NebulOuS

A META OPERATING SYSTEM FOR BROKERING HYPER-DISTRIBUTED APPLICATIONS ON CLOUD COMPUTING CONTINUUMS

OCUC: PAUSA

Application description and initial software architecture

Funded by
the European Union

VERSION HISTORY

Version	Date	Owner	Author(s)	Changes to previous version
0.1	02/07/2024	FORCERA	João Pereira, Tiago Nunes	Initial Draft
0.2	03/07/2024	FORCERA	João Pereira, Tiago Nunes	Initial Work on Section 2
0.3	05/07/2024	FORCERA	João Pereira	Initial Work on Section 3
0.4	09/07/2024	FORCERA	Tiago Nunes	Initial Work on Section 1
1.0	11/07/2024	FORCERA	Tiago Nunes, João Pereira	Final revision
1.1	18/07/2024	FORCERA	João Pereira, Tiago Nunes	Further revision considering suggestions by NebulOuS consortium.
1.2	25/07/2024	FORCERA	João Pereira	Additional minor revision considering comments by NebulOuS consortium.
1.3	02/08/2024	FORCERA	João Pereira	Final Revision
1.4	07/08/2024	FORCERA	João Pereira	New Update after Revision from NebulOuS Consortium.
1.5	12/08/2024	FORCERA	João Pereira	New Revision after meeting with mentoring team.
1.6	12/08/2024	FORCERA	João Pereira	Change IaaS infrastructure from Azure to OVHCloud

TABLE OF CONTENTS

1 INTRODUCTION

The PAUSA project, built on the NebulOuS Meta Operating System, is a pioneering effort to improve urban planning and public health management through advanced environmental monitoring. This initiative will implement serverless features to meet the pressing need for sustainable urban development, leveraging state-of-the-art technology to manage and analyse environmental data efficiently.

PAUSA aims to create a robust platform for city pollution monitoring, predictive air quality modelling, and generating actionable insights for environmental-led urban management. This initiative is essential today, where urban areas face significant environmental challenges that affect public health, regulatory compliance, and quality of life.

By utilising the capabilities of the NebulOuS platform, PAUSA aims to assist in environmental management, making communities smarter, healthier, and more responsive to modern environmental challenges. The project will be implemented in three phases:

- **Phase 1: Infrastructure Setup**
    - This phase will focus on establishing the necessary infrastructure, leveraging NebulOuS serverless capabilities across multiple cloud providers. NebulOuS will ensure the platform is scalable, secure, and compliant with best practices and standards, creating a solid foundation for subsequent developments.
- **Phase 2: Core Functionality Development**
    - During this phase, the core functionalities of the platform will be developed. This will involve deploying serverless functions to handle vast environmental datasets and/or open data streams efficiently and integrating machine learning (ML) models predicting air quality variations.
- **Phase 3: Demonstration and Feedback**
    - The final phase will revolve around demonstrating the platform's capabilities in representative settings, collecting feedback from key stakeholders, and refining the solutions based on this feedback. This stage is crucial for validating the effectiveness of PAUSA and ensuring it meets the practical needs of its intended users.

2 SOFTWARE ARCHITECTURE

The PAUSA project aims to support urban planning and public health management by addressing environmental aspects within cities, regions, and communities. Targeting local governments, the platform seeks to help improve living conditions for city residents through actionable insights. Leveraging the NebulOuS Meta Operating System, PAUSA plans to employ a serverless software architecture for scalable and efficient data processing. It intends to integrate diverse data sources often available under public licenses, such as air quality, weather, traffic flow and noise levels. The platform aims to provide detailed insights into pollution levels and forecast future air quality scenarios. PAUSA sets forth three Primary Objectives (POs):

- **PO1 - Environmental monitoring:** Continuously track air quality and pollution levels to provide up-to-date information.

- **PO2 - Predictive analytics:** Develop models to forecast future air quality and pollution scenarios, enabling proactive management strategies, based on collected datasets under public licenses.
- **PO3 - Data-driven urban planning insights:** Offer detailed data and analysis to support strategic urban expansion, traffic flow optimisation, infrastructure development, among others.

Facilitated by the NebulOuS Meta Operating System, the PAUSA project will use the cloud continuum framework to integrate distributed resources based on major public cloud infrastructures (see Figure 1). This integration aims to establish a serverless back-end within PAUSA, ensuring scalability and cost-effectiveness. Serverless computing will allow for the dynamic allocation and deallocation of resources, which is ideal for managing the fluctuating loads of environmental data processing and/or the fluctuation in user access demand. One of the entities of the PAUSA consortium (FORCERA) has developed technology concerning serverless data processing and orchestration with Technology Readiness Level (TRL) 6 based on Amazon Web Services (AWS) proprietary technology – Lambda and Step Functions. Nevertheless, due to the constraints of NebulOuS open call and associated technology, the team will need to build the PAUSA solution and associated components from scratch. The following sections, images, milestones, and KPIs, show how FORCERA aims to achieve this within the project's time frame.

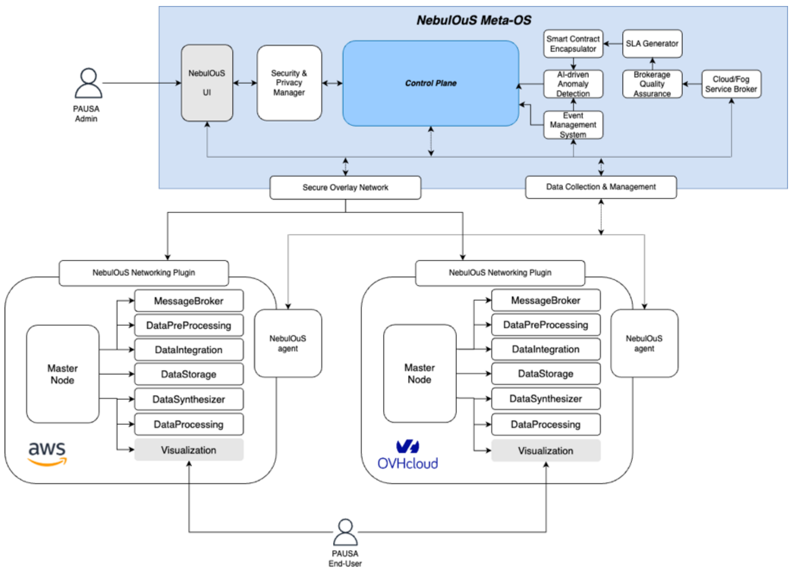

Figure 1 - High-level PAUSA Diagram showing integration with NebulOuS Meta OS.

In detail, the PAUSA project will focus on a robust software architecture that abides by the requirements of the various application components and their functionalities. Special attention will be given to ensuring scalability, with plans to expand resources for specific components and increase instances as necessary. These decisions will be guided by defined metrics sourced from operational data and performance indicators throughout the duration of the project.

Figure 2 provides further details on the technologies and frameworks that will be used in the PAUSA sub-project, using the same terminology as displayed in Figure 1. Each separate cloud environment will leverage its Infrastructure-as-a-Service (IaaS) resources (Elastic Cloud Compute - EC2 instances on AWS, and Virtual Machines in OVHCloud managed via OpenStack) to serve PAUSA to the end-users.

The PAUSA team aims to utilize Knative serverless workflows as much as possible. The NebulOuS agents (coordinated by the NebulOuS master node) will be configured on both environments and will manage data handling, integration, orchestration, among other functions (as listed in Figure 2), leveraging the Knative framework for these processing tasks. The PAUSA serverless workflow begins by collecting data from IoT sources and passing it to a message broker, RabbitMQ, for pre-processing and integration into a PostgreSQL database. Additionally, the NebulOuS agent, leveraging Knative functionalities, will interact with the developed ML model to predict city air quality. Finally, Metabase, the visualization technology, will be containerized and utilized as the endpoint of the PAUSA solution for retrieving environmental analytics. Figure 2 also represents, on the right side, the additional frameworks that will facilitate functional and nonfunctional requirements.

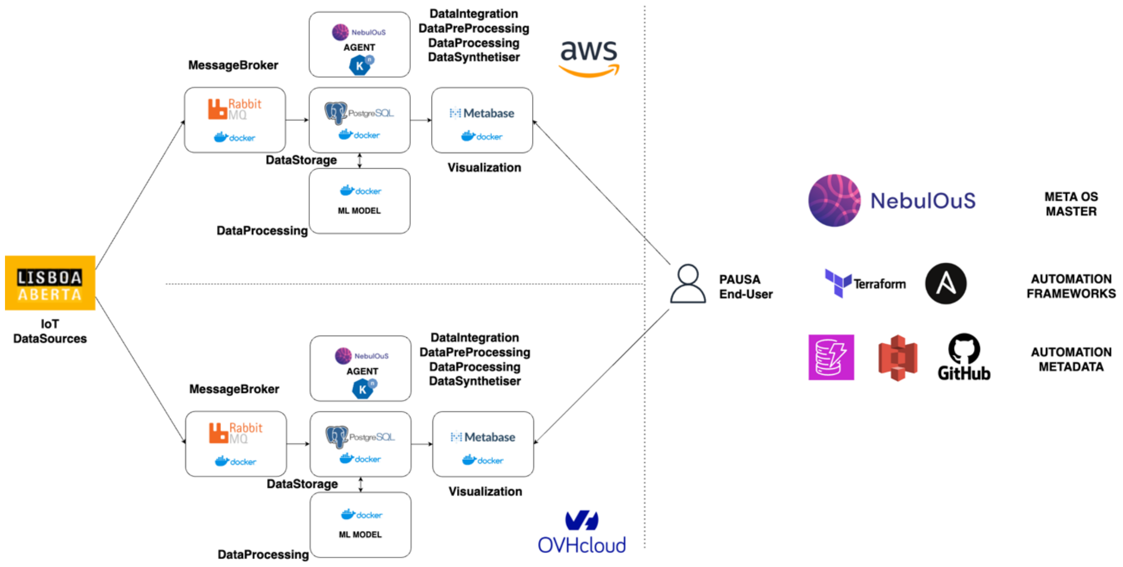

Figure 2 - PAUSA Software Architecture

To complement the software description, the UML sequence diagram depicted in Figure 3 describes a system where IoT devices, labelled as *DataSources*, collect various types of raw data such as air quality, traffic, weather, and noise. Such IoT data sources can be either physical sensorial devices integrated with the PAUSA ecosystem or indirect Edge/Cloud data sources that interact with physical sensorial devices and expose the collected data through an API. A clear example of the latter is weather information that can be collected from multiple geographic locations and from existing API services (e.g., tomorrow.io, OpenWeather).

This raw data is sent to a *MessageBroker*, which acts as a middleman to manage the data flow. The *MessageBroker* forwards the raw data to a serverless component called *DataPreProcessing*, where the data is pre-processed. The pre-processing stage handles data uniformity, ensuring that data from different sources and typologies can be uniformly ingested, treated, and stored by the PAUSA ecosystem. Then, both aggregated and raw data are stored in a *DataStorage* database, which will be of a time series typology, given the nature of the PAUSA ambition. The *MessageBroker* also passes the aggregated data to a *DataIntegration* container, which integrates data from all devices of the same type and stores the integrated data in *DataStorage* as well as sends it back to the *MessageBroker*.

The integrated data is then forwarded to a *DataProcessing* container by the *MessageBroker*. *DataProcessing* processes this integrated data and stores the processed results in *DataStorage*. This processing of information is the more computation-heavy module of the PAUSA framework, as it has the responsibility of not only integrating data from different typologies but, more importantly, generating inference and knowledge based on the ingested information. Such data inference will be strongly tied to the particularities of the different scenarios worked on the set of use cases considered by the PAUSA experiment.

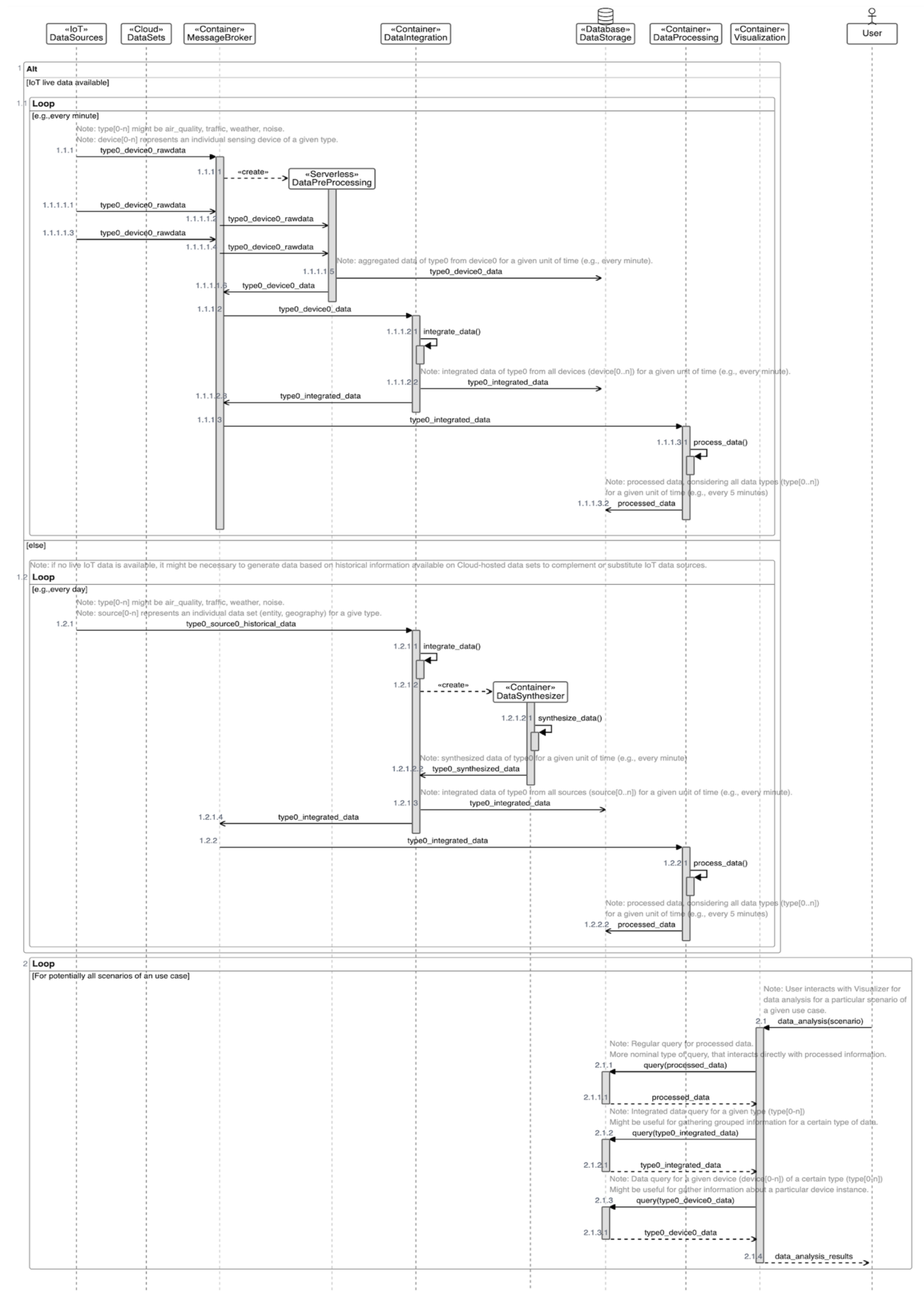
«IoT» DataSources
«Cloud» DataSets
«Container» MessageBroker
«Container» DataIntegration
«Database» DataStorage
«Container» DataProcessing
«Container» Visualization
User
1 Alt
[IoT live data available]
1.1 Loop
[e.g.,every minute]
Note: type[0-n] might be air_quality, traffic, weather, noise.
Note: device[0-n] represents an individual sensing device of a given type.
1.1.1 type0_device0_rawdata
1.1.1.1 «create»
«Serverless» DataPreProcessing
1.1.1.1.1 type0_device0_rawdata
1.1.1.1.2 type0_device0_rawdata
1.1.1.1.3 type0_device0_rawdata
1.1.1.1.4 type0_device0_rawdata
Note: aggregated data of type0 from device0 for a given unit of time (e.g., every minute).
1.1.1.1.5 type0_device0_data
1.1.1.1.6 type0_device0_data
1.1.1.2 type0_device0_data
1.1.1.2.1 integrate_data()
Note: integrated data of type0 from all devices (device[0..n]) for a given unit of time (e.g., every minute).
1.1.1.2.2 type0_integrated_data
1.1.1.2.3 type0_integrated_data
1.1.1.3 type0_integrated_data
1.1.1.3.1 process_data()
Note: processed data, considering all data types (type[0..n]) for a given unit of time (e.g., every 5 minutes)
1.1.1.3.2 processed_data
[else]
Note: if no live IoT data is available, it might be necessary to generate data based on historical information available on Cloud-hosted data sets to complement or substitute IoT data sources.
1.2 Loop
[e.g.,every day]
Note: type[0-n] might be air_quality, traffic, weather, noise.
Note: source[0-n] represents an individual data set (entity, geography) for a give type.
1.2.1 type0_source0_historical_data
1.2.1.1 integrate_data()
1.2.1.2 «create»
«Container» DataSynthesizer
1.2.1.2.1 synthesize_data()
Note: synthesized data of type0 for a given unit of time (e.g., every minute)
1.2.1.2.2 type0_synthesized_data
Note: integrated data of type0 from all sources (source[0..n]) for a given unit of time (e.g., every minute).
1.2.1.3 type0_integrated_data
1.2.1.4 type0_integrated_data
1.2.2 type0_integrated_data
1.2.2.1 process_data()
Note: processed data, considering all data types (type[0..n]) for a given unit of time (e.g., every 5 minutes)
1.2.2.2 processed_data
2 Loop
[For potentially all scenarios of an use case]
Note: User interacts with Visualizer for data analysis for a particular scenario of a given use case.
2.1 data_analysis(scenario)
Note: Regular query for processed data. More nominal type of query, that interacts directly with processed information.
2.1.1 query(processed_data)
2.1.1.1 processed_data
Note: Integrated data query for a given type (type[0-n]) Might be useful for gathering grouped information for a certain type of data.
2.1.2 query(type0_integrated_data)
2.1.2.1 type0_integrated_data
Note: Data query for a given device (device[0-n]) of a certain type (type[0-n]) Might be useful for gather information about a particular device instance.
2.1.3 query(type0_device0_data)
2.1.3.1 type0_device0_data
2.1.4 data_analysis_results

Figure 3 – UML Sequence Diagram of the PAUSA intended usage.

If live IoT data is unavailable, historical data from cloud-hosted *DataSets* is used. This historical data is integrated by *DataIntegration* and stored in *DataStorage*. An example of such cloud-hosted data sets is the Lisboa Aberta[1] portal, which collects and makes available through publicly accessible APIs historical information regarding different urban aspects, such as noise levels, air quality, and traffic information throughout different geographic locations within the city of Lisbon. A *DataSynthesizer* container will be triggered, in such cases, to generate synthesised data from the historical data. The ambition of this data generation procedure is to replace or complement missing information from live sources, both for the purpose of the PAUSA experiment, where it might be useful for validating data sources that cannot be integrated due to the unavailability of data providers or time constraints and also for the post-project nominal usage of the envisioned ecosystem, for complementing information data might be difficult to collect in real-time, particularly for certain geographies. This generated data is then integrated by *DataIntegration* and stored in *DataStorage*, and subsequently processed by *DataProcessing*, with the processed data stored back in *DataStorage*, in an exactly similar way as the previously described live information.

Users interact with the system through a *Visualization* container, analysing data for specific scenarios. The *Visualization* container queries the *DataStorage* for processed data, integrated data of a specific type, and data from specific devices of a certain type. The queried data is returned to the *Visualization* container for analysis, enabling users to gain insights from the processed and integrated data. This setup ensures that the system can handle both live and historical data efficiently, integrating and processing the data to provide valuable insights through visualisation tools for user analysis.

Based on this sequence flow of information and the description of the functionality of the PAUSA components, NebulOuS will play a central role in aiding the scalability and facilitated cross-site deployment of the solution. As NebulOuS operates by dynamically adjusting resources based on workload demands, it can monitor metrics such as processing latency requirements to decide when and how to scale each component. This will mean adding more instances (horizontal scaling) or increasing resources like CPU and RAM (vertical scaling) to handle increased data processing or user requests effectively. In terms of deployment, NebulOuS offers added flexibility that will be used by the PAUSA ecosystem. In detail, it will deploy PAUSA components across a mix of cloud providers, such as AWS or OVHCloud, ensuring that tasks are performed where they are most efficient, with the cloud providing scalability.

For example, the PAUSA *DataIntegration* container will dynamically scale based on data ingestion rates and integration complexity. NebulOuS might deploy it on AWS for flexible scaling nodes closer to IoT devices for faster data aggregation. Similarly, the *DataStorage* database will be managed by NebulOuS to scale storage capacity and adjust read/write throughput as data volumes grow. Similarly, the PAUSA *DataProcessing* container, responsible for heavy computation and data inference, will benefit from NebulOuS's ability to scale compute resources based on workload fluctuations.

Lastly, the PAUSA *Visualization* container, used for data analysis and user interaction, will also be optimised by NebulOuS to ensure responsive performance. It might deploy instances across multiple cloud nodes to minimise latency and maximise usability.

Overall, as NebulOuS's approach involves continuous monitoring of Quality of Service (QoS) metrics and automatic adjustments to maintain performance levels, it ensures that the PAUSA project can handle both live and historical data efficiently, providing valuable insights through scalable and responsive data processing and visualisation capabilities.

[1] Lisboa Aberta Portal. Lisbon Municipality. https://lisboaaberta.cm-lisboa.pt/index.php/pt/. [Accessed 18/07/2024].

Funded by the European Union

2.1 INFRASTRUCTURE SETUP

Initially, the project will concentrate on establishing foundational infrastructure critical for deploying and operating serverless applications. This includes provisioning and configuring cloud resources using Infrastructure as Code (IaC) tools like Terraform[2], aiming to create environments in major public clouds. Ansible[3] will be used for configuration management. It is expected that an IaC approach can be followed to ensure interoperability and seamless integration, configuration, and management of NebulOuS components, notably for the facilitation of NebulOuS Agents (with Knative serverless capabilities) over distinct deployment environments. It is expected that the developed PAUSA serverless architecture, through the employment of NebulOuS, can thus be agnostic to the specificities of public cloud providers. The PAUSA team expects that at least 75% of the infrastructure can be automatically provisioned using an IaC approach. The PAUSA project IaC approach will consider two domains, facilitated by the integration with NebulOuS. These two domains will be presented in the next sub-sections. PAUSA Backend Infrastructure Management

The first domain is related to the provisioning of infrastructure needed for safely establishing the deployments and provisions in Cloud environments to be managed by the NebulOuS tools and services. In detail, the PAUSA project will rely on an approach towards handling backend information for the provisioning of Cloud resources using a GitOps Terraform-based approach. This approach enables efficient, secure, and scalable management of infrastructure resources, allowing the creation and management of NebulOuS-enhanced configurations in a multi-environment capacity. This approach also enables a separation of concerns between the PAUSA backend-infrastructure management, to be presented in the sub-section, and the handling of NebulOuS-enhanced PAUSA components, which will be established within the automatically generated *pausa-infrastructure* repository. In detail, the former deals with securing the necessary cloud-level permissions and the establishment of a systematic manner in which to store Terraform state information. In its turn, the latter deals with the actual creation of resources for the achievement of PAUSA's ambitions, in which the necessary permissions and credentials are automatically set and managed.

The core functional component of this backend infrastructure is the creation of *projects*, that define, for each *project*, the establishment of a collection of Terraform modules for the automated creation of infrastructure-related resources. Within a project, the necessary provisions for creating and managing the infrastructure for holding Terraform's backend and state information are handled.

Listing 1 showcases the invocation of a module that creates an AWS IAM user (*pausa-infrastructure*) that has the necessary IAM policies attached (associated with the *var.backend_infrastructure_iam_group* IAM group) to allow access to an S3 bucket and DynamoDB table used for storing the Terraform state file and state lock information, respectively, for the infrastructure project at hand (*pausa-infrastructure*, in this example). It is important to highlight that the usage of AWS services for storing the Terraform-related state information is independent of the utilisation of other AWS services for the provisioning of infrastructure for the project at hand. The decision to rely on AWS services such as S3 and DynamoDB was taken due to the easiness of integration with the Terraform backend component[4] and will be fundamental for the proper organisation of NebulOuS-enhanced infrastructure to be managed within the PAUSA endeavour.

[2] HashiCorp. "Terraform by HashiCorp." HashiCorp, https://www.terraform.io/. Accessed 25 July 2024.
[3] Red Hat. "Ansible Automation Platform." Red Hat, https://www.ansible.com/. Accessed 25 July 2024.
[4] Further information on the combination of AWS S3 with AWS DynamoDB for storing the Terraform state file and the Terraform state lock information, respectively, can be consulted online at https://developer.hashicorp.com/terraform/language/settings/backends/s3. Accessed 8 August 2024.

Funded by the European Union

```
module "aws_iam_user_pausa_infrastructure" {
  source = "git@github.com:forcera/terraform-modules.git//aws_iam_user"

  user   = "pausa-infrastructure"
  groups = [var.backend_infrastructure_iam_group]

  generate_access_key = true
}
```

Listing 1 – Example of a Backend Infrastructure project's Terraform Backend and State Information module invocation.

In greater detail, the creation of resources that translate to the establishment of the IAM group for managing the Terraform S3 bucket and DynamoDB table that stores the Terraform state file and state lock information for a given project is depicted in Listing 2. The invocation showcased in Listing 2 is within a file defined as part of the aws_iam_groups module, which defines the IAM group for backend infrastructure and attaches a policy to this group for managing Terraform S3 and DynamoDB for Terraform backend operations. This enables the backend infrastructure to provision not only its own Terraform state management, but also the management of Terraform state files and locks for provisioned projects.

In Listing 3, the creation of the IAM policy for allowing access to the attached IAM user/group for storing the Terraform state file (on AWS S3) and state lock information (on a DynamoDB table) is showcased. This declaration is part of the *aws_iam_policies* module, which specifies the policies required for not only managing the Terraform S3 and DynamoDB backend for Terraform state-keeping purposes but also other AWS-related policies. Within this module, other IAM policies can be defined for the creation and management of AWS services, such as the management of AWS EC2 resources.

```
resource "aws_iam_group" "backend-infrastructure" {
  name = var.backend_infrastructure_iam_group
}

resource "aws_iam_group_policy_attachment" "TerraformS3DynamoDBBackend" {
  group      = aws_iam_group.backend-infrastructure.name
  policy_arn = "arn:aws:iam::${var.aws_account_id}:policy/TerraformS3DynamoDBBackend"
}
```

Listing 2 – Terraform resource for the creation of the Backend Infrastructure IAM Group and associated TerraformS3DynamoDBBackend IAM policy attachment.

```
resource "aws_iam_policy" "TerraformS3DynamoDBBackend" {
  name        = "TerraformS3DynamoDBBackend"
  description = "Provides access to S3 and DynamoDB for the storage of Terraform Backend states.
Managed by Terraform."
  policy = jsonencode({
    "Version" : "2012-10-17",
    "Statement" : [
      {
        "Effect" : "Allow",
        "Action" : "s3:ListBucket",
        "Resource" : "arn:aws:s3:::${var.terraform-backend-s3-bucket}"
      },
      {
        "Effect" : "Allow",
        "Action" : [
          "s3:GetObject",
          "s3:PutObject",
          "s3:DeleteObject"
        ],
        "Resource" : "arn:aws:s3:::${var.terraform-backend-s3-bucket}/$${aws:username}/state/
terraform.tfstate"
      },
      {
        "Effect" : "Allow",
        "Action" : [
          "dynamodb:DescribeTable",
          "dynamodb:GetItem",
          "dynamodb:PutItem",
          "dynamodb:DeleteItem"
        ],
        "Resource" : "arn:aws:dynamodb:*:*:table/${var.terraform-backend-s3-bucket}"
      }
    ]
  })
}
```

Listing 3 - Terraform resource for the creation of the TerraformS3DynamoDBBackend IAM policy.

In addition, and more importantly, the creation of a project automatically triggers the creation of a GitHub Terraform infrastructure repository based on a template that is automatically configured with all necessary secrets to kick-start the development and management of an AWS infrastructure deployment. Within this templated repository, all necessary provisions for establishing an Ansible-powered provision of Virtual Private Servers (VPSs) are also declared. It is expected that all provisions related to the installation and configuration of NebulOuS tools and services and the instantiation and management of PAUSA-related infrastructure resources will take place in this automatically created and managed repository. This module invocation is showcased in Listing 4.

In the provided example, the *pausa-infrastructure* GitHub repository is created based on a template repository that already contains the necessary IaC components (Terraform and Ansible) already defined for kickstarting the creation of a project, in line with PAUSA and NebulOuS needs. As observed in Listing 4, this module also links with AWS services by using the access key details from another module, aws_iam_user_pausa_infrastructure (invocation depicted in Listing 1), which sets up an IAM user. This access key is used to configure the GitHub repository with certain AWS-related secrets, including specifying the AWS S3 bucket and DynamoDB table that will be used for backend storage. The S3 bucket and DynamoDB table are defined by variables that are passed into the module. In essence, this module ensures that the GitHub repository is correctly created and configured with the necessary AWS backend settings to facilitate infrastructure management and deployment.

```
module "gh_repository_pausa_infrastructure" {
  source = "git@github.com:forcera/terraform-modules.git//gh_repository"

  name = "pausa-infrastructure"
  description = "pausa Infrastructure, generated by Terraform ☀️⚙️"

  template = var.repository_template

  aws_access_key_id     =
module.aws_iam_user_pausa_infrastructure.aws_access_key_id
  aws_access_key_secret =
module.aws_iam_user_pausa_infrastructure.aws_access_key_secret

  aws_s3_backend_bucket         = var.aws_s3_backend_bucket
  aws_s3_backend_dynamodb_table = var.aws_s3_backend_dynamodb_table
  aws_s3_backend_region         = var.aws_s3_backend_region
}
```

Listing 4 - Example of a Backend Infrastructure project's GitHub repository creation.

In the PAUSA's backend infrastructure GitHub repository, the *projects* module currently contains the *aws_iam_user_infrastructure_template* and *aws_iam_user_pausa_infrastructure* submodules, which define IAM users, create access keys for these users, and manage their group memberships. While the former correlates to the template to be used for instantiating different projects, the latter is actually the declaration of the PAUSA project. Additionally, there are submodules for GitHub repository management. Both the *gh_repository_infrastructure_template* and *gh_repository_pausa_infrastructure* submodules store AWS access key IDs, S3 backend details, and AWS secret access keys as GitHub Actions secrets, as discussed. These secrets are crucial for configuring GitHub repositories to interact with the AWS backend. Each submodule also defines a GitHub repository and ensures the necessary secrets are linked to these repositories.

The relationships and dependencies in this setup are intricate. The IAM user policy attachment depends on both the caller identity data and the policies defined in the IAM policies module. The IAM group and its policy attachments depend on the specified policies for backend operations. Similarly, the defined IAM users and their access keys are interlinked within their respective submodules. The GitHub secrets are carefully connected to their corresponding repositories to ensure secure and efficient operations.

Overall, this Terraform backend infrastructure setup manages IAM users, groups, policies, and GitHub secrets with a focus on Terraform backend infrastructure using AWS S3 and DynamoDB, as well as Cloud operations project simplification, ensuring a secure and efficient environment for managing cloud resources and integrations.

2.1.1 PAUSA INFRASTRUCTURE MANAGEMENT

In order to enable the establishment of AWS resources for the PAUSA project, the PAUSA team has developed a Terraform EC2 creation and configuration module to comply with the indications of the NebulOuS wiki documentation[5]. The created module utilises the official Terraform AWS provider for

[5] NebulOuS Consortium. "NebulOuS Wiki: 2.1 Managing Cloud Providers", https://github.com/eu-nebulous/nebulous/wiki/2.1-Managing-cloud-providers#aws. Accessed 8 August 2024.

the configuration and management of resources[6]. In Listing 5, the Terraform source code that enables the creation of an AWS EC2 and associated Security Group for the PAUSA project is showcased. This machine will use the official Ubuntu 22.04 AMI, as depicted in Listing 6, following indications from the NebulOuS wiki documentation.

Besides the creation of the AWS EC2 machine, the creation, configuration, and management of AWS network components are also established within a module. In Listing 7, the Terraform source code for the creation of an AWS VPC and associated network components is depicted. It starts by defining a Virtual Private Cloud (VPC) with a specified CIDR block, which dictates the range of IP addresses available within the VPC. DNS support and DNS hostnames are enabled for the VPC to allow for easier management and communication within the network. The VPC is tagged with a name derived from a project variable.

Next, a public subnet is created within this VPC. This subnet is assigned its own CIDR block and is set up in a specific availability zone, which is also specified through variables. The subnet is configured to automatically assign public IP addresses to instances launched in it, and it is tagged with a project-specific name. In the future, should a private subnet be required for hosting AWS services such as a Database, it will be incorporated in the present approach.

An Internet Gateway is then created and attached to the VPC. This gateway allows the VPC to connect to the internet, which is necessary for instances that need external internet access. It is also tagged appropriately. A route table is defined for managing how traffic is routed within the VPC. It is associated with the VPC and tagged with a project name. A default route is added to this route table, directing all outbound traffic (0.0.0.0/0) to the Internet Gateway. This ensures that any traffic not destined for the VPC's private network will be routed to the internet. Finally, the route table is associated with the public subnet, effectively applying the route rules to this subnet so that instances in it can access the internet via the Internet Gateway.

[6] Terraform Documentation. "AWS Provider". Hashicorp, https://registry.terraform.io/providers/hashicorp/aws/latest/docs. Accessed 8 August 2024.

```
resource "aws_security_group" "sg" {
  name        = "${var.project}_sg"
  description = "Terraform-managed security group for the ${var.project} project"
  vpc_id      = aws_vpc.vpc.id
}

resource "aws_security_group_rule" "SSH" {
  type              = "ingress"
  from_port         = 22
  to_port           = 22
  protocol          = "tcp"
  cidr_blocks       = ["0.0.0.0/0"]
  ipv6_cidr_blocks  = ["::/0"]
  security_group_id = aws_security_group.sg.id
}

resource "aws_key_pair" "kp" {
  key_name   = "${var.project}_kp"
  public_key = "${var.ec2_public_key}"
}

resource "aws_instance" "ec2" {
  ami                    = data.aws_ami.ubuntu2204.id
  vpc_security_group_ids = [aws_security_group.sg.id]
  subnet_id              = aws_subnet.public_subnet.id
  key_name               = aws_key_pair.kp.id

  root_block_device {
    volume_size = "${var.ec2_disk_size}"
  }

  instance_type = "${var.ec2_instance_type}"

  tags = {
    Name = "${var.project}-${var.environment}"
  }
}
```

Listing 5 - Terraform source code for the creation of an Ubuntu 22.04 AWS EC2 and associated Security Group for the PAUSA infrastructure project.

```
data "aws_ami" "ubuntu2204" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-jammy-22.04-amd64-server-*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}
```

Listing 6 - Terraform data component for retrieving the Ubuntu Server 22.04 image created and managed by Canonical.

```
resource "aws_vpc" "vpc" {
  cidr_block           = "${var.vpc_cidr}"
  enable_dns_hostnames = true
  enable_dns_support   = true

  tags = {
    Name = "${var.project}_vpc"
  }
}

resource "aws_subnet" "public_subnet" {
  vpc_id                  = aws_vpc.vpc.id
  cidr_block              = "${var.public_subnet_cidr}"
  availability_zone       = "${var.region}${var.public_subnet_az}"
  map_public_ip_on_launch = true

  tags = {
    Name = "${var.project}_public_subnet"
  }
}

resource "aws_internet_gateway" "ig" {
  vpc_id = aws_vpc.vpc.id

  tags = {
    Name = "${var.project}_internet_gateway"
  }
}

resource "aws_route_table" "rt" {
  vpc_id = aws_vpc.vpc.id

  tags = {
    Name = "${var.project}_route_table"
  }
}

resource "aws_route" "default_route" {
  route_table_id         = aws_route_table.rt.id
  destination_cidr_block = "0.0.0.0/0"
  gateway_id             = aws_internet_gateway.ig.id
}


resource "aws_route_table_association" "rt_assoc" {
  subnet_id      = aws_subnet.public_subnet.id
  route_table_id = aws_route_table.rt.id
}
```

Listing 7 - Terraform source code for creating an AWS VPC and associated network configurations, created for the PAUSA infrastructure.

In the domain of application development, an Open Container Initiative (OCI) compliant container images framework will be implemented, utilising container clusters for Knative serverless functions and supporting scalable and responsive serverless applications. Configuration management will be automated using Terraform, as described, and Ansible to maintain consistency across environments.

The Ansible configuration will rely on a set of Ansible playbooks that will actuate on Ansible inventory files auto-generated by Terraform. Each set of Ansible inventory files will group the resources associated with a typology of deployment environments. Thus, it is expected to have two distinct inventory files: one for the AWS resources and another for the OVHCloud resources.

Within the Ansible playbooks, community-supported and -tested roles will be employed. These roles will include the base management of the resource, including updating of the resource, through the *robertdebock.update* role[7], the configuration of security policies within the resource, using the *geerlingguy.security* role[8], and the configuration of the resource firewall, relying on the *geerlingguy.firewall* role[9]. Furthermore, roles for the installation of base software and dependencies within the resources will also be employed. These will include the installation of pip, through the *geerlingguy.pip*[10] role, the installation of git, using the *geerlingguy.git* role[11], the installation of Docker using the *geerlingguy.docker* role[12], and, finally, the installation of Kubernetes, using the *geerlingguy.kubernetes* role[13].

At the time of writing, the installation and configuration of NebulOuS-related services are being performed following the provided NebulOuS documentation[14]. It is expected that during the duration of the PAUSA project, as the PAUSA team gets more familiar with the NebulOuS components' usability and as the documentation gets completed, most of the provisioning of NebulOuS-related components can be set following the same IaC principles as the approaches previously described.

Security protocols, including access controls and data protection mechanisms, will be implemented to comply with GDPR, ISO/IEC 27001, and W3C standards, whenever applicable. PAUSA will leverage, whenever possible, the NebulOuS Security and Privacy Manager component to address privacy and security aspects associated with multi-tenant resources. Furthermore, SLAs are expected to be defined during the project, with the PAUSA team expecting to achieve high system stability and uptime (90%+) so as not to incur significant downtimes or system disruptions during the PAUSA project Demonstration and Feedback Phase.

2.2 CORE FUNCTIONALITY DEVELOPMENT

This phase includes monitoring city pollution, predictive air quality modelling, and generating analytical insights crucial for urban management. PAUSA will leverage serverless capabilities to efficiently capture and process diverse datasets, encompassing air quality indices, traffic flow, noise levels, and meteorological data sourced from IoT devices and publicly available datasets. The architecture, based on NebulOuS's framework, will optimise scalability, latency, and bandwidth utilisation through a cloud continuum approach.

Key tools such as OCI-compliant container images, as well as container-level ML tools (e.g., sci-kit-learn[15], Keras[16]) will be instrumental in developing a predictive model. Insights derived from this model will be visualised through intuitive, serverless OCI-compliant dashboards, supporting decision-making. These dashboards utilize container-based deployments of analytics tools like Metabase[17].

Simultaneously, PAUSA will harness NebulOuS architecture components like the Data Integration and Management System, facilitating efficient handling of multi-cloud cluster data and enabling AI-driven

[7] Robert de Bock, "Ansible Role Update". GitHub, https://github.com/robertdebock/ansible-role-update. Accessed 9 August 2024.
[8] Jeff Geerling, "Ansible Role Security". GitHub, https://github.com/geerlingguy/ansible-role-security. Accessed 9 August 2024.
[9] Jeff Geerling, "Ansible Role Firewall". GitHub, https://github.com/geerlingguy/ansible-role-firewall. Accessed 9 August 2024.
[10] Jeff Geerling, "Ansible Role pip". GitHub, https://github.com/geerlingguy/ansible-role-pip. Accessed 9 August 2024.
[11] Jeff Geerling, "Ansible Role git". GitHub, https://github.com/geerlingguy/ansible-role-git. Accessed 9 August 2024.
[12] Jeff Geerling, "Ansible Role Docker". GitHub, https://github.com/geerlingguy/ansible-role-docker. Accessed 9 August 2024.
[13] Jeff Geering, "Ansible Role Kubernetes". GitHub, https://github.com/geerlingguy/ansible-role-kubernetes. Accessed 9 August 2024.
[14] NebulOuS Consoritum, "The main repository of the NebulOuS Meta Operating System project.", GitHub, https://github.com/eu-nebulous/nebulous. Accessed 9 August 2024.
[15] Pedregosa, F., et al. "Scikit-learn: Machine Learning in Python." Scikit-learn, https://scikit-learn.org/stable/index.html. Accessed 25 July 2024.
[16] Chollet, François, et al. "Keras: The Python Deep Learning API." Keras, https://keras.io/. Accessed 25 July 2024.
[17] Metabase, Inc. "Metabase." Metabase, https://www.metabase.com/. Accessed 25 July 2024.

Anomaly Detection for proactive infrastructure monitoring. The project will rely on NebulOuS SLA Generator to ensure adherence to service-level agreements, enhancing platform reliability and user trust. NebulOuS's Cloud & Fog Service Brokerage will play a pivotal role in managing resource provisioning and orchestration across the cloud continuum, leveraging tools like Deployment Manager and Execution Adapter to dynamically handle serverless components and resources based on application needs and data proximity.

Two metrics will be employed to assess PAUSA functionality development. The first metric focuses on Data Integration Coverage, aiming to integrate a minimum of 75% of identified data sources into the PAUSA platform. This metric evaluates the efficacy of the data integration framework in capturing and processing data from planned sources, ensuring comprehensive analytics and insights generation. The second metric, ML Model Accuracy, aims to achieve at least 80% accuracy (based on the regression coefficient of determination - R2 score[18]) in predictive outcomes during testing. This metric assesses the ML models' effectiveness in accurately forecasting and analysing urban data, which is critical for supporting informed decision-making. These metrics serve as benchmarks to gauge the platform's performance and readiness to meet project objectives.

2.3 DEMONSTRATION AND FEEDBACK

The PAUSA consortium foresees substantial benefits from the platform, including a 30% reduction in urban data analysis time, a 20% enhancement in predictive accuracy for environmental changes, and improved responsiveness in public management systems.

Operational scenarios include pilot testing and feedback collection in representative and/or simulated urban environments. PAUSA demonstrates seamless integration of serverless technologies across different cloud environments and prompt responsiveness. Metrics to measure PAUSA's effectiveness include achieving a minimum 50% user engagement rate during demonstrations and reducing the initial average user response time by 20% following a first iteration of user feedback. These metrics ensure platform usability, performance, and user satisfaction.

[18] Wikipedia. “Coefficient of Determination”. Wikipedia, https://en.wikipedia.org/wiki/Coefficient_of_determination. Accessed 7 August 2024.

2.4 KPIS AND MILESTONES

The following tables summarises and provides deeper detail about PAUSA project evaluation indicators, including the means of verification of each KPI and Milestones.

PHASE	KPI	TARGET	ACTUAL	VERIFICATION
Infrastructure Setup	Automated Infrastructure Provision Coverage	> 75%	30%	Simple IaC code inspection or verification of provisioned resources after deployment via automated frameworks.
Infrastructure Setup	Platform Uptime and Stability	> 90%	-	Inspection of system logs for one week of infrastructure uptime.
Core Functionality and Development	Data Integration Coverage	>= 75%	-	Counting the number of different data source typologies (e.g., noise, air quality) that PAUSA uses to generate insights.
Core Functionality and Development	Air Quality Model R2 Score	> 80%	-	Evaluation of the machine learning regression model based on the R2 score.
Demonstration and Feedback	User Engagement Ratio	>= 50%	-	Counting the potential end users (municipal staff) willing to test the platform who would like to use PAUSA in the future, divided by the total willing testers.
Demonstration and Feedback	User Response time after Preliminary Feedback	< 20%	-	Time tracking on how fast a potential end user (municipal staff) accesses specific information on PAUSA before and after first iteration of user feedback.

Table 1 - PAUSA Project KPI Summary.

MILESTONE	DATE
Infrastructure Provisioning Complete	M1
Configuration and Initial Testing Completed	M1
Completion of Data Integration Framework	M3
Launch of Analytics and Dashboard Prototype	M4
An application modelled using NebulOuS definition language that requires the use of at least 3 different serverless functions that are run during normal application operation.	M4
Demonstration of Integrated Systems	M6
System Optimization and Feedback Collection	M6
Successful Deployment and execution of the application with serverless functions	M7

Table 2 - PAUSA Project Milestones

While some metrics and milestones were reviewed during the development of this document, the consortium was unable to meet the defined objectives for month M1. The delay in achieving these goals can be attributed to three main factors. First, the PAUSA consortium may have underestimated the workload for M1 and the learning curve required by NebulOuS Meta Operating System. Second, M1 coincided with the summer break, during which key team members were on vacation. Lastly, the production and multiple revisions of this document were demanding, reducing the time available to achieve the planned milestones and KPIs.

2.5 FUNCTIONAL VS. NON-FUNCTIONAL REQUIREMENTS

The following table summarises the functional and non-functional requirements that PAUSA project aims to fulfil.

Requirement	Type	DETAIL
Data Collection, Processing, and Integration	Functional	The system must collect environmental data from IoT devices, including air quality, weather, traffic, and noise levels.
Data Storage and Management	Functional	The platform must store both raw and processed data in a database for efficient retrieval and analysis. The system should manage the data flow between various components using a message broker.

Predictive Analytics	Functional	The system should use ML models to predict air quality and generate insights on future pollution levels based on historical data.
Data Visualization	Functional	The platform must offer interactive dashboards for users to visualize and analyse environmental data.
Scalability	Non-Functional	PAUSA must be scalable to handle increasing amounts of data from additional IoT devices and new data sources. The serverless back-end should ensure that the platform can dynamically allocate resources based on demand.
Performance	Non-Functional	The system must achieve high uptime, and the platform should process data with minimal latency to provide real-time insights.
Usability	Non-Functional	The user interface of the analytics dashboards should be user-friendly and provide clear, actionable insights. The platform should allow easy navigation and querying capabilities for end-users.
Maintainability	Non-Functional	The system should be easy to maintain with easy-to-use infrastructure setup. PAUSA must allow for easy updates and integration of new features without significant downtime.
Interoperability	Non-Functional	The platform must seamlessly integrate with different cloud providers.

Table 3 - Functional and Non-Functional Requirements

3 USAGE SCENARIOS

The consortium proposes a solution capable of collecting, processing, storing, and presenting intuitive dashboards regarding city environmental data to end-users. PAUSA advocates for serverless technology due to its scalability, cost-effectiveness, and optimisation for active use periods without idle-time costs. This approach aims to set new benchmarks for scalability, reliability, and user engagement in urban planning tools.

Specific scenarios such as tourism impact, traffic congestion, or forest fires, highlight PAUSA's versatility in handling diverse data sources, predicting environmental impacts, and supporting informed decision-making across urban management domains. The platform's dynamic scalability and cloud resource utilisation, powered by NebulOuS Meta OS, ensure it can manage sudden workload peaks and always be ready to provide actionable insights. We reiterate that NebulOuS is a fundamental component of PAUSA as it facilitates the deployment of the platform in different infrastructure environments, ensuring high-availability and performance even when natural disasters or seasonal events take place.

Moreover, ensuring low latency is crucial for providing actionable insights in scenarios like traffic congestion or emergency situations. Increased latency should be one of the signals to scale up PAUSA. Thus, PAUSA scalability will take into account the latency metric, that will trigger another PAUSA environment (e.g., new cloud provider deployment) if the average latency value for HTTP requests within the last 10 minutes exceeds 0.3 seconds. Similarly, average latency below 0.1 seconds should downscale the excedent provisioned environments, as it represents a return to nominal operation.

Additionally, the workload of PAUSA will also be measured taking into account the average CPU load and Memory consumption of the provision resources for a given environment. Average CPU load over 80% or Memory consumption above 90% for the last 10 minutes should trigger the deployment of a new PAUSA environment. For downscaling the infrastructure, average CPU load under 20% and average Memory consumption below 40% (within 10 minutes) should trigger the removal of excedent environments.

Regular monitoring and adjustments based on usage patterns and performance data will help refine this threshold over time.

LOW REQUIREMENTS

Considering an urban environment such as Lisbon, during Autumn and Spring, there will be low requirements for the usage of PAUSA. Typically, there are fewer incidents occurring during these seasons, resulting in a lower number of NebulOuS's workers and services executions and PAUSA environment deployments.

HIGH REQUIREMENTS

In contrast, during Summer and Winter, the Lisbon region typically experiences more events, such as fires, floods, tourism, and local festivities. Therefore, during these periods, additional PAUSA instances (other environments) should be triggered to maintain the same quality standards amid the increased usage of the platform.

4 ANNEX A

4.1 GITHUB REPOSITORY MODULE

```
resource "github_repository" "repository" {
  name        = "${var.name}"
  description = var.description

  visibility = var.visibility

  has_issues = var.has_issues

  is_template = var.is_template

  dynamic "template" {
    for_each = var.template
    content {
      owner = template.value["owner"]
      repository = template.value["repository"]
    }
  }
}

resource "github_actions_secret" "aws_access_key_id" {
  count = var.aws_access_key_id == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_ACCESS_KEY_ID"
  plaintext_value  = var.aws_access_key_id
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_secret_access_key" {
  count = var.aws_access_key_secret == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_ACCESS_KEY_SECRET"
  plaintext_value  = var.aws_access_key_secret
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_s3_backend_bucket" {
  count = var.aws_s3_backend_bucket == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_S3_BACKEND_BUCKET"
  plaintext_value  = var.aws_s3_backend_bucket
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_s3_backend_dynamodb_table" {
  count = var.aws_s3_backend_dynamodb_table == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_S3_BACKEND_DYNAMODB_TABLE"
  plaintext_value  = var.aws_s3_backend_dynamodb_table
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_s3_backend_key" {
  count = var.aws_s3_backend_bucket == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_S3_BACKEND_KEY"
  plaintext_value  = "${var.name}"
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_s3_backend_region" {
  count = var.aws_s3_backend_region == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_S3_BACKEND_REGION"
  plaintext_value  = var.aws_s3_backend_region
  depends_on = [ github_repository.repository ]
}

resource "github_actions_secret" "aws_s3_static_page_bucket" {
  count = var.aws_s3_static_page_bucket == null ? 0 : 1
  repository       = "${var.name}"
  secret_name      = "AWS_S3_BUCKET"
  plaintext_value  = var.aws_s3_static_page_bucket
  depends_on = [ github_repository.repository ]
}
```

Listing 8 – GitHub repository module main Terraform file.

4.2 AWS IAM USER MODULE

```
resource "aws_iam_user" "user" {
  name = "${var.user}"
}

resource "aws_iam_user_group_membership" "user" {
  user = aws_iam_user.user.name
  groups = "${var.groups}"
}

resource "aws_iam_access_key" "user" {
  count = var.generate_access_key ? 1 : 0
  user = aws_iam_user.user.name
  status = var.access_key_status
}
```

Listing 9 - AWS IAM User Module main Terraform file.

CONSORTIUM

NebulOuS

A META OPERATING SYSTEM FOR BROKERING HYPER-DISTRIBUTED APPLICATIONS ON CLOUD COMPUTING CONTINUUMS

Funded by the European Union