Vulcanexus Documentation

Release 1.0.0

eProsima

Feb 28, 2023

INSTALLATION MANUAL

Vulcanexus is a *ROS 2* (Robot Operating System) all-in-one tool set. It allows users to build robotics applications combining the unique *Vulcanexus* elements with the *ROS 2* libraries, having *Fast DDS* as its fixed middleware implementation.

These open source elements include numerous features and tools, providing *Vulcanexus* users customizable solutions while improving overall system performance. With *Vulcanexus*, users have fast access to constantly improving functionalities, such as the latest *Fast DDS* version along with its new features.

Vulcanexus combinable elements are:

1. **VULCANEXUS-CORE**: a set of software libraries that enables users to build the most comprehensive and straightforward robotics application. It consists of eProsima *Fast DDS* and *ROS 2*.
2. **VULCANEXUS-TOOLS**: a set of features and applications which allows users to test, improve and configure the performance of *Vulcanexus* in their systems.
3. **VULCANEXUS-MICRO**: provides access for resource constrained devices (micro-controllers) to the DDS world, bridging the gap between them and *ROS 2*.
4. **VULCANEXUS-CLOUD**: scales and integrates *ROS 2* networks located in geographically spaced environments, and enables the deployment of DDS entities in the cloud in a quick and easy way.
5. **VULCANEXUS-SIMULATION**: enables users to design robotic simulations, providing an end-to-end development environment to model, program, and simulate robots.

Vulcanexus created a collection of downloadable packages that include useful combinations of the previously described elements with *ROS 2*:

The following documentation includes instructions for installing each *Vulcanexus* packages, some tutorials help users to get started, and the supported platforms and releases.

CHAPTER

ONE

LINUX BINARY INSTALLATION

Debian packages for Vulcanexus Galactic Gamble are currently available for Ubuntu Focal. Since Vulcanexus is a ROS 2 all-in-one tool set, certain ROS 2 prerequisites need to be met before installing.

1.1 ROS 2 prerequisites

First of all, set up a *UTF-8* locale as required by ROS 2. Locale settings can be checked and set up with the following commands:

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
# Any UTF-8 locale will work. Using en_US as an example
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

ROS 2 also requires that the Ubuntu Universe repository is enabled. This can be checked and enabled with the following commands:

```
apt-cache policy | grep universe

# This should print something similar to:
#
#  500 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 Packages
# release v=20.04,o=Ubuntu,a=focal,n=focal,l=Ubuntu,c=universe,b=amd64
#
# Otherwise run

sudo apt install software-properties-common
sudo add-apt-repository universe
```

Now download ROS 2 GPG key into the keystore.

```
sudo apt update && sudo apt install curl gnupg lsb-release
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/
↪share/keyrings/ros-archive-keyring.gpg
```

And finally add ROS 2 repository to the repository manager sources list.

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
↪keyring.gpg] http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo
↪$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

1.2 Setup Vulcanexus sources

Once all ROS 2 prerequisites have been met, it is time to start setting up Vulcanexus.

First, add the Qt 5.15 repository, required for the installation of several Fast DDS Monitor dependencies, running the commands:

```
sudo apt install software-properties-common
sudo add-apt-repository ppa:beineri/opt-qt-5.15.2-focal
```

Next, add Vulcanexus GPG key so apt can retrieve the packages:

```
sudo curl -sSL https://raw.githubusercontent.com/eProsima/vulcanexus/main/vulcanexus.key␣
↪-o /usr/share/keyrings/vulcanexus-archive-keyring.gpg
```

Finally, add the eProsima Vulcanexus repository to the repository manager sources list:

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/vulcanexus-
↪archive-keyring.gpg] http://repo.vulcanexus.org/debian $(source /etc/os-release &&␣
↪echo $UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/vulcanexus.list > /dev/
↪null
```

1.3 Install eProsima Vulcanexus packages

Remember to update the apt repository caches after setting up the repositories:

```
sudo apt update
```

Desktop install (Recommended): includes all the simulation tools, demos, and tutorials.

```
sudo apt install vulcanexus-galactic-desktop
```

Base Install: basic installation without simulation tools, demos, and tutorials.

```
sudo apt install vulcanexus-galactic-base
```

For other Vulcanexus packages, please refer to the *Introduction* section for more information.

1.4 Environment setup

In order to use the Vulcanexus installation, the environment must be set up sourcing the following file:

```
source /opt/vulcanexus/galactic/setup.bash
```

1.5 Uninstall eProsima Vulcanexus packages

To uninstall Vulcanexus, it is enough to run the following command :

```
sudo apt autoremove vulcanexus-galactic-desktop
```

CHAPTER

TWO

LINUX INSTALLATION FROM SOURCES

This section explains how to build Vulcanexus in Ubuntu Focal. Since Vulcanexus is a ROS 2 all-in-one tool set, certain ROS 2 prerequisites need to be met before building.

2.1 ROS 2 prerequisites

First of all, set up a *UTF-8* locale as required by ROS 2. Locale settings can be checked and set up with the following commands:

```
locale  # check for UTF-8

sudo apt update && sudo apt install locales
# Any UTF-8 locale will work. Using en_US as an example
sudo locale-gen en_US en_US.UTF-8
sudo update-locale LC_ALL=en_US.UTF-8 LANG=en_US.UTF-8
export LANG=en_US.UTF-8
```

ROS 2 also requires that the Ubuntu Universe repository is enabled. This can be checked and enabled with the following commands:

```
apt-cache policy | grep universe

# This should print something similar to:
#
#  500 http://us.archive.ubuntu.com/ubuntu focal/universe amd64 Packages
# release v=20.04,o=Ubuntu,a=focal,n=focal,l=Ubuntu,c=universe,b=amd64
#
# Otherwise run

sudo apt install software-properties-common
sudo add-apt-repository universe
```

Now download ROS 2 GPG key into the keystore.

```
sudo apt update && sudo apt install curl gnupg lsb-release
sudo curl -sSL https://raw.githubusercontent.com/ros/rosdistro/master/ros.key -o /usr/
↪share/keyrings/ros-archive-keyring.gpg
```

And then add ROS 2 repository to the repository manager sources list.

```
echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/ros-archive-
↪keyring.gpg] http://packages.ros.org/ros2/ubuntu $(source /etc/os-release && echo
↪$UBUNTU_CODENAME) main" | sudo tee /etc/apt/sources.list.d/ros2.list > /dev/null
```

With the ROS 2 repository properly set up the next step is to install the required dependencies and tools for cloning and testing the ROS 2 packages within the workspace.

```
sudo apt update && sudo apt install -y \
build-essential \
cmake \
git \
python3-colcon-common-extensions \
python3-flake8 \
python3-pip \
python3-pytest-cov \
python3-rosdep \
python3-setuptools \
python3-vcstool \
wget
# install some pip packages needed for testing
python3 -m pip install -U \
flake8-blind-except \
flake8-builtins \
flake8-class-newline \
flake8-comprehensions \
flake8-deprecated \
flake8-docstrings \
flake8-import-order \
flake8-quotes \
pytest-repeat \
pytest-rerunfailures \
pytest \
setuptools
```

2.2 Get ROS 2 code

Create a workspace for Vulcanexus and clone the ROS 2 repositories

```
mkdir -p ~/vulcanexus_galactic/src
cd ~/vulcanexus_galactic
wget https://raw.githubusercontent.com/ros2/ros2/galactic/ros2.repos
vcs import src < ros2.repos
```

Now download the required dependencies for these packages.

```
sudo rosdep init
rosdep update
rosdep install --from-paths src --ignore-src -y --skip-keys "fastcdr rti-connext-dds-5.3.
↪1 urdfdom_headers"
```

2.3 Get Vulcanexus code

Add the Vulcanexus repositories and metadata files to the Vulcanexus workspace:

```
cd ~
cd vulcanexus_galactic
wget https://raw.githubusercontent.com/eProsima/vulcanexus/galactic/vulcanexus.repos
wget https://raw.githubusercontent.com/eProsima/vulcanexus/galactic/colcon.meta
vcs import --force src < vulcanexus.repos
```

2.4 Install Vulcanexus dependencies

Some additional dependencies which are required for the Vulcanexus distribution must be installed. Start by adding the Qt 5.15 repository required for the installation of several Fast DDS Monitor dependencies:

```
sudo apt install -y software-properties-common
sudo add-apt-repository -y ppa:beineri/opt-qt-5.15.2-focal
```

Next, install the Vulcanexus required development tools:

```
sudo apt update && sudo apt install -y \
  libp11-dev \
  libengine-pkcs11-openssl \
  libyaml-cpp-dev \
  openjdk-8-jdk \
  qt5-default \
  qt5153d \
  qt515charts-no-lgpl \
  qt515graphicaleffects \
  qt515quickcontrols \
  qt515quickcontrols2 \
  qt515quicktimeline-no-lgpl \
  qt515svg \
  qt515tools \
  qt515translations \
  swig
```

2.5 Build the code in the workspace

If any other Vulcanexus or ROS 2 distribution has been installed from binaries, please ensure that the build is done in a fresh environment (previous installation is not sourced). This can be checked running the following command:

```
printenv | grep 'VULCANEXUS\|ROS'
```

The output should be empty. Please, be aware that in case the environment sourcing has been added to `.bashrc`, it must be removed in order to get a fresh environment.

2.5.1 Build Fast DDS-Gen (Optional)

Fast DDS-Gen is a Java application that generates source code using the data types defined in an IDL file. This tool must be built separately following the instructions below. Please, refer to Fast DDS-Gen documentation for more information about this tool.

```
cd src/eProsima/fastddsgen
./gradlew assemble
```

The generated Java application can be found in `share/fastddsgen`. However, the `scripts` folder provides some user friendly scripts that are recommended to be used. This scripts can be made accessible to the session adding the `scripts` folder path to the `PATH` environment variable.

```
export PATH=~/vulcanexus_galactic/src/eProsima/fastddsgen/scripts:$PATH
```

2.5.2 Build workspace

In order to build the workspace, the command line tool colcon is used. This tool is based on CMake and it is aimed at building sets of software packages, handling ordering and setting up the environment to use them.

```
cd ~/vulcanexus_galactic
colcon build
```

Important: In case that only a set of packages are going to be built, please ensure to include always `vulcanexus_base` package in the set. E.g.:

```
colcon build --packages-up-to demo_nodes_cpp vulcanexus_base
```

This auxiliary package is required to set several environment variables required by the distribution such as `VULCANEXUS_DISTRO` and `VULCANEXUS_HOME`.

2.6 Environment setup

In order to use the Vulcanexus installation, the environment must be set up sourcing the following file:

```
source ~/vulcanexus_galactic/install/setup.bash
```

CHAPTER THREE

DOCKER INSTALLATION

Vulcanexus offers the possibility of running from a containerized environment by providing a Docker image which contains *Vulcanexus*'s Desktop installation. This Docker image can be found in Vulcanexus's Downloads. To run it, first install Docker:

```
sudo apt install docker.io
```

And then load the image with:

```
docker load -i ubuntu-vulcanexus-galactic-desktop.tar
```

Vulcanexus Docker image can be run with:

GUI support

```
xhost local:root
docker run \
    -it \
    --privileged \
    -e DISPLAY=$DISPLAY \
    -v /tmp/.X11-unix:/tmp/.X11-unix \
    ubuntu-vulcanexus:galactic-desktop
```

CLI support

```
docker run -it ubuntu-vulcanexus:galactic-desktop
```

To run more than one session within the same container, *Vulcanexus* installation must be sourced. Given a running container, you can open another session by:

```
docker exec -it <running-container-id> bash
```

Then, within the container, source the *Vulcanexus* installation with:

```
source /opt/vulcanexus/galactic/setup.bash
```

To verify that the sourcing was correct, run:

```
echo $VULCANEXUS_HOME
```

The output should be:

```
/opt/vulcanexus/galactic
```

CHAPTER

FOUR

ROS 2 NETWORK STATISTICS USING VULCANEXUS TOOLS

Table of Contents

4.1 Background

Vulcanexus integrates eProsima Fast DDS Monitor, which is a useful tool for monitoring and studying a ROS 2 network as ROS 2 relies on the DDS specification to communicate the different nodes. The automatic discovery of entities in a local network enables to easily identify the different running Participants, their Endpoints, the Topics that each of them is using, and even the network interfaces they are employing to communicate with one another. Additionally, it is possible to receive statistical data from every endpoint in the network leveraging the Fast DDS Statistics Module. This data is very useful to analyze the DDS network performance and seek possible communication problems in it.

This tutorial provides step-by-step instructions to use Vulcanexus to monitor a ROS 2 talker/listener demo.

4.2 Prerequisites

Ensure that the Vulcanexus installation includes the Vulcanexus tools (either `vulcanexus-galactic-desktop`, `vulcanexus-galactic-tools`, or `vulcanexus-galactic-base`). Also, remember to source the environment in every terminal in this tutorial.

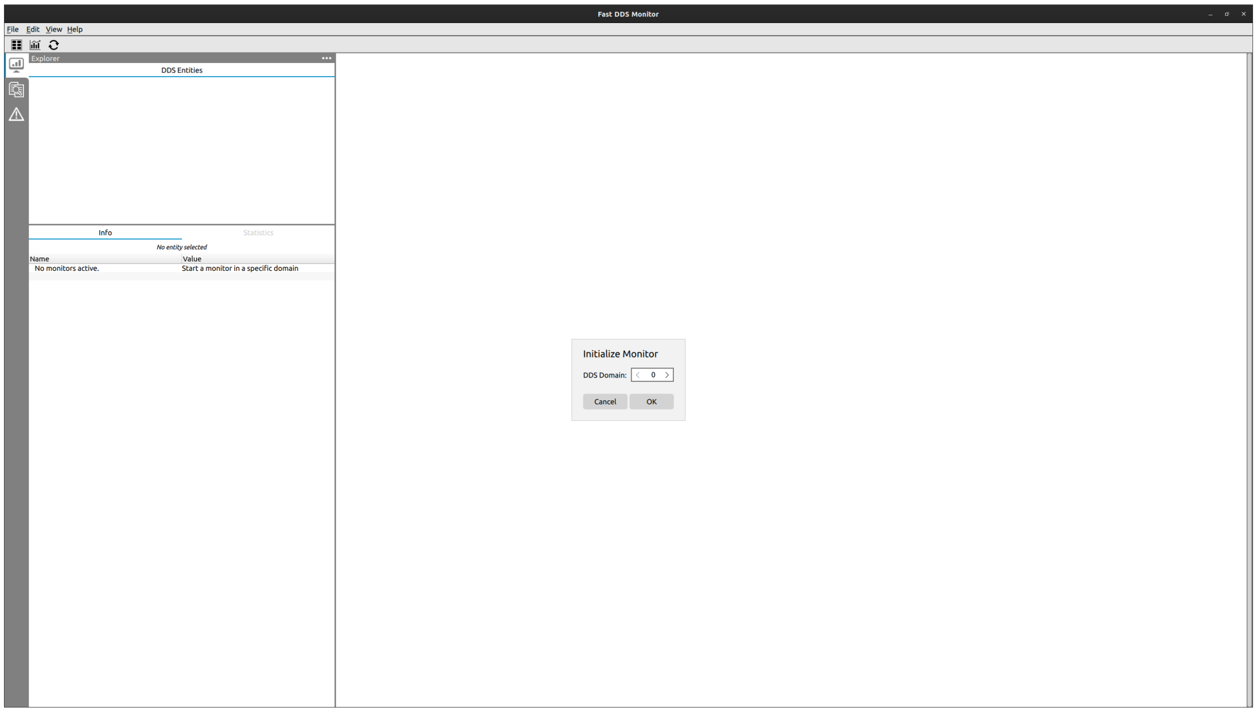
```
source /opt/vulcanexus/galactic/setup.bash
```

4.3 Launch Fast DDS Monitor

Initiate Fast DDS Monitor running the following command:

```
fastdds_monitor
```

Once Fast DDS Monitor is launched, start a monitor in domain `0` (default domain).



4.4 Execute ROS 2 demo nodes with statistics

In order to activate the publication of statistical data, eProsima Fast DDS requires an environment variable specifying which kinds of statistical data are to be reported. Consequently, before launching the ROS 2 nodes, remember to set `FASTDDS_STATISTICS` environment variable. Run the following commands in different terminals (remember to source the Vulcanexus environment):

```
export FASTDDS_STATISTICS="HISTORY_LATENCY_TOPIC;NETWORK_LATENCY_TOPIC;PUBLICATION_
↪THROUGHPUT_TOPIC;\
SUBSCRIPTION_THROUGHPUT_TOPIC;RTPS_SENT_TOPIC;RTPS_LOST_TOPIC;\
HEARTBEAT_COUNT_TOPIC;ACKNACK_COUNT_TOPIC;NACKFRAG_COUNT_TOPIC;\
GAP_COUNT_TOPIC;DATA_COUNT_TOPIC;RESENT_DATAS_TOPIC;SAMPLE_DATAS_TOPIC;\
PDP_PACKETS_TOPIC;EDP_PACKETS_TOPIC;DISCOVERY_TOPIC;PHYSICAL_DATA_TOPIC"

ros2 run demo_nodes_cpp listener
```

```
export FASTDDS_STATISTICS="HISTORY_LATENCY_TOPIC;NETWORK_LATENCY_TOPIC;PUBLICATION_
↪THROUGHPUT_TOPIC;\
SUBSCRIPTION_THROUGHPUT_TOPIC;RTPS_SENT_TOPIC;RTPS_LOST_TOPIC;\
HEARTBEAT_COUNT_TOPIC;ACKNACK_COUNT_TOPIC;NACKFRAG_COUNT_TOPIC;\
```

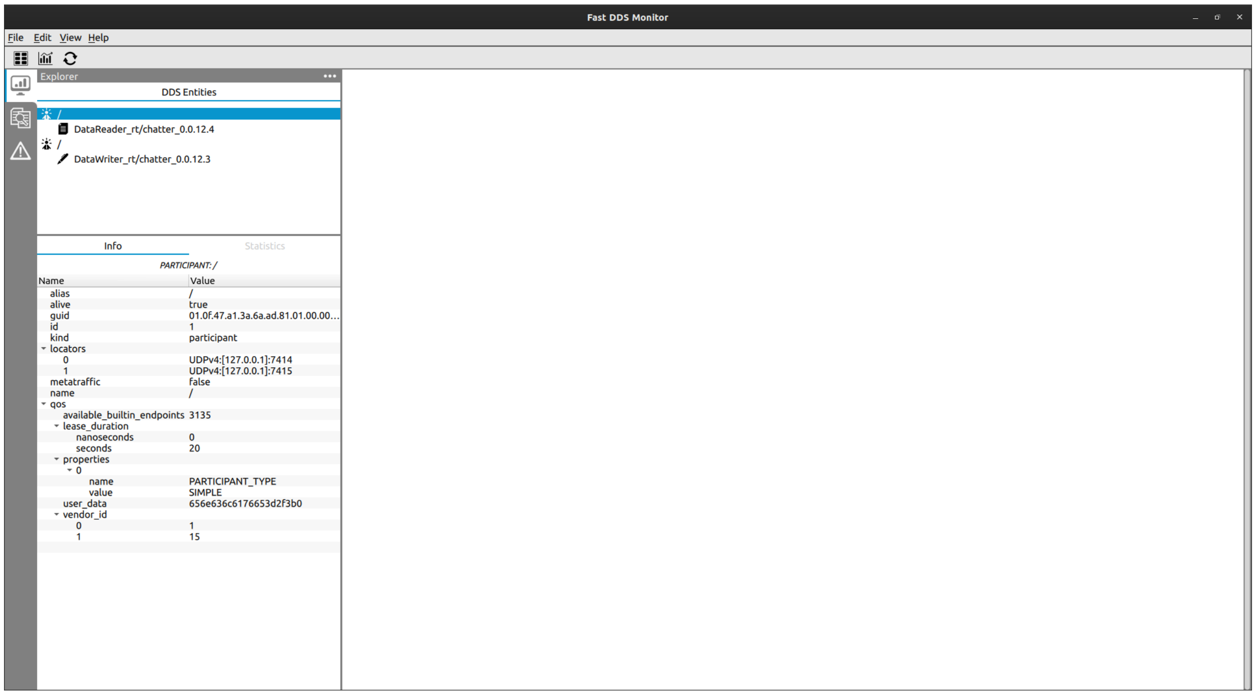
(continues on next page)

(continued from previous page)

```
GAP_COUNT_TOPIC;DATA_COUNT_TOPIC;RESENT_DATAS_TOPIC;SAMPLE_DATAS_TOPIC;\
PDP_PACKETS_TOPIC;EDP_PACKETS_TOPIC;DISCOVERY_TOPIC;PHYSICAL_DATA_TOPIC"

ros2 run demo_nodes_cpp talker
```

4.5 Monitoring network

Now, the two new Participants are visible in the *Fast DDS Monitor*'s DDS Panel.



```
/
    DataReader_rt/chatter_0.0.12.4
/
    DataWriter_rt/chatter_0.0.12.3
```

PARTICIPANT: /

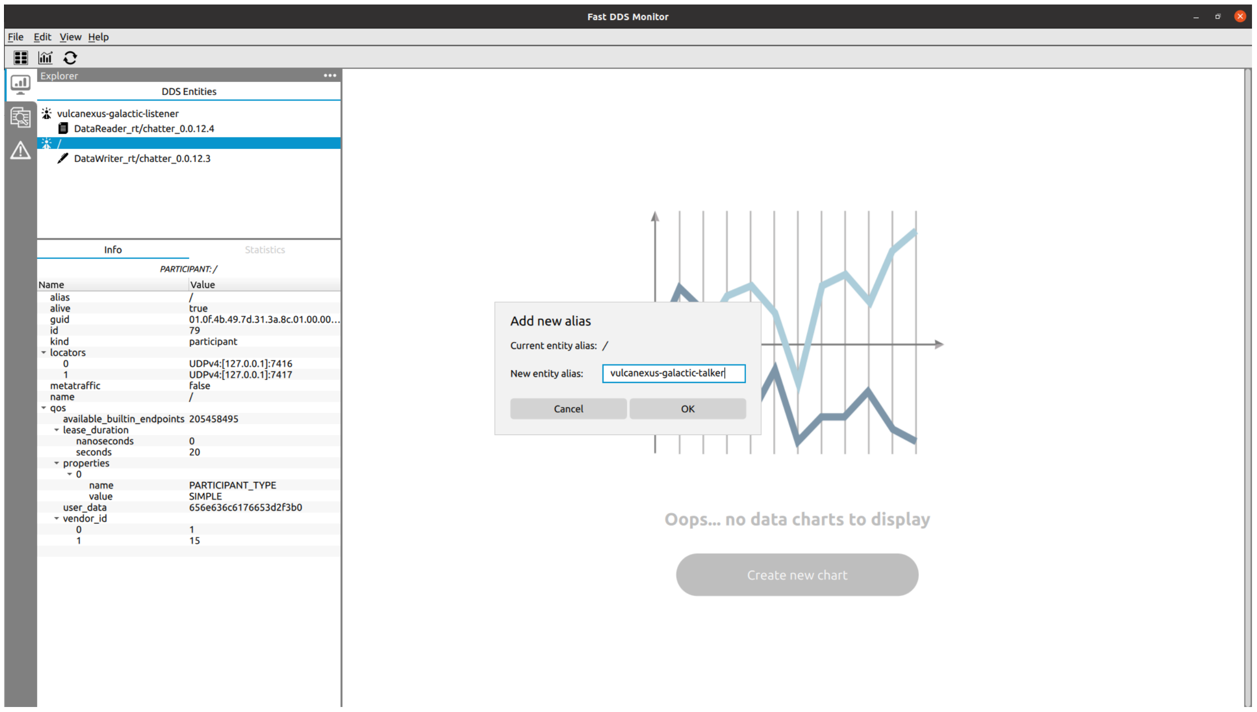
```
Name                          Value
  alias                       /
  alive                       true
  guid                        01.0f.47.a1.3a.6a.ad.81.01.00.00...
  id                          1
  kind                        participant
▾ locators
      0                       UDPv4:[127.0.0.1]:7414
      1                       UDPv4:[127.0.0.1]:7415
  metatraffic                 false
  name                        /
▾ qos
      available_builtin_endpoints 3135
    ▾ lease_duration
          nanoseconds         0
          seconds             20
    ▾ properties
        ▾ 0
              name            PARTICIPANT_TYPE
              value           SIMPLE
      user_data               656e636c6176653d2f3b0
    ▾ vendor_id
          0                   1
          1                   15
```

4.5.1 Alias

Participants in ROS 2 are named `/` by default. In order to differentiate them, it is possible to change the Participant's aliases within the *Fast DDS Monitor*. In this case, the `vulcanexus-galactic-talker` Participant would be the one with a writer, and the `vulcanexus-galactic-listener` Participant would be the one with a reader.

4.5.2 Physical data

In order to see the information of the host and the physical context where every node is running, go to the Explorer Pane and activate the Physical Panel. There, the host, user and process of each node are displayed.



vulcanexus-galactic-listener
DataReader_rt/chatter_0.0.12.4
/
DataWriter_rt/chatter_0.0.12.3

PARTICIPANT: /

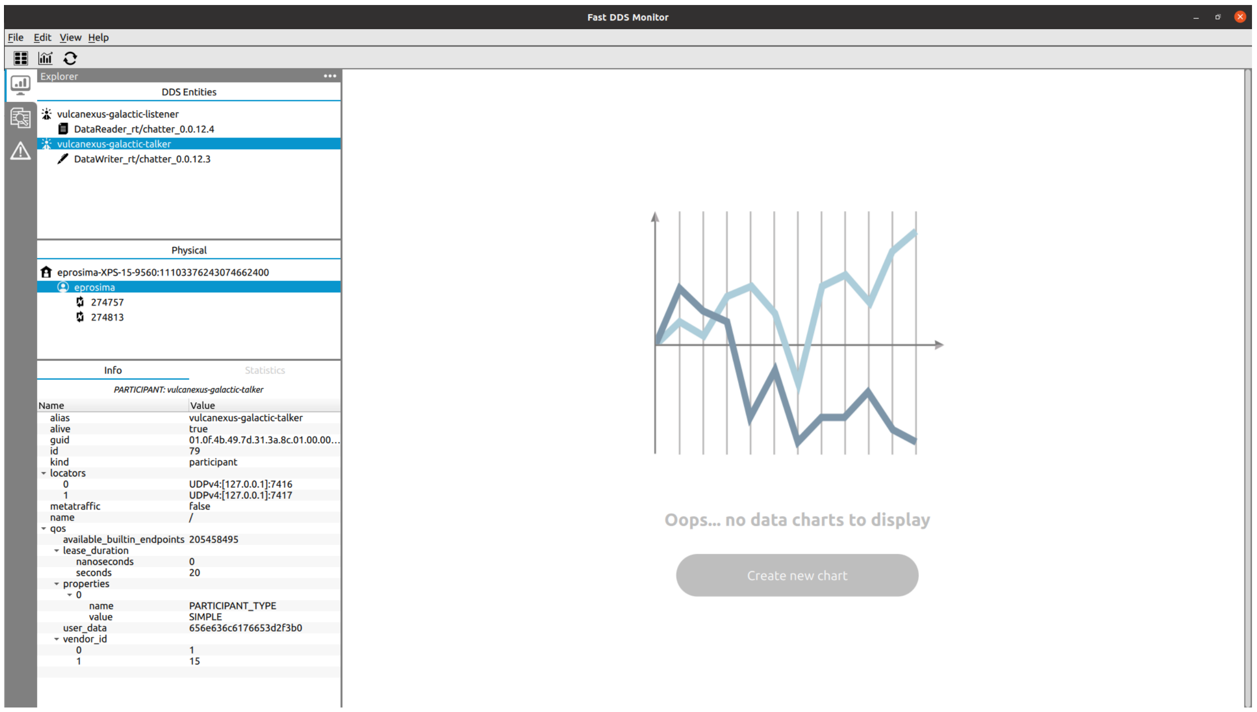
Name	Value
alias	/
alive	true
guid	01.0f.4b.49.7d.31.3a.8c.01.00.00...
id	79
kind	participant
locators	
0	UDPv4:[127.0.0.1]:7416
1	UDPv4:[127.0.0.1]:7417
metatraffic	false
name	/
qos	
available_builtin_endpoints	205458495
lease_duration	
nanoseconds	0
seconds	20
properties	
0	
name	PARTICIPANT_TYPE
value	SIMPLE
user_data	656e636c6176653d2f3b0
vendor_id	
0	1
1	15

Add new alias

Current entity alias: /

New entity alias: vulcanexus-galactic-talker

Cancel OK



vulcanexus-galactic-listener
DataReader_rt/chatter_0.0.12.4
vulcanexus-galactic-talker
DataWriter_rt/chatter_0.0.12.3

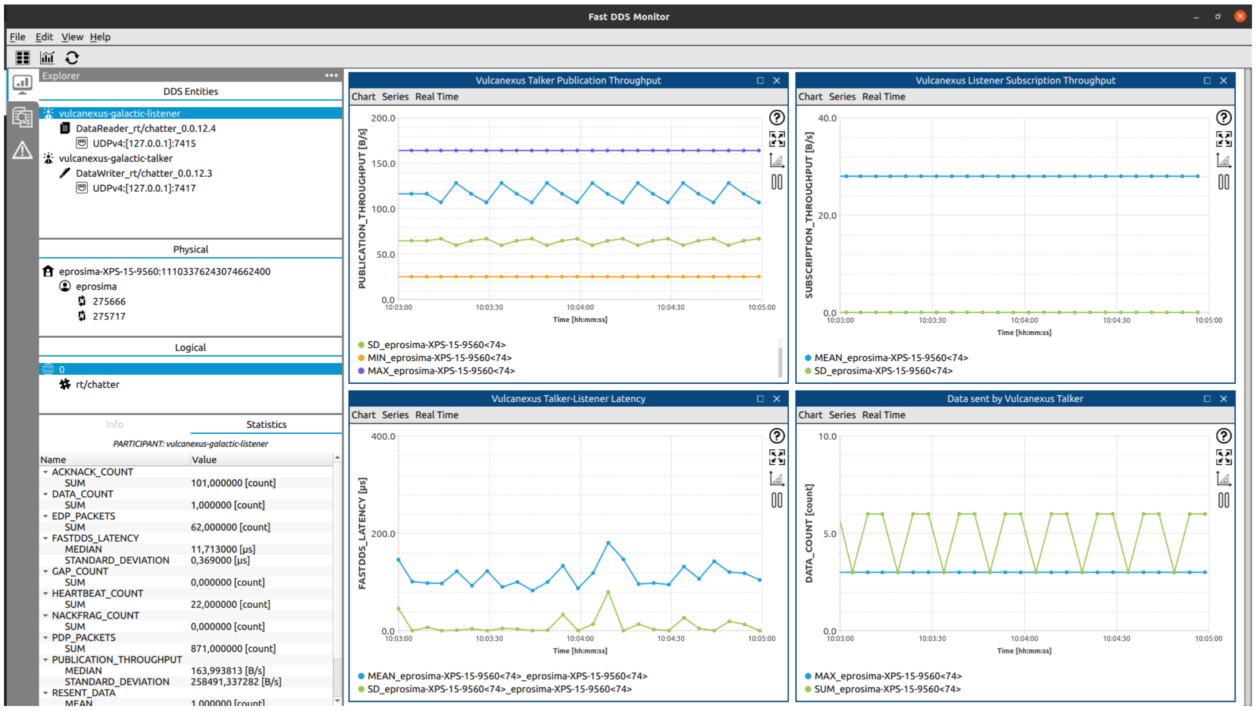
eprosima-XPS-15-9560:11103376243074662400
eprosima
274757
274813

PARTICIPANT: vulcanexus-galactic-talker

Name	Value
alias	vulcanexus-galactic-talker
alive	true
guid	01.0f.4b.49.7d.31.3a.8c.01.00.00...
id	79
kind	participant
locators	
0	UDPv4:[127.0.0.1]:7416
1	UDPv4:[127.0.0.1]:7417
metatraffic	false
name	/
qos	
available_builtin_endpoints	205458495
lease_duration	
nanoseconds	0
seconds	20
properties	
0	
name	PARTICIPANT_TYPE
value	SIMPLE
user_data	656e636c6176653d2f3b0
vendor_id	
0	1
1	15

4.5.3 Statistical data

To show statistical data about the communication between the `vulcanexus-galactic-talker` and the `vulcanexus-galactic-listener`, follow the steps to create dynamic series chart.


Fast DDS Monitor
File Edit View Help
Explorer
DDS Entities
Physical
Logical
Info
Statistics
Vulcanexus Talker Publication Throughput
Chart Series Real Time
PUBLICATION_THROUGHPUT [B/s]
Time [hh:mm:ss]
SD_eprosima-XPS-15-9560<74>
MIN_eprosima-XPS-15-9560<74>
MAX_eprosima-XPS-15-9560<74>
Vulcanexus Listener Subscription Throughput
SUBSCRIPTION_THROUGHPUT [B/s]
MEAN_eprosima-XPS-15-9560<74>
SD_eprosima-XPS-15-9560<74>
Vulcanexus Talker-Listener Latency
FASTDDS_LATENCY [μs]
MEAN_eprosima-XPS-15-9560<74>_eprosima-XPS-15-9560<74>
SD_eprosima-XPS-15-9560<74>_eprosima-XPS-15-9560<74>
Data sent by Vulcanexus Talker
DATA_COUNT [count]
MAX_eprosima-XPS-15-9560<74>
SUM_eprosima-XPS-15-9560<74>

```
vulcanexus-galactic-listener
    DataReader_rt/chatter_0.0.12.4
        UDPv4:[127.0.0.1]:7415
vulcanexus-galactic-talker
    DataWriter_rt/chatter_0.0.12.3
        UDPv4:[127.0.0.1]:7417

eprosima-XPS-15-9560:11103376243074662400
    eprosima
        275666
        275717

0
    rt/chatter
```

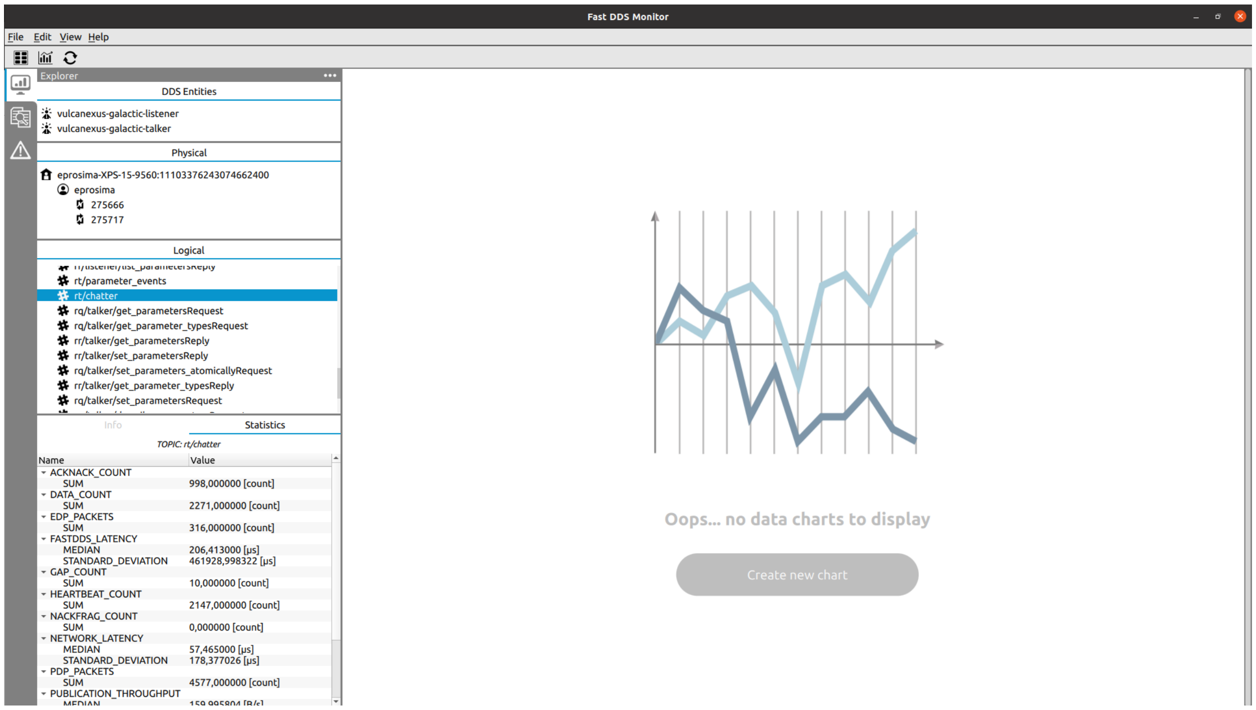
PARTICIPANT: vulcanexus-galactic-listener

Name	Value
ACKNACK_COUNT	
SUM	101,000000 [count]
DATA_COUNT	
SUM	1,000000 [count]
EDP_PACKETS	
SUM	62,000000 [count]
FASTDDS_LATENCY	
MEDIAN	11,713000 [μs]
STANDARD_DEVIATION	0,369000 [μs]
GAP_COUNT	
SUM	0,000000 [count]
HEARTBEAT_COUNT	
SUM	22,000000 [count]
NACKFRAG_COUNT	
SUM	0,000000 [count]
PDP_PACKETS	
SUM	871,000000 [count]
PUBLICATION_THROUGHPUT	
MEDIAN	163,993813 [B/s]
STANDARD_DEVIATION	258491,337282 [B/s]
RESENT_DATA	
MEAN	1,000000 [count]

4.5.4 Introspect metatraffic topics

Fast DDS Monitor filters by default the topics used for sharing metatraffic, as well as the endpoints related to them, so users can inspect their network easily. These topics are the ones that ROS 2 uses for discovery and configuration purposes, such as `ros_discovery_info`, as well as those used by Fast DDS to report statistical data.

In order to see these topics in the monitor, click *View->Show Metatraffic* menu button. Now, these topics are shown in the logical panel. Furthermore, the Readers and Writers associated to them are now listed under their respective Participants.



vulcanexus-galactic-listener
vulcanexus-galactic-talker

eprosima-XPS-15-9560:11103376243074662400
eprosima
275666
275717

rr/listener/list_parametersReply
rt/parameter_events
rt/chatter
rq/talker/get_parametersRequest
rq/talker/get_parameter_typesRequest
rr/talker/get_parametersReply
rr/talker/set_parametersReply
rq/talker/set_parameters_atomicallyRequest
rr/talker/get_parameter_typesReply
rq/talker/set_parametersRequest

TOPIC: rt/chatter

Name	Value
ACKNACK_COUNT	
SUM	998,000000 [count]
DATA_COUNT	
SUM	2271,000000 [count]
EDP_PACKETS	
SUM	316,000000 [count]
FASTDDS_LATENCY	
MEDIAN	206,413000 [μs]
STANDARD_DEVIATION	461928,998322 [μs]
GAP_COUNT	
SUM	10,000000 [count]
HEARTBEAT_COUNT	
SUM	2147,000000 [count]
NACKFRAG_COUNT	
SUM	0,000000 [count]
NETWORK_LATENCY	
MEDIAN	57,465000 [μs]
STANDARD_DEVIATION	178,377026 [μs]
PDP_PACKETS	
SUM	4577,000000 [count]
PUBLICATION_THROUGHPUT	
MEDIAN	159,995804 [B/s]

CHAPTER

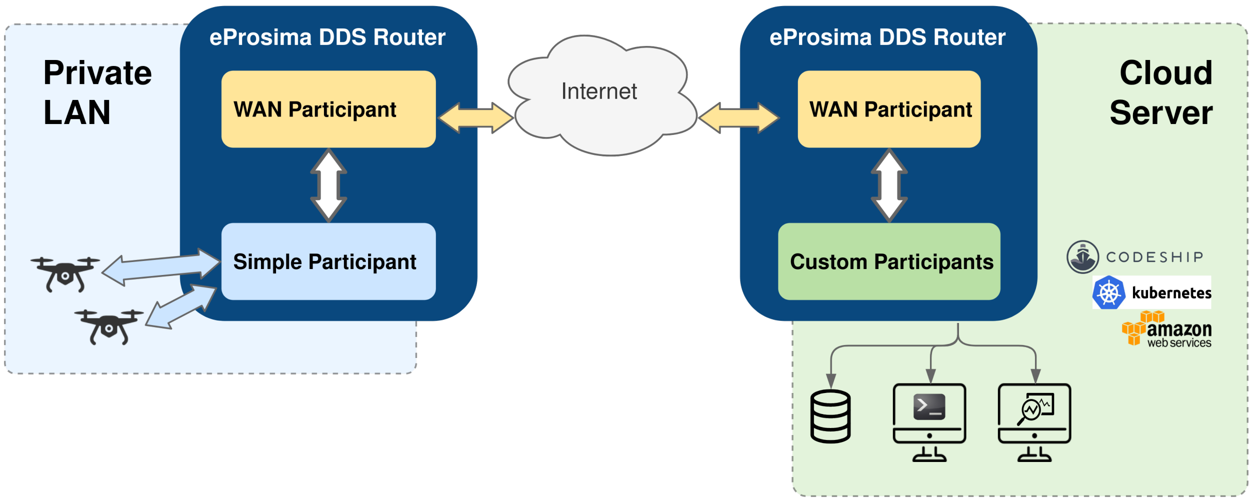
FIVE

VULCANEXUS CLOUD AND KUBERNETES

Table of Contents

5.1 Background

This walk-through tutorial sets up both a *Kubernetes* (*K8s*) network and a local environment in order to establish communication between a pair of ROS nodes, one sending messages from a LAN (talker) and another one receiving them in the Cloud (listener). Cloud environments such as container-oriented platforms can be connected using eProsima DDS Router, and thus, by launching a *DDS Router* instance at each side, communication can be established.



5.2 Prerequisites

Ensure that the Vulcanexus installation includes the cloud and the ROS 2 demo nodes package (it is suggested to use `vulcanexus-galactic-desktop`). Also, remember to source the environment in every terminal in this tutorial.

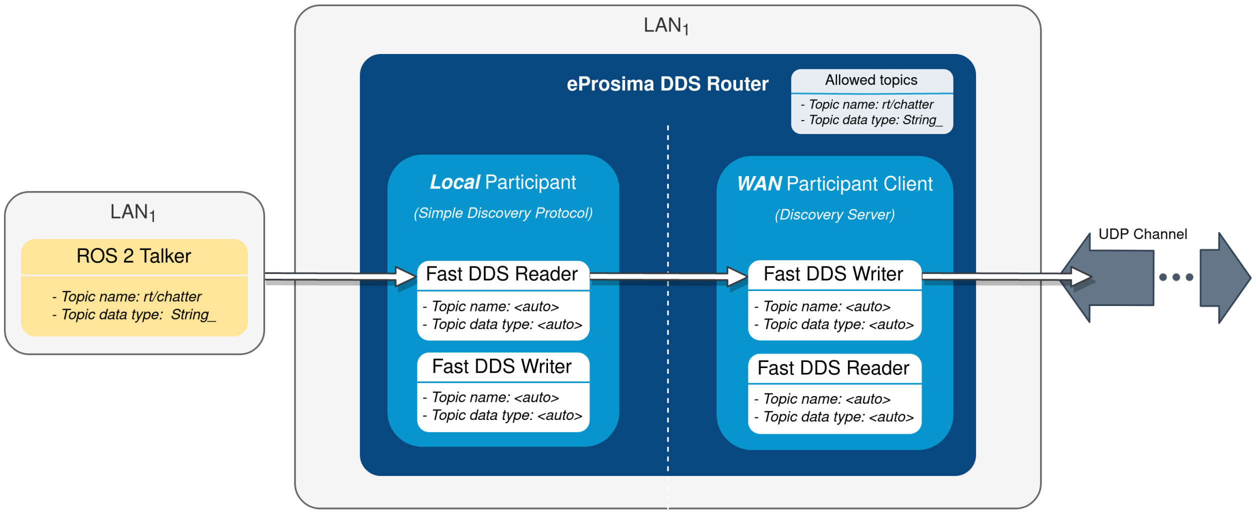
```
source /opt/vulcanexus/galactic/setup.bash
```

Warning: For the full understanding of this tutorial basic understanding of Kubernetes is required.

5.3 Local setup

The local instance of *DDS Router* (local router) only requires to have a Simple Participant and a WAN Participant that will play the client role in the discovery process of remote participants (see Discovery Server discovery mechanism).

After having acknowledged each other's existence through Simple DDS discovery mechanism (multicast communication), the local participant will start receiving messages published by the ROS 2 talker node, and will then forward them to the WAN participant. Next, these messages will be sent to another participant hosted on a *K8s* cluster to which it connects via WAN communication over UDP/IP. Following there is a representation of the above-described scenario:



5.3.1 Local router

The configuration file used by the local router will be the following:

```yaml
# local-ddsrouter.yaml

allowlist:
  - name: "rt/chatter"
    type: "std_msgs::msg::dds_::String_"

SimpleParticipant:
  type: local
  domain: 0
```

(continues on next page)

(continued from previous page)

```yaml
LocalWAN:
  type: wan
  id: 3
  listening-addresses:  # Needed for UDP communication
    - ip: "3.3.3.3"  # LAN public IP
      port: 30003
      transport: "udp"
  connection-addresses:
    - id: 2
      addresses:
        - ip: "2.2.2.2"  # Public IP exposed by the k8s cluster to reach the cloud DDS-
↪Router
          port: 30002
          transport: "udp"
```

Please, copy the previous configuration snippet and save it to a file in your current working directory with name `local-ddsrouter.yaml`.

Note that the simple participant will be receiving messages sent in DDS domain `0`. Also note that, due to the choice of UDP as transport protocol, a listening address with the LAN public IP address needs to be specified for the local WAN participant, even when behaving as client in the participant discovery process. Make sure that the given port is reachable from outside this local network by properly configuring port forwarding in your Internet router device. The connection address points to the remote WAN participant deployed in the *K8s* cluster. For further details on how to configure WAN communication, please have a look at WAN Configuration.

Note: As an alternative, TCP transport may be used instead of UDP. This has the advantage of not requiring to set a listening address in the local router's WAN participant (TCP client), so there is no need to fiddle with the configuration of your Internet router device.

To launch the local router, execute the following command (remember to source the Vulcanexus environment):

```
ddsrouter --config-path local-ddsrouter.yaml
```

5.3.2 Talker

In another terminal, run the following command in order to start the ROS 2 node that publishes messages in DDS domain `0` (remember to source the Vulcanexus environment):

```
ros2 run demo_nodes_cpp talker
```

5.4 Kubernetes setup

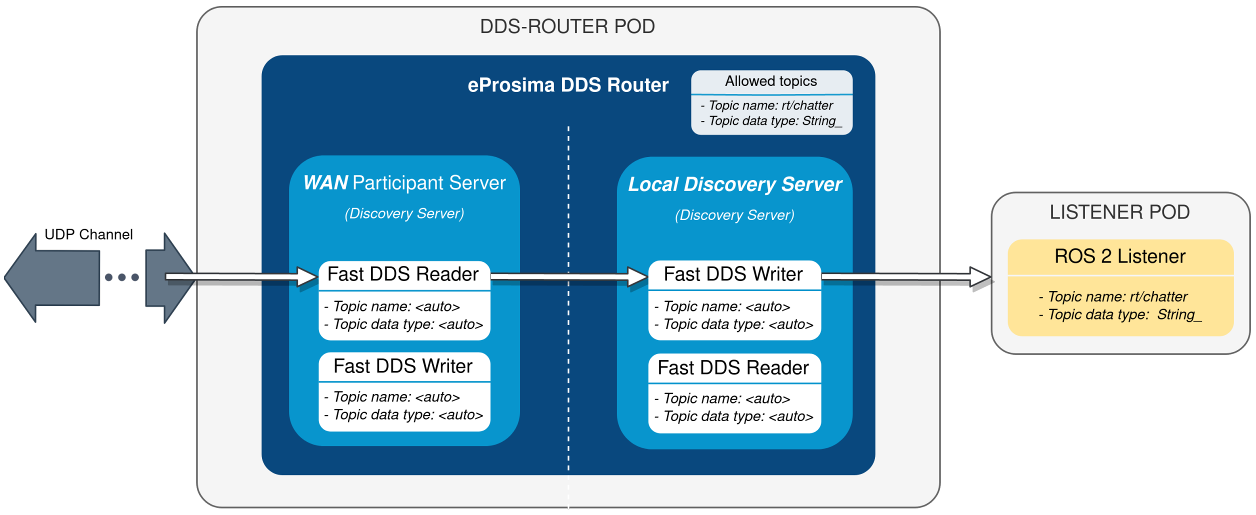
Two different deployments are required to receive the `talker` messages in the Cloud, each in a different *K8s* pod; the first one being a *DDS Router* cloud instance (cloud router), which consists of two participants:

- A WAN Participant that receives the messages coming from our LAN through the aforementioned UDP communication channel.
- A Local Discovery Server (local DS) that propagates them to a ROS 2 listener node hosted in a different *K8s* pod.

Note: The choice of a Local Discovery Server instead of a Simple Participant to communicate with the listener has to do with the difficulty of enabling multicast routing in cloud environments.

The other deployment is the ROS 2 listener node. This node has to be launched as a Client to the local DS running on the first deployment.

The described scheme is represented in the following figure:



In addition to the two mentioned deployments, two *K8s* services are required in order to direct dataflow to each of the pods. A LoadBalancer will forward messages reaching the cluster to the WAN participant of the cloud router, and a ClusterIP service will be in charge of delivering messages from the local DS to the listener pod. Following there are the settings needed to launch these services in *K8s*:

```yaml
kind: Service
apiVersion: v1
metadata:
  name: ddsrouter
  labels:
    app: ddsrouter
spec:
  ports:
    - name: UDP-30002
      protocol: UDP
      port: 30002
      targetPort: 30002
```

(continues on next page)

(continued from previous page)

```
  selector:
    app: ddsrouter
  type: LoadBalancer
```

```
kind: Service
apiVersion: v1
metadata:
  name: local-ddsrouter
spec:
  ports:
    - name: UDP-30001
      protocol: UDP
      port: 30001
      targetPort: 30001
  selector:
    app: ddsrouter
  clusterIP: 192.168.1.11  # Private IP only reachable within the k8s cluster to␣
↪communicate with the ddsrouter application
  type: ClusterIP
```

Note: An Ingress needs to be configured for the LoadBalancer service to make it externally-reachable. In this example we consider the assigned public IP address to be `2.2.2.2`.

The configuration file used for the cloud router will be provided by setting up a ConfigMap:

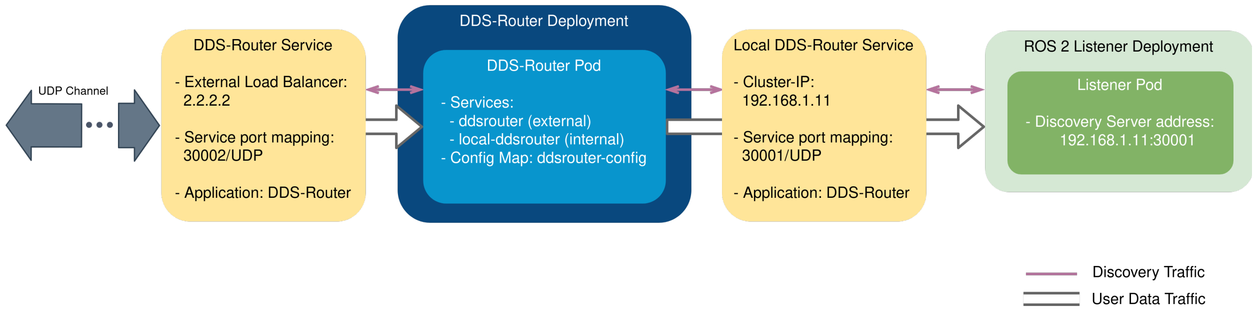
```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ddsrouter-config
data:
  ddsrouter.config.file: |-
    allowlist:
      - name: "rt/chatter"
        type: "std_msgs::msg::dds_::String_"

    LocalDiscoveryServer:
      type: local-discovery-server
      ros-discovery-server: true
      id: 1
      listening-addresses:
        - ip: "192.168.1.11"  # Private IP only reachable within the k8s cluster to␣
↪communicate with the ddsrouter application
          port: 30001
          transport: "udp"

    CloudWAN:
      type: wan
      id: 2
      listening-addresses:
        - ip: "2.2.2.2" # Public IP exposed by the k8s cluster to reach the cloud DDS-
↪Router
```

(continues on next page)

(continued from previous page)

```
        port: 30002
        transport: "udp"
```

Following there is a representation of the overall *K8s* cluster configuration:


UDP Channel
DDS-Router Service
- External Load Balancer: 2.2.2.2
- Service port mapping: 30002/UDP
- Application: DDS-Router
DDS-Router Deployment
DDS-Router Pod
- Services:
- ddsrouter (external)
- local-ddsrouter (internal)
- Config Map: ddsrouter-config
Local DDS-Router Service
- Cluster-IP: 192.168.1.11
- Service port mapping: 30001/UDP
- Application: DDS-Router
ROS 2 Listener Deployment
Listener Pod
- Discovery Server address: 192.168.1.11:30001
Discovery Traffic
User Data Traffic

5.4.1 DDS-Router deployment

The cloud router is launched from within a *Vulcanexus Cloud* Docker image (that can be downloaded in Vulcanexus webpage), which uses as configuration file the one hosted in the previously set up ConfigMap. Assuming the name of the generated Docker image is `ubuntu-vulcanexus-cloud:galactic`, the cloud router will then be deployed with the following settings:

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: ddsrouter
  labels:
    app: ddsrouter
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ddsrouter
  template:
    metadata:
      labels:
        app: ddsrouter
    spec:
      volumes:
        - name: config
          configMap:
            name: ddsrouter-config
            items:
              - key: ddsrouter.config.file
                path: DDSROUTER_CONFIGURATION.yaml
      containers:
        - name: ubuntu-vulcanexus-cloud
          image: ubuntu-vulcanexus-cloud:galactic
          ports:
            - containerPort: 30001
```

(continues on next page)

(continued from previous page)

```
              protocol: UDP
            - containerPort: 30002
              protocol: UDP
          volumeMounts:
            - name: config
              mountPath: /tmp
          args: ["-r", "ddsrouter -r 10 -c /tmp/DDSROUTER_CONFIGURATION.yaml"]
      restartPolicy: Always
```

5.4.2 Listener deployment

Since ROS 2 demo nodes package is not installed by default in *Vulcanexus Cloud*, a new Docker image adding in this functionality must be generated. Also, the IP address and port of the local Discovery Server must be specified, so a custom entrypoint is also provided.

Copy the following snippet and save it to the current directory as `Dockerfile`:

```
FROM ubuntu-vulcanexus-cloud:galactic

# Install demo-nodes-cpp
RUN source /opt/vulcanexus/galactic/setup.bash && \
    apt update && \
    apt install -y ros-galactic-demo-nodes-cpp

COPY ./run.bash /
RUN chmod +x /run.bash

# Setup entrypoint
ENTRYPOINT ["/run.bash"]
```

Copy the following snippet and save it to the current directory as `run.bash`:

```
#!/bin/bash

if [[ $1 == "listener" ]]
then
    NODE="listener"
else
    NODE="talker"
fi

SERVER_IP=$2
SERVER_PORT=$3

# Setup environment
source "/opt/vulcanexus/galactic/setup.bash"

echo "Starting ${NODE} as client of Discovery Server ${SERVER_IP}:${SERVER_PORT}"
ROS_DISCOVERY_SERVER=";${SERVER_IP}:${SERVER_PORT}" ros2 run demo_nodes_cpp ${NODE}
```

Build the docker image running the following command:

```
docker build -t vulcanexus-cloud-demo-nodes:galactic -f Dockerfile
```

Now, the listener pod can be deployed by providing the following configuration:

```yaml
kind: Deployment
apiVersion: apps/v1
metadata:
  name: ros2-galactic-listener
  labels:
    app: ros2-galactic-listener
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ros2-galactic-listener
  template:
    metadata:
      labels:
        app: ros2-galactic-listener
    spec:
      containers:
        - name: vulcanexus-cloud-demo-nodes
          image: vulcanexus-cloud-demo-nodes:galactic
          args:
            - listener
            - 192.168.1.11
            - '30001'
      restartPolicy: Always
```

Once all these components are up and running, communication should have been established between the talker and listener nodes, so that messages finally manage to reach the listener pod and get printed in its STDOUT.

Feel free to interchange the locations of the ROS nodes by slightly modifying the provided configuration files, hosting the talker in the *K8s* cluster while the listener runs in the LAN.

CHAPTER

SIX

VULCANEXUS AND MICRO-ROS

micro-ROS already provides several tutorials that can be also run within Vulcanexus. Please, visit micro-ROS tutorial webpage.

CHAPTER

SEVEN

SUPPORTED PLATFORMS

Vulcanexus ROS 2 all-in-one tool set, is officially available in the platforms specified in the table below.

Table 1: Vulcanexus officially supported platforms

Vulcanexus Version	Architecture	OS
Galactic Gamble	amd64	Ubuntu Focal (20.04)

However, as ROS 2 is officially supported in the platforms stated in the REP 2000 specification, building Vulcanexus for these platforms is expected to succeed. Other platforms not mentioned in the REP 2000 specification may also build successfully and be used.

CHAPTER

EIGHT

VULCANEXUS RELEASES

Vulcanexus maintains several releases with different support cycles. Each year, a new Vulcanexus major version is released. This major versions have a code name composed of an adjective and the name of a volcano, both starting with the same letter, the first of them being **Galactic Gamble** (v1.0.0). Within the support period of any version, there can be both minor and patch releases that either add new functionalities in an ABI compatible way, or fix possible issues. Every other year, a long term release (LTS) is released, the first of them being the H version (May 2022). In between, LTSs a short term release is released which will receive support for a shorter period of time. The following table outlines the Vulcanexus releases and their support cycles:

8.1 Galactic Gamble (v1.0.0)

Table 1: Vulcanexus versions

Name	Version	Release Date	EOL Date
Galactic Gamble	v1	TODO	November 2022