

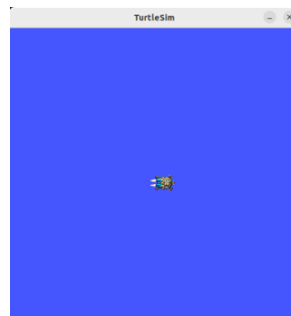
GE5
2023

Tutoriel ROS2

Partie 3

Auteurs :

Arthur BOUILLÉ (arthur.bouille@insa-strasbourg.fr)
Alexandre THOUVENIN (alexandre.thouvenin@insa-strasbourg.fr)



```
#export ROS_DOMAIN_ID=0  
export ROS_DOMAIN_ID=1
```

Table des matières

1	Introduction	3
2	Comprendre les domain_id avec TurtleSim	4
2.1	Essai sans spécification de domain_id	4
2.2	Essai avec spécification de domain_id différents	6
2.3	Essai avec spécification de domain_id identiques	8
3	Comprendre les domain_id avec TurtleSim et sa télécommande	10
3.1	Vérifier le domain_id des deux machines virtuelles	10
3.2	Lancer Turtlesim	11
3.3	Lancer la télécommande	11
4	Bibliographie	12

1 Introduction

Bienvenue dans ce tutoriel dédié à l'exploration du fonctionnement des **domain_id** de ROS 2 en utilisant TurtleSim et sa télécommande sur deux ordinateurs distincts. Dans le domaine de la robotique distribuée, la capacité à contrôler et à coordonner efficacement plusieurs robots simultanément est essentielle. ROS 2, le système d'exploitation pour robots de dernière génération, offre une fonctionnalité puissante appelée **domain_id** qui permet de regrouper les nœuds et les topics en environnements de communication distincts au sein du même réseau.

Dans ce tutoriel, nous allons non seulement explorer comment les **domain_id** facilitent la communication entre les nœuds de TurtleSim sur deux ordinateurs différents, mais aussi comment contrôler simultanément les mouvements de plusieurs tortues virtuelles à l'aide d'une télécommande partageant le même **domain_id**. Suivez les étapes détaillées de ce guide pour comprendre en profondeur comment exploiter cette fonctionnalité puissante de ROS 2 dans vos propres projets de robotique distribuée.

Voici un rappel du fonctionnement de ROS2 et une explication sur les **domain_id** :

- **ROS2** : ROS2 est une nouvelle version de ROS, comportant à peu près les mêmes composants, mais dont l'architecture change fondamentalement. ROS2, tout comme ROS, possède une architecture basée sur le principe de systèmes en temps réel distribués/répartis. Cela signifie que l'architecture est composée de nœuds qui se chargent d'effectuer des calculs ou des opérations simples, reliés entre eux par un réseau en temps réel.

La différence entre les deux vient du protocole réseau utilisé. ROS2 utilise un protocole réseau DDS (Data Distribution Service), qui a pour but de simplifier la programmation réseau. Chaque nœud créé avec ce protocole est à la fois un Publisher et un Subscriber (voir la partie suivante pour plus de détails sur ce que sont les Publishers et Subscribers), et est automatiquement créé sur une fraction du sous-réseau auquel ROS2 a accès.

Cette fraction est appelée domaine, et tous les nœuds d'un domaine sont connectés entre eux lors de leurs créations. Cela rend à la fois la connexion à un réseau ROS2 plus facile, et plus sécuritaire, car la simple création d'un nœud sur le réseau permet d'avoir accès à l'ensemble du réseau, mais il faut connaître à l'avance le sous-réseau ainsi que le domaine sur lequel le réseau est situé.

- **Domain_id** : Dans ROS 2 (Robot Operating System 2), les **domain_id** sont des identifiants utilisés pour grouper les nœuds et les topics en domaines distincts. Ces domaines sont essentiellement des environnements virtuels où les nœuds peuvent communiquer les uns avec les autres via des topics spécifiques à ce domaine.

Imaginons que vous ayez plusieurs robots dans un même réseau. Chacun de ces robots peut fonctionner dans son propre "domaine". Ainsi, les nœuds d'un robot ne communiqueront qu'avec les nœuds de leur propre domaine, à moins qu'ils ne soient explicitement configurés pour communiquer avec d'autres domaines.

En utilisant des **domain_id**, vous pouvez organiser votre réseau de robots en plusieurs groupes indépendants, ce qui facilite la gestion et la communication entre différents sous-systèmes.

En résumé, les **domain_id** dans ROS 2 sont des identifiants qui vous permettent de créer des environnements de communication virtuels distincts, ce qui simplifie la communication et la gestion des différents nœuds au sein de votre système robotique global.

2 Comprendre les domain_id avec TurtleSim

Dans cette partie, nous allons explorer comment les fonctionnalités des domain_id permettent d'observer les différents topics de TurtleSim, lancés sur une première machine virtuelle (VM1), depuis une seconde machine virtuelle (VM2). Cela nous permettra de comprendre comment ces identifiants facilitent la visualisation des données entre différents environnements virtuels, ouvrant la voie à une communication transparente entre les nœuds de différentes machines virtuelles.

2.1 Essai sans spécification de domain_id

1. Lancez vos deux machines virtuelles connectées au même réseau (Utilisez les ordinateurs d'une même salle de cours à l'INSA). *Rappel* : user : **insa** / pass : **insa**.
2. Observez la structure de la ligne de code à ajouter aux fichiers .bashrc afin de spécifier un certain domain_id :

```
export ROS_DOMAIN_ID=<ID_number>
```

3. Ouvrez un terminal dans les deux machines virtuelles et vérifiez que vous êtes dans le répertoire principal du terminal ~ / (tapez la commande `$ cd ~ /`)
4. Ouvrez le fichier .bashrc avec l'éditeur de code nano dans les deux VM (*Rappel* : tapez la commande `$ sudo nano .bashrc`)
5. Descendez à la fin du fichier .bashrc et vérifiez qu'aucune ligne de code correspondant à la spécification d'un domaine_id n'est écrite. La fin de votre fichier .bashrc devrait ressembler à cela :

```

GNU nano 6.2 .bashrc *
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

source /opt/ros/humble/setup.bash
  
```

6. Enregistrez le fichier sur les deux machines virtuelles. Fermez tous les terminaux des deux machines virtuelles pour les mettre à jour
7. Choisissez la machine virtuelle qui sera par la suite la VM1 et l'autre la VM2.
8. Visualisez les topics actifs visibles par chacune des machines virtuelles. (*Rappel* : tapez, dans les 2 VM, la commande `$ ros2 topic list`). Vous devrez obtenir la même liste sur les deux machines virtuelles :

Pour la VM1 :

```

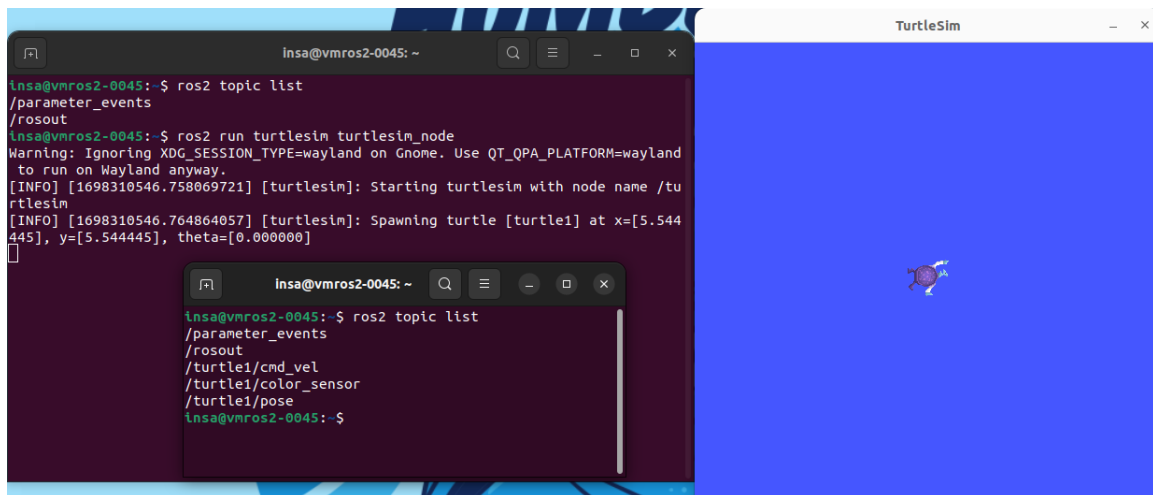
insa@vmros2-0045:~$ ros2 topic list
/parameter_events
/rosout
  
```

Pour la VM2 :

```
insa@vmros2-0044: ~  
insa@vmros2-0044:~$ ros2 topic list  
/parameter_events  
/rosout
```

Ces topics correspondent juste aux topics de fonctionnement de ROS2.

9. Lancez, maintenant, TurtleSim dans la VM1. Visualisez ensuite la liste des topics actifs sur cette même VM (VM1). Vous devriez obtenir :



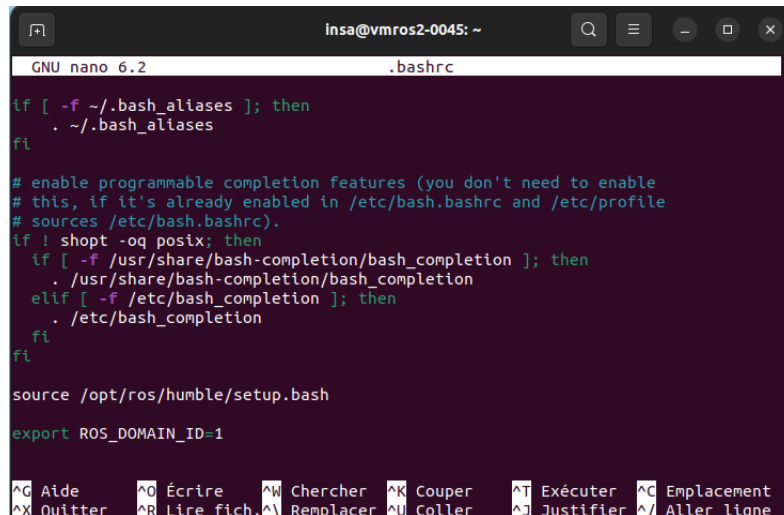
10. Visualisez ensuite la liste des topics actifs sur la VM2. Vous pouvez constater que vous arrivez à voir les topics de TurtleSim sur la VM2 :

```
insa@vmros2-0044: ~  
insa@vmros2-0044:~$ ros2 topic list  
/parameter_events  
/rosout  
/turtle1/cmd_vel  
/turtle1/color_sensor  
/turtle1/pose
```

Cela vient directement du fait que lorsque vous ne spécifiez aucun domain_id, le domain_id par défaut est le 0. Les deux VM étant sur le domain_id par défaut, elles sont donc sûr le même domaine et peuvent donc communiquer. Ce qui implique que tous les noeuds des deux VM peuvent se "voir".

2.2 Essai avec spécification de domain_id différents

1. Ouvrez un terminal dans les deux machines virtuelles et vérifiez que vous êtes dans le répertoire principal du terminal `~ /` (tapez la commande `$ cd ~ /`)
2. Ouvrez le fichier `.bashrc` avec l'éditeur de code nano (*Rappel* : tapez la commande `$ sudo nano .bashrc`)
3. Descendez à la fin du fichier `.bashrc` et ajoutez la ligne de code correspondant à la spécification d'un `domain_id`. Choisissez par exemple `ID=1` pour la VM1 et `ID=2` pour la VM2.
Vous devriez avoir cela pour la VM1 :



```

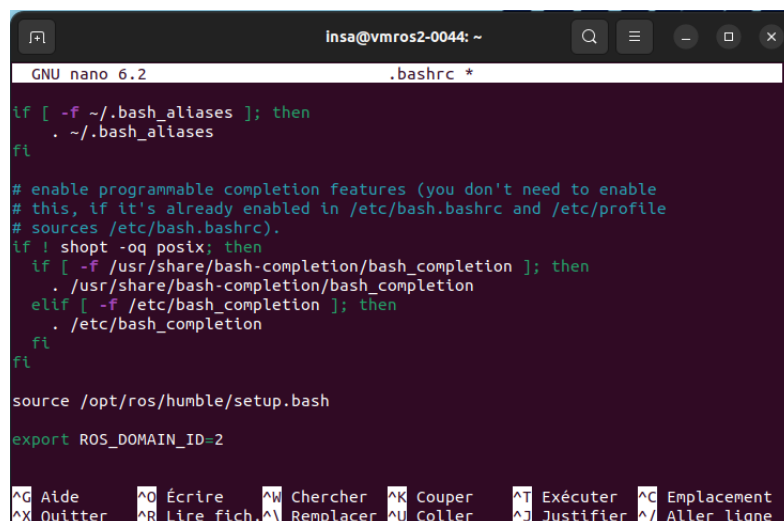
GNU nano 6.2 .bashrc
if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=1
  
```

Vous devriez avoir cela pour la VM2 :



```

GNU nano 6.2 .bashrc *
if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=2
  
```

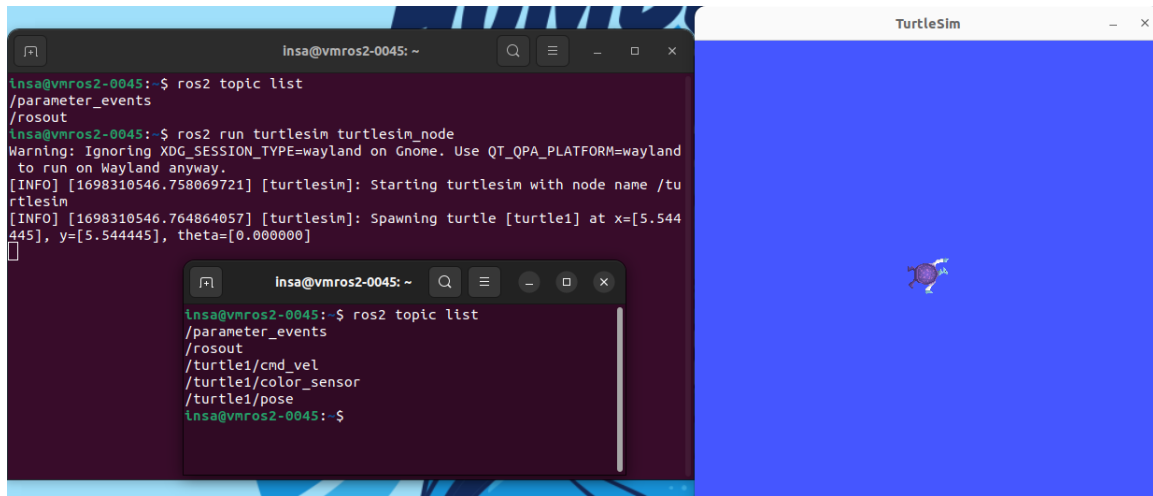
4. Enregistrez le fichier sur les deux machines virtuelles. Fermez tous les terminaux des deux machines virtuelles pour les mettre à jour.
5. Visualisez les topics actifs visibles par chacune des machines virtuelles. (*Rappel* : tapez, dans les 2 VM, la commande `$ ros2 topic list`). Vous devriez obtenir la même liste sur les deux machines virtuelles :

```

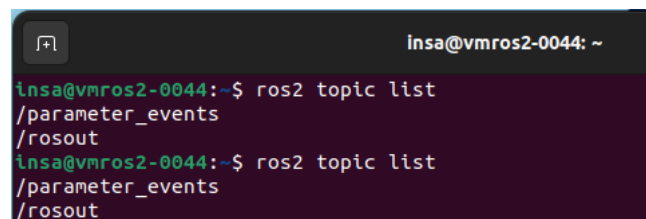
insa@ROS2:~$ ros2 topic list
/parameter_events
/rosout
  
```

Ces topics correspondent juste aux topics de fonctionnement de ROS2.

6. Lancez, maintenant, TurtleSim dans la VM1. Visualisez ensuite la liste des topics actifs sur cette même VM (VM1). Vous devriez obtenir :



7. Visualisez ensuite la liste des topics actifs sur la VM2. Vous pouvez constater que vous n'arrivez pas à voir les topics de TurtleSim sur la VM2 :

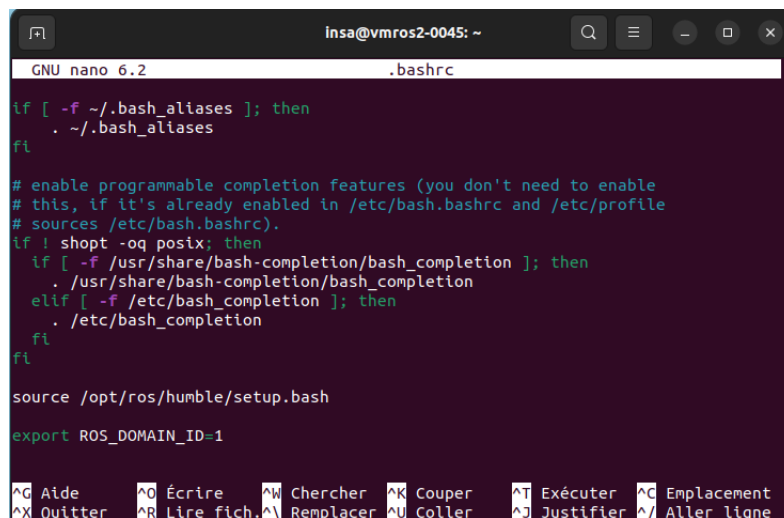


Cela vient directement du fait que lorsque vous spécifiez un `domain_id` différents pour deux systèmes, les noeuds de ces systèmes ne peuvent pas communiquer car il ne seront pas sûr le même sous-réseaux et ne peuvent donc pas se détecter (la VM1 est sur le sous réseau d'ID 1 et la VM2 est sur le sous réseau d'ID 2).

2.3 Essai avec spécification de domain_id identiques

1. Ouvrez un terminal dans les deux machines virtuelles et vérifiez que vous êtes dans le répertoire principal du terminal `~ /` (tapez la commande `$ cd ~ /`)
2. Ouvrez le fichier `.bashrc` avec l'éditeur de code nano (*Rappel* : tapez la commande `$ sudo nano .bashrc`)
3. Descendez à la fin du fichier `.bashrc` et ajoutez la ligne de code correspondant à la spécification d'un `domain_id`. Choisissez par exemple le même ID (`ID=1`) pour la VM1 et aussi pour la VM2.

Vous devriez avoir cela pour la VM1 :



```

GNU nano 6.2 .bashrc

if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

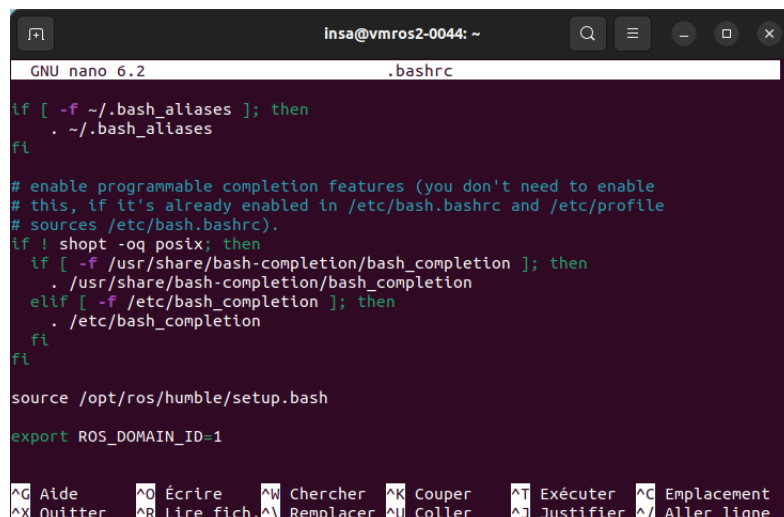
source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=1

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^M Remplacer  ^U Coller    ^J Justifier ^_ Aller ligne

```

Vous devriez avoir cela pour la VM2 :



```

GNU nano 6.2 .bashrc

if [ -f ~/.bash_aliases ]; then
  . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=1

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^M Remplacer  ^U Coller    ^J Justifier ^_ Aller ligne

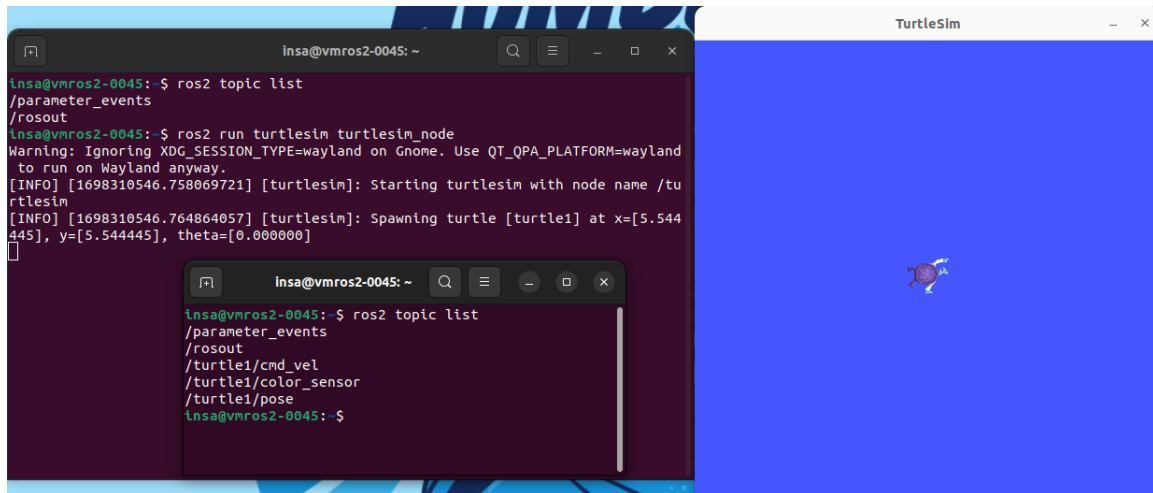
```

4. Enregistrez le fichier sur les deux machines virtuelles. Fermez tous les terminaux des deux machines virtuelles pour les mettre à jour
5. Visualisez les topics actifs visibles par chacune des machines virtuelles. (*Rappel* : tapez, dans les 2 VM, la commande `$ ros2 topic list`). Vous devriez obtenir la même liste sur les deux machines virtuelles :


```
insa@ROS2:~$ ros2 topic list
/parameter_events
/rosout
```

Ces topics correspondent juste aux topics de fonctionnement de ROS2.

6. Lancez, maintenant, TurtleSim dans la VM1. Visualisez ensuite la liste des topics actifs sur cette même VM (VM1). Vous devriez obtenir :



7. Visualisez ensuite la liste des topics actifs sur la VM2. Vous pouvez constater que vous arrivez de nouveau à voir les topics de TurtleSim sur la VM2 :

```
insa@vmros2-0044: ~
insa@vmros2-0044:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Cela vient directement du fait que lorsque vous spécifiez un domain_id identiques pour deux systèmes, les noeuds de ces systèmes peuvent communiquer car ils sont sûr le même sous-réseaux et peuvent donc se détecter (Les deux VM sont sûr le même sous-réseau ID=1).

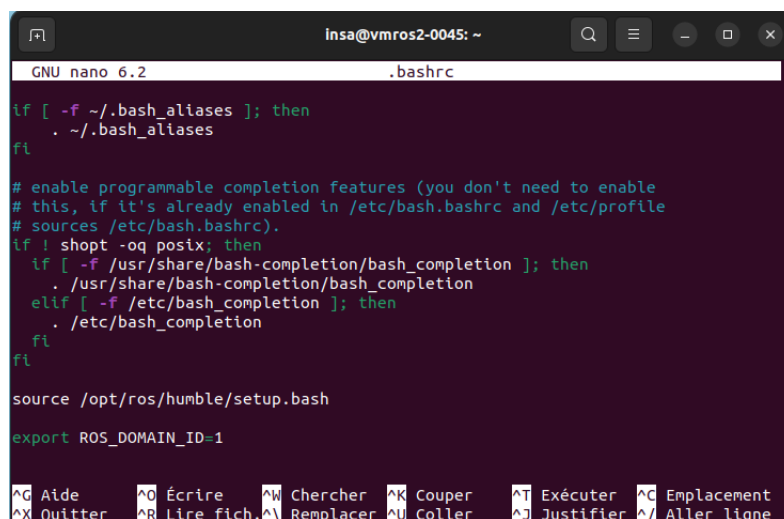
3 Comprendre les domain_id avec TurtleSim et sa télécommande

Dans cette partie vous allez voir qu'il est possible de faire bouger la tortue "/turtle1" de TurtleSim sur la VM1, avec la télécommande "teleop_key" sur la VM2.

3.1 Vérifier le domain_id des deux machines virtuelles

1. Ouvrez un terminal dans les deux machines virtuelles et vérifiez que vous êtes dans le répertoire principal du terminal `~ /` (tapez la commande `$ cd ~ /`)
2. Ouvrez le fichier `.bashrc` avec l'éditeur de code nano (*Rappel* : tapez la commande `$ sudo nano .bashrc`)
3. Descendez à la fin du fichier `.bashrc` et ajoutez la ligne de code correspondant à la spécification d'un `domain_id`. Choisissez par exemple le même ID (`ID=1`) pour la VM1 et aussi pour la VM2.

Vous devriez avoir cela pour la VM1 :



```

GNU nano 6.2 .bashrc

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

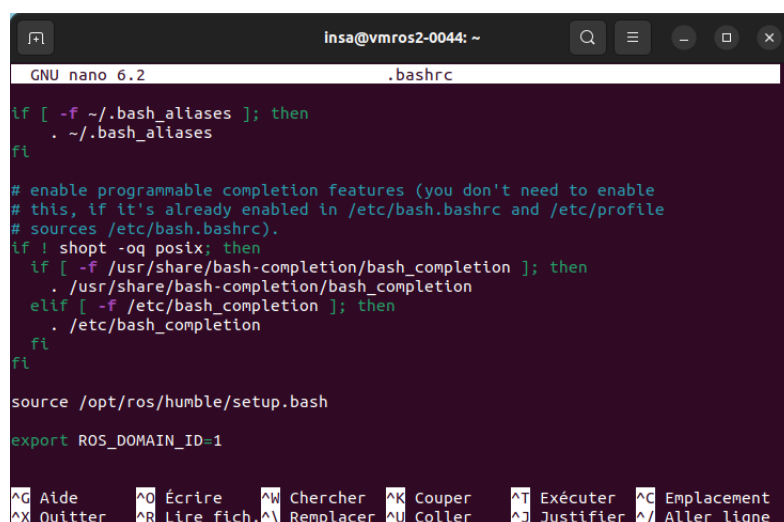
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=1

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^M Remplacer  ^U Coller    ^J Justifier ^_ Aller ligne
  
```

Vous devriez avoir cela pour la VM2 :



```

GNU nano 6.2 .bashrc

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; then
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

source /opt/ros/humble/setup.bash

export ROS_DOMAIN_ID=1

^G Aide      ^O Écrire    ^W Chercher  ^K Couper    ^T Exécuter  ^C Emplacement
^X Quitter   ^R Lire fich.^M Remplacer  ^U Coller    ^J Justifier ^_ Aller ligne
  
```

4. Enregistrez le fichier sur les deux machines virtuelles. Fermez tous les terminaux des deux machines virtuelles pour les mettre à jour

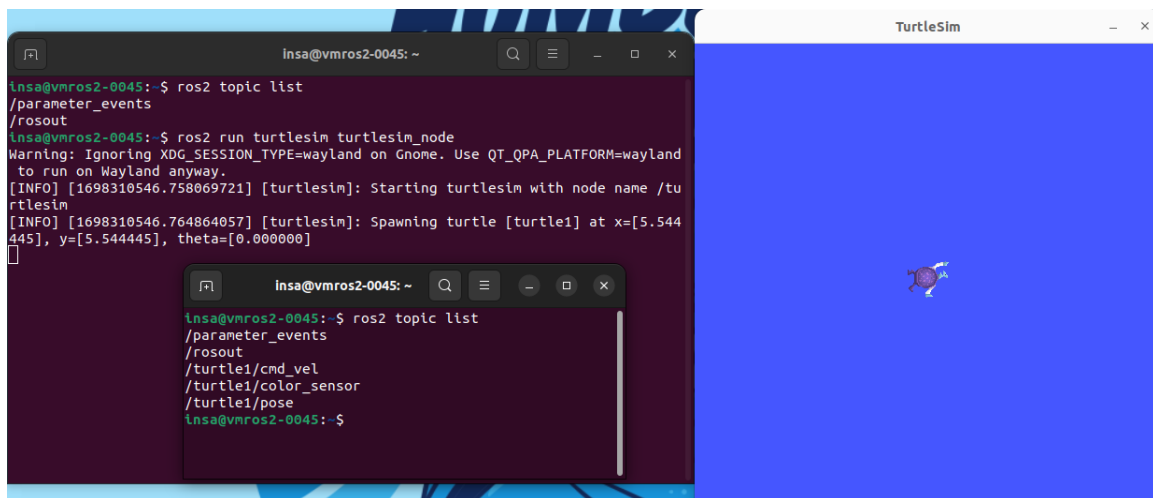
- Visualisez les topics actifs visibles par chacune des machines virtuelles. (*Rappel* : tapez, dans les 2 VM, la commande `$ ros2 topic list`). Vous devriez obtenir la même liste sur les deux machines virtuelles :

```
insa@ROS2:~$ ros2 topic list
/parameter_events
/rosout
```

Ces topics correspondent juste aux topics de fonctionnement de ROS2.

3.2 Lancer TurtleSim

- Lancez, maintenant, TurtleSim dans la VM1. Visualisez ensuite la liste des topics actifs sur cette même VM (VM1). Vous devriez obtenir :



- Visualisez ensuite la liste des topics actifs sur la VM2. Vous pouvez constater que vous arrivez de nouveau à voir les topics de TurtleSim sur la VM2 :

```
insa@vmros2-0044: ~
insa@vmros2-0044:~$ ros2 topic list
/parameter_events
/rosout
/turtle1/cmd_vel
/turtle1/color_sensor
/turtle1/pose
```

Cela vient directement du fait que lorsque vous spécifiez un domain_id identiques pour deux systèmes, les noeuds de ces systèmes peuvent communiquer car ils seront sûr le même sous-réseau et peuvent donc se détecter (Les deux VM sûr le sous-réseau ID=1).

3.3 Lancer la télécommande

- Ouvrez un second terminal, dans la machine virtuelle VM2, et lancez la télécommande "teleop_key" (*Rappel* : tapez la commande : `$ ros2 run turtlesim turtle_teleop_key`).
- Utilisez les flèches pour faire bouger la tortue "/turtle1". Vous pourrez voir la tortue bouger !
- Bonus** : Utiliser le code python de section 7 du "Tutoriel ROS2 : Partie 2" dans la VM2 pour faire bouger la tortue "/turtle1" dans la VM1

4 Bibliographie

- [ROS2 Documentation : TOPIC/SERVICES/ACTIONS](#),
- [ROS2 Design : ACTIONS](#)
- [Différences ROS1/ROS2](#),
- [ROS2 Documentation : Humble](#),
- Tutoriel ROS Noetic, Guillaume Hansen et Alexandre Thouvenin, 2023
- How to Control a Real TurtleBot with ROS through a Remote Raspberry Pi as ROS Master and with an OptiTrack Motion Capture System, Sylvain Durand, 2023
- [Robot Operating System](#), Olivier Stasse