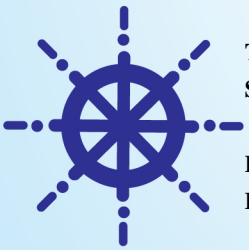
Rapport de Stage M2

Développement de composants d'une plateforme d'orchestration de services composites dans le domaine maritime

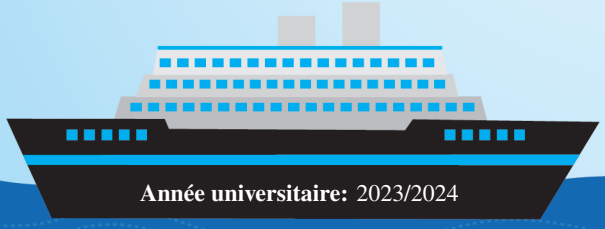
Muhammad AL QABBANI

17 Juillet à 20 Décembre 2023



Tuteur de stage: Dr. Soraya AIT-CHELLOUCHE, Dr. Yassine HADJADJ-AOUL
Superviseur Académique: Dr. César VIHO

Établissement: Université de Rennes - Master en Informatique Parcours Cloud et Réseaux
Établissement d'accueil: INRIA de l'Université de Rennes, France



Année universitaire: 2023/2024

CONTENTS:

For your convenience, a PDF version of this project is available for download :

CHAPTER

ONE

INTRODUCTION

1.1 The Concept

The Concept and Development of a Platform Enabling Automated Service Execution Placement within the Deep Edge Computing Network.

NaviDEC is an "IoT and Image Processing" platform bringing Deep-Edge Computing to the micro-edge of the network. It aims to enhance data processing and transmission efficiency in areas with limited coverage and low latency requirements. Deployed within both stationary and mobile gateways, NaviDEC will undergo testing in maritime use cases.

1.2 The Objectives

The objectives of this project are as follows:

1. Define a "deep-edge" architecture and its interfaces with standard clouds across various use cases.
2. Study and develop distributed video processing algorithms spanning the deep edge and the cloud.
3. Investigate and develop service placement and orchestration strategies between the deep edge and the cloud.
4. Research and develop a coordinated "deep-edge" security and anomaly detection system integrated with the cloud.
5. Deploy and test a prototype in real-world use cases that involve intensive data processing, particularly in the context of maritime scenarios.

1.3 Partners

CHAPTER

TWO

USER GUIDE

2.1 Repository Scope

Within this repository, the team from the Centre Inria de l'Université de Rennes is tasked with the investigation and development of service placement and orchestration strategies between the deep edge and the cloud.

The project entails the capability to manage and transfer resources seamlessly between distinct Kubernetes clusters through the utilization of REST APIs, with the goal of maintaining transparency for the end user.

2.1.1 Quickstart

Installation

Step 1: Bare-metal installation

Before deploying NaviDec, a prerequisite is to prepare the server. Your server must be running Ubuntu **20.04 LTS** and have **Python3** installed.

Firstly, initiate the installation of Ansible on a freshly installed version of Ubuntu 20.xx:

```
sudo apt-get update && sudo apt-get install ansible -y
```

Next, clone the NaviDec repository:

```
git clone -b v4-Auto https://gitlab.inria.fr/fgiarre/navidec.git
```

Lastly, run the Ansible playbook to install the required dependencies:

```
sudo ansible-playbook -i "localhost," -c local navidec/ansible/configure_server.yml &&
↪ exit
```

After the installation is finished, the current active terminal will automatically log you out. You can log back in to ensure that your group membership is re-evaluated.

Step 2: Running NaviDEC code

You can now start NaviDEC directly by following these commands:

```
cd navidec && . run.sh
```

Now, you will be well-prepared to *get started*.

Get started

Documentation

The comprehensive documentation for NaviDEC is available on GitLab INRIA.

License

Distributed under MIT License.

CHAPTER

THREE

DEVELOPER GUIDE

3.1 Ansible

3.1.1 Ansible Playbook Documentation

This document provides an overview of the Ansible playbook *configure_server.yml*.

```yaml
# ansible/configure_server.yml
---
- hosts: localhost
gather_facts: yes
become: yes
vars:
    ansible_python_interpreter: /usr/bin/python3
tasks:
    - name: Check if Ubuntu 20.xx is installed
    assert:
        that:
        - ansible_distribution == 'Ubuntu'
        - ansible_distribution_major_version | int >= 20
        msg: "Please use Ubuntu 20.xx version. Current version: {{ ansible_
↪distribution }} {{ ansible_distribution_version }}"

    - name: Add NOPASSWD rule to sudoers
    block:
        - name: Add NOPASSWD rule to sudoers
        lineinfile:
            path: /etc/sudoers
            line: '%sudo  ALL=(ALL) NOPASSWD: ALL'
        become: yes
        register: result
        ignore_errors: yes

        - name: Print success message
        debug:
            msg: "Sudo NOPASSWD rule added successfully."
        when: result is not failed

        - name: Print error message and stop execution
        block:
            - debug:
                msg: "Error occurred while modifying sudoers file."
            - meta: end_play
        when: result is failed
```

(continues on next page)

(continued from previous page)

```
    - name: Setup Current Directory and Destination Directory
    block:
        - name: Get Current Directory
        command: pwd
        register: current_dir_output
        changed_when: false
        ignore_errors: true

        - name: Debug Current Directory
        debug:
            var: current_dir_output.stdout

        - name: Debug current_dir_output.stdout
        debug:
            var: current_dir_output.stdout

        - name: Set Destination Directory
        set_fact:
            dest_directory: "{{ current_dir_output.stdout | replace('/navidec/ansible
↪', '') }}"
        changed_when: false

        - name: Debug Destination Directory
        debug:
            var: dest_directory

    - name: Copy run.sh from navidec to parent directory
    block:
        - name: Copy run.sh file
        copy:
            src: "{{ dest_directory }}/navidec/run.sh"
            dest: "{{ dest_directory }}/run.sh"
            remote_src: yes
            mode: preserve

        - name: Change owner of run.sh
        file:
            path: "{{ dest_directory }}/run.sh"
            owner: "{{ ansible_env.SUDO_USER }}"
            group: "{{ ansible_env.SUDO_USER }}"
        become: yes

    - name: Install Containernet
    block:
        - name: Check if Containernet repository already exists
        stat:
            path: "{{ dest_directory }}/containernet"
        register: containernet_repo

        - name: Remove existing Containernet repository if it exists
        when: containernet_repo.stat.exists
        file:
            path: "{{ dest_directory }}/containernet"
            state: absent
        ignore_errors: yes
```

(continues on next page)

(continued from previous page)

```
        become: yes

        - name: Clone Containernet repository
        git:
            repo: https://github.com/containernet/containernet.git
            dest: "{{ dest_directory }}/containernet"
            clone: yes
            update: yes
        become: yes

        - name: Remove existing 'openflow' directory if it exists
        file:
            path: "{{ dest_directory }}/openflow"
            state: absent
        ignore_errors: yes
        become: yes

        - name: Run Containernet install playbook
        shell: ansible-playbook -i localhost, -c local "{{ dest_directory }}/
↪containernet/ansible/install.yml"
        become: yes
        register: containernet_output

        - name: Display Containernet install playbook output
        debug:
            var: containernet_output.stdout_lines

    - name: Update package list and Install packages
    apt:
        name: sshpass, latexmk, texlive-fonts-recommended, texlive-latex-extra,␣
↪texlive-fonts-extra
        update_cache: yes
        state: present
    become: yes

    - name: Install Kubectl
    block:
        - name: Download Kubectl
        get_url:
            url: "https://dl.k8s.io/release/{{ lookup('url', 'https://dl.k8s.io/
↪release/stable.txt').split('\n')[0] }}/bin/linux/amd64/kubectl"
            dest: "{{ dest_directory }}/kubectl"
        become: yes

        - name: Install Kubectl
        command: >
            install "{{ dest_directory }}/kubectl" kubectl
        args:
            chdir: /usr/local/bin
        become: yes

        - name: Remove Kubectl setup file
        file:
            path: "{{ dest_directory }}/kubectl"
            state: absent
        become: yes
```

(continues on next page)

(continued from previous page)

```
    - name: Install Sysbox
    block:
        - name: Download Sysbox
        get_url:
            url: "https://downloads.nestybox.com/sysbox/releases/v0.6.2/sysbox-ce_0.6.
↪2-0.linux_amd64.deb"
            dest: "{{ dest_directory }}/sysbox-ce_0.6.2-0.linux_amd64.deb"
        become: yes

        - name: Install jq
        apt:
            name: jq
            state: present
        become: yes

        - name: Install Sysbox
        apt:
            deb: "{{ dest_directory }}/sysbox-ce_0.6.2-0.linux_amd64.deb"
        become: yes

        - name: Remove Sysbox setup file
        file:
            path: "{{ dest_directory }}/sysbox-ce_0.6.2-0.linux_amd64.deb"
            state: absent
        become: yes

    - name: Check Sysbox Service Status
    block:
        - name: Get Sysbox service status
        shell: systemctl is-active sysbox
        register: sysbox_status
        changed_when: false

        - name: Print Sysbox service status
        debug:
            msg: "Sysbox service is {{ sysbox_status.stdout }}"

    - name: Update Docker daemon and restart it
    block:
        - name: Insert line into file
        ansible.builtin.lineinfile:
            path: /etc/docker/daemon.json
            insertafter: '"bip": "172.20.0.1/16",'
            line: '    "default-runtime": "sysbox-runc",'
            state: present
        become: yes

        - name: Restart docker
        ansible.builtin.systemd:
            name: docker
            state: restarted
        become: yes

    - name: Install pip3 and required packages
    command: pip3 install psutil networkx pika fastapi uvicorn requests rich sphinx␣
```

(continues on next page)

(continued from previous page)

```
↪sphinx-rtd-theme sphinx-sitemap myst-parser linkify-it-py
    become: yes

    - name: Install Node.js
    block:
        - name: Download Node.js setup script
        get_url:
            url: "https://deb.nodesource.com/setup_20.x"
            dest: "{{ dest_directory }}/nodesource_setup.sh"
        become: yes

        - name: Run Node.js setup script
        command: "bash {{ dest_directory }}/nodesource_setup.sh"
        become: yes

        - name: Install Node.js
        apt:
            name: nodejs
            state: present
        become: yes

        - name: Remove Node.js setup script
        file:
            path: "{{ dest_directory }}/nodesource_setup.sh"
            state: absent
        become: yes

        - name: Check Node.js version
        command: "node --version"
        register: node_version
        changed_when: false
        become: yes

        - name: Print Node.js version
        debug:
            msg: "Node.js version is {{ node_version.stdout }}"

    - name: Install nodemon
    block:
        - name: Upgrade npm using sudo
        command: "npm install -g npm"
        become: yes
        ignore_errors: yes

        - name: Print npm upgrade result
        debug:
            msg: "npm upgraded successfully."
        when: node_version.stdout is defined and "npm" not in node_version.stdout

        - name: Install nodemon using sudo
        command: "npm install -g nodemon"
        become: yes
        ignore_errors: yes

        - name: Print nodemon installation result
        debug:
```

(continues on next page)

(continued from previous page)

```
            msg: "nodemon installed successfully."
        when: node_version.stdout is defined and "nodemon" not in node_version.stdout

        - name: Install nodemon using npm
        command: "npm install -g nodemon"
        become: yes
        ignore_errors: yes

        - name: Print nodemon installation result
        debug:
            msg: "nodemon installed successfully."
        when: node_version.stdout is defined and "nodemon" not in node_version.stdout

    - name: export COLORTERM=truecolor to current user
    block:
        - name: Get the current user
        set_fact:
            current_user: "{{ ansible_env.SUDO_USER | default(ansible_env.USER) }}"

        - name: Print the username
        ansible.builtin.debug:
            msg: "Current user is: {{ current_user }}"

        - name: Check if '#export COLORTERM=truecolor' is in ~/.bashrc
        command: grep -q "#export COLORTERM=truecolor" "/home/{{ current_user }}/.
↪bashrc"
        register: grep_result
        ignore_errors: yes

        - name: Remove '#' from the line if it exists
        replace:
            path: "/home/{{ current_user }}/.bashrc"
            regexp: '^#export COLORTERM=truecolor'
            replace: 'export COLORTERM=truecolor'
        when: grep_result.rc == 0

        - name: Check if export COLORTERM=truecolor is in ~/.bashrc
        command: grep -q "export COLORTERM=truecolor" "/home/{{ current_user }}/.
↪bashrc"
        register: grep_result
        ignore_errors: yes

        - name: Add export COLORTERM=truecolor to ~/.bashrc if not present
        lineinfile:
            path: "/home/{{ current_user }}/.bashrc"
            line: "export COLORTERM=truecolor"
        when: grep_result.rc != 0


    - name: Create docker group & add current user to it
    block:
        - name: Get the current non root remote user
        set_fact:
            remote_regular_user: "{{ ansible_env.SUDO_USER }}"

        - name: Print the username
```

(continues on next page)

(continued from previous page)

```
        ansible.builtin.debug:
            msg: "Current user is: {{ remote_regular_user }}"

        - name: Add docker group
        ansible.builtin.group:
            name: docker
            state: present

        - name: Add user to docker group
        ansible.builtin.user:
            name: "{{ remote_regular_user }}"
            groups: docker
            append: yes
```

Playbook Overview

The *configure_server.yml* playbook is designed to configure a server. It performs the following tasks:

1. Ensure that the server is running Ubuntu 20.xx.
2. Add an NOPASSWD rule to the sudoers file.
3. Set up the current directory and destination directory.
4. Copy the *run.sh* script from the *navidec* directory to the parent directory.
5. Install Containernet if it's not already installed.
6. Update the package list and install required packages.
7. Install Kubectl.
8. Install Sysbox.
9. Check the status of the Sysbox service.
10. Update the Docker daemon and restart it.
11. Install Python packages using pip3.
12. Install Node.js and nodemon.
13. Export *COLORTERM=truecolor* to the current user's environment.
14. Create a docker group and add the current user to it.

Please refer to the playbook source code for detailed task configurations.

For more information on Ansible, refer to the official documentation: https://docs.ansible.com/

For information on YAML syntax, refer to the YAML documentation: https://yaml.org/

Playbook Usage

You can run the playbook using the following command:

Firstly, initiate the installation of Ansible on a freshly installed version of Ubuntu 20.xx:

```
sudo apt-get update && sudo apt-get install ansible -y
```

Next, clone the NaviDec repository:

```
git clone -b v4-Auto https://gitlab.inria.fr/fgiarre/navidec.git
```

Lastly, run the Ansible playbook to install the required dependencies:

```
sudo ansible-playbook -i "localhost," -c local navidec/ansible/configure_server.yml &&
↪ exit
```

After the installation is complete, the current active terminal session will automatically log you out. You can log back in to ensure that your group membership is re-evaluated.

Note: This playbook assumes that you have Ansible installed on your system.

Warning: Be cautious when making changes to system configurations, especially when using *become* and *sudo* privileges.

For more information on using Ansible playbooks, consult the Ansible documentation.

Playbook Author

Author: INRIA de l'Université de Rennes, IRISA

3.2 API Documentation

3.2.1 Emulations

Activate Ping Network Module

emulations.activate_ping_network.**activate_network**(*d1*, *d2*, *broker*)

Activate network interfaces between containernet devices.

This function activates network interfaces for three docker containers: d1, d2, and broker.

Parameters:

- d1 (mininet.node.Host): The first docker container.
- d2 (mininet.node.Host): The second docker container.
- broker (mininet.node.Host): The broker docker container.

Returns: None

emulations.activate_ping_network.**check_drop_rate**(*net*)

Check the network reachability by pinging all hosts.

This function checks the network reachability by pinging all hosts in the network and calculates the percentage of dropped packets. If the drop rate is not zero, it terminates the process.

Parameters: - net (mininet.net.Mininet): The Mininet network object.

Returns: None

This module provides functions to activate network interfaces and check network reachability through ping tests between Mininet Docker containers.

Bandwidth Module

This module handles the continuous update of bandwidth values and provides a thread for this purpose.

Check Readiness Module

class emulations.check_readiness.**DockerLogsThread**(*container*)

Bases: Thread

A thread class for displaying Docker logs.

This class is used to create a thread that displays the logs of a specified container. It provides methods to start the thread, stop it gracefully, and check if it has stopped.

Attributes:

- container: The Docker container object.

run()

Run the thread.

This method displays the logs of the specified container and terminates the log process when the thread is stopped.

stop()

Stop the thread gracefully.

This method sets the internal stop event to stop the thread.

stopped()

Check if the thread has stopped.

Returns: True if the thread has stopped, False otherwise.

emulations.check_readiness.**check_kubectl_ready**(*container*)

Check the readiness of a Kubernetes container.

This function checks the readiness of a Kubernetes container by executing the "kubectl get all" command repeatedly and waiting for the "ClusterIP" status to appear in the output.

Parameters:

- container: The Mininet Docker container object running Kubernetes.

Returns: None

emulations.check_readiness.**check_rabbitmq_process**()

Check the status of the RabbitMQ server process.

This function checks if the RabbitMQ server process is running inside a specified Docker container.

Returns: None

This module provides functions to check the readiness of Docker containers and processes running inside them, as well as to display Docker container logs.

Find Node.js and npm Paths Module

emulations.find_node_npm_paths.**find_node_npm_paths**()

Find and set the paths to the Node.js and npm commands.

This function uses the *which* command to locate the paths to the Node.js and npm commands on the system and stores them in the *variable.node_path* and *variable.npm_path* variables respectively.

Parameters: None

Returns: None

This module provides a function to find and set the paths to the Node.js and npm commands on the system.

Get H1 Host PID Module

emulations.get_h1_host_pid.**get_h1_host_pid**(*net*, *host_name*)

Get the process ID (PID) of a specified host in the network.

This function takes the Mininet network object *net* and the name of the host *host_name* as input and returns the process ID (PID) of the specified host. It iterates through the list of hosts in the network and matches the host by name.

Parameters:

- net (mininet.net.Mininet): The Mininet network object.
- host_name (str): The name of the host for which the PID is to be retrieved.

Returns: int or None: The PID of the specified host, or None if the host was not found.

This module provides a function to get the process ID (PID) of a specified host in the network.

Get Variables Module

emulations.get_variables.**current_directory**()

Extract and print the current working directory.

This function extracts the current working directory and stores it in the *variable.current_directory* variable. It then prints the current directory path.

Returns: None

emulations.get_variables.**home_path**()

Extract and print the home directory path.

This function extracts the home directory path and stores it in the *variable.home_path* variable. It then prints the home directory path.

Returns: None

emulations.get_variables.**os_path**()

Extract and print the PATH environment variable.

This function extracts the PATH environment variable and stores it in the *variable.os_path* variable. It then prints the value of the PATH variable.

Returns: None

emulations.get_variables.**username**()

Extract and print the username.

This function extracts the username and stores it in the *variable.username* variable. It then prints the username.

Returns: None

emulations.get_variables.**volume_path**()

Extract and print the volume path.

This function extracts the volume path and stores it in the *variable.volume_path* variable. It then prints the volume path.

Returns: None

This module provides functions to extract and print various environment and system variables.

Kill Node Server Module

emulations.kill_node_server.**kill_node_server**(**ports*)

Kill processes listening on specified ports.

This function takes a variable number of port numbers as arguments and attempts to terminate processes that are listening on those ports. It sends a SIGTERM signal to the processes to gracefully terminate them.

Parameters:

*ports (int): Variable number of port numbers to kill processes on.

Returns: None

This module provides a function to terminate processes listening on specified ports.

Kill Process on Ports Module

emulations.kill_process_on_ports.**kill_process_on_ports**(**ports*)

Kill processes listening on specified ports.

This function takes a variable number of port numbers as arguments and attempts to terminate processes that are listening on those ports. It uses the 'lsof' command to identify the processes and sends a 'kill' command to terminate them.

Parameters:

*ports (int): Variable number of port numbers to kill processes on.

Returns: None

This module provides a function to terminate processes listening on specified ports.

Minikube Module

class emulations.minikube.**PortForwardProcess**(*container_name: str*, *event: Event*, *commands: list*, *pod_names: list*, *label_selectors: list*, *namespaces: list*)

Bases: Process

A multiprocessing Process class for managing port forwarding and pod readiness checks.

Args:

- container_name (str): The name of the Docker container.
- event (multiprocessing.Event): An event to signal when port forwarding is complete.
- commands (list): A list of kubectl port-forward commands to execute.
- pod_names (list): A list of pod names corresponding to the commands.
- label_selectors (list): A list of label selectors for pods to check readiness.
- namespaces (list): A list of namespaces for the pods.

Methods:

- stop(): Stop the port forwarding process.
- check_pod_status(label_selector: str, namespace: str) -> bool: Check if pods are ready.

check_pod_status(*label_selector: str*, *namespace: str*) → bool

run()

Method to be run in sub-process; can be overridden in sub-class

stop()

emulations.minikube.**enable_minikube_addons**(*container_name: str*)

Enable Minikube addons for metrics-server, dashboard, ingress, ingress-dns, and list enabled addons.

Args: - container_name (str): The name of the Docker container.

Returns: None

emulations.minikube.**get_minikube_ip**(*container_name: str*)

Get the IP address of the Minikube cluster.

Args: - container_name (str): The name of the Docker container.

Returns: str: The IP address of the Minikube cluster.

emulations.minikube.**get_public_port**(*container_name: str*, *private_port: int*)

Get the public port associated with a private port in a Docker container.

Args: - container_name (str): The name of the Docker container. - private_port (int): The private port number.

Returns: int: The public port number, or None if not found.

emulations.minikube.**kubectl_config**(*container_name: str*, *public_port: int*)

Configure kubectl to connect to the Minikube cluster.

Args:

- container_name (str): The name of the Docker container.
- public_port (int): The public port of the Minikube cluster.

Returns: None

emulations.minikube.**setup_prometheus_monitoring**(*container_name: str*)

Set up Prometheus monitoring in the Kubernetes cluster.

Args: - container_name (str): The name of the Docker container.

Returns: None

emulations.minikube.**skooner_get_deployment_token**(*container_name: str*)

Get the Skooner deployment token.

Args: - container_name (str): The name of the Docker container.

Returns: str: The Skooner deployment token.

emulations.minikube.**start_kubectl_proxy**(*event2*)

Start kubectl proxy in the background.

Args: - event2 (multiprocessing.Event): An event to signal when kubectl proxy is running.

Returns: multiprocessing.Process: The process running kubectl proxy.

emulations.minikube.**terminate_kubectl_proxy**()

Terminate the kubectl proxy process.

Returns: None

This module provides functions for managing Minikube, enabling addons, setting up Prometheus monitoring, getting Skooner deployment tokens, configuring kubectl, and starting kubectl proxy.

Start Bandwidth API Server Module

emulations.start_bandwidthapi_server.**start_bandwidthapi_server**()

Start the bandwidth API server.

This function activates a virtual environment, changes the current working directory to the 'bandwidthapi' folder, and starts the FastAPI server using uvicorn in the background. The stdout and stderr of the server are redirected to log files. It also checks for the readiness of the server and logs any errors.

Global Variables:

- bandwidthapi_server_pid (int): The process ID of the bandwidth API server.

Returns: None

This module contains a function to start the bandwidth API server. It activates a virtual environment, changes the current working directory to the 'bandwidthapi' folder, and starts the FastAPI server using uvicorn in the background. The stdout and stderr of the server are redirected to log files. It also checks for the readiness of the server and logs any errors.

Start Bandwidth Visualization Server Module

emulations.start_bandwidthvisualization_server.**start_bandwidthvisualization_server**()

Start the bandwidth visualization server.

This function starts the bandwidth visualization server. It changes the current working directory to the 'bandwidthvisualization' folder, runs the 'npm ci' command, and starts the server in the background using 'nodemon'. It also monitors the server's stdout and stderr for output.

Global Variables:

- bw_server_pid (int): The process ID of the bandwidth visualization server.

Returns: None

This module provides a function to start the bandwidth visualization server for monitoring network bandwidth. It changes the working directory, installs

Start xterm.js Server Module

emulations.start_xtermjs_server.**start_xtermjs_server**()

Start the xterm.js server for terminal emulation.

This function starts the xterm.js server for terminal emulation. It changes the current working directory to the xterm.js folder, runs the *npm ci* command, and starts the server in the background using *nodemon*. It also monitors the server's stdout and stderr for output.

Global Variables:

- server_pid (int): The process ID of the xterm.js server.

Returns: None

This module contains a function to start the xterm.js server for terminal emulation.

System Info Module

This module provides information about the system and its resources.

Variable Module

emulations.variable.**init**()

Initialize global variables.

This function initializes global variables used in the emulation environment to default None values.

Global Variables:

- h1_pid (int): The PID of host h1.
- node_path (str): The path to the Node.js executable.
- npm_path (str): The path to the npm (Node Package Manager) executable.
- server_pid (int): The PID of the Node.js server process.
- bw_server_pid (int): The PID of the bandwidth control server process.
- os_path (str): The PATH environment variable.
- current_directory (str): The current working directory.
- home_path (str): The path to the user's home directory.
- volume_path (str): The name of the volume used in Docker containers.
- username (str): The username of the current user.
- public_port (int): The public port for exposing services.
- skooner_token (str): The Skooner authentication token.

Returns: None

This module defines and initializes global variables used in the emulation environment.

3.2.2 Windows Chrome

Launch Chrome Module

windows_chrome.launch_chrome.**launch_chrome**()

Launches the Chrome browser with specified URLs on remote Windows machines.

This function is designed to launch the Chrome browser on remote Windows machines, PC1 or PC2, with a set of predefined URLs. The behavior of this function depends on the hostname of the Linux machine where it is run.

URLs to open in Chrome:

- Xterm.js node server: http://<Linux-Hostname>:3001
- Bandwidth Visualization node server: http://<Linux-Hostname>:3002
- FastAPI Swagger UI for Bandwidth Speed API: http://<Linux-Hostname>:8000/docs
- Skooner Dashboard: http://<Linux-Hostname>:4654
- RabbitMQ UI: http://<Linux-Hostname>:8080

Args:
None

Returns:
None

This module provides a function to launch the Chrome browser on remote Windows machines with predefined URLs. The behavior of the function depends on the hostname of the Linux machine where it is run. It can launch Chrome on PC1 or PC2 with the specified URLs.

CHAPTER

FOUR

NAVIDEC LICENSE

NaviDEC is released under the 3-clause BSD license with the following terms:

Copyright (c) [2023] [INRIA de l'Université de Rennes, IRISA]

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of Continuum Analytics, Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL CONTINUUM ANALYTICS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CHAPTER

FIVE

DOCUMENTATION INDICES

5.1 General Index

5.2 Module Index

PYTHON MODULE INDEX

e

w

INDEX

O

P

R

S

T

U

V

W