en ayant besoin.

Les tickets de documentation Il représentent un manquement quant à la compréhension de quelque chose. Ils peuvent être utilisés pour poser une question sur un fonctionnement resté flou. Ces tickets doivent mener à une amélioration de ce document ainsi que des pages de manuels des différents modules Perl mis à disposition via *Floday*.

3.1.2 La rédaction d'un ticket

1ère apparition : v1.0.0

Attention, il est préférable que vous écrivez en anglais lors de la création de votre ticket. Voici une description des informations qui vous seront demandées :

Champ à renseigner	Description
Résumé	Sujet du ticket, soit une phrase qui décrit en une ligne de quoi il parle.
Rapporteur	Laisser la valeur par défaut (Spydemon). Il n'y a que moi qui travaille sur le projet pour le moment de toute façon.
Description	Il faut expliquer avec le plus détails possibles l'objet de votre demande ici afin que les autres (moi ?) aient le plus de chances de la comprendre.
Type	Les différents types disponibles sont listés au paragraphe précédent.
Priorité	Sauf ticket d'anomalie critique, la priorité doit toujours être à *normale*.
Version	La version que vous êtes en train d'utiliser. À noter que pour les tickets d'évolutions, il vaut mieux que celle-ci soit la plus récente possible.
Composant	Défini à quelle partie du logiciel le ticket fait référence. Pour le moment, aucune segmentation n'est présente au niveau du projet, il faudra donc toujours sélectionner *core*.

Copie à	Permet de mettre d'autres destinataires en copie des emails envoyés lors de l'évolution du ticket.
Propriétaire	Il s'agit de vous, vous pouvez laisser la valeur *<default>*.

3.1.3 Le workflow du bug tracker

1ère apparition : v1.0.0

Une fois votre ticket créé, il aura le statut *ouvert*. Il restera dans cet état jusqu'à ce qu'il soit compris et prit en compte par l'équipe concernée (moi ?). Avant cette étape, des échanges auront peut-être lieu dans les commentaires pour essayer d'en savoir plus. Il pourra à ce moment passer soit à l'état *accepté*, ce qui signifiera qu'il sera pris en compte un jour, soit *rejeté*. Dans le second cas, la demande sera ignorée à tout jamais. Si le ticket a été accepté, il prendra l'état *en cours de traitement* lorsque quelqu'un commencera à travailler dessus, avant de finir en *terminé* une fois le développement de publié. À ce moment, il sera ajouté à un jalon définissant à partir de quelle version le contenu relatif au ticket sera intégré à *Floday*.

3.2 Démarrer une instance de développement

1ère apparition : v1.0.0

La première étape pour contribuer est de réussir à lancer *Floday* dans un environnement contrôlé sur lequel des tests peuvent être exécutés sans crainte. Une façon de faire va être décrite dans cette section.

Pour fonctionner, on utilisera deux images *VirtualBox* dont l'une sera un clone de la première. La machine virtuelle de base s'appellera *Floday_Clean* et la copie de travail courante *Floday_Work*. Toute la configuration sera à faire sur *Floday_Clean*, ce qui permettra de réinitiliser *Floday_Work* si un déploiement se passe mal.

Je présente ici la procédure d'installation de cette architecture sur une Debian 8.0 (Jessie) avec OpenRC comme système d'init. Il n'y a cependant aucune restriction quant à celle-ci, vous pouvez utiliser le socle que vous voulez, tant que *LXC* y est correctement supporté. Notez aussi que *riuk*, le jeu de conteneur de test nécessite *OpenRC*.

La création de cette infra reste assez bancale pour le moment, mais elle a le mérite de fonctionner très bien une fois les galères de l'installation passées. Le jeu en vaut donc la chandelle ! Voici les étapes :

- Installation de *VirtualBox* sur l'hôte.
- Création d'un dossier partagé appelé *floday* entre `/opt/floday/` sur la VM et le dépôt Git sur l'hôte. Attention, ce partage doit être en lecture et écriture pour pouvoir passer tous les tests unitaires.
- Configurer les réseaux *VirtualBox* pour avoir un pont sur *eth0* et un réseau privé hôte sur *eth1*.
- Virer *systemd* pour utiliser *Open-RC* à la place : http://linuxmafia.com/kb/Debian/openrc-conversion.html.
- Installation des additions invitées : https://virtualboxes.org/doc/installing-guest-additio
- Création des dossiers `/opt/floday`, `/var/lib/floday` et `/etc/floday`.
- Variables d'environnement à ajouter dans `/root/.bash` :
  ```
  export FLODAY_CONTAINERS="/opt/floday/src/containers/";
  export FLODAY_T="/opt/floday/t/";
  export FLODAY_T_SRC="/opt/floday/src/";
  ```
- `apt-get install -y -no-install-recommends bridge-utils cgroup-tools cgroupfs-mount curl apparmor apparmor-utils lxc`
- Édition du fichier */etc/network/interfaces* :

```
1  source /etc/network/interfaces.d/*
2
3  # The loopback network interface
4  auto lo
5  iface lo inet loopback
6
7  # The primary network interface
8  allow-hotplug eth0
9
10 iface eth0 inet dhcp
11 iface eth1 inet dhcp
12 # This is an autoconfigured IPv6 interface
13 iface eth0 inet6 auto
14
```

```
15 auto lxcbr0
16 iface lxcbr0 inet static
17 address 10.0.3.1
18 netmask 255.255.255.0
```

— Écrire le fichier `/etc.init.d/floday`. Il faudra penser à changer *192.168.1.12* par l'ip de votre VM appartenant à l'interface bridgé avec votre routeur allant vers Internet (vous pouvez facilement la récupérer avec la commande `ip addr list eth0`.

```
1  #!/sbin/openrc-run
2
3  depend() {
4    need cgroupfs-mount
5    need vboxadd
6    before backup_watcher
7  }
8
9  start() {
10   ebegin "Init Floday stuff"
11   brctl addbr lxcbr0
12   ifup lxcbr0
13   dhclient eth1
14   mount -t vboxsf floday /opt/floday
15   mount -t vboxsf perlvirtlxc /opt/perlvirtlxc
16   echo 1 > /proc/sys/net/ipv4/ip_forward
17   iptables -t nat -A POSTROUTING -s 10.0.3.0/24 -o eth0 \
18     -j SNAT --to-source 192.168.1.12
19   eend 0
20 }
```

— `rc-update add floday`
— `ln -s /opt/floday/lxc-templates/lxc-flodayalpine \ /usr/share/lxc/templates/`
— `ln -s /opt/floday/floday.cfg /etc/floday/floday.cfg`
— `ln -s /opt/floday/t/integration/floday.d/runfile.yml /etc/floday/runfile.yml`
— Écrire le fichier `/etc/lxc/default.conf` :

```
1 lxc.network.type = veth
2 lxc.network.link = lxcbr0
3 lxc.network.flags = up
4 lxc.network.hwaddr = 00:16:3e:xx:xx:xx
```

— `cpan Log::Any::Adapter Test::Exception Backticks Moo Config::Tiny File::Slurp YAML::Tiny Template::Alloy Hash::Merge IPC::Run Unix::Syslog`
— Ajouter les paramètres `apparmor=1 security=apparmor` aux variables `GRUB_CMDLINE_LINUX` du fichier */etc/default/grub*.
— `sudo update grub`.
— Ajouter la ligne `10.0.3.5 test.keh.keh test2.keh.keh` dans le fichier */etc/hosts* de la machine virtuelle pour pouvoir réussir les tests d'intégrations.
— Rebooter la machine virtuelle.

Pour valider que la configuration soit correcte, il vous est conseillé de jouer les tests d'intégrations. S'ils ne passent pas malgré le fait que vous avez scrupuleusement suivi ce guide, vous êtes invités à poster un ticket de type *question* afin que nous puissions enrichir la procédure.

À terme, un script de déploiement automatique sera peut-être fourni.

3.3 Apporter des modifications au code source

1ère apparition : v1.0.0

Une fois que vous accédez au code source de l'application, vous pouvez en faire ce que vous voulez. Si participer à l'évolution vous intéresse, vous êtes encouragé à me faire des pull requests que je mergerais dans le dépôt principal si votre travail répond correctement à un ticket ayant l'état *accepté*.

N'ayant jamais effectué ce genre d'actions pour le moment, je ne suis pas sûr qu'il s'agisse de la meilleure façon de faire, ni même qu'elle soit réellement pratique sur le long terme, mais on aura toujours l'occasion de changer ça une fois ce logiciel massivement maintenu ☺.

3.3.1 Convention de code

1ère apparition : v1.0.0

Dans un souci de cohérence, de brèves consignes de style ont été mises en place. Il vous est demandé de les respecter :

— Les commentaires ainsi que les variables doivent être écrits en anglais.
— On utilise le *snake_case*.
— Une tabulation est employée pour l'indentation.
— Deux espaces sont insérés pour les brisements de lignes.
— La virgule touche touche toujours un caractère à gauche, et possède un espace ou un saut de ligne à droite.
— On ne saute pas de ligne avant l'accolade ouvrante.
— Une parenthèse, accolade ou crochet fermant doit être au même niveau d'indentation que son binôme ouvrant.

Le listing 5 est un exemple permettant d'illustrer l'histoire de l'indentation et du brisement de lignes. Les boules grises représentent des espaces, et les flèches une tabulation.

Listing 5: Conventions de code

```
1 sub write_stuff(
2 ··$param_one,
3 ··$param_two
4 ) {
5 —>print "$param_one $param_two";
6 }
```

3.3.2 Workflow *Git*

1ère apparition : v1.0.0

Les types de branches La figure 4 (qui est moche pour le moment, mais sera refaite avec *TikZ* un jour) illustre de façon plus visuelle les explications ci-dessous. Notez qu'il manque néanmoins la branche *master* et les branches *release* sur celle-ci, elle n'est donc pas tout à fait juste non plus...

Attention, pour le moment il s'agit d'un workflow plutôt théorique et qui reste très confus... Je n'ai en effet pas encore eu l'occasion de le mettre en pratique. Celui-ci est susceptible de changer !

Branche *master* Elle pointera toujours sur la dernière version stable. Même si cela nécessite un changement de version majeure.

Branche *development* C'est sur celle-ci qu'on mergera les tickets relatifs à la prochaine version mineure ou majeure en cours de développement. Quand ceux-ci sont terminés, une nouvelle branche stable sera créée depuis celle-ci, ainsi que la branche *bugfixes* correspondante.

Branches stables Une branche est créée pour chaque version majeure et mineure du logiciel. Ces branches doivent être utilisées en production.

Branches bugfixes Permettent la préparation des correctifs. Une branche *bugfixes* n'existe qu'avant la publication d'une nouvelle version de correctifs. Quand nous estimons que celle-ci contient suffisamment d'éléments pour justifier une release, elle sera mergée sur la branche stable correspondante, incrémentant ainsi le numéro z de la version. La branche pourra ensuite être supprimée et recrée à partir de la nouvelle version mineure si besoin est.

Branches d'implémentations Une branche d'implémentation doit être créée pour chaque ticket. La base de cette branche dépendra du type de ticket, on en parle dans un paragraphe sous-jacent. Une fois les développements réalisés et mergés au bon endroit, nous pouvons supprimer la branche pour éviter de polluer le git. Deux types de branches d'implémentation existent :

Implémentation de fonctionnalités Elles partent de l'état courant de *master* et sont à utiliser pour l'accomplissement d'un ticket d'évolution. Une fois le développement considéré comme terminé, la branche pourra être mergée dans *developement*. Si après ce merge, on s'aperçoit d'un bug ou d'une régression entrainée par celui-ci, il faudra continuer à publier les correctifs sur cette branche, puis la remerger dans *development*. Attention à bien la conserver jusqu'à ce qu'elle soit mergée sur une branche stable.

Implémentation de correctifs Ces branches partent de la première version stable ayant le problème. Elles seront ensuite mergées dans les branches *bugfixes* de chaque autre version touché par l'anomalie, voir dans *development* si elle monte aussi haut. Idéalement, le correctif devrait donc se présenter sous la forme d'un seul commit, ce qui rend facile le *cherry-pick*.

Procédure de merge De façon générale, les messages de commit doivent être écrits en anglais. *Git* doit aussi être correctement configuré pour que le nom et l'adresse email de l'utilisateur soient les bons. Une fois une branche de développement prête à être mergée, le commit devra avoir un message dont la première ligne ressemble à :
`[<signe_ticket>] #<numéro_ticket>: <description>.`
Avec :

- *<description>* – dois commencer par une minuscule, et décrire de façon concise ce qui a été réalisé dans le commit.
- *<signe_ticket>* – le signe du ticket dépend grandement du type de branche sur laquelle nous sommes.

Si le message de commit mérite de plus amples informations, on pourra marquer ce texte après un saut de ligne. Le signe du ticket peut ici avoir trois états différents :

- `+` en cas de traitement d'une évolution.
- `-` en cas de correction d'anomalie publiée.
- `*` en cas de correction d'anomalie d'évolution (évolution déjà mergée dans *development* mais qui nécessite un nouveau merge à cause d'une régression).

Voici des indications plus précises concernant les branches en fonction de leurs types :

Branches d'implémentations N'oubliez pas qu'il s'agit là des seules branches pouvant réellement avoir du contenu applicatif. Les commits peuvent ici avoir une forme assez libre. Comme d'habitude, privilégiez d'en faire de petits, ayant des messages précis sur leurs apports.

Branche *development* Une fois une fonctionnalité mature, elle peut être mergée sur *development*. À ce moment, la branche d'implémentation en question ne doit pas être supprimée, car le pointeur sera nécessaire pour merger une seconde fois sur une branche stable. Notez que ce merge doit être fait avec l'option *–no-ff* pour éviter de n'avoir qu'un simple déplacement de pointeur au niveau de *Git*.

Le message de ce commit peut avoir la forme générique présentée ci-dessus, avec un `+` ou `*` comme signe de ticket.

Branches de release et branches stables Une fois des développements suffisamment testés (sur la branche *development*), une nouvelle version pourra être publiée incluant toutes les évolutions validées. Il s'agira d'une incrémentation de version mineure si la rétrocompatibilité n'est pas brisée, ou sinon de la version majeure. Dans ce premier cas, n'oubliez pas d'également merger la nouvelle branche dans la version majeure concernée.

Concrètement parlant, on créera ici une nouvelle branche depuis une branche stable que l'on veut voir évoluer (une branche majeure ou mineur) qui s'appellera *release-vx.y.0*. Sur cette branche nous mergerons toutes les branches d'implémentations que l'on désire inclure dans cette nouvelle version. Finalement, on créera un nouveau commit (vide ? (option *allow-empty*)) à la tête de celle-ci, ayant comme première ligne sur le message de commit : `[RELEASE] vx.y.0` Avec x et y qui représente le nouveau numéro de version (z sera forcément à zéro). Les autres lignes peuvent être un rapide journal des changements présentant les différents tickets intégrés à l'évolution.
Si le package ainsi produit fonctionne bien, la release sera validée, et un merge avec l'option `no-ff` sur la branche stable pourra être effectué. Nous pouvons finalement supprimer la branche de release.
Depuis ce commit, un tag sera créé avec le même numéro de version(`git tag -m x.y.0 -a x.y.0`), mais sans le préfixe *v*.

Branches bugfixes Une branche *bugfixes* devra être créée en parallèle de chaque version stable. Elle se chargera d'agréger les branches d'implémentation correspondant à des corrections à intégrer. Le changelog ne devra présenter que les correctifs publiés. Notons qu'il ne devrait de toute façon rien avoir d'autre de modifié.

Quand la branche *bugfixes* est freezée, elle sera mergée avec comme message de commit `[RELEASE] vx.y.z` dans la branche stable à laquelle elle est liée. À présent, cette branche *bugfixes* pourra être supprimée et pourra être recréée depuis cette nouvelle version stable pour accueillir les futurs correctifs.

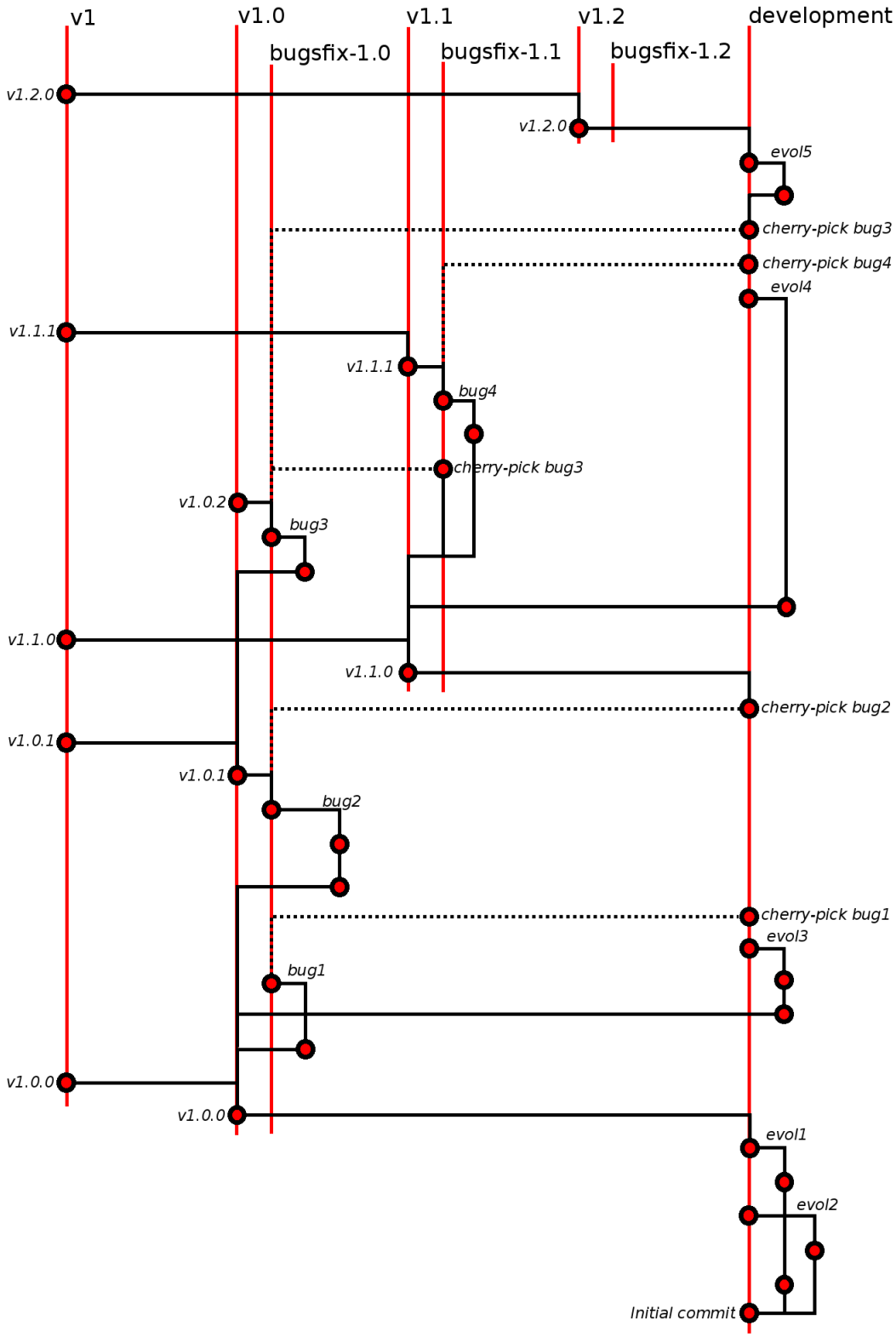


FIGURE 4: Workflow git

3.4 Participer à la documentation

1ère apparition : v1.0.0

La documentation s'écrit de la même façon que le code source. En effet, il s'agit d'une suite de fichiers LaTeX présents au sein du dépôt *Git* du projet. Cette caractéristique permet d'avoir une documentation qui est toujours en adéquation avec la version du logiciel en cours d'utilisation.

3.4.1 Comment compiler soi-même la documentation

1ère apparition : v1.0.0

Normalement, cette étape est assez simple si vous avez correctement installé *pdflatex* et la dizaine de giga-octets de dépendances qu'il embarque. Il ne vous restera plus qu'à lancer les commandes suivantes, dans l'ordre donné :

- `cd <racine du dépôt>/doc`
- `pdflatex main.tex`
- `pdflatex main.tex`
- `makeglossaries main`
- `pdflatex main.tex`

Si vous ne venez pas du monde merveilleusement chiant qu'est LaTeX, vous seriez probablement étonné de voir trois fois la même commande... L'ordre et la fréquence sont importants pour s'assurer une compilation correcte du glossaire, de la table des matières et des numéros de pages.

3.4.2 Quoi documenter ?

1ère apparition : v1.0.0

Tout devrait l'être. Bien entendu, quand une nouvelle fonctionnalité est ajoutée à *Floday*, il faut obligatoirement en parler ici. Après, je conçois qu'elle n'est actuellement pas rédigée par des gens qui savent écrire correctement, elle peut être pleine de fautes, ou à la compréhension douteuse. C'est pour cela que l'incrémentation des versions mineures du logiciel peut aussi s'expliquer par un enrichissement de la documentation. N'oubliez pas qu'il est possible de poster sur le bug tracker des tickets indiquant des problèmes au niveau de celle-ci. Les correctifs devront donc finir sur des branches *bugfixes* ou *release*.

3.4.3 Traductions de la documentation

1ère apparition : v1.0.0

Comme je suis français, il ne s'agit actuellement que de l'unique langue dans laquelle cette documentation est disponible... Si vous voulez la traduire, n'hésitez pas à me contacter pour que l'on voie comment mettre ça en place !

Glossaire

action Une action consiste en un procédé actif réalisé par *Floday* dans le cadre du déploiement. Elles peuvent être des exécutions de commandes, des écritures de configuration, ou de toutes autres formes prévues par la définition du conteneur en cours d'instanciation. . 22

application Service concret rendu par le système informatique de référence, résultant du déploiement d'un conteneur. C'est l'entité pour laquelle *Floday* existe : le processus final utilisé par les clients. Exemple : un blog, un serveur *Mumble*, un service de messagerie instantanée. . 4–6, 8, 14, 22, 23

attribut L'attribut est une entité d'information permettant de définir un conteneur. Il peut avoir des formes très variées, et se situe généralement au sein de la configuration d'un conteneur. Ils ont également des formes hiérarchiques : un attribut peut en contenir d'autres. De façon générale, les attributs peuplent intégralement les fichiers *config.yml* présents au cœur de chaque conteneur. . 4, 7

chemin d'application Définit le nom concret d'une application en fonction de l'imbrication de ses gestionnaires. Un tiret est utilisé pour séparer les différentes imbrications. Par exemple, on peut avoir : *spyzone-web-my_blog* pour une application nommée *my_blog* contraint par *web* sur l'hôte *spyzone*. Il s'agit ici d'une notion similaire à celle du chemin de conteneur, mais avec le nom de l'application à la place du type de conteneur. . 6, 9, 22

chemin de conteneur Définit le type concret d'un conteneur en fonction de l'imbrication de ses gestionnaires. Un tiret est utilisé pour séparer les différentes imbrications. Par exemple, on peut avoir : *riuk-web-wordress* pour un conteneur *wordpress* contraint par *web* dans le jeu de conteneurs *riuk*. Il s'agit ici d'une notion similaire à celle du chemin d'application, mais avec le type de conteneur à la place du nom de l'application. . 6, 7, 12, 22

conteneur Le conteneur est une recette à appliquer pour produire une application. Un même conteneur peut donc être utilisé pour générer plusieurs applications similaires. Par exemple : un conteneur *Wordpress* peut être utilisé pour générer plusieurs blogs. . 4, 22, 23

définition – de conteneurs Une définition de conteneur présente la façon dont le conteneur doit se déployer. Il s'agit d'y définir les scripts à exécuter, dans quel ordre, ainsi que tous autres éléments nécessaires, comme la création des attributs. Cette définition s'effectue dans les fichiers `config.yml` du conteneur courant, de ceux de ses parents ainsi que du runfile (pour les attributs surchargés). . 6, 12, 22

déploiement – d'applications Le déploiement est l'action permettant de transformer la définition d'un conteneur en application. Il est généralement constitué de deux étapes principales : l'initialisation et l'itération des différents scripts d'installation présents au niveau de la définition du conteneur en question. . 8

gestionnaire – de conteneurs Un conteneur est gestionnaire s'il manage ou permet à d'autres conteneurs de fonctionner au travers de lui. Un gestionnaire s'occupe de ses sous-conteneurs. . 4, 5, 22, 23

hôte L'hôte et le niveau zéro de l'imbrication. Il s'agit du système physique sur lequel les applications sont exécutées. Les actions qui y sont effectuées n'ont donc aucun confinement. . 5, 12, 22

imbrication L'imbrication représente un niveau de hiérarchie entre les sous-applications et les gestionnaires. L'imbrication zéro représente l'hôte sur lequel les applications s'exécutent. Une application d'imbrication deux signifie qu'elle est contrainte par une autre application présente entre elle-même et l'hôte. Par exemple : *host-web-wordpress*. . 6, 22

initialisation Il s'agit du minimum à effectuer pour permettre le déploiement d'une application fonctionnelle, c'est-à-dire généralement, avoir un *OS* minimaliste permettant les fonctionnalités de base. Actuellement, les seules initialisations de supportées le sont via des templates *LXC*. Concrètement, il s'agit donc d'exécuter la commande `lxc-deploy` avec le bon template (*flodayalpine* est fourni par défaut). Comme son nom l'indique, il se contente de créer un conteneur *Alpine Linux* minimaliste. Actuellement, l'initialisation à utiliser pour chaque conteneur est définie à l'aide du paramètre applicatif *template*. . 5, 7, 8, 22

instanciation Installation d'une application. C'est-à-dire qu'on déroule tous les scripts d'exécution pour mettre en place l'application et la rendre utilisable. . 5, 6, 22

jeu de conteneurs Le jeu de conteneurs constitue la définition même de ce que *Floday* est capable de déployer. Par défaut, le jeu doit être présent dans le répertoire `/etc/floday/containers`. Il est à la charge de l'utilisateur d'écrire son propre jeu, car seul lui est à même de savoir comment son architecture devrait fonctionner. . 4–6, 14, 22

paramètre applicatif Un paramètre applicatif est une entité de configuration appliquée à une application. Il peut être implicite, s'il est défini au niveau du conteneur ou explicite, s'il l'est plutôt au niveau du runfile. Il peut aussi être redéfini à chaque niveau : dans un conteneur hérité, fils puis dans le runfile. Comme exemple, nous pouvons prendre le cas d'un paramètre *url* présent dans le conteneur *Wordpress* permettant de savoir comment configurer correctement le *CMS*. . 4, 6, 7, 12, 14, 22, 23

propulsion – de version On parlera de propulsion de version l'action de stopper les développements en cours et de les publier. Par exemple, nous pouvons dire :
— La branche *développement* propulsera la prochaine version majeure.
— La branche *bugfixes1.2* propulsera la nouvelle version *v1.2.z*.
Le terme peut également être utilisé pour indiquer dans quelles versions se situent un ticket, ou une fonctionnalité précise. Nous pouvons dire :
— Les corrections #3, #4 et #5 seront propulsées dans la *v1.2.4* et la *v1.3.0*.
— La fonctionnalité de parsage dynamique de la configuration sera propulsée dans la *v1.4*.
. 14

runfile Par défaut à l'emplacement `/etc/floday/runfile`, sont rôle est de définir explicitement toutes les applications à exécuter sur l'ensemble du « parc » informatique géré. Le même fichier est donc utilisé sur plusieurs hôtes pour permettre là encore de voir les interactions (procédés de backup, ou de répartition de la charge, etc.). Il contient donc une liste exhaustive de toutes les applications à déployer, ainsi que la liste de tous les paramètres applicatifs non implicites. . 4, 5, 7, 8, 12, 14, 22, 23

runlist Il s'agit d'un composant interne à *Floday* utilisé pour représenter de façon la plus simple possible (via un tableau multidimensionnel) tout ce qui constitue une application. C'est à dire l'ensemble de sa définition une fois tous les procédés d'héritages et de réécritures effectués. . 12

sous-application Une sous-application représente une instance de sous-conteneur. Il ne s'agit là aussi que d'un terme utile à la clarification de la hiérarchie entre les applications. . 6, 8, 9, 12, 22

sous-conteneur Un sous-conteneur est un conteneur dont son propre fonctionnement est conditionné par un autre, appelé conteneur gestionnaire. Ce terme permet d'établir la hiérarchie entre les différents conteneurs. Exemple : *spyzone-web-wordpress*. Ici, *wordpress* est un sous-conteneur de *web*. Ça signifie que *web* aura, dans la propre définition, des actions à prendre en fonction de ses sous-applications (une configuration adéquat au niveau du serveur Web, par exemple). . 4, 6, 22, 23

évitement – de déploiment Une application peut voir son déploiement évité si la version actuelle de l'application en cours de fonctionnement répond favorablement à une série de critères établis au niveau de la définition du conteneur. Si une application est considérée comme *évitable*, elle ne sera plus initialisée (initialisation) et les scripts de *setups* et *end_setup* marqués comme évitables ne seront pas non plus exécutés. Cette notion permet de fortement optimiser le redéploiement d'un hôte ayant que des modifications mineures dans son runfile. . 8

The GNU General Public Licence

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program–to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software ; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it : responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps : (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how

to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

   The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

   A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

   The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

   The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

   The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

   The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

   All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

   You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

   Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed ; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

   No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

   When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

   You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice ; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code ; keep intact all notices of the absence of any warranty ; and give all recipients a copy of this License along with the Program.

   You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

   You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions :

   (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

   (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

(c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

(d) If the work has interactive user interfaces, each must display Appropriate Legal Notices ; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways :

(a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

(b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

(c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

(d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

(e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms :

(a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License ; or

(b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it ; or

(c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version ; or

(d) Limiting the use for publicity purposes of names of licensors or authors of the material ; or

(e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks ; or

(f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions ; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

    The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

    Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

    If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

    Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

    THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

    IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

    If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

End of Terms and Conditions

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty ; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode :

```
<program>  Copyright (C) <year>  <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
This is free software, and you are welcome to redistribute it
under certain conditions; type ‘show c’ for details.
```

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different ; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see `http://www.gnu.org/licenses/`.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read `http://www.gnu.org/philosophy/why-not-lgpl.html`.