

# Cahier des charges

KostiTeam

17 novembre 2016

# Table des matières

<b>1</b>	<b>Problématique</b>	<b>3</b>
<b>2</b>	<b>Solution</b>	<b>3</b>
<b>3</b>	<b>LXC</b>	<b>3</b>
3.1	Définition . . . . .	3
3.2	Pourquoi LXC ? . . . . .	4
3.3	Fonctionnement de LXC . . . . .	4
3.3.1	Les containers, leur fonctionnement . . . . .	4
3.3.2	Bridges et fonctionnement réseau . . . . .	4
<b>4</b>	<b>Architecture du programme</b>	<b>5</b>
4.1	Vues . . . . .	5
<b>5</b>	<b>Etapes prévues</b>	<b>7</b>
<b>6</b>	<b>Notice d'utilisation de LXC</b>	<b>7</b>
6.1	Bien débiter . . . . .	7
6.1.1	Installer LXC . . . . .	7
6.1.2	Créer un container . . . . .	7
6.1.3	Démarrer un container . . . . .	7
6.1.4	Configurer un container . . . . .	8
6.1.5	Relier les containers avec les ponts (ou bridges) . . . . .	9
<b>7</b>	<b>Rapport de bug</b>	<b>9</b>
<b>8</b>	<b>Annexe</b>	<b>10</b>
8.1	Diagrammes . . . . .	10
8.1.1	Diagramme de cas d'utilisation : . . . . .	10
8.1.2	Diagramme de classe d'analyse . . . . .	11
8.2	Diagrammes de séquence . . . . .	11
8.3	Scénarios . . . . .	14
8.3.1	Premier cas . . . . .	14
8.3.2	Deuxième cas . . . . .	14
8.3.3	Troisième cas . . . . .	15
8.3.4	Quatrième cas . . . . .	15
8.3.5	Cinquième cas . . . . .	16

# 1 Problématique

La virtualisation est le fait de créer une version virtuelle d'une entité physique, ces versions virtuelles sont alors appelées Machines virtuelles ou VM (Virtual Machine). Nous pouvons prendre comme exemple Virtual Box, qui est un outil de virtualisation permettant d'émuler un système d'exploitation (linux ou autre). Les différentes ressources de la machine hôte sont alors partagées et allouées dynamiquement aux différentes machines virtuelles par des logiciels appelés hyperviseur.

La virtualisation peut être utile dans diverses situations. En effet, virtualiser un environnement/un objet permet de trouver ses caractéristiques optimales à moindre coût. De plus, cela permet aussi d'acquérir des compétences sans pour autant disposer de matériel. Dans le cadre d'études informatiques, il peut être pratique de simuler des réseaux dans lesquels des machines dialoguent. Marionnet est un exemple de logiciel de simulation de réseaux. Cependant, après l'avoir utilisé un tant soit peu, il s'avère que plusieurs défauts gênent son utilisation :

- crashes assez fréquents ;
- impossible de changer les configurations des machines en marche ;
- processus d'arrêt des machines trop fastidieux ;
- interface peu ergonomique ;
- logiciel trop gourmand en ressources.

Face à ces problèmes, nous avons donc décidé de créer un nouveau simulateur de réseaux.

## 2 Solution

Pour pallier les problèmes exposés précédemment, il nous semble pertinent de ne pas virtualiser les machines utilisées (VM) mais uniquement les environnements (VE).

Les VM ont un défaut pour nous : elles simulent tout le matériel d'une machine (processeur, RAM...). Nous préférons donc utiliser des VE (Virtual Environment). Les VE permettent de ne simuler que le système d'exploitation, et de partager le même noyau que la machine hôte, et donc, de répartir les ressources entre machine hôte et les différents environnements virtuels. Cette solution correspondrait donc plus à nos besoins, grâce à un gain notable de performance, notamment lorsque le nombre de machines est important.

Pour virtualiser les environnements, nous avons choisi d'utiliser la technologie de LXC.

## 3 LXC

### 3.1 Définition

LXC est un outil de virtualisation permettant de créer des environnements virtuels, différents systèmes d'exploitations sont mis à disposition (Ubuntu, Debian...). Ces Environnements sont appelés containers. Le partage des ressources est assuré par l'outil Cgroups du noyau, qui permet de limiter, compter et isoler l'utilisation des ressources.

Chaque environnement virtuel est isolée, de la même manière que l'isolement d'un programme avec "*chroot*" : chaque environnement est créé de manière à ce qu'ils n'aient pas accès au système d'exploitation de la machine hôte. En revanche, la machine hôte, elle, a accès aux machines virtuelles. Cette isolation entre les machines virtuelles et la machine hôte, permet de garantir une certaine sécurité.

## 3.2 Pourquoi LXC ?

LXC possède sa propre API, notamment en C/C++, ce qui nous permet de l'intégrer ) notre projet efficacement. Il est plus léger que des technologies comme Docker (qui hérite de LXC) et convient à notre utilisation.

## 3.3 Fonctionnement de LXC

### 3.3.1 Les containers, leur fonctionnement

Les environnements virtuels, ou, containers (conteneurs en français) doivent, en premier temps, être créés à l'aide de la commande `"lxc-create ..."` (voir notices pour plus de précisions sur les commandes). De nombreux systèmes d'exploitation seront alors proposés, (nous avons choisi de prendre Debian, Jessie, i386). Par défaut, les machines créées ne sont pas configurées : elles n'ont pas d'interfaces, elles n'ont pas de compilateur, et les utilitaires préinstallés sont très rudimentaire (pas de ping, ifconfig, tcpdump...).

Chaque container possède un fichier de configuration situé à l'emplacement suivant : `" /var/lib/lxc/<nom du container>/config"`. Il est possible de configurer de nombreux aspect du container dans ce fichier (par exemple : modifier les variables d'environnement, ou changer le nom d'hôte ("*hostname*") du container. Voir "*man lxc*" pour en savoir plus). Ce fichier va notamment permettre de paramétrer la configuration réseau du container à son démarrage (c'est ce qui va nous intéresser en priorité avec ce fichier).

Les containers se lancent avec la commande `"lxc-start ..."` ; il est préférable de les lancer en démons (en arrière-plan), puis d'y "*attacher*" un terminal avec `"lxc-attach"`, afin d'être connecté en super utilisateur (ou root) car, premièrement, par défaut, aucun profil d'utilisateur n'est créé sur le container, et, de plus, cela permet d'avoir l'entier contrôle du container afin de, par exemple, modifier son adresse ipv4.

### 3.3.2 Bridges et fonctionnement réseau

#### 3.3.2.1 fichier de configuration

Comme expliqué plus haut, les paramètres réseaux du container peuvent être modifiés via le fichier config. Dans ce fichier, chaque "*block*" permet de définir une interface. Un block commence toujours par la définition du type de réseau, nous allons donc nous intéresser tout d'abord au type de réseau ("*lxc.network.type*"). Plusieurs types de réseaux sont disponibles, mais, nous allons choisir le type veth, qui signifie Virtual Ethernet, ce type permet d'établir un lien entre l'interface virtuelle du container et un pont, préalablement créé sur la machine host (nous verrons cette liaison plus en détail par la suite).

Ce fichier permet aussi d'établir des adresses ipv4 ("*lxc.network.ipv4*"), ipv6 ("*lxc.network.ipv6*"), et mac ("*lxc.network.hwaddr*"), ainsi que leurs Broadcast. Cela permet de mettre en place le réseau virtuel avant de lancer les machines ; bien que ces paramètres puissent être modifiés une fois la machine lancée à l'aide de l'outil "*ifconfig*". Attention, les tables de routage ne peuvent pas être configurées d'avance, il faut les configurer une fois le VE lancé.

Le dernier paramètre que nous verrons pour ce fichier est le lien ("*lxc.network.link*") ; ce paramètre va permettre d'indiquer à quel bridge nous voulons connecter notre interface.

### 3.3.2.2 Bridges

Les bridges (ou pont en français) sont des équipements réseaux qui permettent de relier deux (ou plus) interfaces de manière complètement transparente : en observant les paquets qui transitent, le pont peut connaître les adresses mac des interfaces, et ainsi, rediriger les paquets. Les ponts peuvent par exemple, être utilisés pour rediriger une connexion Ethernet : une machine se connecte en Ethernet, une seconde machine se connecte à la première, elles établissent un pont entre deux de leurs interfaces (une interface de la première machine, et une interface de la deuxième machine), et, ainsi, la deuxième machine peut avoir accès à internet.

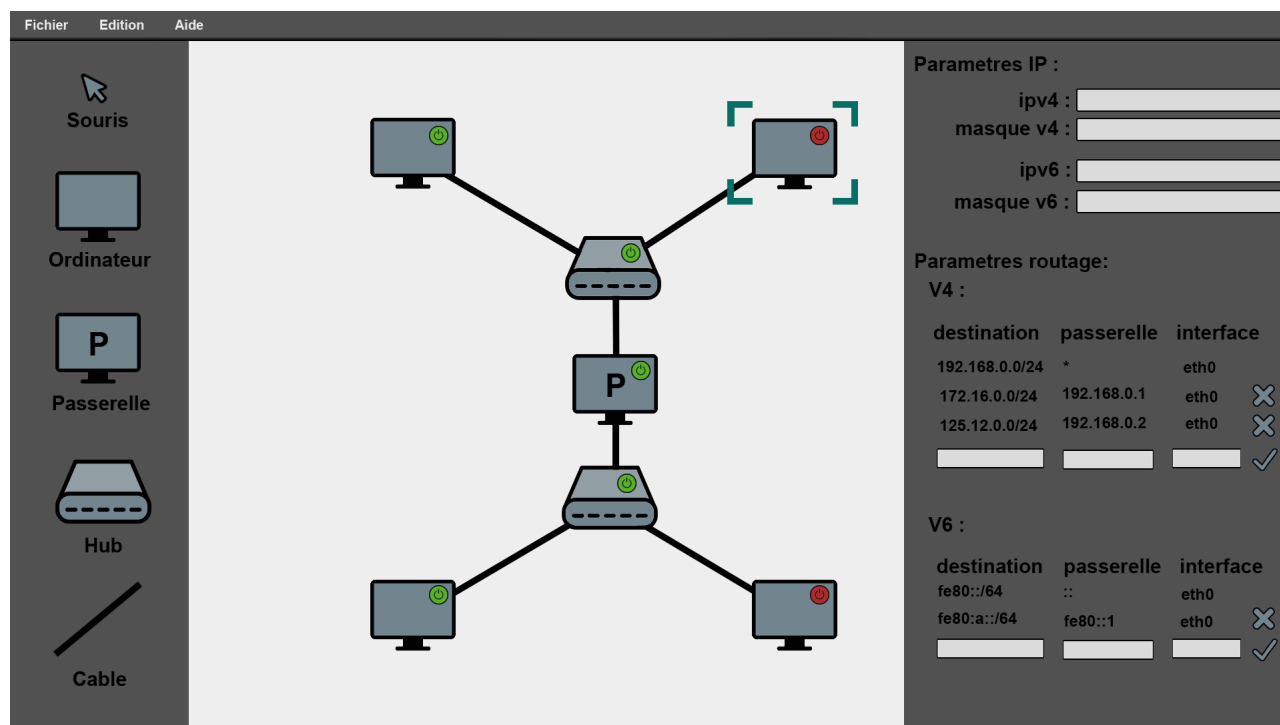
Lorsqu'un VE se lance, une interface se crée sur la machine hôte pour chaque interface présente sur le VE. Ces interfaces créées ont un nom qui commence toujours par "*VETH*" suivi de quatre caractères. Ces interfaces sur la machine hôte représentent les interfaces de l'environnement virtuel, et nous permettent de relier les environnements entre eux ; il est possible de voir la correspondance entre les interfaces d'un container et les interfaces de la machine hôte grâce à la commande "*lxc-info ...*".

Il est donc nécessaire de relier ces interfaces sur la machine hôte à un même bridge pour que les containers puissent communiquer !

## 4 Architecture du programme

### 4.1 Vues

Le logiciel a une seule interface :



Elle est divisible en 4 parties :

- La barre supérieure (Fichier, Éditions, Aide...),
  - la barre de sélection (à gauche) où on peut choisir quel entité ajouter,
  - l'écran principal (au centre),
  - la barre d'information (à droite) où sont affichées les configurations des entités sélectionnées.
- Voici un exemple avec un ping entre deux machines capturé avec tcpdump :

**NetSmooth**  
Fichier Edition Aide

**Souris**  
**Ordinateur**  
**Cable**

**xterm**  
[root - cnt2 ~/] \$ ifconfig eth0  
eth0 Link encap:Ethernet HWaddr 00:80:F7:32:B3:1E  
inet addr:172.16.0.15 Bcast:172.16.0.255 Mask:255.255.255.0  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:864 errors:0 dropped:0 overruns:0 frame:0  
TX packets:654 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0  
[root - cnt2 ~/] \$ route  
Kernel IP routing table  

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
172.16.0.0	*	255.255.255.0	U	0	0	0	eth0
192.168.0.0	172.16.0.1	255.255.255.0	U	0	0	0	eth0

[root - cnt2 ~/] \$ tcpdump  
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode  
15:34:02.625435 IP 192.168.0.15 > 172.16.0.15: ICMP echo request, id 55304, seq 168, length 64  
15:34:02.625726 IP 172.16.0.15 > 192.168.0.15: ICMP echo reply, id 55304, seq 168, length 64  
15:34:03.625458 IP 192.168.0.15 > 172.16.0.15: ICMP echo request, id 55304, seq 169, length 64  
15:34:03.625254 IP 172.16.0.15 > 192.168.0.15: ICMP echo reply, id 55304, seq 169, length 64  
15:34:02.625458 IP 192.168.0.15 > 172.16.0.15: ICMP echo request, id 55304, seq 170, length 64  
15:34:02.625954 IP 172.16.0.15 > 192.168.0.15: ICMP echo reply, id 55304, seq 170, length 64

**xterm**  
[root - cnt1 ~/] \$ ifconfig eth0  
eth0 Link encap:Ethernet HWaddr 00:80:C8:32:C8:1E  
inet addr:192.168.0.15 Bcast:192.168.0.255 Mask:255.255.255.0  
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
RX packets:864 errors:0 dropped:0 overruns:0 frame:0  
TX packets:654 errors:0 dropped:0 overruns:0 carrier:0  
collisions:0  
[root - cnt1 ~/] \$ route  
Kernel IP routing table  

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
192.168.0.0	*	255.255.255.0	U	0	0	0	eth0
172.16.0.0	192.168.0.1	255.255.255.0	U	0	0	0	eth0

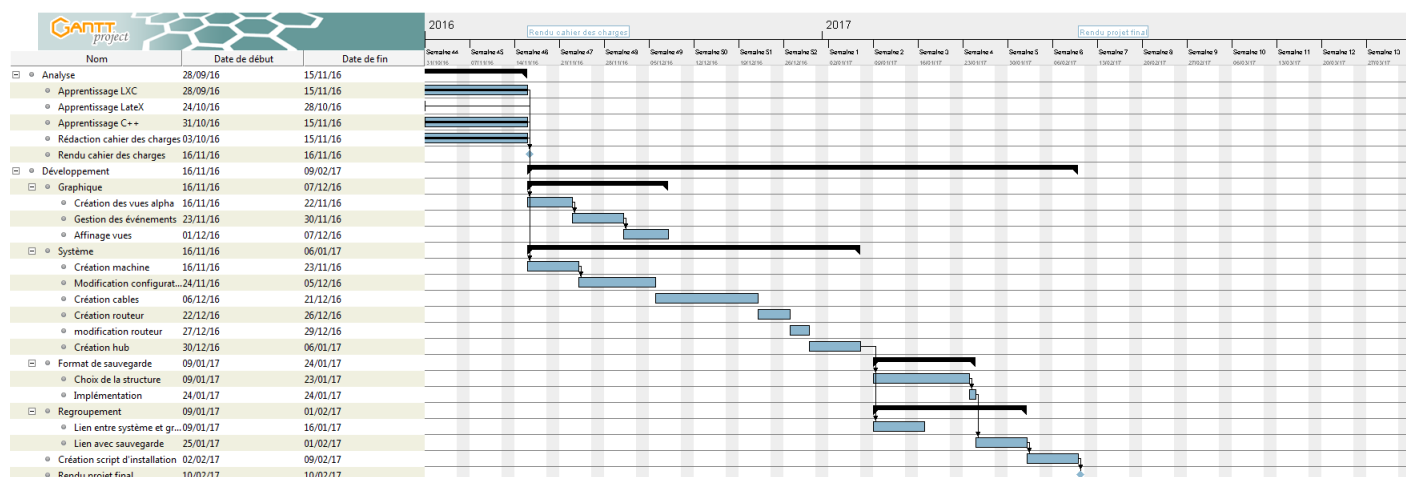
[root - cnt1 ~/] \$ ping 172.16.0.15  
PING 172.16.0.15 (172.16.0.15) 56(84) bytes of data  
64 bytes from 172.16.0.15: icmp\_seq=1 ttl=63 time=0.129 ms  
64 bytes from 172.16.0.15: icmp\_seq=2 ttl=63 time=0.101 ms  
64 bytes from 172.16.0.15: icmp\_seq=3 ttl=63 time=0.066 ms

**V6 :**  

destination	passerelle	interface
fe80::/64	::	eth0
fe80:a::/64	fe80::1	eth0

## 5 Etapes prévues

Voici un diagramme de Gantt qui décrit les étapes et l'avancement actuel du projet :



## 6 Notice d'utilisation de LXC

### 6.1 Bien débiter

Toutes les commandes exécutées sont effectuées en root (super-utilisateur).

#### 6.1.1 Installer LXC

**6.1.1.1 Arch-linux** `#pacman -S lxc arch-install-scripts`

**6.1.1.2 Debian** `#apt-get install lxc`

#### 6.1.2 Créer un container

`#lxc-create -t download -n name` : créer un container de nom *name* en proposant la liste des images d'OS possibles à télécharger.

`#lxc-create -t download -n name -d debian -r jessie -a i386` : créer un container de nom *name* en téléchargeant une image de distribution *debian*, de release *jessie* et d'architecture *i386* (32 bits).<sup>1</sup>

`#lxc-ls` : obtenir la liste des containers créés –fancy pour plus de détails.

#### 6.1.3 Démarrer un container

`#lxc-start -n name -d` : démarrer le container de nom *name* en daemon (*-d*).  
Par défaut, aucun compte utilisateur n'est créé. Il faut donc se connecter en root.

1. Créer un container d'architecture 64 bits sur un host 32 bits **ne fonctionne pas**.

*#lxc-attach -n name* : se connecter en root sur le container *name*.

Une fois un compte utilisateur créé, et le container lancé, il est possible de se connecter a une session en particulier.

*#lxc-console -n name -t 0* : ouvrir un écran de login sur le terminal *tty0* du container *name*.<sup>2</sup>

Ou, si le conatiner est stoppé, il est possible de lancer le container sur un écran de login, en enlevant l'option *-d* a la commande *lxc-start*

### 6.1.4 Configurer un container

**6.1.4.1 Fichier de configuration** Le fichier de configuration d'un container *name* est */var/lib/lxc/name/config*. Voici un exemple de configuration de passerelle :

*RTFM* : *lxc.container.conf*

*/var/lib/lxc/passerelle/config*

```
# Distribution configuration
lxc.include = /usr/share/lxc/config/debian.common.conf
lxc.arch = x86_64
```

```
# Container specific configuration
lxc.rootfs = /var/lib/lxc/passerelle/rootfs
lxc.rootfs.backend = dir
lxc.utsname = passerelle
```

```
# Network configuration
lxc.network.type = veth
lxc.network.name = eth0
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 172.16.1.1
lxc.network.ipv6 = fec00 :0 :0 :2 : :1
```

```
lxc.network.type = veth
lxc.network.name = eth1
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 192.168.1.1
lxc.network.ipv6 = fc00 :0 :0 :1 : :1
```

Ici, le container possède deux interfaces (*eth0* et *eth1*) avec chacune leurs adresses *ipv4* (*lxc.network.ipv4*), *ipv6* (*lxc.network.ipv6*) et *MAC* (*lxc.network.hwaddr*).

*lxc.network.flags* indique quelle action effectuer (*up* active l'interface).

*lxc.network.type* indique quelle type de virtualisation de réseau utiliser (*RTFM*).

*lxc.network.link* indique l'interface à utiliser pour le vrai trafic, cette notion sera vu plus en détail par la

---

2. **BUG** : sur certaines distributions, *-t* différent de 0 ne fonctionne pas



suite.

#### 6.1.4.2 ifconfig, ip \*\*\*Commandes ip/ifconfig pour ipv4,ipv6\*\*\*

### 6.1.5 Relier les containers avec les ponts (ou bridges)

Pour relier les containers entre eux, ou même au host, LXC utilise les bridge (pont en français). Lors du lancement d'un container, l'activation de chacune de ses interfaces virtuelles va créer une interface sur la machine host. Relier ces interfaces à un même pont permet de relier les containers "physiquement".

**6.1.5.1 La méthode manuelle** Il est possible de relier les containers aux bridges apres les avoir lancer :

*#brctl addbr br0* : crée un bridge de nom *br0*

*#ifconfig br0 up* : active l'interface *br0*

*#brctl addif br0 VETH12345* : relie le bridge *br0* à l'interface physique *VETH12345*<sup>3</sup>

**6.1.5.2 La méthode automatique** Il est aussi possible de relier les containers par le fichier de configuration.<sup>4</sup> Attention, si vous essayez de lancer un container, qui dépend d'un bridge non créé, il y aura une erreur.

*#lxc.network.link* permet de préciser le pont auquel relier l'interface créée.

Exemple :

```
lxc.network.type = veth
lxc.network.name = eth1
lxc.network.link = lxcbr0
lxc.network.flags = up
lxc.network.hwaddr = 00 :16 :3e :5b :0e :8f
lxc.network.ipv4 = 192.168.1.1
lxc.network.ipv6 = fc00 :0 :0 :1 : :1
```

L'interface *eth1* sera reliée (indirectement) au pont *lxcbr0*.

## 7 Rapport de bug

---

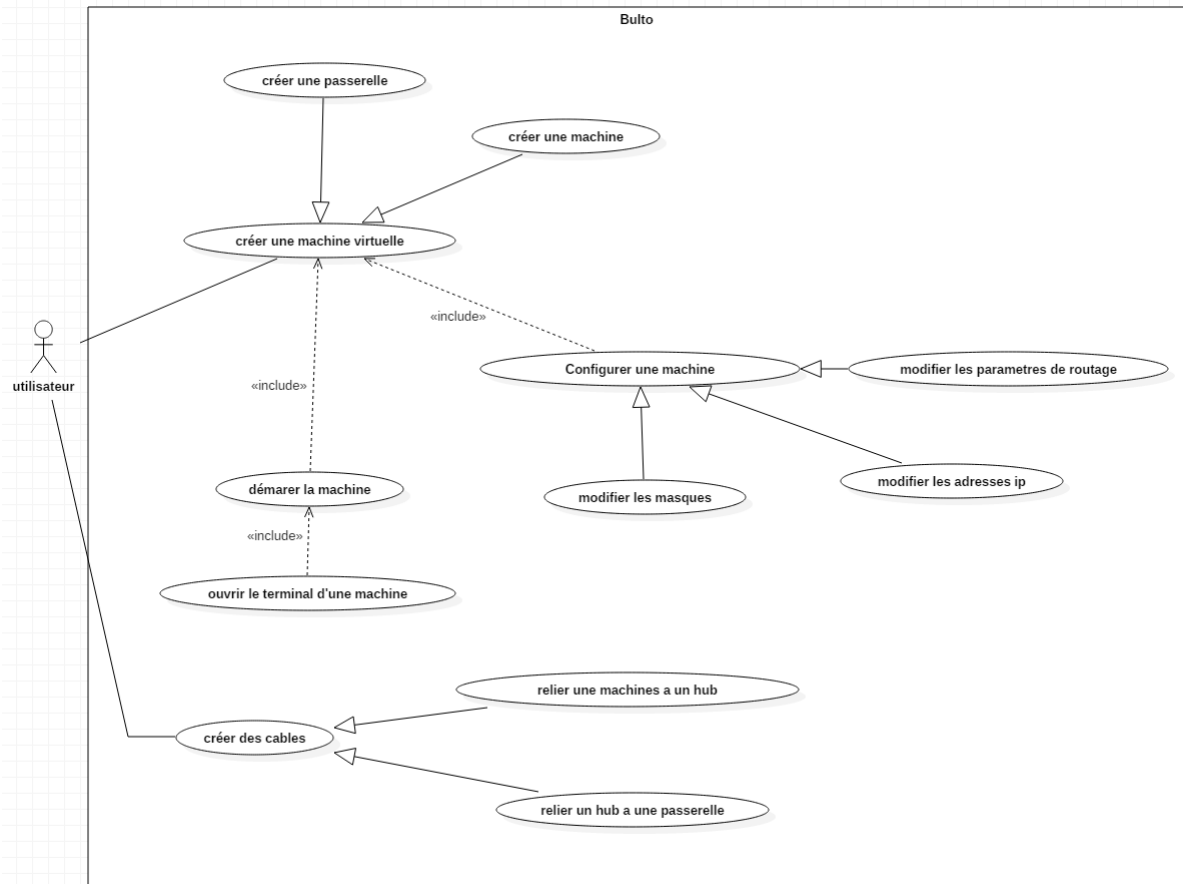
3. Voir l'autre doc pour plus de details sur le nom de l'interface

4. **Attention** : le pont ne peut pas être **créé** par le fichier de configuration. Il faut le créer manuellement.

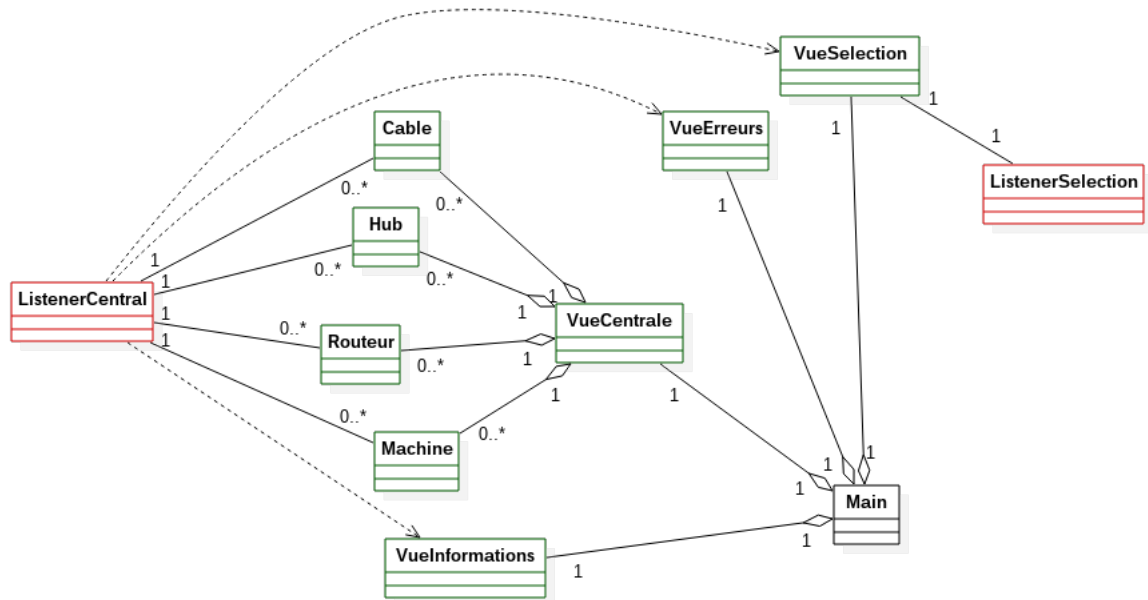
## 8 Annexe

### 8.1 Diagrammes

#### 8.1.1 Diagramme de cas d'utilisation :

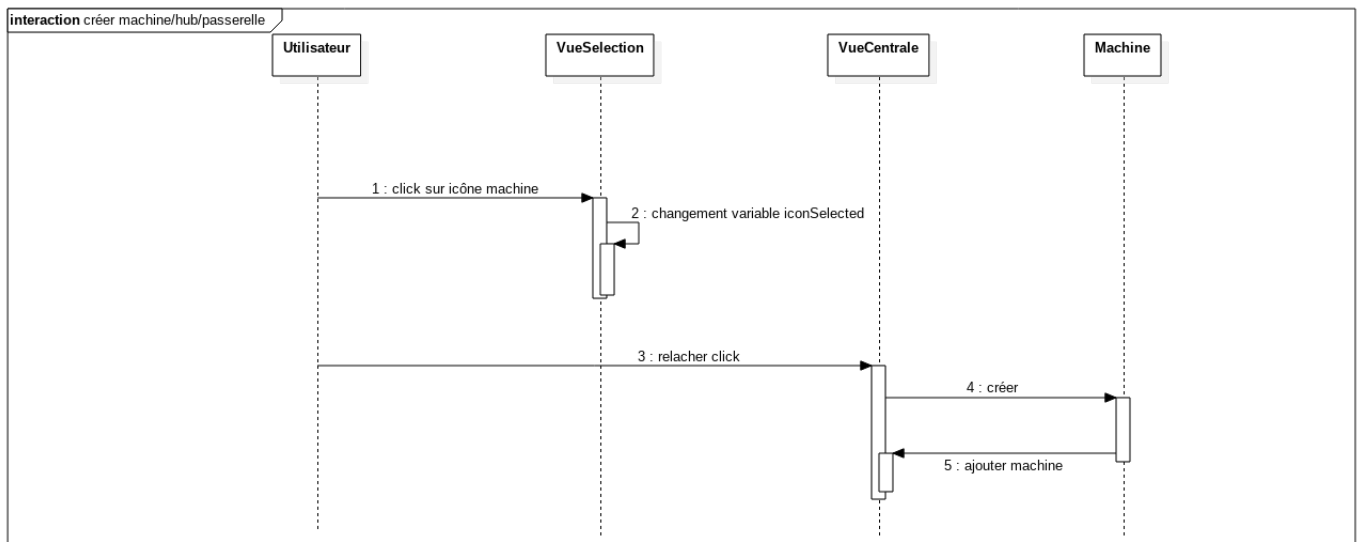


### 8.1.2 Diagramme de classe d'analyse

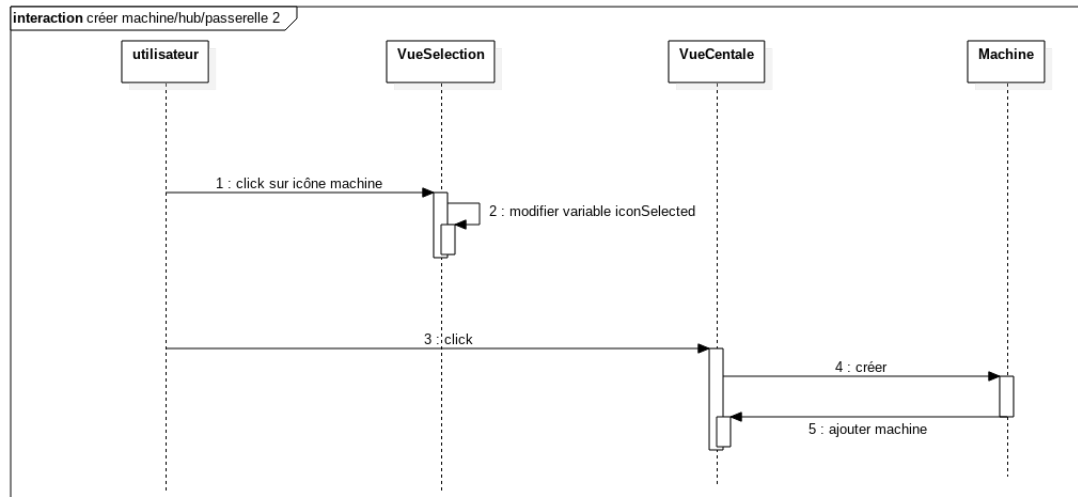


## 8.2 Diagrammes de séquence

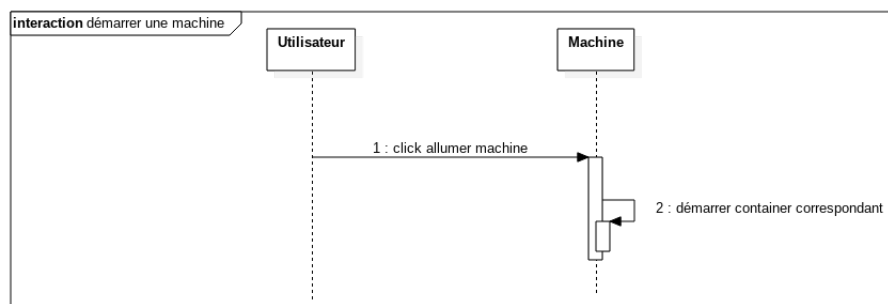
Créer une machine, un hub ou une passerelle - première version



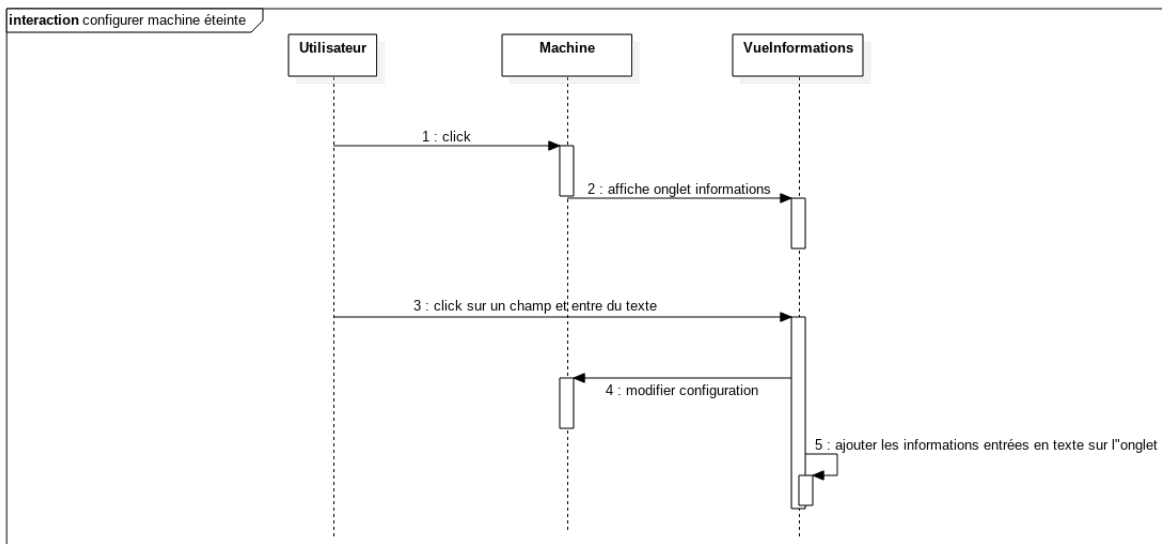
## Créer une machine, un hub ou une passerelle - deuxième version



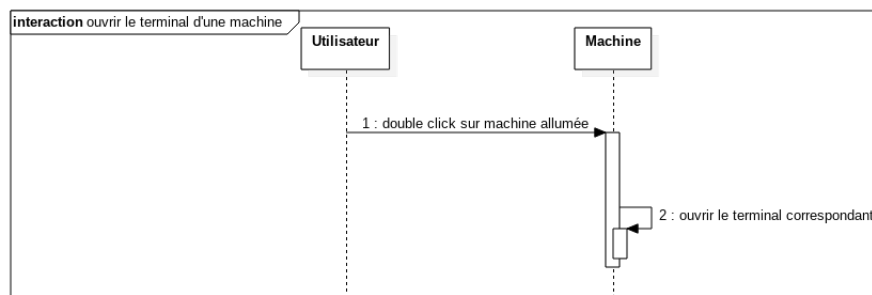
## Démarrer une machine ou une passerelle



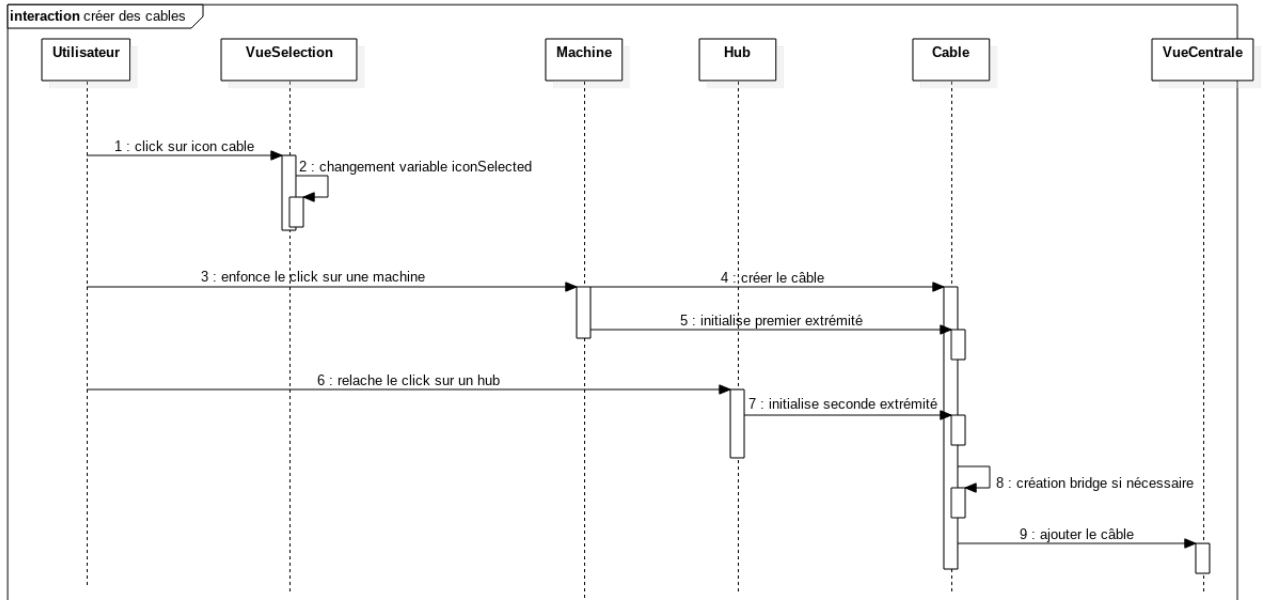
## Configurer une machine/passerelle éteinte



## Ouvrir terminal d'une machine allumée



## Créer un câble



## 8.3 Scénarios

### 8.3.1 Premier cas

**TITRE** : Créer une machine virtuelle.

**OBJECTIF** : Pouvoir créer une machine pour l'utiliser dans un réseau.

**ACTEURS** : L'utilisateur.

**TYPE** : Scénario nominale.

**PRECONDITIONS** : Il faut avoir lancé le logiciel.

**POSTCONDITIONS** : La machine est créée.

**DESCRIPTIF** :

1. L'utilisateur clique sur l'icône de la machine.
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.
3. L'utilisateur saisi l'adresse IP4 et/ou IP6 de la machine.
4. L'utilisateur saisi le masque 4 et/ou 6 de la machine.

**FLUX ALTERNATIFS** :

5. L'utilisateur veut définir des paramètres de routages.
  - 5a. L'utilisateur saisi l'adresse destination en IP4/IP6.
  - 5b. L'utilisateur saisi l'adresse de la passerelle en IP4/IP6.
  - 5c. L'utilisateur saisi l'interface qu'il veut utiliser.

**CAS REFERENCES** : aucun.

### 8.3.2 Deuxième cas

**TITRE** : Créer une passerelle.

**OBJECTIF** : Pouvoir créer une passerelle et l'utiliser dans le réseau.

**ACTEURS** : L'utilisateur.

**TYPE** : Scénario nominale.

**PRECONDITIONS** : Avoir lancé le logiciel.

**POSTCONDITIONS** : La passerelle est créée.

**DESCRIPTIF** :

1. L'utilisateur clique sur l'icône de la passerelle.
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.
3. L'utilisateur choisit le nombre d'interfaces qu'il veut.
4. L'utilisateur saisit l'adresse IP4 et/ou IP6 de la passerelle pour toutes ses interfaces.
5. L'utilisateur saisit le masque 4 et/ou 6 de la passerelle pour toutes ses interfaces.

**FLUX ALTERNATIFS** :

6. L'utilisateur veut définir des paramètres de routages.
  - 6a. L'utilisateur saisit l'adresse destination en IP4/IP6.
  - 6b. L'utilisateur saisit l'adresse de la passerelle en IP4/IP6.
  - 6c. L'utilisateur saisit l'interface qu'il veut utiliser.

**CAS REFERENCES** : aucun.

### 8.3.3 Troisième cas

**TITRE** : Créer un hub.

**OBJECTIF** : Pouvoir créer un hub pour l'utiliser dans un réseau.

**ACTEURS** : L'utilisateur.

**TYPE** : Scénario nominale.

**PRECONDITIONS** : Il faut avoir lancé le logiciel.

**POSTCONDITIONS** : Le hub est créé.

**DESCRIPTIF** :  
1. L'utilisateur clique sur l'icône du hub.  
2. L'utilisateur déplace l'icône sur la zone qui représente le réseau.

**CAS REFERENCES** : aucun.

### 8.3.4 Quatrième cas

**TITRE** : Relier une machine à une machine/hub/passerelle.

**OBJECTIF** : Pouvoir relier une machine avec un autre dispositif pour qu'il puisse se « voir » dans le réseau.

**ACTEURS** : L'utilisateur.

**TYPE** : Scénario nominale.

**PRECONDITIONS** : Il faut avoir lancé le logiciel, créée une machine et un autre dispositif.

**POSTCONDITIONS** : La machine est reliée avec l'autre dispositif.

**DESCRIPTIF** :

1. L'utilisateur clique sur l'icône du câble.
2. L'utilisateur déplace l'icône sur la machine à relier.
3. L'utilisateur clique sur le dispositif auquel il veut relier la machine.

**CAS REFERENCES** : aucun.

### 8.3.5 Cinquième cas

**TITRE** : Allumer un dispositif.

**OBJECTIF** : Pouvoir allumer un dispositif pour qu'il soit visible sur le réseau.

**ACTEURS** : L'utilisateur.

**TYPE** : Scénario nominale.

**PRECONDITIONS** : Il faut avoir lancé le logiciel, et créer un dispositif.

**POSTCONDITIONS** : Le dispositif est allumé.

**DESCRIPTIF** :

1. L'utilisateur clique sur l'icône on/off.

**CAS REFERENCES** : aucun.