



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078.

Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare



Deliverable 7.7
Report on implementation of assessment, requirement criteria, and “stress testing”

Work Package 7
Integrated FeatureCloud health informatics platform and app store

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 826078. Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

© FeatureCloud Consortium, 2023

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.

Document information

Grant Agreement Number: 826078			Acronym: FeatureCloud	
Full title	Privacy preserving federated machine learning and blockchaining for reduced cyber risks in a world of distributed healthcare			
Topic	Toolkit for assessing and reducing cyber risks in hospitals and care centres to protect privacy/data/infrastructures			
Funding scheme	RIA - Research and Innovation action			
Start Date	1 January 2019	Duration	60 months	
Project URL	https://featurecloud.eu/			
EU Project Officer	Christos Maramis, Health and Digital Executive Agency (HaDEA)			
Project Coordinator	Jan Baumbach, University of Hamburg (UHAM)			
Deliverable	D7.7 – Report on implementation of assessment, requirement criteria, and “stress testing”			
Work Package	WP7 – Integrated FeatureCloud health informatics platform and app store			
Date of Delivery	Contractual	31/12/2023	Actual	22/12/2023
Nature	Report	Dissemination Level	Public	
Lead Beneficiary	UHAM			
Responsible Author(s)	Mohammad Bakhtiari (UHAM), Jan Baumbach (UHAM)			
	Dominik Heider (UHAM), Rudolf Mayer (SBA)			
Keywords	Requirement criteria, stress testing, platform stability			

History of changes

Version	Date	Contributions	Contributors (name and institution)
V0.1	06/12/2023	First draft	Mohammad Bakhtiari (UHAM)
V0.2	18/12/2023	Comments	Rudolf Mayer (SBA) Dominik Heider (UMR)
V1.0	22/12/2023	Final version	Mohammad Bakhtiari (UHAM) Nina Donner (concentris) Jan Baumbach (UHAM)

Table of Contents

1 Table of acronyms and definitions	6
2 Objectives of the deliverable based on the Description of Action (DoA)	8
3 Executive Summary	8
4 Introduction (Challenge)	8
5 Methodology	9
5.1 Stress testing	9
5.2 Tools and scripts	10
5.2.1 Test workflow	10
5.2.2 Communication test app	11
5.2.3 Test Environment	11
5.2.4 Security testing	12
5.3 Designed tests	13
5.3.1 FeatureCloud pip package	13
5.3.2 FeatureCloud Controller	16
5.3.3 Web Security and Workflow Execution Platform	18
6 Results	20
6.1 Implementation of assessment and requirement criteria	20
6.2 Pip package	21
6.2.1 Execution	21
6.2.2 Communication memo	26
6.3 Controller	36
6.4 Web Security	39
6.4.1 Unauthenticated user access	39
6.4.2 Common Vulnerabilities and Exposures	41
6.4.3 Brute-force attack	43
6.4.4 Manipulate the confidentiality and integrity of data	45
6.4.5 SSL Stripping attacks	45
7 Open issues	47
8 Deviations	47
9 Conclusion	47
10 Other supporting documents / figures / tables	48
10.1 Communication test app	48
10.2 stress testing the Controller	49
10.2.1 Controller Fails the stress test	49
10.2.2 Controller survives the stress test	50
10.3 Aira Server Test Environment	51
10.4 Stress Testing the Pip Package	55
10.4.1 Incorporation of the feedback into the pip package	55
10.4.2 Data misplacement	60
10.4.3 Simple-SMPC Pickle error	64
10.4.4 Binary-Text Data Decoding Error	67

10.4.5 Pickle Unpickling Stack Underflow Error	74
10.5 Controller’s Dockerfile	86

1 Table of acronyms and definitions

API	application programming interface
App	application
CI/CD	Continuous Integration/Continuous Deployment
CLI	command-line interface
concentris	concentris research management gmbh
CPU	central processing unit
CSRF	Cross-Site Request Forgery
CVE	Common Vulnerabilities and Exposures
D	Deliverable
DDoS	Distributed Denial of Service
DoA	Description of Action
DP	Differential Privacy
FL	federated learning
GB	gigabyte
GHz	gigahertz
GPU	graphics processing unit
HSTS	HTTP Strict Transport Security
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation (a text-based format)
ID	identification (number)
IP	Internet Protocol (address)
KPI	key performance indicator
ML	machine learning
MS	Milestone
MUG	Medizinische Universität Graz
OS	operating system
Patients	In this deliverable, we use the term “patients” for all research subjects. In FeatureCloud, we will focus on patients, as this is already the most vulnerable case scenario and this is where most primary data is available to us. Admittedly, some research subjects participate in clinical trials but not as patients but as healthy individuals, usually on a voluntary basis and are therefore not dependent on the physicians who care for them. Thus, to increase readability, we simply refer to them as “patients”.
RAM	random access memory
RSA	Rivest–Shamir–Adleman (a quick encryption and decryption technique)
SBA	SBA Research Gemeinnützige GmbH
SMPC	Secure Multi-Party Computation
SQL	Structured Query Language
SSD	solid-state drive

SSL	Secure Sockets Layer (security technology for encrypting data sent between a website and a browser or between two servers)
TB	terabyte
TLS	Transport Layer Security
UHAM	University of Hamburg
UI	user interface
UMR	Philipps Universität Marburg
URL	Uniform Resource Locator
WP	Work package
XSS	cross-site scripting

2 Objectives of the deliverable based on the Description of Action (DoA)

This deliverable, reports on implementation of assessment, requirement criteria, and “stress testing”, is closely related to the task 5 of WP 7, “Stress testing”, where we stress test different FeatureCloud platform components. Meanwhile, the successful stress tests entail the incorporation of test feedback into the platform to improve its performance. Accordingly, this deliverable touches task 1, Programming interfaces and platform, due to modification of the pip package. Moreover, the deliverable contributes to task 2, “App store and workflow management”, and task 3, User interfaces and testing, with respect to the applied modifications as a result of stress testing the controller, Docker registry and user interfaces.

3 Executive Summary

This deliverable reports on the successful stress testing of the FeatureCloud platform. We designed tests to stress the platform while following the principles in the literature (see section 5.1). Accordingly, we designed tests with various scenarios for stress testing of the pip package (see section 5.3.1 and 6.2), and elaborated the test scenarios for the Controller (see section 5.3.2 and 6.3), and security (see section 5.3.3 and 6.4). We analyzed the outcomes and logs of the tests and explained the incorporation of test feedback, and its effect on the FeatureCloud platform (see section 6) to be resilient and robust against failure.

4 Introduction (Challenge)

FeatureCloud, a pioneering platform in federated data analysis and computation, integrates various applications from its app store into seamless workflows, crucial for secure data processing involving multiple stakeholders. The platform's front-end, with a user-friendly interface, and back-end are responsible for handling data and workflow coordination, which makes them central to its functionality. In this complex environment, stress testing becomes vital, ensuring both parts work flawlessly under varied, intensive scenarios.

In stress testing FeatureCloud, a significant emphasis is placed on web security. This involves employing advanced tools like Burp Suite, Trivy, and Nessus to simulate real-world threats such as Cross-Site Request Forgery (CSRF) vulnerabilities, Transport Layer Security (TLS) protocol weaknesses, and Secure Sockets Layer (SSL) stripping attacks. By testing the platform's resilience against various threats, we ensure the integrity and confidentiality of data transmissions, a cornerstone for trust in our federated computing environment.

Additionally, we stress test the FeatureCloud controller and pip package using specialized stress tests. For the controller, the focus is on its capability to manage workflows efficiently and securely, even under extreme conditions. The pip package, essential for developers in creating federated applications, is tested for its robustness in data communication and handling. This thorough approach in stress testing different components of FeatureCloud not only highlights potential vulnerabilities but also aids in continuously improving the platform's performance and security, maintaining its position as a reliable and efficient tool in the realm of federated computing.

5 Methodology

In this section, we detail our approach to stress testing the FeatureCloud platform, focusing on thoroughly evaluating its diverse components. We establish a realistic test environment and methodically analyze the results. Our process involves pinpointing workload profiles, defining specific objectives, and choosing relevant metrics, all tailored to different aspects of FeatureCloud. By employing a variety of stress tests, such as application-specific and systemic, we aim to identify and fortify any potential vulnerabilities in each component, ensuring their robustness in challenging scenarios.

5.1 Stress testing

Stress testing is a critical phase in ensuring an application's robustness and reliability. It begins with identifying the test environment, which involves analyzing the hardware, software, networks, and other system configurations where the application will run. Understanding these elements is vital to create a realistic test environment that closely mirrors the production setting. Once the environment is established, the next step is to outline clear objectives for the test. This includes defining what success looks like, such as specific performance thresholds or stability criteria. These objectives should be measurable and closely aligned with user expectations and business requirements. An objective is defined to stress the software with intention of making it fail.

With workload testing objectives, the next task is to determine the workload profile. This involves identifying the various types of user interactions that the application will experience, and simulating these actions to assess how the application behaves under varying levels of stress. After defining the objectives, metrics must be identified to evaluate the application's performance. These metrics could include response times, throughput rates, and error percentages. They will serve as indicators of the application's behavior under stress and help pinpoint areas that require optimization.

Based on the metrics and objectives, detailed test cases can be designed and implemented to satisfy the objectives. The tests are scripts or procedures that simulate specific user actions or series of actions within the application. Each test case should have defined expected outcomes to judge whether the application behaves as anticipated under stress conditions. For workload testing, the test involves simulating the load based on the test cases, where the application is put through the paces of the simulated workload, often using automated tools to apply the stress. The load should be monitored to ensure it reflects the desired stress levels without unintentionally overwhelming the load generators themselves.

Lastly, the results from these tests are thoroughly analyzed. This analysis is compared against the predetermined objectives and metrics to assess whether the application meets the desired performance standards or it fails. If performance issues are detected, further investigation into code, design, or system configurations may be needed to address these bottlenecks. Throughout the process, stress testing can take several forms: Application stress testing for uncovering defects related to the overall application's interactions, transactional stress testing to focus on specific components or transactions, and systemic stress testing to evaluate the application's behavior in a multi-application environment. Additionally, exploratory stress testing may be employed to investigate how the system responds to unlikely or extreme scenarios, thereby providing insights into potential areas of improvement that aren't obvious through structured testing alone.

5.2 Tools and scripts

In this section, we will explore the tools and methodologies employed for stress testing the FeatureCloud platform. These include the Test Workflow for simulating complex user scenarios, the Communication Test App for assessing the system's response to diverse data types and network conditions, and a detailed Test Environment script for evaluating server capabilities.

5.2.1 Test workflow

The FeatureCloud platform enables end-users and developers to construct and execute linear workflows through its front-end website. Users can sequentially run various apps, where the output of one app serves as input for the next. This requires that the output data is formatted and extended properly for subsequent apps. Workflows can include multiple clients (data owners) collaborating in a federated manner using one or more applications. Although the online workflow meets end-user needs, developers often face challenges in implementing non-linear, flexible workflows for innovative solutions. These developers range from app developers extending existing centralized applications to researchers addressing federated challenges, necessitating arbitrary operations on data for research purposes. The FeatureCloud test workflow is introduced for developers to create arbitrary workflows for research, which are not supported in the app store. Once a final solution is developed, it must be presented as an app for inclusion in the broader workflow ecosystem.

The FeatureCloud platform includes a Controller class within its pip package, essential for developing test apps and workflows. This class facilitates basic interactions with the FeatureCloud controller, enabling operations like start, stop, delete, list, traffic, logs, and info on test runs. Developers can run multiple controller instances on the same machine, enhancing the flexibility and scope of app development. The TestApp class addresses the needs of individual apps within a workflow, offering features for data handling, result extraction, and interaction with the controller. Additionally, the TestWorkFlow class provides an abstract framework for creating general workflows, requiring developers to implement methods for registering apps and running workflows. This class maintains a list of TestApp instances, along with a default results directory for extracted data useful for subsequent apps. Developers looking to implement specific workflows must extend the TestWorkFlow class, implementing 'register_apps' and 'run' methods. Workflows can be executed using FeatureCloud's pip package with super-user access, ensuring data and results are appropriately managed in the controller's data directory.

The Test Workflow in FeatureCloud offers a unique opportunity for developers to conduct comprehensive stress tests across various components of the platform, such as the pip package, controller, and relay server. By designing and implementing non-linear and complex workflows within the Test Workflow framework, developers can simulate real-world scenarios and heavy loads. This approach enables them to evaluate the performance and robustness of individual components under stress. For example, creating workflows with a high number of client interactions or data-intensive tasks can test the scalability and reliability of the relay server and the controller's ability to manage multiple concurrent operations. Similarly, by utilizing a variety of functions provided by the FeatureCloud pip package within these test workflows, developers can assess the package's stability and responsiveness under different stress conditions. This process not only helps in identifying potential bottlenecks or weaknesses in the system but also aids in ensuring that the platform remains efficient and reliable even when faced with demanding scenarios typical in federated computing environments. Accordingly, in the test designed to stress test the FeatureCloud platform we will implement test workflows to run the designed test in federated simulated scenarios.

5.2.2 Communication test app

The Communication Test App¹ is a FeatureCloud application developed for stress testing. It embodies a comprehensive and modular approach to evaluating the resilience and efficiency of the relay server, controller, and pip package. The application's architecture, as shown in Appendix A11.1, offers a sophisticated setup that enables a variety of stress testing scenarios. The application conducts different stress testing scenarios which are applicable to the stress testing of the pip package, particularly in terms of communicating different data types and formats, e.g., nested data types. The communications can pass through the relay server, enabling us to stress test it using different data volumes from varying numbers of clients. Specifically, the following factors contribute to executing different stress testing scenarios in the app:

- Scenario Selection (sub_Scen in config.yml): This allows for choosing between different custom scenarios, each likely simulating unique operational conditions or data processing tasks. Depending on the scenario, the application might execute different algorithms, handle data differently, or simulate specific cloud computing challenges.
- Secure Multi-Party Computation (SMPC) (smpc_ in config.yml): This tests the application's behavior with SMPC enabled or disabled, crucial for applications requiring collaborative computations while preserving data privacy.
- Differential Privacy (DP) (dp_ in config.yml): This evaluates the application's integration and performance with Differential Privacy mechanisms. The DP engine, implemented inside the controller, adds noise to data or query results to ensure individual data privacy while still providing useful aggregate information.
- Maximum Number of Sub-Scenarios (max_n_sub_scen in config.yml): This sets the limit for variations in testing within each selected scenario, affecting the depth and breadth of stress testing. It allows us to vary data types, computational complexities, or interaction patterns to rigorously test application resilience.
- Traffic Test (traffic_test in config.yml): Focused on assessing the application's performance under high data traffic conditions, it enables heavy data transfer operations, both upload and download, to test network and processing capabilities under load.
- Data Size for Traffic Test (Data_Size in config.yml): This specifies the volume of data used in traffic tests, allowing simulation of different load intensities. Meanwhile, large data sizes test the application's ability to handle, process, and transfer large datasets efficiently.

In summary, the FeatureCloud Communication Test App is designed to robustly test various aspects of the platform, from handling network traffic and data processing on a relay server to orchestrating complex scenarios via a controller and the pip package. This comprehensive approach ensures that the application can be rigorously tested under a wide range of conditions, making it an invaluable tool in assessing and enhancing the reliability and efficiency of the platform.

5.2.3 Test Environment

For describing the test environment for different tests, we implemented a shell script to provide a comprehensive overview of a Linux servers' test environment. It displays the server's hostname, operating system details, kernel version, central processing unit (CPU) specifications, memory usage, disk space utilization, network interfaces and internet protocol (IP) addresses, system load and uptime, currently running processes, available system updates, and currently logged-in users. This information is crucial for understanding the server's capabilities and state, which is essential for planning and conducting stress tests.

```
#!/bin/bash
```

¹ <https://github.com/FeatureCloud/Communication-Test>

```
# Display the hostname of the server
echo "Hostname:"
hostname

# Show the Linux distribution and version
echo -e "\nOperating System Info:"
cat /etc/*release

# Display Kernel information
echo -e "\nKernel Info:"
uname -r

# Show CPU information like model, cores, and speed
echo -e "\nCPU Info:"
lscpu

# Display Memory information
echo -e "\nMemory Info:"
free -h

# Show Disk usage
echo -e "\nDisk Usage:"
df -h

# Display network interfaces and their IPs
echo -e "\nNetwork Interfaces:"
ip addr

# Show current system load and uptime
echo -e "\nSystem Load and Uptime:"
uptime

# Check for available updates
echo -e "\nAvailable System Updates:"
apt list --upgradable 2>/dev/null | grep -v 'Listing...'

# Show current users logged in
echo -e "\nCurrent Logged-In Users:"
who
```

5.2.4 Security testing

In the security assessment of the FeatureCloud platform, we focused on the web interface and the workflow execution. We utilized a set of well-known tools to identify and analyze potential vulnerabilities. These tools, selected for their relevance and effectiveness in various aspects of security testing, were instrumental in examining different facets of the platform’s web service. We

will briefly outline each of these tools - Burp Suite², Nessus³, and Trivy⁴ - and their general applications in the context of our stress testing process:

- Burp Suite (Application Security Testing Software) is an application security testing software that provides a range of tools for performing security testing of web applications. It includes functionalities for automated scanning, manual testing, and exploitation of web application vulnerabilities. In the FeatureCloud platform’s assessment, Burp Suite was used for tests like probing for Cross-Site Request Forgery (CSRF) vulnerabilities, testing for SSL stripping attacks, and conducting penetration tests on endpoints or authentication mechanisms. Its ability to intercept and modify Hypertext Transfer Protocol (HTTP) requests makes it particularly useful for such tests.
- Nessus Vulnerability Scanner, developed by Tenable, is a widely-used network vulnerability scanner. It detects and reports potential vulnerabilities in networked systems, including unpatched software, misconfigurations, and exposure to common exploits. In the stress testing of the FeatureCloud platform, Nessus was instrumental in scanning the network for vulnerabilities, particularly for assessing the platform’s TLS protocol usage, encryption practices, and overall configuration and credential security.
- Trivy (Open Source Security Scanner for Vulnerability) is an open-source tool designed for comprehensive vulnerability scanning. Its primary functionality is to detect vulnerabilities in container images, file systems, and even source code repositories. Trivy can be used for identifying a wide range of security issues, including operating system (OS)-level vulnerabilities and application dependencies. In the context of the FeatureCloud platform assessment, Trivy employed to scan container images or file systems used by the platform, aiding in the identification of known vulnerabilities that might impact the platform’s security.

5.3 Designed tests

In this section, we elaborate the designed stress test on the FeatureCloud platform. Each test is designed and executed independently, however, we tried to follow the same principles. The test environment should be isolated from production to prevent any impact. Accordingly, we used different servers than the production server to prevent possible disruptions for FeatureCloud platform services. We used appropriate monitoring tools to capture metrics such as CPU load, memory usage which helps to find the breaking points for successful tests. Meanwhile, we analyze the system after incorporation of stress tests feedback into the platform to show how it survives the stress testing.

5.3.1 FeatureCloud pip package

The FeatureCloud pip package is a comprehensive tool designed for developers, particularly with a focus on privacy preservation, to facilitate the creation, testing, and deployment of federated applications through a range of commands and interfaces. In this section, we elaborate on the tests designed for stress testing the pip package in regard to data communication. For app creation and management, the pip package offers basic commands including building Docker images, downloading images from the repository, and even plotting app states and transitions, providing a visual representation of the app’s workflow. Meanwhile, the engine package simplifies app development by handling state registration and transitions, requiring minimal developer intervention. This transparency is key to encouraging wider adoption and innovation in federated learning.

The FeatureCloud pip package offers various communication methods to facilitate data exchange in a federated environment. These methods are essential for managing data flows between different clients (participants and coordinators) within the federated network, ensuring efficient and secure

² <https://portswigger.net/burp>

³ <https://www.tenable.com/products/nessus>

⁴ <https://trivy.dev/>

data handling. The package provides functionalities like 'aggregate_data' and 'gather_data'. The former automates the handling of SMPC usage and serialization, returning aggregated data that maintains the original data structure and shape. The 'await_data' method is designed to wait for data arrival from a specified number of clients, polling at regular intervals. This is particularly useful in stress testing scenarios where the responsiveness and efficiency of data receipt are critical metrics. Furthermore, the package includes 'send_data_to_participant', 'send_data_to_coordinator', and 'broadcast_data' methods.

The 'configure_smpc' function allows developers to set up the SMPC component, which affects some of the communication methods. Meanwhile, the package enables apps to communicate their operational states (such as app.status_available, app.status_finished, app.status_message, and app.status_progress) to the controller and, indirectly, to the front-end. This feature is essential for providing real-time updates and feedback to users during the app's operation.

In stress testing scenarios, these communication methods play a pivotal role. They allow for the simulation of real-world data exchange patterns by including different data transfer rates and methods.

5.3.1.1 Test Environment Setup

For stress testing the FeatureCloud pip package we run the controller and the communication test app locally in a test-bed. Therefore, the test does not involve any of FeatureCloud servers. We conducted the test using a MacBook Pro with an Apple M1 chip, featuring 8 cores (4 performance and 4 efficiency cores) and 16 gigabyte (GB) of memory. The system is running macOS 12.6.9 on a Darwin 21.6.0 kernel. Storage-wise, it has a 494.38 GB solid-state drive (SSD), with 261.77 GB free space.

5.3.1.2 Test Scenarios

To design the test for stressing the FeatureCloud pip package we considered the scope of the interactions, and since the pip package interactions are limited to the controller, e.g., reporting status and data communication, we use the testbed for stress testing the pip package to isolate the stress on the pip package and refrain from stressing the relay server, which was done in a separate test, see section 5.3.1, while sufficiently increasing the stress. The pip package test workflow runs the communication test app:

```
class WorkFlow(TestWorkFlow):
    def __init__(self, controller_host: str, channel: str, query_interval:
int):
        super().__init__(controller_host, channel, query_interval)
        self.controller_path = "/home/bbb1037/data"
        self.ctrl_data_path = f"{self.controller_path}"
        self.ctrl_test_path = f"{self.controller_path}/tests"

        self.generic_dir = {}
        self.n_clients = 2
        self.TestApp = partial(TestApp,
                                n_clients=self.n_clients,
                                ctrl_data_path=self.ctrl_data_path,
                                ctrl_test_path=self.ctrl_test_path,
                                controller_host=controller_host,
```

```
channel=channel,
query_interval=int(query_interval))

def register_apps(self):
    app_id = 0
    app1 = self.TestApp(app_id=app_id,
app_image="featurecloud.ai/communication_app")
    self.register(app1)

def run(self):
    print("Workflow execution starts ...")
    i = 0
    app = self.apps[0]
    id, _ = app.start()
    app.set_id(id)
    print(f"{app.app_image}(ID: {app.test_id}) is running ...")
    app.wait_until_finishes()
    print("App execution is finished!")
    app.extract_results(self.default_res_dir_name)
    print("extracting the data...")
    while not app.results_ready:
        sleep(5)
    print("Delete the app container...")
    app.delete()
    print("Workflow execution is finished!")
```

We executed the test workflow using the local channel as follows:

```
featurecloud workflow start --wf-dir ./data --wf-file
PipPackageTestWorkflow.py --controller-host localhost:8000 --channel local --
query-interval 1
```

For stressing the pip package, we simulate different combinations of supported communication methods in the pip package while using different settings for applying SMPC or DP. We designed four scenarios, where each is continuance of possible communication methods:

- Simple aggregation: Aggregation using `send_data_to_coordinator`, `aggregate_data`, and `broadcast_data` methods.
- Gathering: Gathering the data using `send_data_to_coordinator`, `gather_data`, and `broadcast_data` methods.
- Awaiting: Awaiting the data using `send_data_to_coordinator` + `await_data` + `broadcast_data` methods.
- Peer-to-peer communication: `send_data_to_participant` + `await_data`
- Mix communication: Immediate uninterrupted communication of data with different methods and privacy enhancing technologies.

For all supported methods, we call them with and without SMPC or DP. Meanwhile, we communicate different data types using different methods to stress the platform:

- I: Scalar Integer
- L: Python List including integer and floating numbers
- F: Float data type
- S: Python string

Furthermore, to ensure the platform survives arbitrary immediate combinations of requests, in a loop we call different scenarios without waiting for the aggregation.

```
elif sub_Scen == 5:
    for i in range(10):
        smpc_ = False
        dp_ = False
        if np.random.rand() > 0.5:
            smpc_ = True
            self.configure_smpc()

        if np.random.rand() > 0.5:
            dp_ = True

        self.send_data_to_coordinator(data_to_send, use_smpc=smpc_ ,
use_dp=dp_)

        self.log(f'Sub_Scen: {sub_Scen} Round: {count_} client send
data {data_to_send} to coordinator .... SMPC: {smpc_}, DP: {dp_}')
```

5.3.2 FeatureCloud Controller

In managing workflows, the controller orchestrates various applications, overseeing their order, execution, and supervision. It is responsible for initializing and setting up these applications, which can be structured into workflows or operated in a testing environment. The controller itself functions within a Docker container and manages the operation of applications in separate Docker instances. The controller operates and oversees network servers, facilitating incoming connections and the transfer of data among app containers. It employs TLS and Rivest–Shamir–Adleman (RSA) to ensure secure communication, with thorough testing of both encrypted and unencrypted network exchanges. The controller also acquires data or status updates from active applications, utilizing JavaScript Object Notation (JSON) for structuring data during HTTP-based exchanges. Stress testing the controller is crucial for ensuring its robustness and reliability in real-world scenarios. By simulating unusual or extreme use cases, stress testing helps identify potential bottlenecks or vulnerabilities in the Controller. This process is essential for verifying that the controller can efficiently handle heavy network traffic, manage multiple concurrent workflows, and maintain secure communication under varying levels of demand. Additionally, stress testing provides insights into the system's scalability and resilience, ensuring that the controller remains stable and functional even under challenging conditions.

We used the “aira” server as a test environment. The server “aira” is configured with openSUSE Tumbleweed (20231012 version), featuring a 6.5.6-1-default kernel. It's powered by two Intel Xeon Gold 6258R CPUs (56 cores, 112 threads in total), supporting both 32-bit and 64-bit operations. It boasts 503 GB of random access memory (RAM), and a 2 GB swap space. The system's disk setup

includes a 1.1 (terabyte) TB root filesystem with only 2% usage, an 8.8 TB /home partition with 40% usage, and various other mounts including a 80 TB network file system. Network-wise, it has multiple interfaces, including a bonded interface (bond0) with an IP of 134.100.85.17/26, and a Docker network interface. The server had been running for 44 days and at the test time experienced a moderate load (12.27, 13.29, 12.97). A complete log for the “aira” server is in 11.3.

5.3.2.1 Test Scenarios

We design the stress tests for the FeatureCloud controller to ensure that the system can handle scenarios involving unmatching applications in a workflow and unsupported input data and configurations. Even though the controller might not support certain use cases, encountering them should not lead to system crashes or significant disruptions. One key area of testing involves application compatibility. Here, the controller would be exposed to a mix of compatible and incompatible applications. The aim is to observe whether the system can effectively handle the incompatible applications without affecting its overall stability. This test would reveal the robustness of the controller's application management protocols and its ability to maintain operational integrity in the face of unexpected application behaviors.

Another important test scenario is the introduction of unsupported data formats. We do not expect the controller to process this data correctly, but rather to ensure that it can reject or ignore these formats without experiencing crashes or critical errors while providing proper logs for the end-users. Such a test is crucial for understanding the system's resilience against data anomalies. Furthermore, we simulate network failures or disconnections during data transmission to evaluate the controller's ability to handle and recover from network interruptions to stress test the platform's network resiliency.

Meanwhile, since the workflow execution is not supported in the FeatureCloud command-line interface (CLI) due to security and privacy concerns, we run various workflows manually. These workflows include various applications published in the app store. To ensure that the tests will not affect the platform, specially in terms of residuals of the test, we use the staging server and import the applications. We have created template workflows to encompass the incompatible applications and include partially or fully unsupported data or configurations in the platform. Since the stress test is not concerned with scalability, we run the workflows with two clients. Evidently, all the executed scenarios will result in errors, however, in the tests we want to observe how the platform behaves under the stress.

5.3.3 Web Security and Workflow Execution Platform

FeatureCloud offers a comprehensive web service platform designed to facilitate federated data analysis and computation. Its core functionality allows users to use and combine federated applications from the AppStore into federated workflow. Those workflows can then be to securely run involving multiple data owners, leveraging a federated approach to data processing. The platform's front-end provides an intuitive interface for users to construct and execute linear workflows, where the output from one application feeds directly into the next, ensuring seamless data flow and integration. This interactive front-end is crucial for user engagement, allowing them to easily navigate and utilize the platform's features for their federated computing needs without the need for deep knowledge of federated learning. The back-end of FeatureCloud, equally essential, handles the intricate processes of data management, application execution, and workflow coordination. It ensures that the computations are performed efficiently and securely, maintaining the integrity and privacy of the federated data.

Given the complex nature of federated data processing and the diverse user interactions with the platform, stress testing both the front-end and back-end of FeatureCloud is vital. Stress testing the front-end involves simulating various user interactions at scale, such as creating, running, and managing workflows, to ensure that the user interface remains responsive and stable under heavy load. This helps identify potential bottlenecks in the interface design or workflow management systems. For the back-end, stress tests focus on the system's ability to handle large volumes of data, maintain consistent performance during simultaneous workflow executions, and efficiently manage network traffic and data processing tasks. This is crucial for assessing the platform's scalability and resilience, especially in handling complex federated computations and ensuring data privacy. General methods of stress testing these aspects include injecting high volumes of simulated user requests, executing multiple workflows concurrently, and processing large datasets to evaluate system performance and reliability under various load conditions.

5.3.3.1 Test Environment Setup

For stress testing the FeatureCloud website we used the test environment on a server named "FeatureCloud-staging," running Ubuntu 22.04.3 LTS with the 5.15.0-88-generic kernel. It is powered by an Intel Core i7-6700 CPU with 8 cores, operating at a maximum frequency of 4.00 GHz. The server boasts a substantial memory capacity of 31 GB RAM, along with a sizable disk space of 905 GB, of which 570 GB is used. Network-wise, it has multiple interfaces, including a primary one with the IP address 138.201.198.53. The server's uptime indicates stable performance, having been up for 15 days with minimal load, as shown by the load averages. The environment includes Docker for running containerized applications. This setup provides a robust and reliable platform for conducting comprehensive stress tests in federated learning scenarios.

5.3.3.2 Test Scenarios

FeatureCloud, leveraging Hetzner's infrastructure for hosting, does not require stress testing for Distributed Denial of Service (DDoS) attacks. Hetzner, known for its dependable cloud services, offers extensive DDoS protection as a fundamental part of its infrastructure, adept at detecting and neutralizing large-scale DDoS attacks. This safeguards services like FeatureCloud from such threats. The advanced security solutions provided by Hetzner enable FeatureCloud to concentrate on optimizing other vital aspects of its platform, including performance and stability. Hetzner's effectiveness in handling DDoS threats ensures the continuous security and functionality of FeatureCloud, eliminating the need for FeatureCloud to perform DDoS-specific stress tests. Additionally, stress tests like network saturation, traffic floods, application layer attacks, protocol attacks, and volume-based attacks are not conducted on FeatureCloud, as Hetzner's infrastructure already offers robust protection against these. With Hetzner ensuring network capacity and system integrity, FeatureCloud can channel its efforts towards enhancing application resilience and

performance, rather than focusing on the stress tests associated with the security aspect of deploying FeatureCloud as a web service that employs Docker containerization.

It has to be noted that providing mitigation against DDOS attacks is what nowadays most reliable server hosting providers offer; thus, a change from Hetzner to another provider with an equivalent level of service, will mean that a similar protection will be achieved.

Security specialists at SBA, trained for performing penetration tests in multiple settings and against various types of systems, conducted a comprehensive assessment of the FeatureCloud platform by employing a variety of methodologies to detect potential security vulnerabilities. SBA investigated the platform for Cross-Site Request Forgery (CSRF) vulnerabilities which involve simulating requests from authenticated users to test if the platform executed actions without proper authorization. In fact the approach was to mimic typical user behaviors under compromised conditions, checking for any unauthorized command transmissions. Moreover, to stress the platform's use cases of TLS protocols, SBA conducted tests attempting to establish connections using various deprecated TLS versions to determine if the platform would accept insecure or outdated encryption protocols, thereby exposing it to potential data breaches. In fact, we tested the platform's encryption by attempting to breach encrypted data channels and assessing the response to cryptographic attacks like ciphertext-only attacks. This aspect of testing was crucial in evaluating the integrity of encrypted communications within the platform. Another key focus was on SSL stripping attacks by intercepting and altering communications between the client and the server; For instance, connections are downgraded to insecure (HTTP) ones. SBA assessed the platform's ability to maintain Hypertext Transfer Protocol Secure (HTTPS) connections. This testing was vital in understanding the platform's resilience against efforts to downgrade secure connections to less secure ones.

Furthermore, SBA undertook efforts to exploit session cookies or tokens, to stress the platform by aiming to gain unauthorized access or maintain persistent access to user sessions. This test is designed to scrutinize the session management and exploit possible weaknesses by attempting to gain unauthorized access or maintain persistent access to user sessions, e.g., using cross-site scripting (XSS). By supplying various forms of malicious input, such as a Structured Query Language (SQL) script, or command injections, into input fields, we stress the platform's ability in handling and sanitizing user input. This was critical in assessing the platform's defense mechanisms against injection attacks.

In another test scenario, we stressed the encryption practices in the platform by using the data storage and transmission methods of the platform's to undermine the security measures in place for protecting user's data, particularly focusing on exposure risks. SBA also stresses the platform's endpoints or authentication mechanism by making malformed or irregular request patterns. Also, SBA stressed the platform's configurations and credentials using flawed configs and abusing the credentials. The test was designed to uncover any misconfigurations or overlooked default settings that could potentially lead to security breaches. The application programming interface (API) security was another area of our focus in stress testing the security aspects of the platform. SBA interactively engaged with the platform's API in unintended ways to stress its robustness.

Throughout the stress testing, we maintained a balance between automated scanning tools and manual probing. This comprehensive approach was instrumental in successfully pushing the platform to fail the stress tests on a range of security risks.

6 Results

In this section we discuss the results of implementing the assessment and requirement criteria that was reported in D7.1. Furthermore, we will elaborate on the outcomes of stress testing different components in FeatureCloud platform in terms of successful tests which managed to fail the platform, incorporation of the feedback to the FeatureCloud core development team, and execution of the tests. We show that after implementing the required measures, the platform manages to survive the stress tests.

6.1 Implementation of assessment and requirement criteria

FeatureCloud implemented its assessments and requirements criteria through several key processes and methodologies:

- **Federated Learning (FL) for Security and Privacy:** We emphasize the use of FL as a fundamental component for ensuring security and privacy which allows multiple parties to collaboratively train a machine learning (ML) model without sharing sensitive training data and reduces privacy and security concerns compared to centralized models by keeping training data at local sites. Additionally, FeatureCloud developed key performance indicator (KPI) 3 Privacy Requirements for Federated Algorithms to maintain a balance between shared parameters and raw patient data.
- **App Certification Process:** We implemented a certification process for apps in the FeatureCloud app store by asking for a two-step verification that involves automated and manual verification of security and privacy aspects. Only certified apps are allowed in the FeatureCloud workflow without raising a notification, ensuring that the apps meet certain standards before being published in the app store.
- **Automated Pipeline for Code Review:** We employed a CI/CD (Continuous Integration/Continuous Deployment) pipeline to review developed code for the platform after each commit. This pipeline includes linting for code readability, testing to cover a high percentage of the code, and a build stage to ensure error-free compilation. This process helps ensure that the code meets the project's standards and requirements before updating the production services.
- **Balancing Privacy and Performance:** We emphasize on achieving comparable results to traditional cloud-based in published federated applications while maintaining privacy. Our efforts manifested in developing applications like sPLINK, Flimma, and Partea, as peer reviewed publications, which deliver similar results in federated environments as in aggregated analysis, demonstrating the effectiveness of FeatureCloud as a federated platform.
- **Additional Privacy Measures:** FeatureCloud encourages AI developers to integrate additional privacy enhancing technologies like differential privacy (DP) or Secure Multi-Party Computation (SMPC) to further enhance privacy. These implementations are fully integrated into the featureCloud platform to facilitate the usage. Such technologies were used in different apps and underwent a certification process to assess the potential privacy leakage.
- **Security Controlled Communications:** We included controlled communications and data access protocols in the platform to run inside Docker containers, ensuring that only privacy-aware results are exported, and sensitive intermediate results are not shared.
- **Developer Review Process:** for developing the platform, after the CI/CD pipeline, another internal developer reviews the code and functionalities to ensure that the implementation meets all defined requirements. We fix all the raised issues before merging the new commits into the production branch.
- **Implementation Process and Sprint Planning:** FeatureCloud's implementation process involved splitting user stories into development tasks, prioritizing them into sprints, and

assigning them to developers. In regular retrospective meetings with stakeholders we discussed the progress and control to ensure alignment with project goals.

These steps showcase our comprehensive approach to implement the assessments and requirements while focusing on security, privacy, performance, and stakeholder involvement.

6.2 Pip package

In this section we stress test the pip package and highlight various scenarios that led to platform failures. These scenarios include Aggregation, Gathering, Awaiting, and Peer-to-peer communication methods using different data types and privacy-enhancing technologies. We show occurrences of data mismatches during Aggregation, and serialization errors in mix communication scenarios involving Simple-SMPC, Simple-DP, and DP-SMPC. We addressed these issues by enhancing our pip package to manage data and status information more efficiently, introducing a unique memo feature for communication rounds, and ensuring backward compatibility.

6.2.1 Execution

While stressing the pip package and controller, we manage to fail the platform on multiple occasions. Here, we discuss different scenarios independently and later we will show how all are caused by the same issue. For aggregation scenarios while running the Aggregation scenario using the simple communication method, i.e., using neither SMPC nor DP, for two clients, the platform is stressed by receiving numerous communication requests at the participants side and corresponding aggregation requests on the coordinator side which causes the received data pieces being misplaced with each other. The expected values can be seen in the log entries where “[State: client_state]” is printed. In comparison to this, it can be seen that in the lines where “[State: coordinator_state]” is printed, data from different communication rounds is seen as the incoming data for specific communication rounds. For example, communication=0 receives [1,11] instead of [1,1]. Each client sends out exactly the same series of data to the coordinator, however, corresponding values are not aggregated together due to the mismatch. For instance, [1, 1] should be results of gathering clients send out 1, however the mismatch causes the gathering output be [1, 11].]

```
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 4, client send data 41 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 5, client send data 51 to coordinator .... SMPC:
False, DP: False
```



```
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 6, client send data 61 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 7, client send data 71 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:39] "GET /api/status HTTP/1.0" 200 151
[Time: 30.11.23 17:44:40] [Level: info] state: coordinator_state
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Coordinator
is launched...
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 11] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 1 gather data: [21, 31] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 2 gather data: [41, 51] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 3 gather data: [61, 71] SMPC: False, DP: False
[CTRL] GET /status
```

For more detailed logs, see section 11.4.2

The same problem is observed for other data types with simple scenarios. The platform also shows the same behavior while we test it using Awaiting and Gathering scenario with the same setting as Aggregation scenario. Later we show the persistent issue for all the scenarios stems from the same design flaw.

For all the Aggregation scenarios, for simple, DP, and SMPC options, once we run the test for String data type, we receive an error in the coordinator's log regarding the fact that aggregation method does not support String data. This is intended behavior which is well explained in the relevant documentations.

```
[Time: 30.11.23 18:49:08] [Level: info] Traceback (most recent call last):
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 226, in guarded_run
    self.run()
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 241, in run
    transition = self.current_state.run()
  File "/app/states.py", line 461, in run
    data_to_collect = self.aggregate_data(use_smpc=smpc_, use_dp=dp_)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 599, in aggregate_data
    return _aggregate(data, operation) # Data needs to be aggregated
```

```
according to operation
File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 1000, in _aggregate
    aggregate = aggregate + d
numpy.core._exceptions._UFuncNoLoopError: ufunc 'add' did not contain a loop
with signature matching types (dtype('<U5'), dtype('<U5')) -> None
```

We also stressed the platform using mix communication scenario, where we used aggregation methods with setting the DP and SMPC functions as true or False to create these three sub-scenarios:

- Simple-SMPC: We alternately set SMPC to True, while DP is always set to False.
- Simple-DP: We alternately set DP to True while always setting SMPC to False.
- DP-SMPC: We alternately set either DP or SMPC to True or False.

By stressing the platform in Simple-SMPC sub-scenario, we send out the same data series using the same order of communication methods and options from both clients. As it is shown in the coordinators logs, the controller of the coordinator displaces the received data which result in an runtime error inside the app:

```
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:07:06] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:06] "GET /api/status HTTP/1.0" 200 219
127.0.0.1 - - [30/Nov/2023 18:07:06] "GET /api/data HTTP/1.0" 200 2
[CTRL] GET /data
[Time: 30.11.23 18:07:07] [Level: info] state: coordinator_state
[Time: 30.11.23 18:07:07] [Level: info] [State: coordinator_state] Coordinator
is launched...
[Time: 30.11.23 18:07:07] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 21] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:09] "GET /api/status HTTP/1.0" 200 219
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:07:09] "GET /api/data HTTP/1.0" 200 2
```

```
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:07:11] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:12] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:07:15] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[Time: 30.11.23 18:07:15] [Level: info] Traceback (most recent call last):
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 226, in guarded_run
    self.run()
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 241, in run
    transition = self.current_state.run()
  File "/app/states.py", line 454, in run
    data_to_collect = self.gather_data(is_json=flg)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 618, in gather_data
    return self.await_data(len(self._app.clients), unwrap=False,
is_json=is_json)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 648, in await_data
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 648, in <listcomp>
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 974, in _deserialize_incoming
    return pickle.loads(data)
_pickle.UnpicklingError: could not find MARK
```

The error arises from a mismatch in data serialization formats, where the data sent to the Python pickle module for deserialization have been serialized using JSON, the dedicated serialization method for simple communications. Therefore, attempting to deserialize data that was serialized in JSON (a text-based format) causes an incompatibility, leading to the `_pickle.UnpicklingError: could not find MARK` error. This error indicates that pickle is unable to find the specific markers it uses to recognize and decode serialized objects, as these markers are not present in JSON-formatted data. For a more detailed log see 11.4.3.

For the Simple-DP sub-scenario, the pip package uses the wrong status of another sub scenario without DP for the data meant to be sent with DP, which results in a runtime error inside the controller:

```
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': "Error while
adding noise: Error while reading sent data: invalid character '\\u0080'
looking for beginning of value", 'time': '2023-11-30T18:14:03Z'}
```



```
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': "Error while reading sent data: invalid character '\\u0080' looking for beginning of value", 'time': '2023-11-30T18:14:03Z'}
```

The presence of “\u0080” character implies that the data being processed was serialized in a binary format using pickle, while the controller attempting to read this data, expecting JSON serialization, which is text-based and does not typically handle raw binary data well. In this case, this means that data that was supposed to be sent without DP, and therefore with binary serialization, was sent before the DP data alongside with the status of the DP data. For more detailed logs see section 11.4.4.

And for the DP-SMPC sub-scenario, the controller of the coordinator again displaces the received data which result in an runtime error inside the coordinator’s app:

```
[Time: 30.11.23 18:30:18] [Level: info] Traceback (most recent call last):
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 226, in guarded_run
    self.run()
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 241, in run
    transition = self.current_state.run()
  File "/app/states.py", line 465, in run
    data_to_collect = self.gather_data(is_json=flg)
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 618, in gather_data
    return self.await_data(len(self._app.clients), unwrap=False,
is_json=is_json)
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 648, in await_data
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 648, in <listcomp>
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-packages/FeatureCloud/app/engine/app.py", line 974, in _deserialize_incoming
    return pickle.loads(data)
_pickle.UnpicklingError: unpickling stack underflow
```

For more detailed logs see section 11.4.5

All the test scenarios, settings, and data types are summarized in Table 1. Where, for String data, DP and SMPC are not supported, same as simple aggregation; however, for simple gathering and awaiting, the test is successful due to the misplacement issue. The same misplacement issue makes the test successful to fail the platform for other data types, operations and communications methods, except for the peer-to-peer communication method.

6.2.2 Communication memo

After receiving the feedback from the stress test, we analyze the logs to discover the possible vulnerabilities contributing to the platform’s failure. Accordingly, we found the misplacement issue which requires implementing the memo mechanism to keep the track of communications. Implementing the memo affects the controller and pip package.

Tabel 1. Stress testing four different scenarios: different communication methods using various data types and privacy enhancing technologies: (✓) indicates the test was successful, (X) shows the test was unsuccessful, (-) shows the data or operation is not supported.

		Aggregation	Gathering	Awaiting	Peer-to-peer
Simple	Int	✓	✓	✓	X
	Float	✓	✓	✓	X
	List	✓	✓	✓	X
	String	-	✓	✓	X
DP	Int	✓	✓	✓	-
	Float	✓	✓	✓	-
	List	✓	✓	✓	-
	String	-	-	-	-
SMPC	Int	✓	✓	✓	-
	Float	✓	✓	✓	-
	List	✓	✓	✓	-
	String	-	-	-	-

As it is shown in Table 2., for all scenarios the test was successful to break the platform.

Tabel 2. Stress testing mix communication scenario: different communication methods using various data types and privacy enhancing technologies: (✓) Successful test, (X) Unsuccessful test, (-) Doesn't support.

Scenarios	Simple-SMPC	Simple-DP	DP-SMPC
Mix	✓	✓	✓

6.2.2.1 Pip package

To fix the problems found in our stress testing of the pip package, we applied two different strategies. Before, shared status_variables, defining how the next data piece should be sent, were used which was prone to race condition problems. To fix this, we now use the given function “get_current_status” to ensure that different threads don’t change the status information from other threads, also saving the status information alongside the data for each data piece sent between clients.

```
def get_current_status(self, **kwargs):
    status = dict()
    status["available"] = self.status_available
    status["finished"] = self.status_finished
    status["message"] = self.status_message
    status["progress"] = self.status_progress
    status["state"] = self.status_state
    status["destination"] = self.status_destination
    status["smpc"] = self.status_smpc
    status["dp"] = self.status_dp
    status["memo"] = self.status_memo
    for key, value in kwargs.items():
        # set whatever is wanted from the arguments
        status[key] = value
    return status
```

The status information of how a certain data piece is to be sent is saved as a JSON object representing the answer to the GET / status request of the controller. Alongside this JSON string, the data object is saved.

Whenever the status is requested, the oldest added data, status combination is looked at and its status is sent as an answer. For the following GET / data request, it is ensured that the last send status is the same as the one of the data objects about to be sent out.

This ensures in total that data is sent out exactly in the way it is wanted by the user, as is shown in this extract from one of the sending methods of the pip package

```
smpc = self._app.default_smpc if use_smpc else None
dp = self._app.default_dp if use_dp else None
status = self._app.get_current_status(message=message,
                                     destination=destination, smpc=smpc, dp=dp, memo=memo,
                                     available=True)
self._app.data_outgoing.append((data, json.dumps(status, sort_keys=True)))
```

For each communication round, we added the possibility to add a memo to tag the communication round. This memo ensures that each communication round is uniquely identifiable so that any race conditions of individual communications taking a longer time due to e.g. higher data load are not problematic, as with an added memo, the order of incoming and outgoing data is not relevant for the identification of individual communication rounds. We added the memo string to the GET /STATUS call of the controller by adding it to the `get_current_status(**kwargs)` method.

We roughly categorize the communication methods into send and receive methods. For send methods, including `send_data_to_participant`, `send_data_to_coordinator`, and `broadcast_data` we check for the status and include the memo into the status and communicate the data and status together by appending them as a tuple into the same element of the `data_outgoing` list.

```
def send_data_to_coordinator(self, data, send_to_self=True, use_smpc=False,
                             use_dp=False, memo=None):
    message = self._app.status_message if self._app.status_message else
        (self._app.current_state.name if self._app.current_state else None)
    self._app.status_message = message
    smpc = self._app.default_smpc if use_smpc else None
    dp = self._app.default_dp if use_dp else None
    status = self._app.get_current_status(message=message,
                                          destination=destination, smpc=smpc, dp=dp, memo=memo,
                                          available=True)
    self._app.data_outgoing.append((data, json.dumps(status,
                                                       sort_keys=True)))
```

On the other hand, for all receive methods, including `aggregate_data` and `gather_data`, we call the `await` method which is responsible for handling the incoming data. In fact, the `await_data` method looks for receiving specific tuples of data and memo from all awaited clients. In this way, each data and memo will be identifiable for the app.

As older apps that do not use memos will not send any memo, we must ensure backwards compatibility by assuming potentially, no memo is given when receiving data. In that case, we simply use the default memo (`None`).

```
if "memo" in request.query:
    memo = request.query["memo"]
else:
    memo = None
```

For more detail about the changes regarding the pip package see action 11.4.1.

As discussed in 6.2.1, various problems were discovered. The stress test was then done again on the updated pip package and the following logs were produced considering the following errors

The misplacement of values from different communication rounds was fixed by the implementation of memos, ensuring that values can be assigned to the correct communication round. The following log shows no misplacement anymore after the fix. Here, two clients sent different data in each communication round which then got aggregated. While before, the aggregations would have been incorrect due to misplacement, e.g. 1 and 11 would have been added to 12 instead of 1 and 1, now the aggregation returns the correct result.

```
[Time: 05.12.23 18:14:42] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
```

```
[Time: 05.12.23 18:14:42] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: False
[Time: 05.12.23 18:14:42] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 05.12.23 18:14:42] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: False
...
[Time: 05.12.23 18:14:44] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 aggregate data: 2 SMPC: False, DP: False
[Time: 05.12.23 18:14:47] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 1 aggregate data: 22 SMPC: False, DP: False
[Time: 05.12.23 18:14:50] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 2 aggregate data: 42 SMPC: False, DP: False
[Time: 05.12.23 18:14:53] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 3 aggregate data: 62 SMPC: False, DP: False
```

The other issue that was found during stress testing was the sending of data serialized in one way alongside of status that declared the data to be serialized in another way. This was fixed by keeping the data together with their status, furthermore comparing the last used status with the status the data should be sent with. In the following log, it is shown that even when mixing communication, which before due to shared status variables led to serialization related errors, the communication app works.

```
[Time: 06.12.23 17:11:13] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 06.12.23 17:11:13] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: True
[Time: 06.12.23 17:11:13] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 06.12.23 17:11:13] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: True
...
127.0.0.1 - - [06/Dec/2023 17:11:13] "POST
/api/data?client=fee5ff399468436f&memo=GATHERROUND2 HTTP/1.0" 200 0
127.0.0.1 - - [06/Dec/2023 17:11:16] "POST
/api/data?client=fee5ff399468436f&memo=GATHERROUND4 HTTP/1.0" 200 0
127.0.0.1 - - [06/Dec/2023 17:11:18] "POST
/api/data?client=1c3278743a14209a&memo=GATHERROUND1 HTTP/1.0" 200 0
```

```
[Time: 06.12.23 17:11:18] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 1] SMPC: False, DP: False
127.0.0.1 - - [06/Dec/2023 17:11:21] "POST
/api/data?client=1c3278743a14209a&memo=GATHERROUND2 HTTP/1.0" 200 0
[Time: 06.12.23 17:11:21] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 1 gather data: [48.962946450250456, -
15.404591992555652] SMPC: False, DP: True
127.0.0.1 - - [06/Dec/2023 17:11:22] "GET /api/status HTTP/1.0" 200 165
[CTRL] POST /data
127.0.0.1 - - [06/Dec/2023 17:11:24] "POST
/api/data?client=1c3278743a14209a&memo=GATHERROUND3 HTTP/1.0" 200 0
[Time: 06.12.23 17:11:24] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 2 gather data: [21, 21] SMPC: False, DP: False
127.0.0.1 - - [06/Dec/2023 17:11:25] "GET /api/status HTTP/1.0" 200 165
[Time: 06.12.23 17:11:27] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 3 gather data: [50.419946977926884, -
17.18074216073728] SMPC: False, DP: True
```

Furthermore, also when mixing SMPC with plain sending of data, no problems occur. Please notice that when using SMPC, data is aggregated by design:

```
[Time: 06.12.23 20:13:18] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 06.12.23 20:13:18] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
True, DP: False
[Time: 06.12.23 20:13:18] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 06.12.23 20:13:18] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
...
127.0.0.1 - - [06/Dec/2023 20:13:21] "POST
/api/data?client=5cbaecb01a49dc14&memo=GATHERROUND1 HTTP/1.0" 200 0
[Time: 06.12.23 20:13:21] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 1] SMPC: False, DP: False
127.0.0.1 - - [06/Dec/2023 20:13:24] "POST
/api/data?client=0000000000000000&memo=GATHERROUND2 HTTP/1.0" 200 0
[Time: 06.12.23 20:13:24] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 1 gather data: [22] SMPC: True, DP: False
127.0.0.1 - - [06/Dec/2023 20:13:27] "POST
/api/data?client=5cbaecb01a49dc14&memo=GATHERROUND3 HTTP/1.0" 200 0
```



```
[Time: 06.12.23 20:13:27] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 2 gather data: [21, 21] SMPC: False, DP: False
127.0.0.1 - - [06/Dec/2023 20:13:30] "POST
/api/data?client=0000000000000000&memo=GATHERROUND4 HTTP/1.0" 200 0
[Time: 06.12.23 20:13:30] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 3 gather data: [62] SMPC: True, DP: False
```

6.2.2.2 Controller

Since the memo must be propagated from one client to another, the memo also had to be implemented into the controller. For any communication of a data piece between two clients, the controller of a client sends the information as given in the code block below. Here, we added one byte to inform about the memo size and then up to 255 byte of the memo itself.

```
func (s *Server) writePacket(cmd byte, b []byte, dest int, memosize byte, memo
[]byte) error {
    // writes byte cmd, [4]byte destination, [8]byte contentLength,
    // byte memosize, [memosize]byte memo, [contentlength] b

    // write cmd
    s.mutexOut.Lock()
    if err := s.WriteByte(cmd); err != nil {
        s.mutexOut.Unlock()
        return err
    }

    // write destination
    if dest != -1 {
        destinationBytes := [4]byte{}
        binary.BigEndian.PutUint32(destinationBytes[:], uint32(dest))
        if err := s.writeBytes(destinationBytes[:]); err != nil {
            return err
        }
    }

    // write contentLength
    contentLength := int64(len(b))

    chunkSize := [8]byte{}
    binary.BigEndian.PutUint64(chunkSize[:], uint64(contentLength))

    if err := s.writeBytes(chunkSize[:]); err != nil {
        s.mutexOut.Unlock()
        return err
    }
}
```

```
// write memosize and the memo itself
if err := s.WriteByte(memosize); err != nil {
    s.mutexOut.Unlock()
    return err
}
if memosize > 0 {
    if err := s.writeBytes([]byte(memo)); err != nil {
        s.mutexOut.Unlock()
        return err
    }
}

// write b (the data itself)
if err := s.writeBytes(b); err != nil {
    s.mutexOut.Unlock()
    return err
}

s.mutexOut.Unlock()

return nil
}
```

Then, on the receiving side, the global relay server reads the memo

```
// read memosize
memosize, err := c.ReadByte()
if err != nil {
    return err
}

// read memo
memo := make([]byte, int(memosize))
if err := c.readBytes(memo); err != nil {
    return err
}
```

and relays it in the same way to the receiving client.

```
func (c *Client) SendData(msg []byte, senderID shared.ClientID, tp string,
memosize byte, memo []byte) error {
    c.mutexOut.Lock()
    defer c.mutexOut.Unlock()

    var b byte
    switch tp {
    case "plain":
        b = shared.CMD_SEND_PLAIN_DATA
```

```
case "p2p":
    b = shared.CMD_SEND_P2P_DATA
case "smpc":
    b = shared.CMD_SEND_SMPD_DATA
case "smpc_agg":
    b = shared.CMD_SEND_SMPD_AGG
}

// write the command byte
if err := c.WriteByte(b); err != nil {
    return err
}

// write the senderID so the receiver can identify the sender
if err := c.WriteBytes(senderID[:]); err != nil {
    return err
}

// write the contentlength
remainingBytes := [8]byte{}
binary.BigEndian.PutUint64(remainingBytes[:], uint64(len(msg)))

if err := c.WriteBytes(remainingBytes[:]); err != nil {
    return err
}

// write the memosize
if err := c.WriteByte(memosize); err != nil {
    return err
}

// write the memo
if err := c.WriteBytes(memo); err != nil {
    return err
}

// write the content
err := c.WriteBytes(msg)
if err != nil {
    return err
}

return nil
}
```

Lastly, the receiving client reads it again in the same manner

```
// read the memo
    if cmd == shared.CMD_SEND_PLAIN_DATA || cmd ==
shared.CMD_SEND_P2P_DATA || cmd == shared.CMD_SEND_SMPC_DATA || cmd ==
shared.CMD_SEND_SMPC_AGG {
        // read the memosize and memo, for CMD_SEND_SETUP this is
not send, this is why we need this if statement
        memosize, err := s.ReadByte()
        if err != nil {
            return s.evaluateError(err)
        }
        if memosize > 0 {
            memoBytes = make([]byte, memosize)
            if err := s.ReadBytes(memoBytes); err != nil {
                return s.evaluateError(err)
            }
            memo = string(memoBytes)
        }
    }
```

And finally sends the data to the app instance. The memo is given as part of the Uniform Resource Locator (URL) of the POST Request to the app Instance.

```
func (s *Server) postData(dataBytes []byte, clientID shared.ClientID, memo
string) error {
    maxTries := 3
    // Relay data (POST data)
    for tries := 0; tries < maxTries; tries++ {
        var targetURL string
        if len(memo) > 0 {
            targetURL = fmt.Sprintf("%s/data?client=%x&memo=%s",
s.localURL, clientID, memo)
        } else {
            targetURL = fmt.Sprintf("%s/data?client=%x", s.localURL,
clientID)
        }
    }
```

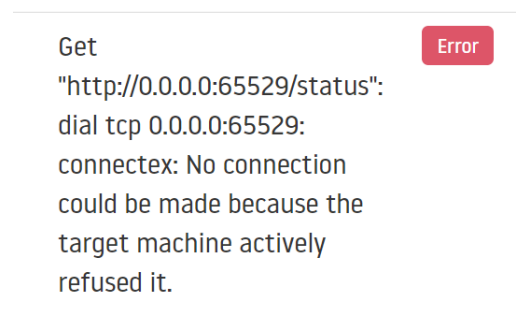
6.3 Controller

In the process of stress testing the FeatureCloud controller, we evaluated the platform's robustness against a variety of scenarios. One significant aspect of our testing involved running workflows with a mix of both compatible and incompatible applications. The platform is able to manage these application inconsistencies effectively, maintaining operational stability throughout the test. When faced with incompatible applications the system intelligently raised runtime errors. These errors were indicative of the platform's inability to process data due to format mismatches, thus providing valuable feedback in the application logs.

Another key area of our testing focused on the controller's response to unsupported data formats. The controller rejects or ignores these formats without crashing or resulting in critical system errors. In fact, the pip package, used in app development, generates error logs to inform the users about the nature of the data format issues. This aspect of the testing underscored the controller's resilience in handling data anomalies.

We also conducted tests involving incompatible configurations. The system's response was to raise specific errors tailored to the nature of the configuration issue. For example, a "File Not Found" error was generated when the system tried to access a file that did not exist according to the given configuration. Similarly, a "Key Error" was raised during the reading of the configuration file, highlighting problems related to missing or incorrect keys necessary for processing. The logs will appear in output logs of the workflow.

Lastly, our stress tests included simulations of network failures and disconnections. These tests were crucial in evaluating the platform's capability to cope with and recover from network-related disruptions. During such network interruptions, the workflow was programmed to halt, and this stoppage was clearly communicated to the user through an error message displayed on the frontend. This error message informed users of the workflow's interruption due to network issues, thereby helping them understand the cause of the disruption.



```
Get
"http://0.0.0.0:65529/status":
dial tcp 0.0.0.0:65529:
connectex: No connection
could be made because the
target machine actively
refused it.
```

Figure 1. Network disruption error during workflow execution

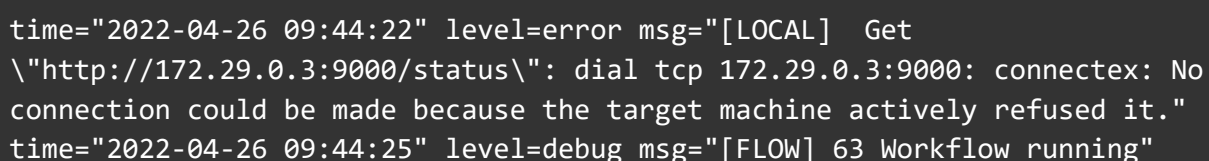
Executing workflows including incompatible apps, configurations, and data can be handled by the platform differently since the response will be highly dependent on the error and exception handling of the apps. If the app faces unhandled exceptions the app will crash, which is an expected behavior in the platform design, however, the controller recognizes the app status and informs the end user via error logs in the frontend.



```
Stopping application
```

Figure 2. Error logs for app crashes in a workflow

However, during the stress testing the controller, we discovered a failure. In fact, the controller continuously polls the app for status updates. If the app stops responding for any reason (e.g. wrong input data, wrong config file caused the app to crash), the controller doesn't properly react to this and the workflow stays in “running” state endlessly.



```
time="2022-04-26 09:44:22" level=error msg="[LOCAL] Get
\\"http://172.29.0.3:9000/status\\": dial tcp 172.29.0.3:9000: connectex: No
connection could be made because the target machine actively refused it."
time="2022-04-26 09:44:25" level=debug msg="[FLOW] 63 Workflow running"
```

This message indicates an attempt by the Controller to connect to the app at the IP address 172.29.0.3 on port 9000 in the local Docker network to retrieve a status update. However, the connection attempt failed because the target app actively refused the connection. This happens once the app crashes. Despite this error, the workflow remains in a "running" state, for more detailed logs see section 11.2.1 In fact, once the app failed, the overall workflow process continued to run, which is not an expected behavior by the platform.

Unhandled errors may occur inside the FeatureCloud app, and the platform should be able to handle them. Below we show a series of changes we implemented in the major platform components (see D7.2, chapter 4.1. System Architecture) which improved the platform’s robustness. Here we cover the code blocks of the error handling in chronological order.

The app template catches any runtime errors, and reports the error state to the controller as a status update:

```
def guarded_run(self):
    """ run the workflow while trying to catch possible exceptions
    """
    try:
        self.run()
    except Exception as e:
        self.log(traceback.format_exc())
        self.status_message = e.__class__.__name__
        self.status_state = State.ERROR.value
        self.status_finished = True
```

The Controller, acting on behalf of one workflow participant, propagates the error to the Global Backend. The `updateFunc` function is invoked each time a status update is received from the FeatureCloud app and `w.setAppStatus` sends the update to the Global Backend.

```
// This function produces a callback function, creating a closure for the
projectId
updateFunc := func(projectId int) func(message *string, progress *float64,
state *string) {
    return func(message *string, progress *float64, state *string) {
        w.setAppStatus(projectId, message, progress, state)
    }
}(info.ProjectId)

// Now we configure the link between the container and the socket server
// It will perform the querying (hence the interval) and relaying, so
basically this struct
// implements the FeatureCloud controller API
lastInfo.LocalLink = local.NewServer(
    time.Duration(w.queryInterval)*time.Second,
    info.Coordinator,
    true,
    updateFunc,
)
```

Furthermore, if the communication with the FeatureCloud app fails (i.e. status update cannot be acquired), the controller reports it as a status update with “error” state, which is handled by the above code and propagated to the Global Backend.

```
// Start relaying
go func() {
    err := s.relayBinaryLocal()
    if err != nil {
        if s.updateCB != nil {
            msg := err.Error()
            state := "error"
            s.updateCB(&msg, nil, &state)
        }
        logger.Error(LOCAL, "", err.Error())
    }
}()
```

In case any participant reports an error, it means the workflow cannot continue and the Global Backend will set the entire workflow to “error” state.

```
if 'state' in request.data:
    current_state = request.data.get('state')
    membership.state = current_state
    # Set error status at project level too
    if current_state == 'error':
        project.status = 'error'
        project.finished_at = timezone.now()
        project.save()
```

The controller will handle the workflow “error” state and stop the FeatureCloud application:

```
case "shutdown", "error":
    if info.Stage == "error" {
        logger.Error(FLOW, strconv.Itoa(info.ProjectId), "Workflow is shutting
down because of an error")
    } else {
        logger.Info(FLOW, strconv.Itoa(info.ProjectId), "Workflow is shutting
down")
    }
    lastInfo.Run = w.ensureRunNumber(info.ProjectId)
    shutdownLocalLink(lastInfo.LocalLink)
    w.stopApplication(lastInfo.LocalApp, info.ProjectId, lastInfo.Step)
```

After the robustness improvements by the FeatureCloud core development team, we executed the stress test to validate that the platform survives the test. As it is shown in the logs below, the “error” entries are extended to not only report on the unreachability of the app, but also shutting down the workflow (for more detailed logs see 11.2.2). Consequently, the workflow stop logs, see Figure 2, will inform the end-user about the status of the app and workflow. Further detailed logs can be provided by the app on the reasons for the app crashing.

```
time="2022-05-04 10:24:05" level=error msg="[LOCAL] Get  
\"http://172.29.0.3:9000/status\": dial tcp 172.29.0.3:9000: connectex: No  
connection could be made because the target machine actively refused it."  
time="2022-05-04 10:24:08" level=info msg="[FLOW] 63 Info change for project  
63: stage: error, step: 0"  
time="2022-05-04 10:24:08" level=error msg="[FLOW] 63 Workflow is shutting  
down because of an error"
```

6.4 Web Security

We stress tested the security aspect of FeatureCloud platform as a web application in different scenarios using different tools:

- Cross-Site Request Forgery (CSRF) Vulnerabilities: We used Burp Suite to modify and replay web requests.
- TLS Protocol Usage and Encryption Testing: We used Trivy for scanning vulnerabilities in encryption protocols, and we used Nessus to scan network-level TLS protocol implementations.
- SSL Stripping Attacks: We used Burp Suite to intercept and modify HTTP/HTTPS traffic.
- Session Management and XSS Testing: We used Burp Suite to exploit possible session management vulnerabilities and conduct XSS attacks.
- Input Validation Tests (SQL, Script, Command Injections): We used Burp Suite for stressing the FeatureCloud web applications.
- Data Storage and Transmission Methods Testing: We used Trivy to scan for possible vulnerabilities in the platform’s data handling components.
- Endpoints and Authentication Mechanism Stress Testing: We used Burp Suite to create and modify web requests.
- Configuration and Credential Testing: We used Nessus to scan for misconfigurations and credential weaknesses.
- API Security Testing: We used Burp Suite for stressing the platform’s API.

6.4.1 Unauthenticated user access

We designed a test to access data out of the intended scope while exploiting the root user privileges inside the controller's Docker container.

During the stress tests, the controller failed to robustly act on the path traversal aspect caused by insufficient validation of file name input which allows an unauthenticated user to write or overwrite arbitrary files on the server. This is implemented in the file upload function of a project in the controller, where the file name is passed to the server via a URL parameter:

```
fPath := filepath.Join(internalDir, fileName)
```

Where, `fileName` is concatenated directly with `directoryPath` to form a file path, without checking if `fileName` contains any directory traversal characters like “../”. Thereby, the test was successful to manipulate the file system of the server by unauthorized data modification, e.g., overwriting critical system files or configuration files.

We incorporated the feedback into the platform by validating and normalizing the path. In fact, we implemented validation checks. We ensure the path is resolved to its absolute canonical form then we normalize and validate the resolved path points only to authorized directories, thus fixing this vulnerability.


```
fPath := getPath(internalDir, fileName)
if err != nil {
    return err
}
...
func (m manager) getPath(topDir string, relPath string) (string, error) {
    relPath = filepath.Clean(relPath)
    if filepath.IsAbs(relPath) {
        return "", errors.New(fmt.Sprintf("need relative path, topDir: %s,
relPath: %s", topDir, relPath))
    }
    if strings.Contains(relPath, "..") {
        return "", errors.New(fmt.Sprintf("relative path must not escape top
path: %s", relPath))
    }
    p := filepath.Join(topDir, relPath)
    if rp, err := filepath.Rel(topDir, p); err != nil || rp != relPath {
        return "", errors.New(fmt.Sprintf("path %s should be inside %s", p,
topDir))
    }
    return p, nil
}
```

In another test scenario, the FeatureCloud controller exhibited vulnerabilities in its Docker configuration, where the Docker image for the fc-controller did not specify a non-root user, resulting in the application running with elevated root permissions. This was evident from the Docker run script used to start the fc-controller.

The fc-controller Docker image lacks a specific USER directive, causing it to default to using the root user. Consequently, the image's ENTRYPOINT, which launches the application /go/bin/controller, operates with root permissions 11.5.1 To support running the controller with a specific user, we extended app.cli.controller.commands in a way that the start command accepts a new --user option. This option allows specifying the username with which the controller should run.

```
@controller.command('start')
@click.option('--user', default='root', help='Username to run the controller.
Optional parameter (e.g. --user=non_root_user).', required=False)
# ... (other options)
def start(user: str, name: str, port: int, data_dir: str, controller_image,
gpu: bool, mount: str) -> None:
    # ... (existing code)
    Try:
        commands.start(name, port, data_dir, controller_image, gpu, mount,
user)
        click.echo(f'Started controller: {name} with user: {user}')
```

Also, In `app.imp.controller.commands`, modify the `start` function to accept the `user` parameter and use it when creating the Docker container.

```
def start(name: str, port: int, data_dir: str, controller_image: str,
with_gpu: bool, mount: str, user: str = 'root'):
    if os.geteuid() != 0 and user == 'root':
        raise EnvironmentError("Only the `root` user can run FeatureCloud
controller by setting the user as `root`.")

    Try:
        client.containers.run(

            user=user,
```

Finally, after implementing these changes, we start the FeatureCloud controller with a specified non-root user by using the following command:

```
$ featurecloud controller start --user non_root_user
```

6.4.2 Common Vulnerabilities and Exposures

In another scenario, we tried to exploit possible outdated libraries with known vulnerabilities inside the controller. These Docker images, which are based on standard Linux distribution base images, were not regularly updated, leading to potential security risks. The Docker image for the “`featurecloud.ai/controller`” was found to contain multiple outdated libraries with high to medium severity vulnerabilities:

- github.com/docker/distribution: Vulnerabilities like CVE-2023-2253 (Common Vulnerabilities and Exposures)
- github.com/docker/docker
- golang.org/x/net

The vulnerabilities within these libraries primarily threatened the integrity and availability of the system. Notably, they did not pose a direct threat to confidentiality but could lead to system instability and unauthorized data modification or deletion. To mitigate these risks, we have enhanced our CI/CD pipeline with a robust routine focused on continuous monitoring of libraries used in our Docker images. This proactive approach enables us to quickly identify known vulnerabilities and implement necessary countermeasures. Our response primarily involves updating the affected libraries to their secured, fixed versions, thereby ensuring the ongoing stability and security of the FeatureCloud platform.

To address the issue of outdated libraries with known vulnerabilities in the Docker images, we improved our CI/CD pipeline to include automated checks for library vulnerabilities by integrating Trivy as a vulnerability scanning tool and updating our CI/CD pipeline configuration. The new stage runs after the build stage, but before the deployment stage:

```
check:vulnerabilities:
  image: registry.blitzhub.io/trivy
```

```
stage: check
script:
  - trivy image $REGISTRY_URL/featurecloud/controller:staging
dependencies:
  - build_controller
```

We also modify the deployment stages to depend on the vulnerability check. This ensures that deployment only happens if the vulnerability check passes.

```
deploy:gitlab:staging:
  # ...
  dependencies:
    - check:lint
    - check:test
    - check:vulnerabilities
  # ...
deploy:main:staging:
  # ...
  dependencies:
    - check:lint
    - check:test
    - check:vulnerabilities
```

We also add regular base image updates to ensure that the base images used in the Dockerfile are regularly updated to the latest versions.

```
before_script:
  - docker pull golang:latest # Ensure the latest base image is used
```

And we add library update checks to include steps in the pipeline to check for and update the dependencies in the project to their latest, secure versions:

```
setup:
  stage: setup
  script:
    - go get -u ./... # Update all dependencies
```

The “./...” in the “go get -u ./...” command ensures that the update operation applies recursively to all packages in the current directory and its subdirectories, covering all project dependencies. If the project uses Go modules, as standard from Go 1.11 onwards, this command will update dependencies within the constraints defined in the “go.mod” file, ensuring compatibility and version control. Once the version deployed on the staging server shows stability without producing any errors, we manually deploy the updates on the production server.

6.4.3 Brute-force attack

In the next scenario, we systematically attempted various password combinations to gain unauthorized access which resulted in identifying a significant security vulnerability in terms of the absence of brute-force attack protection on the FeatureCloud platform. The staging.featurecloud.ai system lacked measures to prevent or mitigate brute-force attacks, posing a high risk to both confidentiality and integrity. The test stressed the platform by submitting numerous incorrect login attempts within a short time without triggering any account lockout or alert mechanisms.

We incorporated the feedback into the platform for this issue by implementing an account lockout policy including temporarily locking the target account after five consecutive failed login attempts for a duration of five minutes. We introduced two new fields, “failed_login_attempts” and “lockout_until”, to the FCUser model to enhance security. These fields help us keep track of consecutive failed login attempts and specify a time period during which the user account remains locked. Additionally, we added the “lockout_expired” method to determine if the lockout period has passed, allowing the user to attempt logging in again.

```
class FCUser(AbstractUser):
    site = models.ForeignKey('Site', on_delete=models.SET_NULL, null=True,
blank=True)
    is_site_admin = models.BooleanField(default=False)
    # ... (other fields of FCUser) ...

    # Add fields for tracking failed login attempts and lockout timestamp
    failed_login_attempts = models.IntegerField(default=0)
    lockout_until = models.DateTimeField(null=True, blank=True)

    def lockout_expired(self):
        # Check if the lockout period has expired
        if self.lockout_until and now() >= self.lockout_until:
            return True
        return False
```

We modified the “login_view” function to incorporate our new account lockout mechanism. Now, when a user tries to log in, we first check if their account is locked. If it is, we redirect them to an error page. For successful logins, we reset the “failed_login_attempts” to zero and clear the “lockout_until” field, effectively unlocking the account. In case of failed login attempts, we increment the “failed_login_attempts” and, if necessary, set the lockout_until field to lock the account for a period. We made sure to render appropriate views based on whether the login attempt succeeds, fails, or if the account is locked.

```
def login_view(request):

    def post(self, request):
        username = request.POST.get('username')
        password = request.POST.get('password')
        user = authenticate(username=username, password=password)

        if user is not None:
```

```
# Check if the account is locked
if user.lockout_expired() or user.lockout_until is None:
    # Reset failed login attempts and lockout timestamp upon successful
login
    user.failed_login_attempts = 0
    user.lockout_until = None
    user.save()
    # Proceed with login process
    login(request, user)
    return Response({'success': True})
else:
    content = {'success': False, 'detail': 'Account is locked'}
    return Response(content, status=status.HTTP_401_UNAUTHORIZED)
else:
    # Failed login attempt
    try:
        user = User.objects.get(username=username)
        user.failed_login_attempts += 1
        if user.failed_login_attempts >= 5:
            user.lockout_until = timezone.now() + timezone.timedelta(minutes=5)
            user.save()
    except User.DoesNotExist:
        pass

    content = {'success': False, 'detail': 'No active account found with the
given credentials'}
    return Response(content, status=status.HTTP_401_UNAUTHORIZED)
```

We implemented “BruteForceProtectionMiddleware” to handle preemptive checks on login attempts. This middleware intervenes during POST requests to the login URL. It checks whether the username exists and if the corresponding account is locked. If an account is locked, it raises a PermissionDenied exception, thus preventing further processing of the login attempt. For login attempts with non-existent usernames, we log these events for security monitoring and analysis.

```
class BruteForceProtectionMiddleware:
    def __init__(self, get_response):
        self.get_response = get_response

    def __call__(self, request):
        response = self.get_response(request)
        return response

    def process_view(self, request, view_func, view_args, view_kwargs):
        if request.method == 'POST' and '/login' in request.path:
            username = request.POST.get('username')
```

```
try:
    user = User.objects.get(username=username)
    if not user.lockout_expired() and user.lockout_until is not None:
        raise PermissionDenied('Account is temporarily locked due to multiple
failed login attempts.')
    except User.DoesNotExist:
        logger.info(f"Login attempt with non-existing username: {username}")
    return None
```

6.4.4 Manipulate the confidentiality and integrity of data

We stressed the platform by attempting to manipulate the confidentiality and integrity of data transmission in the platform by exploiting the usage of insecure TLS versions in the platform. In fact, the platform uses TLS 1.0 and 1.1 for secure communication. These versions are considered insecure due to vulnerabilities that cannot be remediated:

- TLS 1.0: Utilizes the SHA-1 hash algorithm, which is no longer secure, and supports outdated ciphers susceptible to multiple known vulnerabilities like FREAK, DROWN, BEAST, CRIME, and POODLE.
- TLS 1.1: Similar to TLS 1.0, it uses the SHA-1 algorithm for signature creation and lacks significant support among modern browsers and clients.

Consequently, we upgraded the platform to use modern TLS versions 1.2 while completely disabling all older versions (SSLv2, SSLv3, TLS 1.0, TLS 1.1). We updated the Apache server configurations in both staging and production servers:

```
SSLProtocol all -SSLv2 -SSLv3 -TLSv1 -TLSv1.1
```

6.4.5 SSL Stripping attacks

Moreover, we stressed the platform by SSL Stripping attacks to manipulate confidentiality and integrity. We attempt to intercept the initial HTTP request and prevent redirection to a secure HTTPS connection, gaining the ability to see and manipulate the data. The test was successful because of the lack of HTTP Strict Transport Security (HSTS) Headers.

```
HTTP/1.1 200 OK
Date: Tue, 21 Nov 2023 10:37:26 GMT
Server: nginx/1.19.4
Content-Type: text/html
Last-Modified: Tue, 31 Oct 2023 13:14:58 GMT
ETag: "6540fdd2-1386-gzip"
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Length: 4998
Connection: close
```

Therefore, to improve the platform, we configured the web servers to send the HSTS Header with a “max-age” of one year and include all subdomains. This measure ensures that browsers will automatically upgrade all HTTP requests to HTTPS, even if the user enters an HTTP URL.

Recommended configuration for Apache:

```
Header always set Strict-Transport-Security "max-age=31536000;  
includeSubDomains"
```

Recommended configuration for NGINX:

```
add_header Strict-Transport-Security "max-age=31536000;  
includeSubDomains" always;
```

Recommended configuration for IIS:

```
<system.webServer>  
...  
<httpProtocol>  
<customHeaders>  
<add name="Strict-Transport-Security" value="max-age=31536000;  
includeSubDomains"/>  
</customHeaders>  
</httpProtocol>  
...  
</system.webServer>
```


7 Open issues

No open issues remaining.

8 Deviations

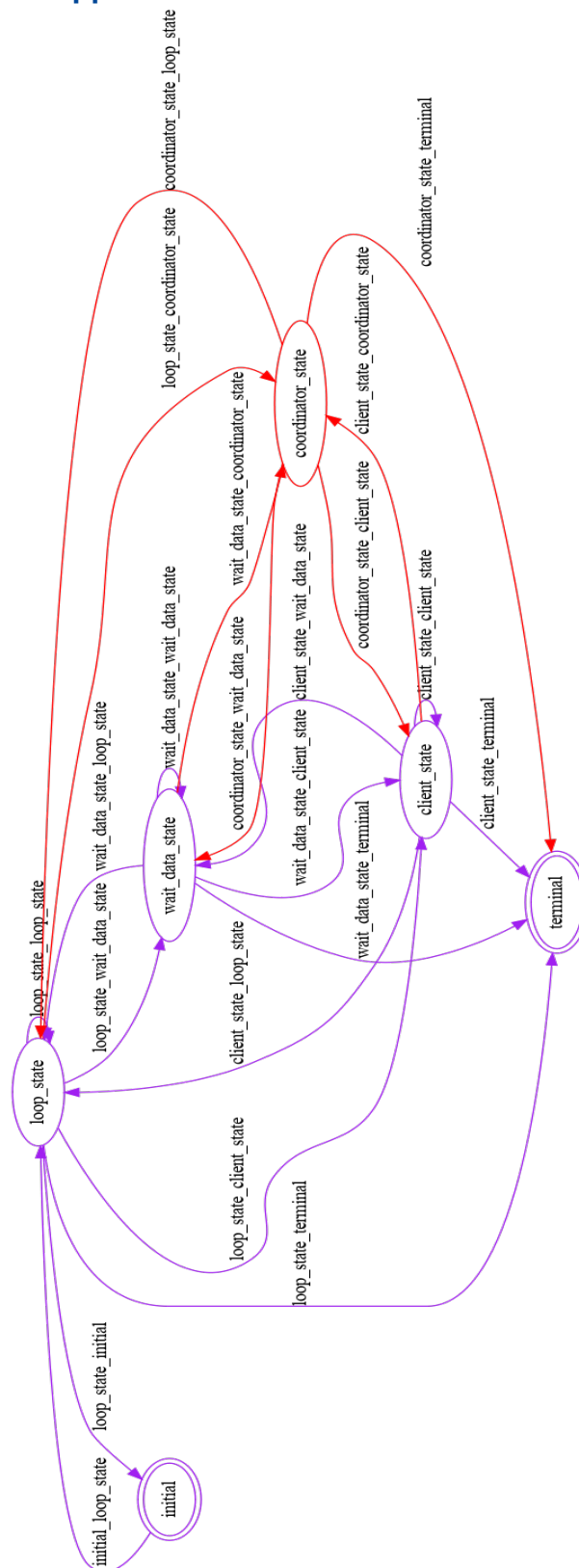
As it was discussed in D7.4 and D7.5, the patients' user interface (UI) is deployed on hospitals' infrastructure due to privacy concerns. This affects a considerable aspect of blockchain integration because all the patient's consent management alongside workflow auditing will take place on the hospital's side where various technologies, including hardware and software, can be used. Accordingly, designing tests to stress the blockchain integration could not capture real world aspects of the deployment on hospitals platform since we could not simulate the production environment.

9 Conclusion

In conclusion, in this deliverable, we reported the FeatureCloud platform's resilience to have a successful performance in a series of stress tests. Specifically, the platform demonstrated robust web security, effectively countering threats like Cross-Site Request Forgery (CSRF), SSL stripping, and vulnerabilities in TLS protocols. These tests, conducted using tools such as Burp Suite, Trivy, and Nessus, established the platform's capability to safeguard data integrity and confidentiality. Moreover, the controller component of FeatureCloud showcased its efficiency in managing complex workflows under high-stress conditions. Equally impressive was the pip package, which proved its reliability in data communication, a critical aspect for developers in federated environments. The platform's ability to withstand these diverse tests not only confirms its operational resilience but also assures users of its reliability in managing federated data processing and analysis securely and efficiently.

10 Other supporting documents / figures / tables

10.1 Communication test app



10.2 stress testing the Controller

This section provides more detailed logs on the controllers under stress.

10.2.1 Controller Fails the stress test

Controller logs before the robustness improvements, see the “error” entry, after which the workflow is still in “running” state:

```
time="2022-04-26 09:43:51" level=info msg="[FLOW] 63 Start workflow"
time="2022-04-26 09:43:55" level=info msg="[FLOW] 63 Info change for project
63: stage: running, step: 0"
time="2022-04-26 09:43:56" level=info msg="[FLOW] 63 Start application:
fc_stagingfeaturecloudaitestappsha256ca6cef25fae208ea3ef9db4df49d81e4e369dc1c5
0c1939ad514ea7bd326397f_579646700"
time="2022-04-26 09:43:56" level=info msg="[FLOW] 63 Start relaying
fc_stagingfeaturecloudaitestappsha256ca6cef25fae208ea3ef9db4df49d81e4e369dc1c5
0c1939ad514ea7bd326397f_579646700"
time="2022-04-26 09:43:59" level=info msg="[FLOW] 63 Trigger setup"
time="2022-04-26 09:43:59" level=info msg="[LOCAL] Received setup trigger"
time="2022-04-26 09:44:03" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:06" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:09" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:12" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:16" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:19" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:22" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:22" level=error msg="[LOCAL] Get
\"http://172.29.0.3:9000/status\": dial tcp 172.29.0.3:9000: connectex: No
connection could be made because the target machine actively refused it."
time="2022-04-26 09:44:25" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:28" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:31" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:34" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:37" level=debug msg="[FLOW] 63 Workflow running"
time="2022-04-26 09:44:40" level=debug msg="[FLOW] 63 Workflow running"
```

10.2.2 Controller survives the stress test

Controller logs after the robustness improvements, see the “error” entry, after which the workflow is still in “running” state:

```
time="2022-05-04 10:23:34" level=info msg="[FLOW] 63 Start workflow"
time="2022-05-04 10:23:37" level=info msg="[FLOW] 63 Info change for project
63: stage: running, step: 0"
time="2022-05-04 10:23:38" level=info msg="[FLOW] 63 Start application:
fc_stagingfeaturecloudaitestappsha256ca6cef25fae208ea3ef9db4df49d81e4e369dc1c5
0c1939ad514ea7bd326397f_316668300"
time="2022-05-04 10:23:39" level=info msg="[FLOW] 63 Start relaying
fc_stagingfeaturecloudaitestappsha256ca6cef25fae208ea3ef9db4df49d81e4e369dc1c5
0c1939ad514ea7bd326397f_316668300"
time="2022-05-04 10:23:39" level=debug msg="[DATA] 63 Updated project state
for project 63"
time="2022-05-04 10:23:42" level=debug msg="[FLOW] 63 Trying to advance
project 63... 1/1"
time="2022-05-04 10:23:42" level=info msg="[FLOW] 63 Trigger setup"
time="2022-05-04 10:23:42" level=info msg="[LOCAL] Received setup trigger"
time="2022-05-04 10:23:46" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:23:49" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:23:52" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:23:55" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:23:58" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:24:01" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:24:04" level=debug msg="[FLOW] 63 Workflow running"
time="2022-05-04 10:24:05" level=debug msg="[DATA] 63 Updated project state
for project 63"
time="2022-05-04 10:24:05" level=error msg="[LOCAL] Get
\"http://172.29.0.3:9000/status\": dial tcp 172.29.0.3:9000: connectex: No
connection could be made because the target machine actively refused it."
time="2022-05-04 10:24:08" level=info msg="[FLOW] 63 Info change for project
63: stage: error, step: 0"
time="2022-05-04 10:24:08" level=error msg="[FLOW] 63 Workflow is shutting
down because of an error"
```

10.3 Aira Server Test Environment

Test environment logs for the “aira” server.

```
Hostname:
aira

Operating System Info:
NAME="openSUSE Tumbleweed"
# VERSION="20211012"
ID="opensuse-tumbleweed"
ID_LIKE="opensuse suse"
VERSION_ID="20211012"
PRETTY_NAME="openSUSE Tumbleweed"
ANSI_COLOR="0;32"
CPE_NAME="cpe:/o:opensuse:tumbleweed:20211012"
BUG_REPORT_URL="https://bugzilla.opensuse.org"
SUPPORT_URL="https://bugs.opensuse.org"
HOME_URL="https://www.opensuse.org"
DOCUMENTATION_URL="https://en.opensuse.org/Portal:Tumbleweed"
LOGO="distributor-logo-Tumbleweed"

Kernel Info:
6.5.6-1-default

CPU Info:
Architecture:                x86_64
CPU op-mode(s):              32-bit, 64-bit
Address sizes:                46 bits physical, 48 bits virtual
Byte Order:                   Little Endian
CPU(s):                       112
On-line CPU(s) list:         0-111
Vendor ID:                    GenuineIntel
Model name:                   Intel(R) Xeon(R) Gold 6258R CPU @ 2.70GHz
CPU family:                   6
Model:                        85
Thread(s) per core:           2
Core(s) per socket:           28
Socket(s):                    2
Stepping:                     7
BogoMIPS:                     5402.00
Flags:                         fpu vme de pse tsc msr pae mce cx8 apic
sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts
rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64
monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1
```

```
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand
lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 invpcid_single
intel_ppin ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow flexpriority ept
vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm mpx rdt_a
avx512f avx512dq rdseed adx smap clflushopt clwb intel_pt avx512cd avx512bw
avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total
cqm_mbm_local dtherm ida arat pln pts vnmi pku ospke avx512_vnni md_clear
flush_lld arch_capabilities
```

```
Virtualization: VT-x
L1d cache: 1.8 MiB (56 instances)
L1i cache: 1.8 MiB (56 instances)
L2 cache: 56 MiB (56 instances)
L3 cache: 77 MiB (2 instances)
NUMA node(s): 2
NUMA node0 CPU(s):
0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46,48,50,52,54
,56,58,60,62,64,66,68,70,72,74,76,78,80,82,84,86,88,90,92,94,96,98,100,102,104
,106,108,110
NUMA node1 CPU(s):
1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47,49,51,53,55
,57,59,61,63,65,67,69,71,73,75,77,79,81,83,85,87,89,91,93,95,97,99,101,103,105
,107,109,111
Vulnerability Gather data sampling: Mitigation; Microcode
Vulnerability Itlb multihit: KVM: Mitigation: VMX disabled
Vulnerability L1tf: Not affected
Vulnerability Mds: Not affected
Vulnerability Meltdown: Not affected
Vulnerability Mmio stale data: Mitigation; Clear CPU buffers; SMT
vulnerable
Vulnerability Retbleed: Mitigation; Enhanced IBRS
Vulnerability Spec rstack overflow: Not affected
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass
disabled via prctl
Vulnerability Spectre v1: Mitigation; usercopy/swaps barriers and
__user pointer sanitization
Vulnerability Spectre v2: Mitigation; Enhanced / Automatic IBRS,
IBPB conditional, RSB filling, PBRB-eIBRS SW sequence
Vulnerability Srbds: Not affected
Vulnerability Tsx async abort: Mitigation; TSX disabled
```

Memory Info:

	total	used	free	shared	buff/cache
available					
Mem:	503Gi	30Gi	114Gi	5.0Mi	361Gi
472Gi					

```
Swap:          2.0Gi          55Mi          1.9Gi

Disk Usage:
Filesystem                                Size  Used Avail Use% Mounted on
/dev/sdb2                                1.1T   21G   1.1T   2% /
devtmpfs                                4.0M   8.0K   4.0M   1% /dev
tmpfs                                    252G   4.0K   252G   1% /dev/shm
efivarfs                                304K   115K   185K   39%
/sys/firmware/efi/efivars
tmpfs                                    101G   2.6M   101G   1% /run
/dev/sdb2                                1.1T   21G   1.1T   2%
/boot/grub2/i386-pc
/dev/sdb2                                1.1T   21G   1.1T   2%
/boot/grub2/x86_64-efi
/dev/sdb2                                1.1T   21G   1.1T   2% /opt
/dev/sdb2                                1.1T   21G   1.1T   2% /root
/dev/sdb2                                1.1T   21G   1.1T   2% /srv
/dev/sdb2                                1.1T   21G   1.1T   2%
/usr/local
/dev/sdb2                                1.1T   21G   1.1T   2% /var
tmpfs                                    252G   164K   252G   1% /tmp
/dev/sda1                                8.8T   3.5T   5.4T   40% /home
/dev/sdb1                                511M   5.9M   506M   2%
/boot/efi
fs-s-nas04.rrz.uni-hamburg.de:/nfs-min/cosybio  80T   40T   41T   50% /cosybio
tmpfs                                    51G    4.0K   51G    1%
/run/user/7001780
tmpfs                                    51G    4.0K   51G    1%
/run/user/7021401
tmpfs                                    51G    4.0K   51G    1%
/run/user/7031037
tmpfs                                    51G    4.0K   51G    1%
/run/user/6880865

Network Interfaces:
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host proto kernel_lo
        valid_lft forever preferred_lft forever
2: em1: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond0
state UP group default qlen 1000
```



```
link/ether aa:84:ed:ed:a3:f4 brd ff:ff:ff:ff:ff:ff permaddr
5c:6f:69:25:63:5a
    altname eno1np0
    altname enp25s0f0np0
3: em2: <BROADCAST,MULTICAST,SLAVE,UP,LOWER_UP> mtu 1500 qdisc mq master bond0
state UP group default qlen 1000
    link/ether aa:84:ed:ed:a3:f4 brd ff:ff:ff:ff:ff:ff permaddr
5c:6f:69:25:63:5b
    altname eno2np1
    altname enp25s0f1np1
4: idrac: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
    link/ether ec:2a:72:2b:47:4d brd ff:ff:ff:ff:ff:ff
    altname enp0s20f0u14u3
5: em3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
    link/ether 5c:6f:69:25:63:58 brd ff:ff:ff:ff:ff:ff
    altname eno3
    altname enp1s0f0
6: em4: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default
qlen 1000
    link/ether 5c:6f:69:25:63:59 brd ff:ff:ff:ff:ff:ff
    altname eno4
    altname enp1s0f1
7: bond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
    link/ether aa:84:ed:ed:a3:f4 brd ff:ff:ff:ff:ff:ff
    inet 134.100.85.17/26 brd 134.100.85.63 scope global bond0
        valid_lft forever preferred_lft forever
533: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state
DOWN group default
    link/ether 02:42:4f:11:eb:4d brd ff:ff:ff:ff:ff:ff
    inet 172.172.0.1/24 brd 172.172.0.255 scope global docker0
        valid_lft forever preferred_lft forever
34: br-130ce407edc7: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc
noqueue state DOWN group default
    link/ether 02:42:ea:b2:6a:71 brd ff:ff:ff:ff:ff:ff
    inet 172.172.3.1/24 brd 172.172.3.255 scope global br-130ce407edc7
        valid_lft forever preferred_lft forever
```

System Load and Uptime:

16:29:42 up 44 days 6:40, 4 users, load average: 12.27, 13.29, 12.97

Available System Updates:



Current Logged-In Users:

bbb1037	pts/1	2021-11-27	16:27	(134.100.40.3)
bam0865	pts/7	2021-11-27	16:23	(134.100.40.24)
bay1780	pts/6	2021-11-27	16:17	(134.100.40.30)
bay1780	pts/5	2021-11-27	14:01	(134.100.40.30)

10.4 Stress Testing the Pip Package

This section contains more detailed logs of stress testing the pip package and detailed codes of improving the robustness.

10.4.1 Incorporation of the feedback into the pip package

Extract from *http_ctrl.py* defining the changed API that serves the controller. *app.handle_status()* includes the memo.

```
@api_server.get('/status')
def ctrl_status():
    print(f'[CTRL] GET /status')
    return app.handle_status()
@api_server.route('/data', method='POST')
def ctrl_data_in():
    print(f'[CTRL] POST /data')
    if "memo" in request.query:
        memo = request.query["memo"]
    else:
        memo = None
    return app.handle_incoming(request.body.read(), request.query['client'],
                               memo=memo)
```

The *app.py* file defining the connector for users to the API. It now includes a more robust status handling that uses an individual status object for each sending operation. Furthermore, it is ensured that each data piece when sent to the controller had the previous status call answered with the correct data object. The *app.py* file provides all the required functionalities for app development, and covering all its content is out of scope for this deliverable. Therefore, here we only include the parts affected or implemented to address the platform robustness against stress testing. Accordingly, the code documentation is also not covered. For more detail visit our public repository on GitHub⁵.

```
class App:
    def __init__(self):
        self.id = None
        self.coordinator = None
        self.clients = None
        self.default_memo = None
        self.thread: Union[threading.Thread, None] = None
```

⁵ <https://github.com/FeatureCloud/FeatureCloud>

```
self.data_incoming = {}
self.data_outgoing = []
self.default_smpc: SMPCType = {'operation': 'add', 'serialization':
'json', 'shards': 0, 'exponent': 8}
self.default_dp: DPTType = {'serialization': 'json', 'noisetype':
'laplace',
                             'epsilon': 1.0, 'delta': 0.0,
                             'sensitivity': None, 'clippingVal': 10.0}

self.current_state: Union[AppState, None] = None
self.states: Dict[str, AppState] = {}
self.transitions: Dict[
    str, Tuple[AppState, AppState, bool, bool, str]] = {} # name =>
(source, target, participant, coordinator, label)

self.transition_log: List[Tuple[datetime.datetime, str]] = []

self.internal = {}

self.status_available: bool = False
self.status_finished: bool = False
self.status_message: Union[str, None] = None
self.status_progress: Union[float, None] = None
self.status_state: Union[str, None] = None
self.status_destination: Union[str, None] = None
self.status_smpc: Union[SMPCType, None] = None
self.status_dp: Union[DPTType, None] = None
self.status_memo: Union[str, None] = None

self.last_send_status = self.get_current_status()

# Add terminal state
@app_state('terminal', Role.BOTH, self)
class TerminalState(AppState):
    def register(self):
        pass

    def run(self) -> str:
        pass

def get_current_status(self, **kwargs):
    status = dict()
    status["available"] = self.status_available
    status["finished"] = self.status_finished
```

```
status["message"] = self.status_message
status["progress"] = self.status_progress
status["state"] = self.status_state
status["destination"] = self.status_destination
status["smpc"] = self.status_smpc
status["dp"] = self.status_dp
status["memo"] = self.status_memo
for key, value in kwargs.items():
    # set whatever is wanted from the arguments
    status[key] = value
return status

def handle_incoming(self, data, client, memo):
    if memo not in self.data_incoming:
        self.data_incoming[memo] = [(data, client)]
    else:
        self.data_incoming[memo].append((data, client))

class AppState(abc.ABC):

    def aggregate_data(self, operation: SMPCOperation = SMPCOperation.ADD,
use_smpc=False,
                        use_dp=False, memo=None):

        if use_smpc:
            return self.await_data(n=1, unwrap=True, is_json=True, memo=memo)
            # Data is aggregated already
        else:
            data = self.gather_data(is_json=use_dp, memo=memo)
            return _aggregate(data, operation)
            # Data needs to be aggregated according to operation

    def gather_data(self, is_json=False, use_smpc=False, use_dp=False,
memo=None):
        if not self._app.coordinator:
            self._app.log('must be coordinator to use gather_data',
level=LogLevel.FATAL)
            n = len(self._app.clients)
            if use_smpc or use_dp:
                is_json = True
            if use_smpc:
                n = 1
            return self.await_data(n, unwrap=False, is_json=is_json,
use_dp=use_dp,
                                use_smpc=use_smpc, memo=memo)
```

```

def await_data(self, n: int = 1, unwrap=True, is_json=False,
               use_dp=False, use_smpc=False, memo=None):
    if use_smpc:
        n = 1
        is_json = True
    if use_dp:
        is_json = True

    while True:
        # print(f"Current Incoming Data: {self._app.data_incoming}")
        num_data_pieces = 0
        if memo in self._app.data_incoming:
            num_data_pieces = len(self._app.data_incoming[memo])
        if num_data_pieces >= n:
            # warn if too many data pieces came in
            if num_data_pieces > n:
                self._app.log(
                    f"await was used to wait for {n} data pieces, " +
                    f"but more data pieces ({num_data_pieces}) were found.
" +
                    f"Used memo is <{memo}>",
                    LogLevel.ERROR)

            # extract and deseralize the data
            data = self._app.data_incoming[memo][:n]
            self._app.data_incoming[memo] =
self._app.data_incoming[memo][n:]
            if len(self._app.data_incoming[memo]) == 0:
                # clean up the dict regularly
                del self._app.data_incoming[memo]
            if n == 1 and unwrap:
                return _deserialize_incoming(data[0][0], is_json=is_json)
            else:
                return [_deserialize_incoming(d[0], is_json=is_json) for d
in data]

        sleep(DATA_POLL_INTERVAL)

def send_data_to_participant(self, data, destination, use_dp=False,
                             memo=None):

    data = _serialize_outgoing(data, is_json=use_dp)
    if destination == self._app.id and not use_dp:
        # In no DP case, the data does not have to be sent via the

```

```
controller
    self._app.handle_incoming(data, client=self._app.id, memo=memo)
    else:
        # update the status variables and get the status object
        message = self._app.status_message if self._app.status_message
    else (self._app.current_state.name if self._app.current_state else None)
        dp = self._app.default_dp if use_dp else None
        self._app.status_message = message
        status = self._app.get_current_status(message=message,
                                              destination=destination, dp=dp, memo=memo,
                                              available=True)
        self._app.data_outgoing.append((data, json.dumps(status,
sort_keys=True)))

    def send_data_to_coordinator(self, data, send_to_self=True,
use_smpc=False,
                                use_dp=False, memo=None):

        if use_smpc or use_dp:
            data = _serialize_outgoing(data, is_json=True)

        else:
            data = _serialize_outgoing(data, is_json=False)

        if self._app.coordinator and not use_smpc and not use_dp:
            if send_to_self:
                self._app.handle_incoming(data, self._app.id, memo)
        else:
            # for SMPC and DP, the data has to be sent via the controller
            if use_dp and self._app.coordinator:
                destination = self._app.id
            else:
                destination = None
            # this is interpreted as to the coordinator
            message = self._app.status_message if self._app.status_message
    else (self._app.current_state.name if self._app.current_state else None)
        self._app.status_message = message
        smpc = self._app.default_smpc if use_smpc else None
        dp = self._app.default_dp if use_dp else None
        status = self._app.get_current_status(message=message,
                                              destination=destination, smpc=smpc, dp=dp, memo=memo,
                                              available=True)
        self._app.data_outgoing.append((data, json.dumps(status,
sort_keys=True)))
```

```
def broadcast_data(self, data, send_to_self=True, use_dp = False,
                  memo = None):
    if not self._app.coordinator:
        self._app.log('only the coordinator can broadcast data',
level=LogLevel.FATAL)

    is_json = False
    if use_dp:
        is_json = True

    # serialize before broadcast
    data = _serialize_outgoing(data, is_json=False)

    message = self._app.status_message if self._app.status_message else
(self._app.current_state.name if self._app.current_state else None)
    self._app.status_message = message
    dp = self._app.default_dp if use_dp else None
    status = self._app.get_current_status(message=message,
        destination=None, dp=dp, memo=memo,
        available=True)
    if send_to_self:
        self._app.handle_incoming(data, client=self._app.id, memo=memo)
        self._app.data_outgoing.append((data, json.dumps(status,
sort_keys=True)))
```

10.4.2 Data misplacement

Client one (Coordinator) log:

```
[Time: 30.11.23 17:44:36] [Level: info] id: 5eea47b3e422a904
[Time: 30.11.23 17:44:36] [Level: info] coordinator: True
[Time: 30.11.23 17:44:36] [Level: info] clients: ['91ff362b71b912fd',
'5eea47b3e422a904']
[Time: 30.11.23 17:44:36] [Level: info] state: initial
[Time: 30.11.23 17:44:36] [Level: info] [State: initial] App is started...
[Time: 30.11.23 17:44:36] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[Time: 30.11.23 17:44:36] [Level: info] [State: initial] Coordinator broadcast
test settings...
[Time: 30.11.23 17:44:36] [Level: info] transition: loop_state
127.0.0.1 - - [30/Nov/2023 17:44:36] "POST /api/setup HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:36] "GET /api/status HTTP/1.0" 200 143
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 17:44:36] "GET /api/data HTTP/1.0" 200 40
```



```
[Time: 30.11.23 17:44:37] [Level: info] state: loop_state
[Time: 30.11.23 17:44:37] [Level: info] transition: wait_data_state
[Time: 30.11.23 17:44:38] [Level: info] state: wait_data_state
[Time: 30.11.23 17:44:38] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 17:44:38] [Level: info] transition: client_state
[Time: 30.11.23 17:44:39] [Level: info] state: client_state
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 4, client send data 41 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 5, client send data 51 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 6, client send data 61 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 7, client send data 71 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:39] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:39] "GET /api/status HTTP/1.0" 200 151
[Time: 30.11.23 17:44:40] [Level: info] state: coordinator_state
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Coordinator
is launched...
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 11] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 1 gather data: [21, 31] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 2 gather data: [41, 51] SMPC: False, DP: False
[Time: 30.11.23 17:44:40] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 3 gather data: [61, 71] SMPC: False, DP: False
```

```
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:42] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:44] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:45] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:47] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[Time: 30.11.23 17:44:47] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 4 gather data: [1, 11] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:48] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:50] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:51] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:53] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[Time: 30.11.23 17:44:53] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 5 gather data: [21, 31] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:54] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:56] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:57] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:59] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[Time: 30.11.23 17:44:59] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 6 gather data: [41, 51] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:45:00] "GET /api/status HTTP/1.0" 200 151
127.0.0.1 - - [30/Nov/2023 17:45:02] "POST /api/data?client=91ff362b71b912fd
HTTP/1.0" 200 0
[CTRL] POST /data
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:45:03] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:45:05] "POST /api/data?client=91ff362b71b912fd
```

```
HTTP/1.0" 200 0
[Time: 30.11.23 17:45:05] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 7 gather data: [61, 71] SMPC: False, DP: False
[Time: 30.11.23 17:45:05] [Level: info] transition: loop_state
```

Client two (participant) log:

```
[Time: 30.11.23 17:44:38] [Level: info] id: 91ff362b71b912fd
[Time: 30.11.23 17:44:38] [Level: info] coordinator: False
[Time: 30.11.23 17:44:38] [Level: info] clients: ['91ff362b71b912fd',
'5eea47b3e422a904']
[Time: 30.11.23 17:44:38] [Level: info] state: initial
[Time: 30.11.23 17:44:38] [Level: info] [State: initial] App is started...
[Time: 30.11.23 17:44:38] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
127.0.0.1 - - [30/Nov/2023 17:44:38] "POST /api/setup HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:38] "GET /api/status HTTP/1.0" 200 141
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 17:44:38] "POST /api/data?client=5eea47b3e422a904
HTTP/1.0" 200 0
[Time: 30.11.23 17:44:38] [Level: info] transition: loop_state
[Time: 30.11.23 17:44:39] [Level: info] state: loop_state
[Time: 30.11.23 17:44:39] [Level: info] transition: wait_data_state
[Time: 30.11.23 17:44:40] [Level: info] state: wait_data_state
[Time: 30.11.23 17:44:40] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 17:44:40] [Level: info] transition: client_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 17:44:41] "GET /api/status HTTP/1.0" 200 146
[Time: 30.11.23 17:44:41] [Level: info] state: client_state
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 4, client send data 41 to coordinator .... SMPC:
```

```
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 5, client send data 51 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 6, client send data 61 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 7, client send data 71 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 17:44:41] [Level: info] transition: loop_state
[Time: 30.11.23 17:44:42] [Level: info] state: loop_state
[Time: 30.11.23 17:44:42] [Level: info] transition: wait_data_state
```

10.4.3 Simple-SMPC Pickle error

Client one (Coordinator) log:

```
[Time: 30.11.23 18:07:03] [Level: info] id: b47aaefed78d6811
[Time: 30.11.23 18:07:03] [Level: info] coordinator: True
[Time: 30.11.23 18:07:03] [Level: info] clients: ['b47aaefed78d6811',
'5c0d492b0866d4a7']
[Time: 30.11.23 18:07:03] [Level: info] state: initial
[Time: 30.11.23 18:07:03] [Level: info] [State: initial] App is started...
[Time: 30.11.23 18:07:03] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[Time: 30.11.23 18:07:03] [Level: info] [State: initial] Coordinator broadcast
test settings...
[Time: 30.11.23 18:07:03] [Level: info] transition: loop_state
127.0.0.1 - - [30/Nov/2023 18:07:03] "POST /api/setup HTTP/1.0" 200 0
127.0.0.1 - - [30/Nov/2023 18:07:03] "GET /api/status HTTP/1.0" 200 143
[CTRL] GET /status
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:07:03] "GET /api/data HTTP/1.0" 200 40
[Time: 30.11.23 18:07:04] [Level: info] state: loop_state
[Time: 30.11.23 18:07:04] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:07:05] [Level: info] state: wait_data_state
[Time: 30.11.23 18:07:05] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:07:05] [Level: info] transition: client_state
[Time: 30.11.23 18:07:06] [Level: info] state: client_state
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
```

```
True, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:07:06] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:07:06] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:06] "GET /api/status HTTP/1.0" 200 219
127.0.0.1 - - [30/Nov/2023 18:07:06] "GET /api/data HTTP/1.0" 200 2
[CTRL] GET /data
[Time: 30.11.23 18:07:07] [Level: info] state: coordinator_state
[Time: 30.11.23 18:07:07] [Level: info] [State: coordinator_state] Coordinator
is launched...
[Time: 30.11.23 18:07:07] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 21] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:09] "GET /api/status HTTP/1.0" 200 219
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:07:09] "GET /api/data HTTP/1.0" 200 2
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:07:11] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:07:12] "GET /api/status HTTP/1.0" 200 151
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:07:15] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[Time: 30.11.23 18:07:15] [Level: info] Traceback (most recent call last):
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 226, in guarded_run
    self.run()
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 241, in run
    transition = self.current_state.run()
  File "/app/states.py", line 454, in run
    data_to_collect = self.gather_data(is_json=flg)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 618, in gather_data
    return self.await_data(len(self._app.clients), unwrap=False,
is_json=is_json)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 648, in await_data
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
```

```
File "/root/.local/lib/python3.8/site-  
packages/FeatureCloud/app/engine/app.py", line 648, in <listcomp>  
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]  
File "/root/.local/lib/python3.8/site-  
packages/FeatureCloud/app/engine/app.py", line 974, in _deserialize_incoming  
    return pickle.loads(data)  
_pickle.UnpicklingError: could not find MARK
```

```
[CTRL] GET /status  
127.0.0.1 - - [30/Nov/2023 18:07:15] "GET /api/status HTTP/1.0" 200 151
```

Client two (Participant) log:

```
[Time: 30.11.23 18:07:05] [Level: info] id: 5c0d492b0866d4a7  
[Time: 30.11.23 18:07:05] [Level: info] coordinator: False  
[Time: 30.11.23 18:07:05] [Level: info] clients: ['b47aaefed78d6811',  
'5c0d492b0866d4a7']  
[Time: 30.11.23 18:07:05] [Level: info] state: initial  
[Time: 30.11.23 18:07:05] [Level: info] [State: initial] App is started...  
[Time: 30.11.23 18:07:05] [Level: info] [State: initial] traffic_test: False ,  
Data_Size: 100  
127.0.0.1 - - [30/Nov/2023 18:07:05] "POST /api/setup HTTP/1.0" 200 0  
[CTRL] GET /status  
127.0.0.1 - - [30/Nov/2023 18:07:05] "GET /api/status HTTP/1.0" 200 141  
127.0.0.1 - - [30/Nov/2023 18:07:05] "POST /api/data?client=b47aaefed78d6811  
HTTP/1.0" 200 0  
[CTRL] POST /data  
[Time: 30.11.23 18:07:06] [Level: info] transition: loop_state  
[Time: 30.11.23 18:07:07] [Level: info] state: loop_state  
[Time: 30.11.23 18:07:07] [Level: info] transition: wait_data_state  
[Time: 30.11.23 18:07:08] [Level: info] state: wait_data_state  
[Time: 30.11.23 18:07:08] [Level: info] [State: wait_data_state] Round: 1 -  
Scenario: 5 - DataType: 1 - SMPC: False - DP: False  
[Time: 30.11.23 18:07:08] [Level: info] transition: client_state  
[CTRL] GET /status  
127.0.0.1 - - [30/Nov/2023 18:07:08] "GET /api/status HTTP/1.0" 200 146  
[Time: 30.11.23 18:07:09] [Level: info] state: client_state  
[Time: 30.11.23 18:07:09] [Level: info] [State: client_state] Sub_Scen: 5,  
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:  
False, DP: False  
[Time: 30.11.23 18:07:09] [Level: info] [State: client_state] Sub_Scen: 5,  
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:  
True, DP: False  
[Time: 30.11.23 18:07:09] [Level: info] [State: client_state] Sub_Scen: 5,  
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
```



```
False, DP: False
[Time: 30.11.23 18:07:09] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:07:09] [Level: info] transition: loop_state
[Time: 30.11.23 18:07:10] [Level: info] state: loop_state
[Time: 30.11.23 18:07:10] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:07:11] [Level: info] state: wait_data_state
```

10.4.4 Binary-Text Data Decoding Error

Client one (coordinator) log:

```
Time: 30.11.23 18:13:55] [Level: info] id: 731caacdfca736bf
[Time: 30.11.23 18:13:55] [Level: info] coordinator: True
[Time: 30.11.23 18:13:55] [Level: info] clients: ['f192abe67545f698',
'731caacdfca736bf']
[Time: 30.11.23 18:13:55] [Level: info] state: initial
[Time: 30.11.23 18:13:55] [Level: info] [State: initial] App is started...
[Time: 30.11.23 18:13:55] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[Time: 30.11.23 18:13:55] [Level: info] [State: initial] Coordinator broadcast
test settings...
[Time: 30.11.23 18:13:55] [Level: info] transition: loop_state
127.0.0.1 - - [30/Nov/2023 18:13:55] "POST /api/setup HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:13:55] "GET /api/status HTTP/1.0" 200 143
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:13:55] "GET /api/data HTTP/1.0" 200 40
[Time: 30.11.23 18:13:56] [Level: info] state: loop_state
[Time: 30.11.23 18:13:56] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:13:57] [Level: info] state: wait_data_state
[Time: 30.11.23 18:13:57] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:13:57] [Level: info] transition: client_state
[Time: 30.11.23 18:13:58] [Level: info] state: client_state
[Time: 30.11.23 18:13:58] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:13:58] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:13:58] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:13:58] [Level: info] [State: client_state] Sub_Scen: 5,
```



```
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:13:58] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:13:58] "GET /api/status HTTP/1.0" 200 281
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:13:58] "GET /api/data HTTP/1.0" 200 2
[CTRL] /data
127.0.0.1 - - [30/Nov/2023 18:13:58] "POST /api/data?client=731caacdfca736bf
HTTP/1.0" 200 0
[Time: 30.11.23 18:13:59] [Level: info] state: coordinator_state
[Time: 30.11.23 18:13:59] [Level: info] [State: coordinator_state] Coordinator
is launched...
[Time: 30.11.23 18:13:59] [Level: info] [State: coordinator_state] Sub_Scen:
5, Round: 1, Communication: 0 gather data: [1, 21] SMPC: False, DP: False
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:14:01] "GET /api/status HTTP/1.0" 200 267
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:14:01] "GET /api/data HTTP/1.0" 200 2
```

Client two (participant) log:

```
2023-11-30 18:13:55,430 INFO success: app entered RUNNING state, process has
stayed up for > than 1 seconds (startsecs)
2023-11-30 18:13:55,471 INFO success: nginx entered RUNNING state, process has
stayed up for > than 1 seconds (startsecs)
[CTRL] POST /setup
[Time: 30.11.23 18:13:57] [Level: info] id: f192abe67545f698
[Time: 30.11.23 18:13:57] [Level: info] coordinator: False
[Time: 30.11.23 18:13:57] [Level: info] clients: ['f192abe67545f698',
'731caacdfca736bf']
[Time: 30.11.23 18:13:57] [Level: info] state: initial
[Time: 30.11.23 18:13:57] [Level: info] [State: initial] App is started...
127.0.0.1 - - [30/Nov/2023 18:13:57] "POST /api/setup HTTP/1.0" 200 0
[Time: 30.11.23 18:13:57] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:13:57] "GET /api/status HTTP/1.0" 200 141
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:13:57] "POST /api/data?client=731caacdfca736bf
HTTP/1.0" 200 0
[Time: 30.11.23 18:13:57] [Level: info] transition: loop_state
[Time: 30.11.23 18:13:58] [Level: info] state: loop_state
[Time: 30.11.23 18:13:58] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:13:59] [Level: info] state: wait_data_state
```

```
[Time: 30.11.23 18:13:59] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:13:59] [Level: info] transition: client_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:14:00] "GET /api/status HTTP/1.0" 200 146
[Time: 30.11.23 18:14:00] [Level: info] state: client_state
[Time: 30.11.23 18:14:00] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:14:00] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:14:00] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: False
[Time: 30.11.23 18:14:00] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:14:00] [Level: info] transition: loop_state
[CTRL] POST /data
```

The controller's log:

```
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start
application', 'time': '2023-11-30T18:13:48Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:13:48Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Append app
to list', 'time': '2023-11-30T18:13:50Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_input_1_1701368030', 'time': '2023-11-30T18:13:50Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_output_1_1701368030', 'time': '2023-11-30T18:13:50Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move data
to workspace skipped, no source provided', 'time': '2023-11-30T18:13:50Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move
generic data to input volume from /Users/me/Featurecloud/communication-
test/data/gdir', 'time': '2023-11-30T18:13:50Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
workspace', 'time': '2023-11-30T18:13:50Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:13:50Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:13:51Z'}
```

```
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Intermediary container used for moving data from host to volume has been removed', 'time': '2023-11-30T18:13:52Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start application', 'time': '2023-11-30T18:13:52Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to internal network', 'time': '2023-11-30T18:13:52Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Append app to list', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link fc_communicationtest_157122670 with global socket server', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link fc_communicationtest_777203380 with global socket server', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start local socket server', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start local socket server', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link to global address', 'time': '2023-11-30T18:13:54Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link to global address', 'time': '2023-11-30T18:13:54Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Registered new coordinator controller', 'time': '2023-11-30T18:13:54Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Registered new participant controller', 'time': '2023-11-30T18:13:54Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received setup trigger', 'time': '2023-11-30T18:13:54Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received setup trigger', 'time': '2023-11-30T18:13:54Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': 'Got error during setup to http://172.24.0.4:9000: Post "http://172.24.0.4:9000/setup": dial tcp 172.24.0.4:9000: connect: connection refused', 'time': '2023-11-30T18:13:54Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Setup 53ed44c56459ee1a30295fd627fd44c800e67e2b068b1f0dc5145f019e213792', 'time': '2023-11-30T18:13:55Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Setup triggered', 'time': '2023-11-30T18:13:55Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': '2 apps and server created and running', 'time': '2023-11-30T18:13:55Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'http://172.24.0.3:9000/status: New data available', 'time': '2023-11-30T18:13:55Z'}
```

```
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: plain',  
'time': '2023-11-30T18:13:55Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':  
'http://172.24.0.3:9000/data: Fetched 40 bytes', 'time': '2023-11-  
30T18:13:55Z'}  
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Coordinator  
(ID: 731caacdfca736bf) broadcasting 40 bytes', 'time': '2023-11-30T18:13:55Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 40  
bytes from Client 731caacdfca736bf', 'time': '2023-11-30T18:13:57Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to  
http://172.24.0.4:9000/data?client=731caacdfca736bf [Try 1/3]', 'time': '2023-  
11-30T18:13:57Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':  
'http://172.24.0.3:9000/status: New data available', 'time': '2023-11-  
30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: p2p',  
'time': '2023-11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Applying DP',  
'time': '2023-11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'DP was applied  
sucessfully', 'time': '2023-11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':  
'http://172.24.0.3:9000/data: Fetched 74 bytes', 'time': '2023-11-  
30T18:13:58Z'}  
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]  
Participant (ID: 731caacdfca736bf) sending 74 bytes to Participant (index:  
1)', 'time': '2023-11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 74  
bytes from Client 731caacdfca736bf', 'time': '2023-11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to  
http://172.24.0.3:9000/data?client=731caacdfca736bf [Try 1/3]', 'time': '2023-  
11-30T18:13:58Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':  
'http://172.24.0.3:9000/status: New data available', 'time': '2023-11-  
30T18:14:01Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: plain',  
'time': '2023-11-30T18:14:01Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Applying DP',  
'time': '2023-11-30T18:14:01Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'DP was applied  
sucessfully', 'time': '2023-11-30T18:14:01Z'}  
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':  
'http://172.24.0.3:9000/data: Fetched 18 bytes', 'time': '2023-11-  
30T18:14:01Z'}  
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Coordinator
```

```
(ID: 731caacdfca736bf) broadcasting 18 bytes', 'time': '2023-11-30T18:14:01Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 18
bytes from Client 731caacdfca736bf', 'time': '2023-11-30T18:14:01Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to
http://172.24.0.4:9000/data?client=731caacdfca736bf [Try 1/3]', 'time': '2023-
11-30T18:14:01Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.24.0.4:9000/status: New data available', 'time': '2023-11-
30T18:14:03Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: plain',
'time': '2023-11-30T18:14:03Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Applying DP',
'time': '2023-11-30T18:14:03Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': "Error while
adding noise: Error while reading sent data: invalid character '\\u0080'
looking for beginning of value", 'time': '2023-11-30T18:14:03Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': "Error while
reading sent data: invalid character '\\u0080' looking for beginning of
value", 'time': '2023-11-30T18:14:03Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Shutdown',
'time': '2023-11-30T18:14:04Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'error', 'msg': 'Participant
(ID: 731caacdfca736bf): relaying stopped due to: EOF', 'time': '2023-11-
30T18:14:04Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Shutdown',
'time': '2023-11-30T18:14:04Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'error', 'msg': 'Participant
(ID: f192abe67545f698): relaying stopped due to: EOF', 'time': '2023-11-
30T18:14:04Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
test id: 1', 'time': '2023-11-30T18:14:04Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
container fc_communicationtest_157122670', 'time': '2023-11-30T18:14:04Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
host', 'time': '2023-11-30T18:14:04Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:14:04Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:14:05Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'debug', 'msg': 'Intermediary
container used for moving data from volume to host has been removed', 'time':
'2023-11-30T18:14:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Saved
output volume content to
/data/tests/results_test_1_client_0_fc_communicationtest_157122670.zip',
```

```
'time': '2023-11-30T18:14:07Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_output_0_1701368024', 'time': '2023-11-30T18:14:07Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_output_0_1701368024', 'time': '2023-11-30T18:14:07Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_input_0_1701368024', 'time': '2023-11-30T18:14:07Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_input_0_1701368024', 'time': '2023-11-30T18:14:07Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
container fc_communicationtest_777203380', 'time': '2023-11-30T18:14:07Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
host', 'time': '2023-11-30T18:14:07Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:14:07Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:14:08Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'debug', 'msg': 'Intermediary
container used for moving data from volume to host has been removed', 'time':
'2023-11-30T18:14:09Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Saved
output volume content to
/data/tests/results_test_1_client_1_fc_communicationtest_777203380.zip',
'time': '2023-11-30T18:14:09Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_output_1_1701368030', 'time': '2023-11-30T18:14:09Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_output_1_1701368030', 'time': '2023-11-30T18:14:09Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_input_1_1701368030', 'time': '2023-11-30T18:14:09Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_input_1_1701368030', 'time': '2023-11-30T18:14:09Z'}
```

10.4.5 Pickle Unpickling Stack Underflow Error

Client one (coordinator) log:

```
[Time: 30.11.23 18:30:07] [Level: info] id: e0e38bcda48fbb08
[Time: 30.11.23 18:30:07] [Level: info] coordinator: True
[Time: 30.11.23 18:30:07] [Level: info] clients: ['46cd4c65ff9b419a',
'e0e38bcda48fbb08']
[Time: 30.11.23 18:30:07] [Level: info] state: initial
[Time: 30.11.23 18:30:07] [Level: info] [State: initial] App is started...
[Time: 30.11.23 18:30:07] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[Time: 30.11.23 18:30:07] [Level: info] [State: initial] Coordinator broadcast
```



```
test settings...
[Time: 30.11.23 18:30:07] [Level: info] transition: loop_state
127.0.0.1 - - [30/Nov/2023 18:30:07] "POST /api/setup HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:07] "GET /api/status HTTP/1.0" 200 143
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:30:07] "GET /api/data HTTP/1.0" 200 40
[Time: 30.11.23 18:30:08] [Level: info] state: loop_state
[Time: 30.11.23 18:30:08] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:30:09] [Level: info] state: wait_data_state
[Time: 30.11.23 18:30:09] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:30:09] [Level: info] transition: client_state
[Time: 30.11.23 18:30:10] [Level: info] state: client_state
[Time: 30.11.23 18:30:10] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:30:10] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:30:10] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:30:10] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:30:10] [Level: info] transition: coordinator_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:10] "GET /api/status HTTP/1.0" 200 219
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:30:10] "GET /api/data HTTP/1.0" 200 1
[Time: 30.11.23 18:30:11] [Level: info] state: coordinator_state
[Time: 30.11.23 18:30:11] [Level: info] [State: coordinator_state] Coordinator
is launched...
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:13] "GET /api/status HTTP/1.0" 200 219
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:30:13] "GET /api/data HTTP/1.0" 200 2
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:30:15] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:16] "GET /api/status HTTP/1.0" 200 267
[CTRL] GET /data
127.0.0.1 - - [30/Nov/2023 18:30:16] "GET /api/data HTTP/1.0" 200 2
```



```
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:30:18] "POST /api/data?client=0000000000000000
HTTP/1.0" 200 0
[Time: 30.11.23 18:30:18] [Level: info] Traceback (most recent call last):
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 226, in guarded_run
    self.run()
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 241, in run
    transition = self.current_state.run()
  File "/app/states.py", line 465, in run
    data_to_collect = self.gather_data(is_json=flg)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 618, in gather_data
    return self.await_data(len(self._app.clients), unwrap=False,
is_json=is_json)
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 648, in await_data
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 648, in <listcomp>
    return [_deserialize_incoming(d[0], is_json=is_json) for d in data]
  File "/root/.local/lib/python3.8/site-
packages/FeatureCloud/app/engine/app.py", line 974, in _deserialize_incoming
    return pickle.loads(data)
_pickle.UnpicklingError: unpickling stack underflow
```

Client two (participant) log:

```
[Time: 30.11.23 18:30:09] [Level: info] id: 46cd4c65ff9b419a
[Time: 30.11.23 18:30:09] [Level: info] coordinator: False
[Time: 30.11.23 18:30:09] [Level: info] clients: ['46cd4c65ff9b419a',
'e0e38bcda48fbb08']
[Time: 30.11.23 18:30:09] [Level: info] state: initial
[Time: 30.11.23 18:30:09] [Level: info] [State: initial] App is started...
127.0.0.1 - - [30/Nov/2023 18:30:09] "POST /api/setup HTTP/1.0" 200 0
[Time: 30.11.23 18:30:09] [Level: info] [State: initial] traffic_test: False ,
Data_Size: 100
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:09] "GET /api/status HTTP/1.0" 200 141
[CTRL] POST /data
127.0.0.1 - - [30/Nov/2023 18:30:09] "POST /api/data?client=e0e38bcda48fbb08
HTTP/1.0" 200 0
[Time: 30.11.23 18:30:09] [Level: info] transition: loop_state
```

```
[Time: 30.11.23 18:30:10] [Level: info] state: loop_state
[Time: 30.11.23 18:30:10] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:30:11] [Level: info] state: wait_data_state
[Time: 30.11.23 18:30:11] [Level: info] [State: wait_data_state] Round: 1 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:30:11] [Level: info] transition: client_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:12] "GET /api/status HTTP/1.0" 200 146
[Time: 30.11.23 18:30:12] [Level: info] state: client_state
[Time: 30.11.23 18:30:12] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 0, client send data 1 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:30:12] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 1, client send data 11 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:30:12] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 2, client send data 21 to coordinator .... SMPC:
False, DP: True
[Time: 30.11.23 18:30:12] [Level: info] [State: client_state] Sub_Scen: 5,
Round: 1, Communication: 3, client send data 31 to coordinator .... SMPC:
True, DP: False
[Time: 30.11.23 18:30:12] [Level: info] transition: loop_state
[Time: 30.11.23 18:30:13] [Level: info] state: loop_state
[Time: 30.11.23 18:30:13] [Level: info] transition: wait_data_state
[Time: 30.11.23 18:30:14] [Level: info] state: wait_data_state
[Time: 30.11.23 18:30:14] [Level: info] [State: wait_data_state] Round: 2 -
Scenario: 5 - DataType: 1 - SMPC: False - DP: False
[Time: 30.11.23 18:30:14] [Level: info] transition: client_state
[CTRL] GET /status
127.0.0.1 - - [30/Nov/2023 18:30:15] "GET /api/status HTTP/1.0" 200 214
```

The controller log:

```
{'component': 'MAIN', 'instance': '', 'level': 'info', 'msg': 'Controller
start', 'time': '2023-11-30T18:19:14Z'}
{'component': 'MAIN', 'instance': '', 'level': 'info', 'msg': 'Mode:
Dockerized (isolated)', 'time': '2023-11-30T18:19:14Z'}
{'component': 'MAIN', 'instance': '', 'level': 'info', 'msg': "Configuration
value 'HasGPU': false", 'time': '2023-11-30T18:19:14Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Removing
leftover Docker entities for label: fc-controller-label', 'time': '2023-11-
30T18:19:15Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Removed 0 of
0 leftover containers', 'time': '2023-11-30T18:19:15Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Removed 0 of
```

```
1 leftover volumes', 'time': '2023-11-30T18:19:15Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Removed 1 of
1 leftover networks', 'time': '2023-11-30T18:19:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Listening on
port 9151', 'time': '2023-11-30T18:19:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Listening on
port 9150', 'time': '2023-11-30T18:19:15Z'}
{'component': 'TESTBED', 'instance': '', 'level': 'info', 'msg': 'Create new
test', 'time': '2023-11-30T18:29:58Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Create
volumes and containers...', 'time': '2023-11-30T18:29:58Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_input_0_1701368998', 'time': '2023-11-30T18:29:58Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_output_0_1701368998', 'time': '2023-11-30T18:29:58Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move data
to workspace skipped, no source provided', 'time': '2023-11-30T18:29:58Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move
generic data to input volume from /Users/me/Featurecloud/communication-
test/data/gdir', 'time': '2023-11-30T18:29:58Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
workspace', 'time': '2023-11-30T18:29:58Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:29:58Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:30:00Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Intermediary
container used for moving data from host to volume has been removed', 'time':
'2023-11-30T18:30:01Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start
application', 'time': '2023-11-30T18:30:01Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:30:01Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Append app
to list', 'time': '2023-11-30T18:30:02Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_input_1_1701369002', 'time': '2023-11-30T18:30:02Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Create volume
test_1_output_1_1701369002', 'time': '2023-11-30T18:30:02Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move data
to workspace skipped, no source provided', 'time': '2023-11-30T18:30:02Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Move
generic data to input volume from /Users/me/Featurecloud/communication-
test/data/gdir', 'time': '2023-11-30T18:30:02Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
```

```
workspace', 'time': '2023-11-30T18:30:02Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:30:02Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:30:03Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Intermediary
container used for moving data from host to volume has been removed', 'time':
'2023-11-30T18:30:04Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start
application', 'time': '2023-11-30T18:30:04Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:30:04Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Append app
to list', 'time': '2023-11-30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link
fc_communicationtest_83010092 with global socket server', 'time': '2023-11-
30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link
fc_communicationtest_825977803 with global socket server', 'time': '2023-11-
30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start local
socket server', 'time': '2023-11-30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Start local
socket server', 'time': '2023-11-30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link to
global address', 'time': '2023-11-30T18:30:06Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Link to
global address', 'time': '2023-11-30T18:30:06Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Registered
new coordinator controller', 'time': '2023-11-30T18:30:06Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Registered
new participant controller', 'time': '2023-11-30T18:30:06Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received setup
trigger', 'time': '2023-11-30T18:30:06Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received setup
trigger', 'time': '2023-11-30T18:30:06Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'error', 'msg': 'Got error
during setup to http://172.25.0.4:9000: Post "http://172.25.0.4:9000/setup":
dial tcp 172.25.0.4:9000: connect: connection refused', 'time': '2023-11-
30T18:30:06Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Setup
62d3d31c8111d096aab651193aa24541680bf9846808331e3a2dfbbe5821f4c7', 'time':
'2023-11-30T18:30:07Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Setup
triggered', 'time': '2023-11-30T18:30:07Z'}
```

```
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': '2 apps and
server created and running', 'time': '2023-11-30T18:30:07Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: New data available', 'time': '2023-11-
30T18:30:07Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: plain',
'time': '2023-11-30T18:30:07Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':
'http://172.25.0.3:9000/data: Fetched 40 bytes', 'time': '2023-11-
30T18:30:07Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Coordinator
(ID: e0e38bcda48fbb08) broadcasting 40 bytes', 'time': '2023-11-30T18:30:07Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 40
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:09Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to
http://172.25.0.4:9000/data?client=e0e38bcda48fbb08 [Try 1/3]', 'time': '2023-
11-30T18:30:09Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: New data available', 'time': '2023-11-
30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: smpc',
'time': '2023-11-30T18:30:10Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 82 bytes to Participant (index:
0)', 'time': '2023-11-30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':
'http://172.25.0.3:9000/data: Fetched 1 bytes', 'time': '2023-11-
30T18:30:10Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 83 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:2299814188351802168" from Client e0e38bcda48fbb08', 'time': '2023-11-
30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:-2299814188251802168" from Client e0e38bcda48fbb08', 'time': '2023-11-
30T18:30:10Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: New data available', 'time': '2023-11-
30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: smpc',
```

```
'time': '2023-11-30T18:30:13Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL] Participant (ID: e0e38bcda48fbb08) sending 82 bytes to Participant (index: 0)', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82 bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard "i64:-903095290487264450" from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg': 'http://172.25.0.3:9000/data: Fetched 2 bytes', 'time': '2023-11-30T18:30:13Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL] Participant (ID: e0e38bcda48fbb08) sending 81 bytes to Participant (index: 1)', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 81 bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard "i64:903095291587264450" from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:13Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'http://172.25.0.4:9000/status: New data available', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: smpc', 'time': '2023-11-30T18:30:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL] Participant (ID: 46cd4c65ff9b419a) sending 82 bytes to Participant (index: 0)', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82 bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard "i64:8693413466922187797" from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL] Participant (ID: 46cd4c65ff9b419a) sending 83 bytes to Participant (index: 1)', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg': 'http://172.25.0.4:9000/data: Fetched 1 bytes', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83 bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard "i64:-8693413466822187797" from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
```



```
Participant (ID: 46cd4c65ff9b419a) sending 83 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83
bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:15Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 82 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received
aggregated shard "i64:-7453516418435561651" from Client 46cd4c65ff9b419a',
'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received
aggregated shard "i64:7453516418635561651" from Client e0e38bcda48fbb08',
'time': '2023-11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to
http://172.25.0.3:9000/data?client=0000000000000000 [Try 1/3]', 'time': '2023-
11-30T18:30:15Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: New data available', 'time': '2023-11-
30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: plain',
'time': '2023-11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Applying DP',
'time': '2023-11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'DP was applied
sucessfully', 'time': '2023-11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':
'http://172.25.0.3:9000/data: Fetched 17 bytes', 'time': '2023-11-
30T18:30:16Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': 'Coordinator
(ID: e0e38bcda48fbb08) broadcasting 17 bytes', 'time': '2023-11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 17
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to
http://172.25.0.4:9000/data?client=e0e38bcda48fbb08 [Try 1/3]', 'time': '2023-
11-30T18:30:16Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.4:9000/status: New data available', 'time': '2023-11-
30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: smpc',
'time': '2023-11-30T18:30:18Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: 46cd4c65ff9b419a) sending 82 bytes to Participant (index:
0)', 'time': '2023-11-30T18:30:18Z'}
```



```
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82
bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:6181063008565853832" from Client 46cd4c65ff9b419a', 'time': '2023-11-
30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':
'http://172.25.0.4:9000/data: Fetched 2 bytes', 'time': '2023-11-
30T18:30:18Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: 46cd4c65ff9b419a) sending 83 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83
bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:-6181063007465853832" from Client 46cd4c65ff9b419a', 'time': '2023-11-
30T18:30:18Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: 46cd4c65ff9b419a) sending 82 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82
bytes from Client 46cd4c65ff9b419a', 'time': '2023-11-30T18:30:18Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 83 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received
aggregated shard "i64:5277967718078589382" from Client 46cd4c65ff9b419a',
'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received
aggregated shard "i64:-5277967715878589382" from Client e0e38bcda48fbb08',
'time': '2023-11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'POST to
http://172.25.0.3:9000/data?client=0000000000000000 [Try 1/3]', 'time': '2023-
11-30T18:30:18Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: New data available', 'time': '2023-11-
30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Mode: smpc',
'time': '2023-11-30T18:30:19Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 82 bytes to Participant (index:
0)', 'time': '2023-11-30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 82
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:19Z'}
```

```
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:8222125889749937863" from Client e0e38bcda48fbb08', 'time': '2023-11-
30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'debug', 'msg':
'http://172.25.0.3:9000/data: Fetched 2 bytes', 'time': '2023-11-
30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg':
'http://172.25.0.3:9000/status: Finished', 'time': '2023-11-30T18:30:19Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) sending 83 bytes to Participant (index:
1)', 'time': '2023-11-30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'received 83
bytes from Client e0e38bcda48fbb08', 'time': '2023-11-30T18:30:19Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'info', 'msg': '[GLOBAL]
Participant (ID: e0e38bcda48fbb08) finished the current step', 'time': '2023-
11-30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Received shard
"i64:-8222125886649937863" from Client e0e38bcda48fbb08', 'time': '2023-11-
30T18:30:19Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Shutdown',
'time': '2023-11-30T18:30:20Z'}
{'component': 'LOCAL', 'instance': '', 'level': 'info', 'msg': 'Shutdown',
'time': '2023-11-30T18:30:20Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'error', 'msg': 'Participant
(ID: e0e38bcda48fbb08): relaying stopped due to: EOF', 'time': '2023-11-
30T18:30:20Z'}
{'component': 'GLOBAL', 'instance': '', 'level': 'error', 'msg': 'Participant
(ID: 46cd4c65ff9b419a): relaying stopped due to: EOF', 'time': '2023-11-
30T18:30:20Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
test id: 1', 'time': '2023-11-30T18:30:20Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
container fc_communicationtest_83010092', 'time': '2023-11-30T18:30:20Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
host', 'time': '2023-11-30T18:30:20Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:30:21Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:30:22Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'debug', 'msg': 'Intermediary
container used for moving data from volume to host has been removed', 'time':
'2023-11-30T18:30:23Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Saved
output volume content to
/data/tests/results_test_1_client_0_fc_communicationtest_83010092.zip',
```

```
'time': '2023-11-30T18:30:23Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_output_0_1701368998', 'time': '2023-11-30T18:30:23Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_output_0_1701368998', 'time': '2023-11-30T18:30:23Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_input_0_1701368998', 'time': '2023-11-30T18:30:23Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_input_0_1701368998', 'time': '2023-11-30T18:30:23Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Stopping
container fc_communicationtest_825977803', 'time': '2023-11-30T18:30:23Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Move data to
host', 'time': '2023-11-30T18:30:24Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Attach to
internal network', 'time': '2023-11-30T18:30:24Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove
application', 'time': '2023-11-30T18:30:25Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'debug', 'msg': 'Intermediary
container used for moving data from volume to host has been removed', 'time':
'2023-11-30T18:30:26Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Saved
output volume content to
/data/tests/results_test_1_client_1_fc_communicationtest_825977803.zip',
'time': '2023-11-30T18:30:26Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_output_1_1701369002', 'time': '2023-11-30T18:30:26Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_output_1_1701369002', 'time': '2023-11-30T18:30:26Z'}
{'component': 'DOCKER', 'instance': '', 'level': 'info', 'msg': 'Remove volume
test_1_input_1_1701369002', 'time': '2023-11-30T18:30:26Z'}
{'component': 'TESTBED', 'instance': '1', 'level': 'info', 'msg': 'Deleted
volume test_1_input_1_1701369002', 'time': '2023-11-30T18:30:26Z'}
```

10.5 Controller's Dockerfile

```
ENV FC_DOCKERIZED=1

WORKDIR /go/src/fc_controller

COPY go.* /go/src/fc_controller/
RUN go mod download

COPY cmd/controller cmd/controller
COPY pkg pkg

RUN go install fc_controller/cmd/controller

COPY config.docker.yml config.yml

RUN rm -r pkg && rm -r cmd

EXPOSE 8000

ENTRYPOINT ["/go/bin/controller"]
```