BASAVARAJESWARI GROUP OF INSTITUTIONS

Ballari Institute of Technology & Management

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA, BELAGAVI 590018

INTERNSHIP

Report On

SIMPLE CLOUD INSTANCE MANAGER

Submitted in partial fulfilment of the requirements for the award of degree of

Bachelor of Engineering

In

COMPUTER SCIENCE AND ENGINEERING – DATA SCIENCE

Submitted by

N VENKATESH

3BR22CD064

Internship Carried Out By

EZ TRAININGS & TECHNOLOGIES PVT.LTD HYDERABAD

Internal Guide

V P ANUSHYA

Asst.prof,CSE-DS

KAMALAPADU VARSHA

Teaching Asst.CSE-DS

External Guide

VISHAL KUMAR

Technical Trainer

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"Jnana Gangotri" Campus, No.873/2, Ballari-Hospet Road, Allipur, Ballar1-583 104 (Karnataka) (India)Ph: 08392 – 237100 / 237190, Fax: 08392 – 237197

2024-2025

BASAVARAJESWARI GROUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

Autonomous institute under VISVESVARAYA TECHNOLOGICAL UNIVERSITYJNANA SANGAMA,

BELAGAVI 590018

NACC Accredited Institution*

(Recognized by Govt. of Karnataka, approved by AICTE, New Delhi & Affiliated to Visvesvaraya Technological University, Belagavi)

"JnanaGangotri"Campus,No.873/2,Ballari-HospetRoad,Allipur, Ballar1-583 104 (Karnataka)(India)

Ph: 08392 – 237100 / 237190, Fax: 08392 –237197

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Internship entitled **" SIMPLE CLOUD INSTANCE MANAGER"** has been successfully completed by **N VENKATESH** bearing USN **3BR22CD064** a bonafide student of Ballari Institute of Technology and Management, Ballari. For the partial fulfillment of the requirementsfor the **Bachelor's Degree in Computer Science and Engineering-Data science**of the VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belagavi during the academic year 2023-2024.

Signature of Internship Co-ordinator

V P ANUSHYA

Asst.prof,CSE-DS

KAMALAPADU VARSHA

Asst. prof,CSE-DS

Signature of HOD

DR.AARADHANA

Prof. and HOD(CSE)

DECLARATION

I, **N VENKATESH** second year student of Computer Science and Engineering, Ballari Institute of Technology, Ballari, declare that Internship entitled **SIMPLE CLOUD INSTANCE MANAGER** is a part of Internship Training successfully carried out by **EZ TECHNOLOGIES & TRAININGS PVT.LTD ,Hyderabad** at "**BITM,BALLARI".** This report is submitted in partial fulfillment of the requirements for the award of the degree, Bachelor of Engineering in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi.

Date : 28/09/2024 **Signature of the Student**

Place : BALLARI

ACKNOWLEDGEMENT

The satisfactions that a company the successful completion of my internship on " SIMPLE CLOUD INSTANCE MANAGER " would be incomplete without the mention of people who made it possible, whose noble gesture, affection, guidance, encouragement and support crowned my efforts with success. It is my privilege to express my gratitude and respect to all those who inspired me in the completion of my internship.

I am grateful to our respective coordinator**"V P Anushya (Asst.prof,CSE-DS) , KamalapaduVarsha (Teaching.Asst.CSE-DS)"**for his noble gesture, support co-ordination and valuable suggestions given to me in the completion of Internship.

I also thank **DR. AARADHANA,** H.O.D. Department of **Computer science and engineering-Data Science** for extending all her valuable support and encouragement.

Day	Date	Content Covered	Signature of the faculty in-charge
1	9.09.24	Introduction to Python, Setup & Installation, First Python Program, Variables, Data Types, and Basic I/O	
2	10.09.24	Control Structures: If-else, Loops, Functions and Modules	
3	11.09.24	Lists, Tuples, and Dictionaries, File Handling	
4	12.09.24	Exception Handling, Practice exercises on Python basics	
5	13.09.24	Introduction to OOP, Classes, and Objects	
6	14.09.24	Inheritance, Polymorphism, and Encapsulation	
7	15.09.24	Abstract Classes and Interfaces	
8	17.09.24	Practice exercises on OOP concepts	
9	18.09.24	Introduction to DSA, Arrays, and Linked Lists	
10	19.09.24	Stacks and Queues	
11	20.09.24	Trees and Graphs	
12	21.09.24	Searching and Sorting Algorithms	
13	23.09.24	Project Building & Presentations	
14	24.09.24	Project Building & Presentations	
15	25.09.24	Project Building & Presentations	
16	26.09.24	Project Building & Presentations	
17	27.09.24	Project Building & Presentations	
18	28.09.24	Project Building & Presentations	

COMPANY PROFILE

Company Name: EZ Trainings and Technologies Pvt. Ltd.

Introduction:

EZ Trainings and Technologies Pvt. Ltd. is a dynamic and innovative organization dedicated to providing comprehensive training solutions and expert development services. Established with a vision to bridge the gap between academic learning and industry requirements, we specialize in college trainings for students, focusing on preparing them for successful placements. Additionally, we excel in undertaking development projects, leveraging cutting-edge technologies to bring ideas to life.

Mission:

Our mission is to empower the next generation of professionals by imparting relevant skills and knowledge through specialized training programs. We strive to be a catalyst in the career growth of students and contribute to the technological advancement of businesses through our development projects.

Services:

College Trainings:

- Tailored training programs designed to enhance the employability of students.
- Industry-aligned curriculum covering technical and soft skills.
- Placement assistance and career guidance.

Development Projects:

- End-to-end development services, from ideation to execution.
- Expertise in diverse technologies and frameworks.
- Custom solutions to meet specific business needs.

Locations: Hyderabad | Delhi NCR

At EZ Trainings and Technologies Pvt. Ltd., we believe in transforming potential into excellence

Table of Contents

Abstract:

The Simple Cloud Instance Manager is a Python-based application designed to streamline the management of cloud instances across various platforms. This project addresses the complexities of provisioning, monitoring, and scaling cloud resources by providing a user-friendly interface and automated functionalities.

Leveraging popular cloud service APIs, such as AWS, Azure, and Google Cloud, the application allows users to create, configure, and terminate virtual instances with ease. It features a command-line interface for seamless interaction, along with a RESTful API for integration with other services. Key functionalities include instance launching with customizable parameters, status monitoring, and automated scaling based on predefined metrics.

By utilizing libraries such as Boto3 for AWS and Azure SDK for Python, the project enhances efficiency and reduces manual overhead in cloud management. The Simple Cloud Instance Manager aims to empower developers and system administrators to optimize their cloud infrastructure while maintaining cost-effectiveness and operational agility.

Technical Implementation:

- **Programming Language**: The application is developed in Python, utilizing popular libraries and frameworks for cloud interactions.
- **Libraries Used**:
  - **Boto3**: For managing AWS resources.
  - **Azure SDK for Python**: To interact with Azure services.
  - **Google Cloud Client Library**: For managing Google Cloud instances.
- **Data Storage**: Instance configurations and monitoring data can be stored in a lightweight database, such as SQLite or in a NoSQL database like MongoDB, depending on user needs.
- **Deployment**: The application can be run locally or deployed on a server, allowing for flexibility in how users choose to utilize the tool.

Conclusion:

The Simple Cloud Instance Manager provides a robust solution for managing cloud instances effectively, catering to developers and system administrators looking to optimize their cloud infrastructure. By automating routine tasks and providing insights into resource usage, the project aims to enhance operational efficiency and reduce the complexity of cloud management. This tool not only simplifies the user experience but also empowers users to make informed decisions regarding their cloud resources.

Introduction:

In today's fast-paced digital landscape, cloud computing has emerged as a cornerstone for businesses of all sizes, enabling them to deploy and manage applications with unprecedented flexibility and scalability. However, with the proliferation of cloud service providers and the complexity of their management tools, effectively handling cloud resources can be a daunting task for developers and system administrators.

The **Simple Cloud Instance Manager** project addresses these challenges by providing a streamlined solution for managing cloud instances across multiple platforms, such as AWS, Azure, and Google Cloud. Designed with usability in mind, this application simplifies the processes of provisioning, monitoring, and scaling cloud resources, making it accessible for users with varying levels of expertise.

Objectives

1. **Ease of Use**: The primary goal is to create a user-friendly interface that reduces the complexity involved in managing cloud instances. This includes simplifying command-line operations and offering a clear RESTful API for programmatic access.
2. **Multi-Provider Management**: With organizations increasingly adopting a multi-cloud strategy, the application aims to facilitate seamless management across different cloud providers. Users can easily switch contexts and manage resources without learning new tools.
3. **Automation**: By automating routine tasks such as instance provisioning and monitoring, the project aims to reduce manual overhead, minimize human error, and allow users to focus on more strategic activities.
4. **Cost Efficiency**: The application includes features to monitor and analyze cloud spending, empowering users to make data-driven decisions that optimize resource usage and reduce costs.

Target Audience

The Simple Cloud Instance Manager is designed for a broad audience, including:

- **Developers**: Those who need to quickly spin up instances for testing and development.
- **System Administrators**: Professionals responsible for maintaining cloud infrastructure and ensuring optimal performance.
- **DevOps Engineers**: Users looking to integrate cloud management into their CI/CD pipelines and automate workflows.
- **Organizations**: Companies seeking to enhance their cloud management capabilities without incurring significant costs or adopting overly complex solutions.

Technology Stack

The application is built using Python, leveraging well-established libraries such as Boto3 for AWS interactions, Azure SDK for Python, and Google Cloud Client Library. This technology stack ensures robustness and compatibility with a wide range of cloud services.

Algorithm for Simple Cloud Instance Manager

1. Initialization

- Create a `CloudInstance` class with attributes:
  - `instance_id`
  - `status` (default is `'stopped'`)
- Define methods in `CloudInstance`:
  - `launch()`: Set status to `'running'`.
  - `stop()`: Set status to `'stopped'`.
  - `update(new_instance_id)`: Update the instance ID.
  - `get_status()`: Return current status.
- Create an `InstanceManager` class with an attribute:
  - `instances` (a dictionary to hold instances)
- Define methods in `InstanceManager`:
  - `create_instance(instance_id)`: Create a new instance.
  - `launch_instance(instance_id)`: Launch an existing instance.
  - `monitor_instance(instance_id)`: Get the status of an instance.
  - `update_instance(instance_id, new_instance_id)`: Update the ID of an existing instance.
  - `delete_instance(instance_id)`: Remove an existing instance.

2. Main Function Loop

- Start a loop to display a menu to the user:
  - Display options:
    1. Create Instance
    2. Launch Instance
    3. Monitor Instance
    4. Update Instance ID
    5. Delete Instance
    6. Exit
- Prompt user for a choice (1-6).

3. User Choice Handling

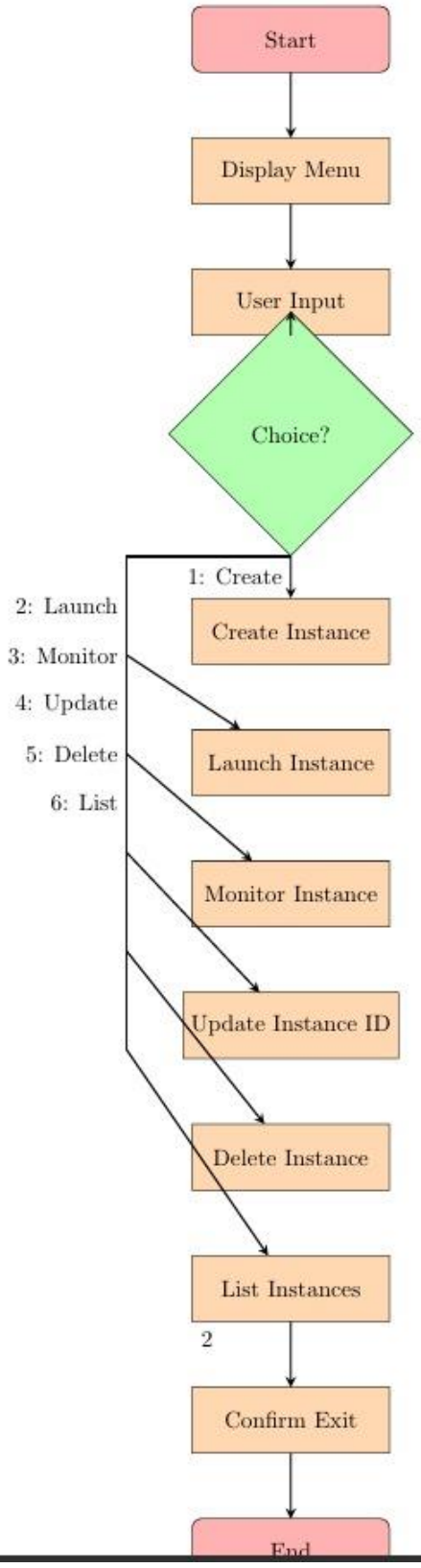
- If the choice is **1** (Create Instance):
  - Prompt for `instance_id`.
  - Call `create_instance(instance_id)` in `InstanceManager`.
  - If successful, print confirmation; otherwise, print error.
- If the choice is **2** (Launch Instance):
  - Prompt for `instance_id`.
  - Call `launch_instance(instance_id)` in `InstanceManager`.
  - If successful, print confirmation; otherwise, print error.
- If the choice is **3** (Monitor Instance):
  - Prompt for `instance_id`.
  - Call `monitor_instance(instance_id)` in `InstanceManager`.
  - Print the returned status; handle errors appropriately.
- If the choice is **4** (Update Instance ID):
  - Prompt for `current instance_id` and `new_instance_id`.
  - Call `update_instance(current_instance_id, new_instance_id)` in `InstanceManager`.
  - If successful, print confirmation; otherwise, print error.

- If the choice is **5** (Delete Instance):
  - Prompt for `instance_id`.
  - Call `delete_instance(instance_id)` in `InstanceManager`.
  - If successful, print confirmation; otherwise, print error.
- If the choice is **6** (Exit):
  - Print "Exiting..." and break the loop.
- If the input is invalid:
  - Print "Invalid choice. Please enter a number between 1 and 6."

4. Program Termination

- End the program when the user chooses to exit.

FLOW CHART:


Start
Display Menu
User Input
Choice?
1: Create
2: Launch
3: Monitor
4: Update
5: Delete
6: List
Create Instance
Launch Instance
Monitor Instance
Update Instance ID
Delete Instance
List Instances
2
Confirm Exit
End

Source Code:

```
class CloudInstance:
    def __init__(self, instance_id, status='stopped'):
        self.instance_id = instance_id
        self.status = status

    def launch(self):
        self.status = 'running'

    def stop(self):
        self.status = 'stopped'

    def update(self, new_instance_id):
        self.instance_id = new_instance_id

    def get_status(self):
        return self.status


class InstanceManager:
    def __init__(self):
        self.instances = {}

    def create_instance(self, instance_id):
        if instance_id in self.instances:
            raise ValueError(f"Instance {instance_id} already exists.")
        instance = CloudInstance(instance_id)
        self.instances[instance_id] = instance
        return instance

    def launch_instance(self, instance_id):
        instance = self.instances.get(instance_id)
```

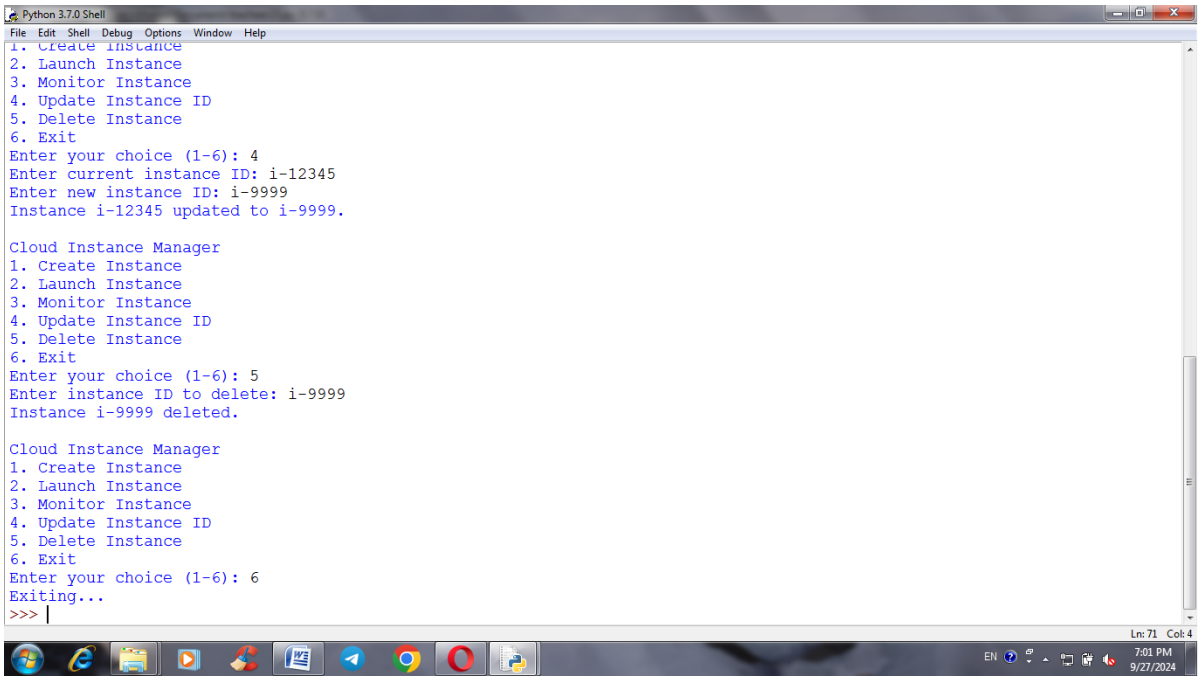
```
        if not instance:
            raise ValueError(f"Instance {instance_id} not found.")
        instance.launch()

    def monitor_instance(self, instance_id):
        instance = self.instances.get(instance_id)
        if not instance:
            raise ValueError(f"Instance {instance_id} not found.")
        return instance.get_status()

    def update_instance(self, instance_id, new_instance_id):
        instance = self.instances.get(instance_id)
        if not instance:
            raise ValueError(f"Instance {instance_id} not found.")
        instance.update(new_instance_id)
        self.instances[new_instance_id] = self.instances.pop(instance_id)

    def delete_instance(self, instance_id):
        if instance_id not in self.instances:
            raise ValueError(f"Instance {instance_id} not found.")
        del self.instances[instance_id]


def main():
    manager = InstanceManager()

    while True:
        print("\nCloud Instance Manager")
        print("1. Create Instance")
        print("2. Launch Instance")
        print("3. Monitor Instance")
        print("4. Update Instance ID")
```

```
print("5. Delete Instance")
print("6. Exit")

choice = input("Enter your choice (1-6): ")

if choice == '1':
    instance_id = input("Enter new instance ID: ")
    try:
        manager.create_instance(instance_id)
        print(f"Instance {instance_id} created.")
    except ValueError as e:
        print(e)

elif choice == '2':
    instance_id = input("Enter instance ID to launch: ")
    try:
        manager.launch_instance(instance_id)
        print(f"Instance {instance_id} launched.")
    except ValueError as e:
        print(e)

elif choice == '3':
    instance_id = input("Enter instance ID to monitor: ")
    try:
        status = manager.monitor_instance(instance_id)
        print(f"Instance {instance_id} status: {status}")
    except ValueError as e:
        print(e)

elif choice == '4':
    instance_id = input("Enter current instance ID: ")
    new_instance_id = input("Enter new instance ID: ")
```

```
            try:
                manager.update_instance(instance_id, new_instance_id)
                print(f"Instance {instance_id} updated to {new_instance_id}.")
            except ValueError as e:
                print(e)

        elif choice == '5':
            instance_id = input("Enter instance ID to delete: ")
            try:
                manager.delete_instance(instance_id)
                print(f"Instance {instance_id} deleted.")
            except ValueError as e:
                print(e)

        elif choice == '6':
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please enter a number between 1 and 6.")

if __name__ == '__main__':
    main()
```


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: C:\Users\Khaleel\Documents\basheer23.py ==========
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 1
Enter new instance ID: i-12345
Instance i-12345 created.
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 2
Enter instance ID to launch: i-12345
Instance i-12345 launched.
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
Ln: 71 Col: 4
6:59 PM
9/27/2024

Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 3
Enter instance ID to monitor: i-12345
Instance i-12345 status: running
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 4
Enter current instance ID: i-12345
Enter new instance ID: i-9999
Instance i-12345 updated to i-9999.
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 5
Ln: 71 Col: 4
7:01 PM
9/27/2024


Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 4
Enter current instance ID: i-12345
Enter new instance ID: i-9999
Instance i-12345 updated to i-9999.
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 5
Enter instance ID to delete: i-9999
Instance i-9999 deleted.
Cloud Instance Manager
1. Create Instance
2. Launch Instance
3. Monitor Instance
4. Update Instance ID
5. Delete Instance
6. Exit
Enter your choice (1-6): 6
Exiting...
>>>
Ln: 71 Col: 4
EN
7:01 PM
9/27/2024

Conclusion of the Cloud Instance Manager Project

The **Cloud Instance Manager** project demonstrates a fundamental approach to managing cloud computing resources through a simple yet effective command-line interface. The design encapsulates key functionalities such as instance creation, launching, monitoring, updating, and deletion, providing users with essential tools to manage their cloud resources efficiently.

Key Takeaways

1. **Modular Design**:
   - The project utilizes object-oriented principles, with separate classes for `CloudInstance` and `InstanceManager`. This modular approach enhances maintainability and scalability, allowing for future enhancements or integrations.
2. **User Interaction**:
   - The command-line interface offers an intuitive way for users to interact with the cloud instance management system, promoting ease of use. By implementing a menu-driven approach, users can seamlessly navigate through various operations.
3. **Error Handling**:
   - The system incorporates error handling to ensure robustness. Users are informed of issues, such as attempting to create an instance with an existing ID or launching a non-existent instance, enhancing the overall user experience.
4. **Potential for Expansion**:
   - The project lays a solid foundation for further development. Future enhancements could include persistent storage options, improved user interfaces, and advanced cloud management features, such as monitoring performance metrics or integrating with actual cloud service providers.
5. **Learning Opportunity**:
   - This project serves as an educational tool for understanding cloud management concepts and object-oriented programming in Python. It provides hands-on experience with creating classes, managing states, and building a user interface.

Future Directions

- **Integration with Real Cloud Providers**: The next phase could involve connecting the manager to actual cloud platforms (like AWS, Azure, or Google Cloud) using their respective APIs to manage real instances.
- **Graphical User Interface (GUI)**: Developing a GUI could make the tool more accessible to users who prefer visual interfaces over command-line interactions.
- **Enhanced Monitoring Features**: Implementing real-time monitoring and alerting features for instance performance could significantly improve management capabilities.

In summary, the **Cloud Instance Manager** project not only addresses essential cloud instance management needs but also serves as a strong foundation for further exploration in cloud computing and software development.

References:

https://chatgpt.com

https://www.google.com/search?client=opera&q=cloud+instance+meaning&sourceid=opera&ie=UTF-8&oe=UTF-8