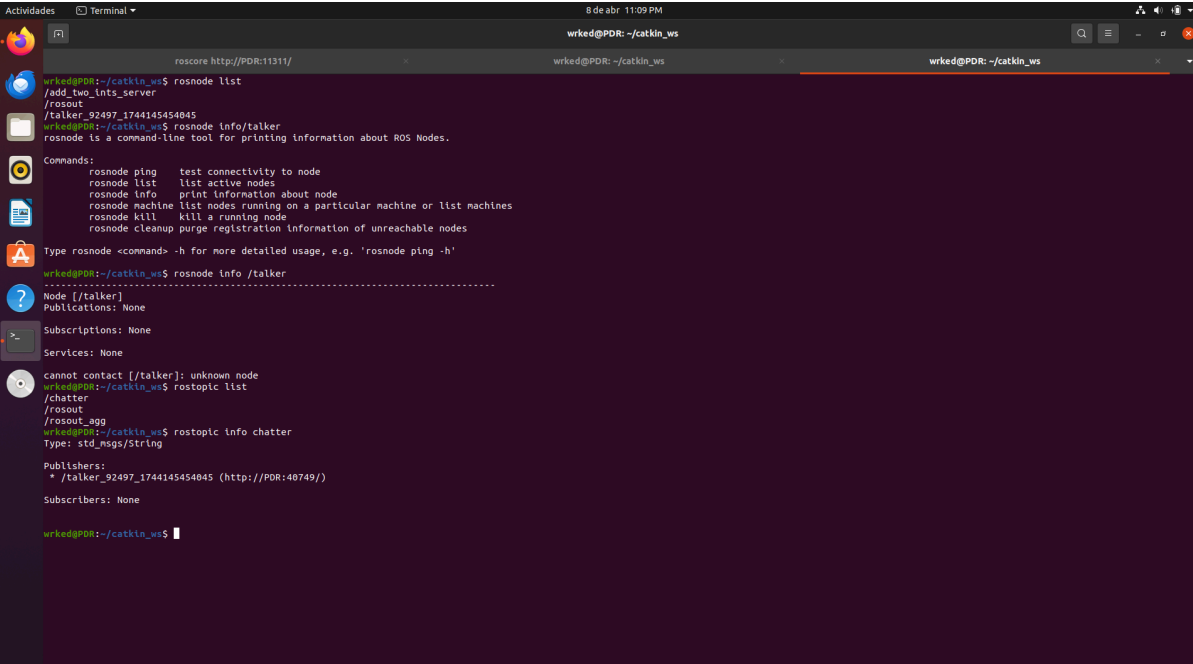


P1. ¿Qué podemos saber de este nodo? (Suscriptores, topics)

Al ejecutar `rostopic info /talker`, obtenemos información detallada del nodo `/talker`, incluyendo:

- Publicaciones: Generalmente pública en el topic `/chatter`.
- Suscriptores: Es probable que no tenga suscriptores inicialmente si no hay otro nodo escuchando.
- Tipo de mensajes: A través de `rostopic info /chatter` podemos saber que se están enviando mensajes tipo `std_msgs/String`.



```
8 de abr 11:09 PM
wrked@PDR: ~/catkin_ws

wrked@PDR:~/catkin_ws$ roscore http://PDR:11311/
roscore http://PDR:11311/

wrked@PDR:~/catkin_ws$ rosnode list
/add_two_ints_server
/rosout
/talker_92497_1744145454045

wrked@PDR:~/catkin_ws$ rosnode info /talker
rosnode is a command-line tool for printing information about ROS Nodes.

Commands:
rosnode ping      test connectivity to node
rosnode list      list active nodes
rosnode info      print information about node
rosnode machine   list nodes running on a particular machine or list machines
rosnode kill      kill a running node
rosnode cleanup   purge registration information of unreachable nodes

Type rosnode <command> -h for more detailed usage, e.g. 'rosnode ping -h'

wrked@PDR:~/catkin_ws$ rosnode info /talker
-----
Node [/talker]
Publications: None

Subscriptions: None

Services: None

cannot contact [/talker]: unknown node
wrked@PDR:~/catkin_ws$ rostopic list
/chatter
/rosout
/rosout_agg
wrked@PDR:~/catkin_ws$ rostopic info chatter
Type: std_msgs/String

Publishers:
 * /talker_92497_1744145454045 (http://PDR:40749/)

Subscribers: None

wrked@PDR:~/catkin_ws$
```

P2. ¿Qué podemos saber sobre el topic que publica el nodo que acabamos de lanzar?

Usando `rostopic info /chatter`, se puede saber:


- Tipo de mensaje: `std_msgs/String`.
- Publicadores: El nodo `/talker`.
- Suscriptores: Inicialmente ninguno, pero pueden aparecer al lanzar el nodo `/listener`.

Además, con `rostopic echo /chatter` puedes ver los mensajes en tiempo real

P3. ¿Qué podemos saber sobre este nuevo nodo?

Después de lanzar `roslaunch rospy_tutorials listener`, con `rostopic info /listener` se puede observar:

- El nodo está suscrito al topic / chatter.
- No publica nada (en este caso).
- Es un consumidor de datos del nodo / talker.



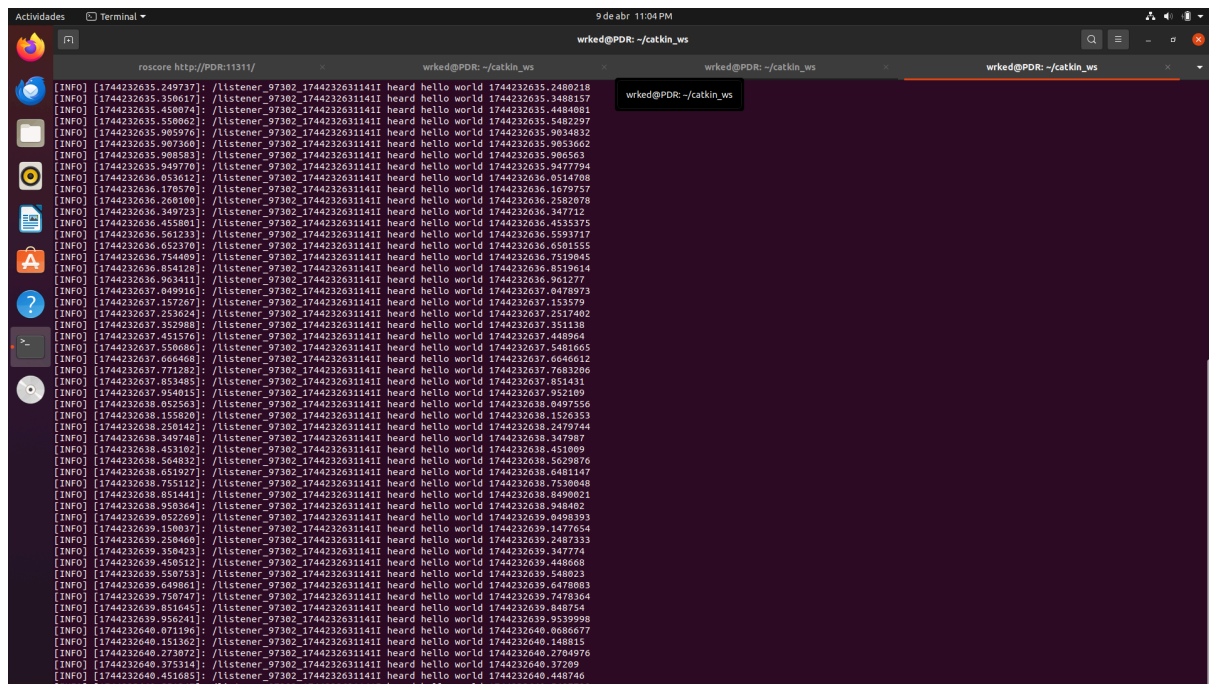
The screenshot shows a terminal window with a dark background. The title bar at the top reads "Actividades" and "Terminal". The terminal output shows a sequence of "hello world" messages from a service client, each followed by a timestamp. The messages are: "hello world 1744232654.247958", "hello world 1744232654.354804", "hello world 1744232654.458749", "hello world 1744232654.551291", "hello world 1744232654.647797", "hello world 1744232654.7518356", "hello world 1744232654.855917", "hello world 1744232654.968208", "hello world 1744232655.0599487", "hello world 1744232655.1496177", "hello world 1744232655.2479823", "hello world 1744232655.3480144", "hello world 1744232655.4480002", "hello world 1744232655.5479171", "hello world 1744232655.6478655", "hello world 1744232655.7478607", "hello world 1744232655.8480918", "hello world 1744232655.9479465", "hello world 1744232656.0481172", "hello world 1744232656.1480508", "hello world 1744232656.2479944", "hello world 1744232656.3479083", "hello world 1744232656.4479542", "hello world 1744232656.547953", "hello world 1744232656.648675", and "hello world 1744232656.7551384". The terminal window has a sidebar on the left with icons for various applications. The top of the window shows the system clock as "9 de abr 11:04 PM".

```
roscore http://PDR:11311/
data: "hello world 1744232654.247958"
---
data: "hello world 1744232654.354804"
---
data: "hello world 1744232654.458749"
---
data: "hello world 1744232654.551291"
---
data: "hello world 1744232654.647797"
---
data: "hello world 1744232654.7518356"
---
data: "hello world 1744232654.855917"
---
data: "hello world 1744232654.968208"
---
data: "hello world 1744232655.0599487"
---
data: "hello world 1744232655.1496177"
---
data: "hello world 1744232655.2479823"
---
data: "hello world 1744232655.3480144"
---
data: "hello world 1744232655.4480002"
---
data: "hello world 1744232655.5479171"
---
data: "hello world 1744232655.6478655"
---
data: "hello world 1744232655.7478607"
---
data: "hello world 1744232655.8480918"
---
data: "hello world 1744232655.9479465"
---
data: "hello world 1744232656.0481172"
---
data: "hello world 1744232656.1480508"
---
data: "hello world 1744232656.2479944"
---
data: "hello world 1744232656.3479083"
---
data: "hello world 1744232656.4479542"
---
data: "hello world 1744232656.547953"
---
data: "hello world 1744232656.648675"
---
data: "hello world 1744232656.7551384"
---
```

P4. ¿Qué ha sucedido ahora en el topic /chatter?

Al lanzar el nodo `/listener`, este se suscribe al topic `/chatter`, lo que significa que:

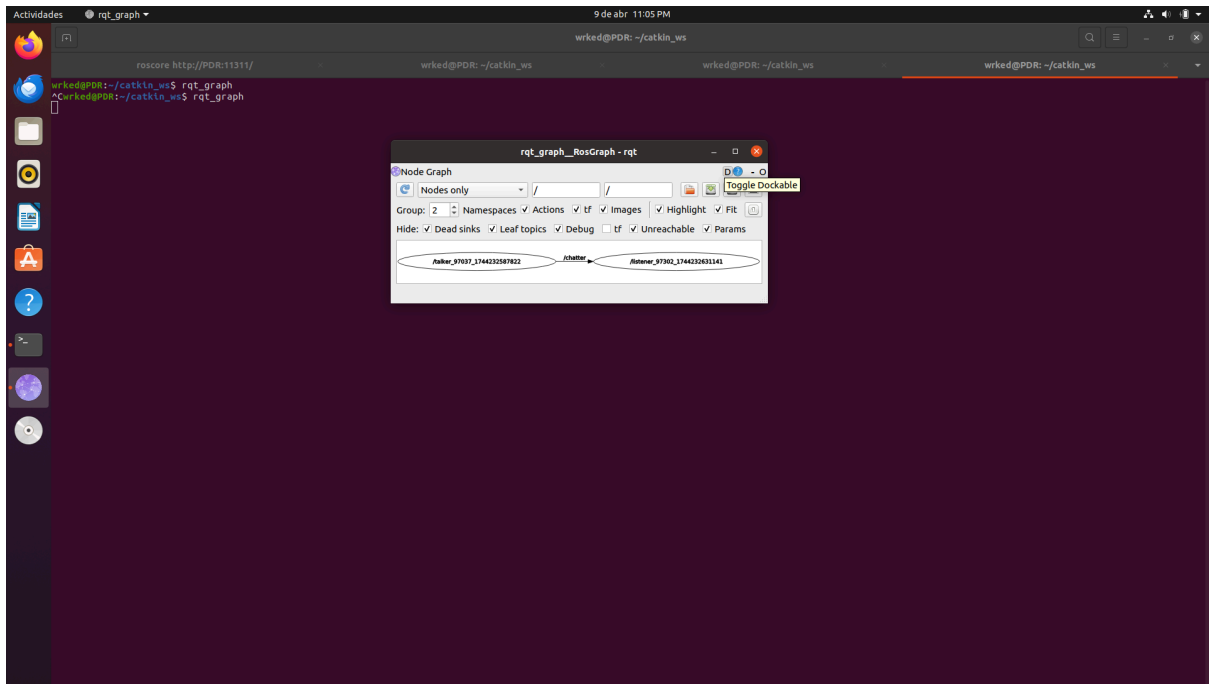
- Ahora el mensaje publicado por `/talker` es recibido y procesado por `/listener`.
- `/chatter` tiene un suscriptor `/listener`.



P5. ¿Qué podemos saber a partir de este gráfico?

Con `rqt_graph`, puedes ver:

- Visualización de nodos y topics como un grafo.
- Conexiones entre nodos publicadores y suscriptores.
- Por ejemplo, `/talker` → `/chatter` → `/listener`, lo que representa el flujo de mensajes.



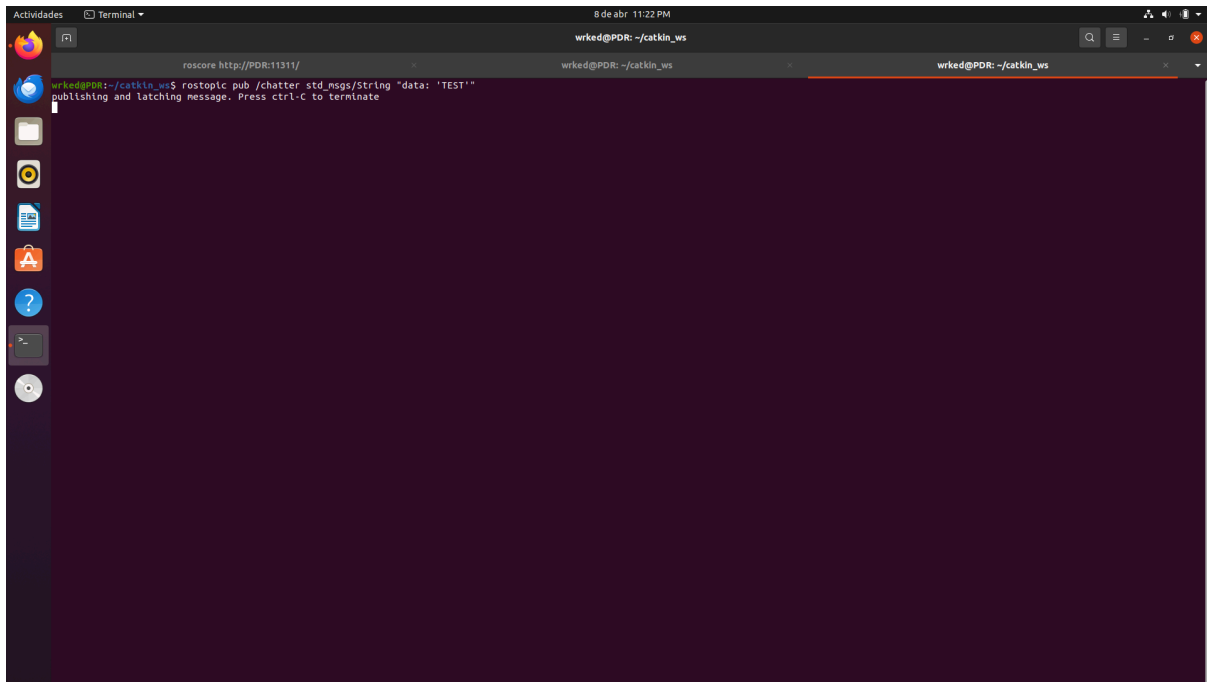
Es útil para depurar y entender la arquitectura del sistema ROS.

P6. ¿Qué puedes comentar acerca del servicio que acabamos de lanzar?

Se observa que:

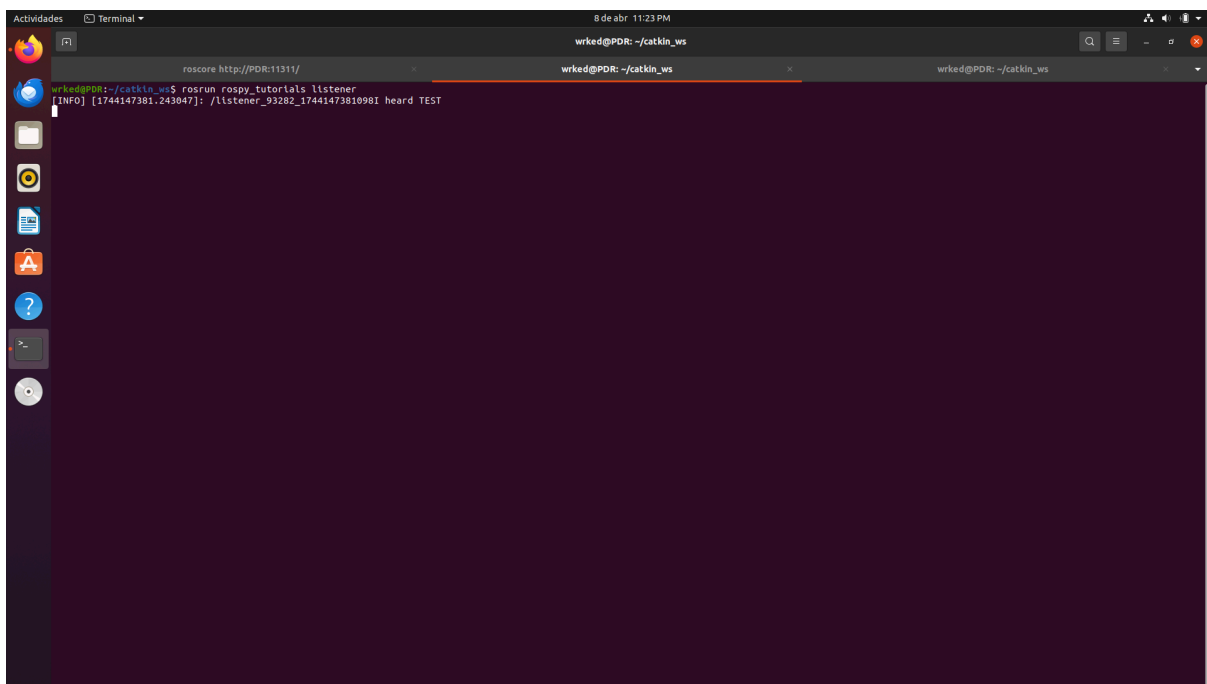
- Se lanza un servicio llamado AddTwoInts.
- El cliente envía dos enteros (3 y 7).
- El servidor responde con la suma (10).
- Puedes usar `rossrv list` y `rossrv show` para ver la definición del servicio.

Ejercicio 1.



A terminal window titled "Terminal" with a dark background and light text. The window shows the execution of a ROS command. The prompt is "wrked@PDR: ~/catkin_ws". The command entered is "rostopic pub /chatter std_msgs/String "data: 'TEST'" publishing and latching message. Press ctrl-C to terminate". The output shows the message being published.

```
wrked@PDR:~/catkin_ws$ rostopic pub /chatter std_msgs/String "data: 'TEST'"
publishing and latching message. Press ctrl-C to terminate
```



A terminal window titled "Terminal" with a dark background and light text. The window shows the execution of a ROS command. The prompt is "wrked@PDR: ~/catkin_ws". The command entered is "roscore http://PDR:11311/". The output shows the message being published. The prompt is "wrked@PDR: ~/catkin_ws". The command entered is "roscore http://PDR:11311/". The output shows the message being published.

```
wrked@PDR:~/catkin_ws$ roscore http://PDR:11311/
[INFO] [1744147381.243047]: /listener_93282_1744147381098I heard TEST
```

Ejercicio 2.

```
Actividades Terminal 9 de abr 10:28 PM
wrked@PDR: ~/catkin_ws/src/servicio_suma/src
roscore http://PDR:11311/
add_three_ints_client.py

GNU nano 4.8
#!/usr/bin/env python3

from __future__ import print_function

import sys
import rospy
from servicio_suma.srv import *

def add_three_ints_client(x, y):
    rospy.wait_for_service('add_three_ints')
    try:
        add_three_ints = rospy.ServiceProxy('add_three_ints', AddThreeInts)
        resp1 = add_three_ints(x, y)
        return resp1.sum
    except rospy.ServiceException as e:
        print("Service call failed: %s"%e)

def usage():
    return "%s [x y z]"%sys.argv[0]

if __name__ == "__main__":
    if len(sys.argv) == 3:
        x = int(sys.argv[1])
        y = int(sys.argv[2])
        z = int(sys.argv[3])
    else:
        print(usage())
        sys.exit(1)
    print("Requesting %s+%s+%s"%(x, y, z))
    print("%s + %s + %s = %s"%(x, y, add_three_ints_client(x, y)))
```

```
Actividades Terminal 9 de abr 10:28 PM
wrked@PDR: ~/catkin_ws/src/servicio_suma/src
roscore http://PDR:11311/
add_three_ints_server.py

GNU nano 4.8
#!/usr/bin/env python3

from __future__ import print_function

from servicio_suma.srv import AddThreeInts, AddThreeIntsResponse
import rospy

def handle_add_three_ints(req):
    print("Returning [%s + %s + %s]"%(req.a, req.b, req.c, (req.a + req.b + req.c)))
    return AddThreeIntsResponse(req.a + req.b + req.c)

def add_three_ints_server():
    rospy.init_node('add_three_ints_server')
    s = rospy.Service('add_three_ints', AddThreeInts, handle_add_three_ints)
    print("Ready to add three ints.")
    rospy.spin()

if __name__ == "__main__":
    add_three_ints_server()
```

