DTG

Generated by Doxygen 1.9.8

Chapter 1

DTG (DTN Testbed GUI)

Lightweight Python-based GUI for configuring network channel emulators (using tc qdisc) and simplify the management of Docker containers on the host machine. It aims to enhance **transparency** and reduce the complexity of command-line configuration.

1.1 Main Features

- Start or stop containers on the host machine.
- Select a container and modify the parameters of its channel emulator using `tc qdisc` command.
- Perform connectivity tests using the integrated `ping` command.
- Open a terminal collected to the specified node, (intended for low-level management purposes only)
- Simple and intuitive interface, designed to be easy to use.

1.2 File Structure

- `main.py` → main file to start the app.
- `app.py` → brain of GUI: handle life cycle and app state
- `core/` → core logic of the application
- `gui/` → windows displayed to user
- `utils/` → utility tools (like OperationLock class)
- `images/` → folder containing images used by the GUI.
- `requirements.txt` → required python libraries

Note: All folders (and files also) must be in the same directory as main.py.

1.3 DTG - Setup Guide

This guide covers the requirements and installation steps required to run DTG on Windows, macOS and Debian/↩ Ubuntu-based systems. It is also compatible with different Linux distros.

1.3.1 System Requirements

1.3.1.1 1. Docker

- Docker Engine
- Docker Compose plugin

1.3.1.2 2. Python

- **python3** — Python interpreter
- **python3-tk** — For the graphical environment (Tkinter)
- **python3-pip** — Tool to install Python libraries

1.3.1.3 3. Python Libraries

- **docker** — Communicates with Docker daemon
- **pillow** — Loads icons for GUI
- **sv_ttk** — Provides prettier themes for Tkinter

1.3.2 Install Docker and Docker Compose

> **Note:** Skip this passage if you have Docker & Docker-compose installed

> **Note:** Docker Compose comes with Docker Desktop on Windows and MacOS

For this passage you can either follow the Docker_Testbeds_guide.pdf file in
https://gitlab.com/unibo-dtn-docker-environment/DTN2hops
or, otherwise, follow the official guide for Docker
https://docs.docker.com/desktop/
and Docker Compose
https://docs.docker.com/compose/install/

1.4 GUI installation

1.4.1 Linux installation (Debian/Ubuntu-based)

> **Note:** Having Docker and Docker Compose installed (and running) is crucial for the GUI!

1.4.1.1 Step 1: Install Python, tkinter, and pip

```
# Update repositories' dependancies and upgrade packages
sudo apt update
sudo apt upgrade -y # -y to automatically say yes

# Install needed python packages
sudo apt install python3 python3-tk python3-pip
```

1.4.1.2 Step 2: Install required libraries

> **Note:** From Ubuntu 22.04 python packages are managed externally by apt, this means you have to either install them globally or create a python venv.

Virtual environment

```
# In the project's root folder (where main.py is)
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

Global installation

Note: sv_ttk is a library for style. Its not included in apt packages and can only be installed with pip!

```
# global installation using apt
sudo apt install python3-docker
sudo apt install python3-pil
sudo apt install python3-pil.imagetk
pip3 install sv_ttk --break-system-packages     # this may sound scary but this just ignores the protection
      and install this library into system's python
```

1.4.1.3 Step 3: Give the user the permission needed for Docker

In order to connect with Docker, the user must have the proper permissions: running the GUI with `sudo` does not work, because the user must be in the '`docker`' group to allow a proper connection to the Docker daemon.

```
# Add $USER to group docker
sudo usermod -aG docker $USER
```

`IMPORTANT` You must **reboot your pc** in order to make this change permanent.
After this, you can use docker cmds without root permissions, try:

```
docker ps
```

1.4.2 Windows Installation

1.4.2.1 Step 1: Download Python

Go to the Python official website: https://www.python.org/downloads/windows/ and download latest version, Windows Installer(64 bit)

1.4.2.2 Step 2: Install Python, tkinter and pip should be included

1. Open `python-xx.exe`
2. `IMPORTANT`Make sure to check "Add Python to PATH" during installation
3. Follow displayed instructions

1.4.2.3 Step 3: Verify Python Installation

Open PowerShell or Command Prompt and type:

```
python --version
pip --version
```

You should see version numbers for both commands if the installation was successful.
If for some reason pip is not installed run:

```
python -m ensurepip --upgrade
```

1.4.2.4 Step 4: Install required libraries

```
# In the project's root folder (where main.py is)

# Install requirements
pip3 install -r requirements.txt
```

1.4.3 MacOS installation

1.4.3.1 Step 1: Download and install python

Download Python 3 from https://www.python.org/downloads/macos/ or install via Homebrew:

```
brew install python3-tk@3.14
```

Tkinter – Included with Python 3 (verify with python3 -m tkinter –version).
pip – Included with Python 3 (verify with python3 -m pip –version).
Python Libraries – Install the required dependencies:

```
pip3 install -r requirements.txt
```

1.5 Running the GUI

Once the environment is ready:

```
# In the project's root folder (where main.py is)
python main.py
```

1.5.0.1 Linux

Note: On Linux command is `python3`, furthermore if you created a venv during installation phase you have to activate it everytime you want to run the GUI.

```
# In the project's root folder (where main.py is)

# activate virtual env
source venv/bin/activate
python3 main.py

# when you are done, deactivate venv
deactivate
```

Chapter 2

Namespace Index

2.1 Package List

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 app Namespace Reference

Classes

- class DTGApp

6.1.1 Detailed Description

```
\file app.py

\brief The central orchestrator of whole application

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author          |   Description
| ---------- | ---------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore   |  Initial implementation for DTG project.
```

6.2 assets Namespace Reference

Functions

- load_image (path, size=(20, 20))

Variables

- BASE_DIR = Path(sys._MEIPASS)
- str IMAGE_DIR = BASE_DIR / "images"

6.2.1 Detailed Description

```
\file gui/assets.py

\brief Asset management utility functions for DTG GUI

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.2.2 Function Documentation

6.2.2.1 load_image()

```
load_image (
              path,
              size = (20, 20) )
```

6.2.3 Variable Documentation

6.2.3.1 BASE_DIR

```
BASE_DIR = Path(sys._MEIPASS)
```

6.2.3.2 IMAGE_DIR

```
str IMAGE_DIR = BASE_DIR / "images"
```

6.3 config_manager Namespace Reference

Functions

- get_config_dir ()
- load_recent_projects ()

- save_recent_project (path)
- load_configs (project_name, container_name)
- save_configs (project_name, container_name, interface, delay_spinbox, loss_spinbox, band_spinbox, limit↩ _spinbox, win, config_status, save_btn, all_container_configs)

Variables

- str APP_NAME = "DTG"
- CONFIG_DIR = get_config_dir()
- str RECENT_PROJECTS_FILE = CONFIG_DIR / "recent_projects.json"

6.3.1 Detailed Description

```
\file core/config_manager.py

\brief Configuration management utility functions for DTG

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author          |   Description
| ---------- | ---------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore   |  Initial implementation for DTG project.
```

6.3.2 Function Documentation

6.3.2.1 get_config_dir()

```
get_config_dir ( )
```

6.3.2.2 load_recent_projects()

```
load_recent_projects ( )

    \brief Utility function to load recent projects from a file

    \return (list) List of recent projects or empty list

    \throws Exception If recent projects file is corrupted
```

6.3.2.3 save_recent_project()

```
save_recent_project (
              path )

    \brief Utility function to save recent projects to a file

    \param path (str) Path of the project to save

    \return (void)
```

6.3.2.4 load_configs()

```
load_configs (
              project_name,
              container_name )

\brief Utility function to laod configs for a specific container from file

\param project_name (str) The name of the project

\param container_name (str) The name of the container

\return (dict) Dictionary of configurations or empty dict
```

6.3.2.5 save_configs()

```
save_configs (
              project_name,
              container_name,
              interface,
              delay_spinbox,
              loss_spinbox,
              band_spinbox,
              limit_spinbox,
              win,
              config_status,
              save_btn,
              all_container_configs )

\brief Utility function to laod configs for a specific container to file

It requires project name, container name and the the current spinbox values.

\return (dict) Dictionary of configurations or empty dict
```

6.3.3 Variable Documentation

6.3.3.1 APP_NAME

```
str APP_NAME = "DTG"
```

6.3.3.2 CONFIG_DIR

```
CONFIG_DIR = get_config_dir()
```

6.3.3.3 RECENT_PROJECTS_FILE

```
str RECENT_PROJECTS_FILE = CONFIG_DIR / "recent_projects.json"
```

6.4 docker_ops Namespace Reference

Functions

- Container get_container (DockerClient client, str container_id)
- List[Container] get_project_containers (DockerClient client, str project_name)
- List[str] get_container_interfaces (DockerClient client, str container_id)
- start_container_by_id (DockerClient client, str container_id)
- stop_container_by_id (DockerClient client, str container_id)
- restart_container_by_id (DockerClient client, str container_id)
- apply_tc_rules (DockerClient client, str container_id, str eth, str delay, str loss, str band, str limit)
- run_container_ping (DockerClient client, str container_id, str ipaddr)

6.4.1 Detailed Description

```
\file core/docker_ops.py

\brief Docker operations utility functions for DTG

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.4.2 Function Documentation

6.4.2.1 get_container()

```
Container get_container (
             DockerClient client,
             str container_id )
```

6.4.2.2 get_project_containers()

```
List[Container] get_project_containers (
             DockerClient client,
             str project_name )

    \brief Utility function to get all containers belonging to the specified project

    \param client (DockerClient) Docker Client instance
```

```
    \param project_name (str) The name of the project

    \return (list) List of containers of the specified project

    \throws Exception If no containers are found
```

6.4.2.3 get_container_interfaces()

```
List[str] get_container_interfaces (
             DockerClient client,
             str container_id )

    \brief Utility function to get interfaces names and associated ips of the specified container

    This function executes commands inside the container
    to list network interfaces and retrieve their IP addresses.

    \param client (DockerClient) Docker Client instance

    \param container_id (str) The id of the container

    \return (list) List of {interface_name - ip} entries
```

6.4.2.4 start_container_by_id()

```
start_container_by_id (
             DockerClient client,
             str container_id )

\brief Utility function to start a container by its id

This fuction uses the Docker SDK for Python to interact with the Docker daemon
and start a certain container. It is invoked by the GUI when the user requests a container start.

\param client (DockerClient) Docker Client instance

\param container_id (str) The id of the container to start

\return (None)
```

6.4.2.5 stop_container_by_id()

```
stop_container_by_id (
             DockerClient client,
             str container_id )

\brief Utility function to stop a container by its id

This fuction uses the Docker SDK for Python to interact with the Docker daemon
and stop a certain container. It is invoked by the GUI when the user requests a container stop.

\param client (DockerClient) Docker Client instance

\param container_id (str) The id of the container to stop

\return (None)
```

6.4.2.6 restart_container_by_id()

```
restart_container_by_id (
             DockerClient client,
             str container_id )
```

\brief Utility function to restart a container by its id

This fuction uses the Docker SDK for Python to interact with the Docker daemon
and restart a certain container. It is invoked by the GUI when the user requests a container restart.

\param client (DockerClient) Docker Client instance

\param container_id (str) The id of the container to restart

\return (None)

6.4.2.7 apply_tc_rules()

```
apply_tc_rules (
            DockerClient client,
            str container_id,
            str eth,
            str delay,
            str loss,
            str band,
            str limit )
```

\brief Utility function to execute tc command inside a container to apply network emulation rules

This fuction uses the Docker SDK for Python to interact with the Docker daemon
and apply tc command to a certain container.
It is invoked by the GUI when the user requests a container restart.

\param client (DockerClient) Docker Client instance

\param container_id_ (str) The id of the container

\param eth (str) The network interface name

\param delay (str) The delay value in ms

\param loss (str) The packet loss percentage

\param band (str) The bandwidth limit in Mbit

\param limit (str) The queue limit in packets

\return (Docker.models.exec.ExecResult) The result of the command execution

6.4.2.8 run_container_ping()

```
run_container_ping (
            DockerClient client,
            str container_id,
            str ipaddr )
```

\brief Utility function to execute ping command inside a container to a specified IP address

This fuction uses the Docker SDK for Python to interact with the Docker daemon
and apply tc command to a certain container.
It is invoked by the GUI when the user requests a container restart.

\param client (DockerClient) Docker Client instance

\param container_id_ (str) The id of the container

\param ipaddr (str) The target IP address to ping

\return (Docker.models.exec.ExecResult) The result of the command execution

6.5 lock_manager Namespace Reference

Classes

- class OperationLock

6.5.1 Detailed Description

```
\file utils/lock_manager.py

\brief Lock manager for asynchronous operations

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.6 main Namespace Reference

Variables

- root = tk.Tk()
- app = DTGApp(root)
- file

6.6.1 Detailed Description

```
\file main.py

\brief Entry point of DTG project. It creates the main window and starts the application.

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.6.2 Variable Documentation

6.6.2.1 root

```
root = tk.Tk()
```

6.6.2.2 app

```
app = DTGApp(root)
```

6.6.2.3 file

```
file
```

6.7 main_window Namespace Reference

Classes

- class MainWindow

6.7.1 Detailed Description

```
\file gui/main_window.py

\brief Main window class for DTG GUI

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
```

```
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author          |   Description
| ---------- | ---------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore   |  Initial implementation for DTG project.
```

6.8 node_window Namespace Reference

Classes

- class NodeWindow

6.8.1 Detailed Description

```
\file gui/node_window.py

\brief Sub window for controlling individual Docker containers (DTN nodes).

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author          |   Description
| ---------- | ---------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore   |  Initial implementation for DTG project.
```

6.9 startup_window Namespace Reference

Functions

- choose_project_popup (parent, icons)

6.9.1 Detailed Description

```
\file gui/startup_window.py

\brief The startup window for DTG application. Used to choose or create a Docker Compose project.

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License

    This file is part of DTG (DTN Testbed GUI).
```

```
    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.9.2 Function Documentation

6.9.2.1 choose_project_popup()

```
choose_project_popup (
              parent,
              icons )

    \brief Popup window to choose or create a Docker Compose project.

    This fucntion creates a popup window that allows the user to either select an existing
    Docker Compose project from a list of recent projects or browse the filesystem to select a
    new Docker Compose YAML file.

    \param parent The parent tkinter window.
    \param icons A dictionary containing icons for buttons.

    \return The path to the selected Docker Compose project file, or None if the user exits.
```

6.10 system_ops Namespace Reference

Classes

- class ComposeNotFoundError
- class DockerComposeError
- class TerminalError

Functions

- exec_compose (compose_file)
- open_terminal (str container_name)

6.10.1 Detailed Description

```
\file core/system_ops.py

\brief Implementation of system operations for Docker Compose and terminal management.

\copyright Copyright (c) 2025, Alma Mater Studiorum, University of Bologna, All rights reserved.

\par License
```

```
    This file is part of DTG (DTN Testbed GUI).

    DTG 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.

    DTG 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 DTG.  If not, see <http://www.gnu.org/licenses/>.

\author Matteo Biancofiore <matteo.biancofiore2@studio.unibo.it>
\date 13/11/2025

\par Supervisor
   Carlo Caini <carlo.caini@unibo.it>


\par Revision History:
| Date       |  Author         |   Description
| ---------- | --------------- | ----------------------------------------------
| 13/11/2025 | M. Biancofiore  |  Initial implementation for DTG project.
```

6.10.2 Function Documentation

6.10.2.1 exec_compose()

```
exec_compose (
              compose_file )

\brief Execute Docker Compose command to start the environment

This fuction search for the version of Docker Compose installed on your system and
execute the precise command with '-d' flag (detatched), containers will run in background.

\param compose_file (Path or str) Absolute path to .yml/.yaml file.

\return (list) List of argument of cmd (e.g. `['docker', 'compose', ...]`).

\throws ComposeNotFoundError If Docker Compose is not installed.
\throws DockerComposeError If errors occurs (e.g. invalid file, docker image not found)
```

6.10.2.2 open_terminal()

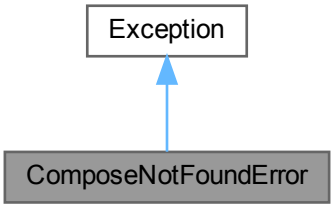
```
open_terminal (
             str container_name )

\brief Opens a terminal attached to the specified container

This fuction detects the host OS (via 'platform' module) and attemps
to launch a shell session using the first terminal found on the system.

\param container_name (str) The name of the Docker container

\return (subprocess.Popen) The process object opened, used for tracking

\throws TerminalError If no compatible terminal emulator is found.
```

Chapter 7

Class Documentation

7.1 ComposeNotFoundError Class Reference

Inheritance diagram for ComposeNotFoundError:

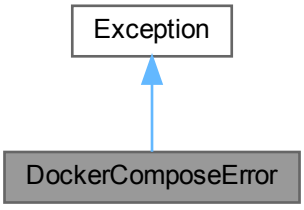


The documentation for this class was generated from the following file:

- core/system_ops.py

7.2 DockerComposeError Class Reference

Inheritance diagram for DockerComposeError:



The documentation for this class was generated from the following file:

- core/system_ops.py

7.3 DTGApp Class Reference

Public Member Functions

- __init__ (self, root)
- run (self)
- open_container_window (self, container_name)
- show_exiting_popup (self)
- on_main_window_close (self)

Public Attributes

- root
- open_windows
- open_terminals
- lock_manager
- project_name
- compose_file
- client
- running_icon
- exited_icon
- other_icon
- refresh_icon
- start_icon
- stop_icon
- folder_icon
- open_icon
- exit_icon
- main_window
- on_main_window_close

Protected Member Functions

- _run_startup (self)

7.3.1 Detailed Description

```
\brief Main Controller that handle DTG lifecycle.

The DTGApp class is responsible for initializing the environment,
connecting to Docker daemon and coordinating between MainWindow
and individual nodes windows.

Its purpose is to store the app's state which includes:
- Docker client.
- Active docker-compose file.
- List of open windows.
- Lock for asynchronous operations.
```

7.3.2 Constructor & Destructor Documentation

7.3.2.1 __init__()

```
__init__ (
              self,
              root )
```

7.3.3 Member Function Documentation

7.3.3.1 run()

```
run (
              self )
```

7.3.3.2 _run_startup()

```
_run_startup (
              self )  [protected]
```

7.3.3.3 open_container_window()

```
open_container_window (
              self,
              container_name )
```

```
\brief Open a new window for the given container.

This method checks if the container is running, if a window
for the container is already open, and if the container is locked.
If all checks pass, it creates a new NodeWindow for the container
and tracks it in the open_windows dictionary.

\param container_name The name of the container to open the window for.

\return None
```

7.3.3.4 show_exiting_popup()

```
show_exiting_popup (
              self )
```

7.3.3.5 on_main_window_close()

```
on_main_window_close (
              self )
```

7.3.4 Member Data Documentation

7.3.4.1 root

```
root
```

7.3.4.2 open_windows

```
open_windows
```

7.3.4.3 open_terminals

```
open_terminals
```

7.3.4.4 lock_manager

```
lock_manager
```

7.3.4.5 project_name

```
project_name
```

7.3.4.6 compose_file

compose_file

7.3.4.7 client

client

7.3.4.8 running_icon

running_icon

7.3.4.9 exited_icon

exited_icon

7.3.4.10 other_icon

other_icon

7.3.4.11 refresh_icon

refresh_icon

7.3.4.12 start_icon

start_icon

7.3.4.13 stop_icon

stop_icon

7.3.4.14 folder_icon

folder_icon

7.3.4.15 open_icon

open_icon

7.3.4.16 exit_icon

exit_icon

7.3.4.17 main_window

main_window

7.3.4.18 on_main_window_close

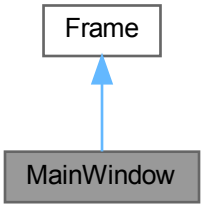
on_main_window_close

The documentation for this class was generated from the following file:

- app.py

7.4 MainWindow Class Reference

Inheritance diagram for MainWindow:



Public Member Functions

- __init__ (self, parent, controller)
- refresh_containers (self)
- start_container (self, row_id)
- stop_container (self, row_id)
- restart_container (self, row_id)
- start_all_containers (self)
- stop_all_containers (self, on_done=None)
- open_terminal (self, row_id)
- show_context_menu (self, event)
- close_context_menu (self, event=None)
- set_buttons_state (self, state)
- reset_operation_flag (self)
- on_tree_select (self, event)

Public Attributes

- parent
- controller
- tree
- refresh_btn
- start_button
- stop_button
- context_menu
- on_tree_select
- show_context_menu
- close_context_menu
- reset_operation_flag
- refresh_containers

Protected Member Functions

- _build_main_ui (self)

7.4.1 Detailed Description

```
\brief Main Window of DTG GUI.

This class represents the main window of the DTG GUI application.
It shows the list of Docker containers and allows users to
start, stop, restart containers, and open terminal windows.

\param parent The parent tkinter widget.
\param controller The main application controller (DTGApp instance).
```

7.4.2 Constructor & Destructor Documentation

7.4.2.1 __init__()

```
__init__ (
              self,
              parent,
              controller )
```

7.4.3 Member Function Documentation

7.4.3.1 _build_main_ui()

```
_build_main_ui (
              self )  [protected]

\brief Utility function to build the main UI components.

This fuction builds the Treeview for displaying Docker containers,
the control buttons, and sets up the context menu for container actions.

\return None
```

7.4.3.2 refresh_containers()

```
refresh_containers (
              self )

    \brief Utility function to refresh the list of Docker containers shown in the GUI.

    This fuction queries Docker via docker_ops for the current status of containers related
    to the active project and updates the Treeview accordingly.
    It also handles closing any open windows or terminals for containers
    that have been removed or stopped.

    \return None

    \throw DockerError if Docker is not reachable
```

7.4.3.3 start_container()

```
start_container (
              self,
              row_id )

\brief Utility function to start a Docker container from the GUI.

This function attempts to start the specified Docker container via docker_ops.
It checks if the container is already running or locked,
and updates the GUI accordingly.
The starting operation is performed in a separate thread to keep the GUI responsive.

\param row_id The identifier of the container to start (container name).

\throw DockerError if Docker is not reachable
```

7.4.3.4 stop_container()

```
stop_container (
              self,
              row_id )
```

```
\brief Utility function to stop a Docker container from the GUI.

This fuction attempts to stop the specified Docker container via docker_ops.
It checks if the container is already stopped or locked,
and updates the GUI accordingly.
The stopping operation is performed in a separate thread to keep the GUI responsive.

\param row_id The identifier of the container to start (container name).

\return None

\throw DockerError if Docker is not reachable
```

7.4.3.5 restart_container()

```
restart_container (
              self,
              row_id )
```

```
\brief Utility function to restart a Docker container from the GUI.

This fuction attempts to restart the specified Docker container via docker_ops.
It checks if the container is locked, and updates the GUI accordingly.
The restarting operation is performed in a separate thread to keep the GUI responsive.

\param row_id The identifier of the container to start (container name).

\return None

\throw DockerError if Docker is not reachable
```

7.4.3.6 start_all_containers()

```
start_all_containers (
              self )
```

7.4.3.7 stop_all_containers()

```
stop_all_containers (
              self,
              on_done = None )
```

7.4.3.8 open_terminal()

```
open_terminal (
              self,
              row_id )
```

```
\brief Utility function to open a terminal window for a Docker container from the GUI.

This fuction uses system_ops to open a terminal window attached
to the specified Docker container.
It checks if the container is locked or already has an open terminal.

\param row_id The identifier of the container to open a terminal for (container name).

\return None

\throw TerminalError if the terminal could not be opened
```

7.4.3.9 show_context_menu()

```
show_context_menu (
              self,
              event )
```

7.4.3.10 close_context_menu()

```
close_context_menu (
              self,
              event = None )
```

7.4.3.11 set_buttons_state()

```
set_buttons_state (
              self,
              state )
```

7.4.3.12 reset_operation_flag()

```
reset_operation_flag (
              self )
```

7.4.3.13 on_tree_select()

```
on_tree_select (
              self,
              event )
```

7.4.4 Member Data Documentation

7.4.4.1 parent

```
parent
```

7.4.4.2 controller

```
controller
```

7.4.4.3 tree

```
tree
```

7.4.4.4 refresh_btn

```
refresh_btn
```

7.4.4.5 start_button

```
start_button
```

7.4.4.6 stop_button

```
stop_button
```

7.4.4.7 context_menu

```
context_menu
```

7.4.4.8 on_tree_select

`on_tree_select`

7.4.4.9 show_context_menu

`show_context_menu`

7.4.4.10 close_context_menu

`close_context_menu`

7.4.4.11 reset_operation_flag

`reset_operation_flag`

7.4.4.12 refresh_containers

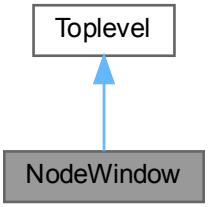
`refresh_containers`

The documentation for this class was generated from the following file:

- gui/main_window.py

7.5 NodeWindow Class Reference

Inheritance diagram for NodeWindow:



Public Member Functions

- __init__ (self, parent, controller, container_name)
- clear_console (self)
- do_tc (self)
- do_ping (self)

Public Attributes

- controller
- container_name
- config_status
- current_iface_tracker
- all_container_configs
- force_close
- interface_var
- delay_spinbox

- loss_spinbox
- band_spinbox
- limit_spinbox
- save_btn
- ipaddr_entry
- ping_btn
- toggle_btn
- console_frame
- output_box
- show_console

Protected Member Functions

- _build_ui (self)
- _update_spinboxes_for_interface (self, event=None)
- _on_close (self)
- _force_close (self)
- _set_config_dirty (self, *args)
- _save_configs_action (self)
- _toggle_console (self)
- _clear_focus (self, event)

Protected Attributes

- _clear_focus
- _on_close
- _update_spinboxes_for_interface
- _set_config_dirty

7.5.1 Detailed Description

```
\brief Main Window of DTN node.

This class represents the control window associated with a specific Docker container (DTN node).
It provides functionalities to apply traffic control rules, perform network tests (ping),
and view console output related to the container.

\param parent The parent Tkinter window.
\param controller The main application controller.
\param container_name The name of the Docker container associated with this window.
```

7.5.2 Constructor & Destructor Documentation

7.5.2.1 __init__()

```
__init__ (
            self,
            parent,
            controller,
            container_name )
```

7.5.3 Member Function Documentation

7.5.3.1 _build_ui()

```
_build_ui (
            self )  [protected]
```

\brief Utility function to build the node UI componetns.

This fuction builds the various UI components of the node window,
including the title, traffic control section, network test section,
and console output area.

\return None

7.5.3.2 _update_spinboxes_for_interface()

_update_spinboxes_for_interface (
self,
event = None) [protected]

\brief Utility function to update spinboxes when interface selection changes.

This fuction handles the event when the user selects a different network interface
from the dropdown. It saves the current configuration for the previously selected interface
(if modified) and loads the saved configuration for the newly selected interface into the spinboxes

\return None

7.5.3.3 _on_close()

_on_close (
self) [protected]

\brief Utility function to notify unsaved changes on close.

This fuction is called when the user attempts to close the node window.

\return None

7.5.3.4 _force_close()

_force_close (
self) [protected]

7.5.3.5 _set_config_dirty()

_set_config_dirty (
self,
** args*) [protected]

\brief Utility function to set configuration status to dirty(unsaved) when modified.

\return None

7.5.3.6 _save_configs_action()

_save_configs_action (
self) [protected]

\brief Utility function to save current configurations.

This fuction saves the current traffic control configurations
for the selected interface using config_manager.

\return None

7.5.3.7 _toggle_console()

```
_toggle_console (
              self )  [protected]
```

7.5.3.8 _clear_focus()

```
_clear_focus (
              self,
              event )  [protected]
```

7.5.3.9 clear_console()

```
clear_console (
              self )
```

7.5.3.10 do_tc()

```
do_tc (
              self )

\brief Utility function to run tc command inside the container.

This fuction retrieves the parameters from the spinboxes, validates them,
and then applies the traffic control rules inside the Docker container using docker_ops.
It also handles displaying the output or any errors in the console area.
Tc command is executed in a separate thread to keep the UI responsive.

\return None
```

7.5.3.11 do_ping()

```
do_ping (
              self )

\brief Utility function to run ping command inside the container.

This fuction retrieves the parameters from the entry box, validates them,
and then performs the ping command inside the Docker container using docker_ops.
It also handles displaying the output or any errors in the console area.
Ping command is executed in a separate thread to keep the UI responsive.

\return None
```

7.5.4 Member Data Documentation

7.5.4.1 controller

```
controller
```

7.5.4.2 container_name

```
container_name
```

7.5.4.3 config_status

```
config_status
```

7.5.4.4 current_iface_tracker

```
current_iface_tracker
```

7.5.4.5 all_container_configs

all_container_configs

7.5.4.6 _clear_focus

_clear_focus [protected]

7.5.4.7 _on_close

_on_close [protected]

7.5.4.8 force_close

force_close

7.5.4.9 interface_var

interface_var

7.5.4.10 _update_spinboxes_for_interface

_update_spinboxes_for_interface [protected]

7.5.4.11 delay_spinbox

delay_spinbox

7.5.4.12 loss_spinbox

loss_spinbox

7.5.4.13 band_spinbox

band_spinbox

7.5.4.14 limit_spinbox

limit_spinbox

7.5.4.15 save_btn

save_btn

7.5.4.16 _set_config_dirty

_set_config_dirty [protected]

7.5.4.17 ipaddr_entry

ipaddr_entry

7.5.4.18 ping_btn

ping_btn

7.5.4.19 toggle_btn

toggle_btn

7.5.4.20 console_frame

```
console_frame
```

7.5.4.21 output_box

```
output_box
```

7.5.4.22 show_console

```
show_console
```

The documentation for this class was generated from the following file:

- gui/node_window.py

7.6 OperationLock Class Reference

Public Member Functions

- __init__ (self)
- lock (self, container_id, op)
- unlock (self, container_id)
- is_locked (self, container_id)
- has_active_locks (self)

Protected Attributes

- _locks

7.6.1 Detailed Description

```
\brief Manager used to handle concurrency and operation locks on containers

The OperationLock prevents race conditions and conflictual action on containers
by ensuring that only one operation (start, stop, restart) can be performed
at a time on a specific container.
```

7.6.2 Constructor & Destructor Documentation

7.6.2.1 __init__()

```
__init__ (
            self )
```

7.6.3 Member Function Documentation

7.6.3.1 lock()

```
lock (
            self,
            container_id,
            op )
```

7.6.3.2 unlock()

```
unlock (
            self,
            container_id )
```

7.6.3.3 is_locked()

```
is_locked (
              self,
              container_id )
```

7.6.3.4 has_active_locks()

```
has_active_locks (
              self )
```

7.6.4 Member Data Documentation

7.6.4.1 _locks

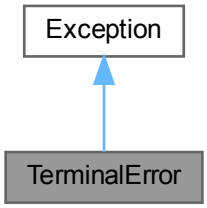
```
_locks  [protected]
```

The documentation for this class was generated from the following file:

- utils/lock_manager.py

7.7 TerminalError Class Reference

Inheritance diagram for TerminalError:



The documentation for this class was generated from the following file:

- core/system_ops.py

Chapter 8

File Documentation

8.1 app.py File Reference

Classes

- class DTGApp

Namespaces

- namespace app

8.2 core/config_manager.py File Reference

Namespaces

- namespace config_manager

Functions

- get_config_dir ()
- load_recent_projects ()
- save_recent_project (path)
- load_configs (project_name, container_name)
- save_configs (project_name, container_name, interface, delay_spinbox, loss_spinbox, band_spinbox, limit↩ _spinbox, win, config_status, save_btn, all_container_configs)

Variables

- str APP_NAME = "DTG"
- CONFIG_DIR = get_config_dir()
- str RECENT_PROJECTS_FILE = CONFIG_DIR / "recent_projects.json"

8.3 core/docker_ops.py File Reference

Namespaces

- namespace docker_ops

Functions

- Container get_container (DockerClient client, str container_id)
- List[Container] get_project_containers (DockerClient client, str project_name)
- List[str] get_container_interfaces (DockerClient client, str container_id)
- start_container_by_id (DockerClient client, str container_id)

- stop_container_by_id (DockerClient client, str container_id)
- restart_container_by_id (DockerClient client, str container_id)
- apply_tc_rules (DockerClient client, str container_id, str eth, str delay, str loss, str band, str limit)
- run_container_ping (DockerClient client, str container_id, str ipaddr)

8.4 core/system_ops.py File Reference

Classes

- class ComposeNotFoundError
- class DockerComposeError
- class TerminalError

Namespaces

- namespace system_ops

Functions

- exec_compose (compose_file)
- open_terminal (str container_name)

8.5 Doxyfile.txt File Reference

8.6 gui/assets.py File Reference

Namespaces

- namespace assets

Functions

- load_image (path, size=(20, 20))

Variables

- BASE_DIR = Path(sys._MEIPASS)
- str IMAGE_DIR = BASE_DIR / "images"

8.7 gui/main_window.py File Reference

Classes

- class MainWindow

Namespaces

- namespace main_window

8.8 gui/node_window.py File Reference

Classes

- class NodeWindow

Namespaces

- namespace node_window

8.9 gui/startup_window.py File Reference

Namespaces

- namespace startup_window

Functions

- choose_project_popup (parent, icons)

8.10 main.py File Reference

Namespaces

- namespace main

Variables

- root = tk.Tk()
- app = DTGApp(root)
- file

8.11 README.md File Reference

8.12 requirements.txt File Reference

8.13 utils/lock_manager.py File Reference

Classes

- class OperationLock

Namespaces

- namespace lock_manager