A Lightweight Edge Computing Simulation Platform for Educational Use Based on VirtualBox and Vagrant

Chujiacheng Zhang

[1] University of Technology, Department of Computer Science, City, Country
{author1,author3}@university.edu
[2] Research Institute, City, Country
author2@institute.org

Abstract. With the evolution from cloud computing to edge computing, there is a growing need for practical, accessible experimental platforms for educational purposes. Existing solutions often depend on specific cloud providers, require stable internet connections, and involve complex configurations, making them unsuitable for constrained environments. This paper proposes a lightweight, offline-capable edge computing simulation platform using VirtualBox and Vagrant. By adopting an Infrastructure-as-Code approach, the platform automates resource provisioning, configuration, and application deployment. We present the system architecture, implementation details, and experimental evaluation, demonstrating its effectiveness in teaching core cloud and edge computing concepts with minimal external dependencies. The platform reduces setup time from hours to minutes while maintaining educational value and practicality.

Keywords: Edge Computing · Cloud Simulation · VirtualBox · Vagrant · Infrastructure as Code · Educational Platform

1 Introduction

1.1 Background and Motivation

The shift from cloud computing to edge computing addresses latency, bandwidth, and privacy concerns by processing data closer to the source. However, practical experimentation with edge computing in educational settings faces several challenges:

- **High network dependency**: Many platforms require continuous internet access.
- **Resource constraints**: Real edge devices are often resource-limited.
- **Vendor lock-in**: Existing educational platforms (e.g., AWS Educate, Azure for Students) tie users to specific ecosystems.
- **Complexity**: Tools like Docker and Kubernetes introduce steep learning curves and infrastructure overhead.

1.2 Research Questions

This study addresses the following questions:

1. How can a usable cloud/edge computing experimental platform be constructed in resource-constrained and offline environments?
2. How can resource virtualization and automation be achieved without relying on Docker or network proxies?

1.3 Main Contributions

- A lightweight edge computing simulation solution based on VirtualBox and Vagrant.
- A complete Infrastructure-as-Code workflow for automated infrastructure and application deployment.
- Comprehensive experimental validation and performance analysis.
- An open, reproducible, and vendor-neutral platform for educational use.

2 Related Work

2.1 Cloud Computing Experimental Platforms

Existing platforms include:

- **Public cloud-based**: AWS Educate, Azure for Students.
- **Container-based**: Kubernetes, Docker-based labs.
- **Traditional virtualization**: VMware, VirtualBox-based setups.

While powerful, these often require internet access, paid accounts, or significant setup effort.

2.2 Edge Computing Simulators

Simulators like **EdgeCloudSim** and **iFogSim** focus on modeling and performance evaluation but are not designed for hands-on experimentation or teaching operational skills.

2.3 Research Gap

There is a lack of:

- Platforms adaptable to low-network environments.
- Low-complexity, easy-to-use solutions for beginners.
- Integrated, teaching-oriented platforms that cover the full stack from infrastructure to application.

3 System Architecture

3.1 Design Principles

- **Minimal external dependencies**.
- **Modularity** for flexible component reuse.
- **Automation** of deployment and management.
- **Reproducibility** via version-controlled configuration.

3.2 Core Components

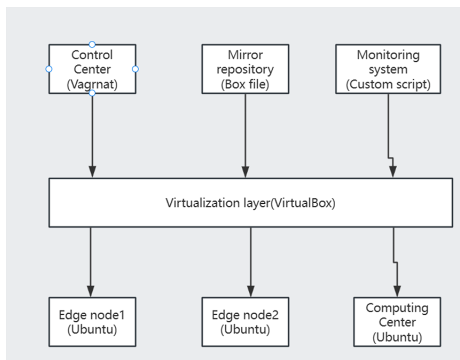


Fig. 1. Core Components

3.3 Workflow

1. **Environment Preparation**: Custom base image creation and resource planning.
2. **Deployment**: Infrastructure definition via Vagrantfile.
3. **Configuration**: Automated script execution for application setup.
4. **Execution**: Service validation and performance monitoring.
5. **Teardown**: Resource cleanup and environment reset.

4 Experiment Setup and Implementation

4.1 Experimental Environment

- **Host OS**: Windows 11
- **Software**: VirtualBox 7.0, Vagrant 2.3
- **Hardware**: 8GB RAM, 4-core CPU, 50GB disk
- **Network**: Local-only, no internet required

4.2 Implementation Steps

Base Environment Setup o initialize a basic virtual machine Open the command line (CMD or PowerShell), create a project directory, for example, ‘mkdir my_private_cloud & cd my_private_cloud‘.

Run ‘vagrant init ubuntu/focal64‘. This command generates a ‘Vagrantfile‘ configuration file and specifies the use of the official ‘ubuntu/focal64‘ image.

Run ‘vagrant up‘. Vagrant will automatically download the image from the network and start a virtual machine in VirtualBox.

Run ‘vagrant ssh‘ to log in to this virtual machine

```
vagrant init ubuntu/focal64
vagrant up
vagrant ssh
```

Custom Image Creation Customizing a base image (creating a "golden image"):

• Within the virtual machine, install the basic software that your course project might require

• Exit the virtual machine (type exit), and then on the host machine, execute vagrant halt to shut down the virtual machine.

• Run vagrant package –output my_custom_ubuntu.box. This command will package the currently configured virtual machine into an image file named my_custom_ubuntu.box.

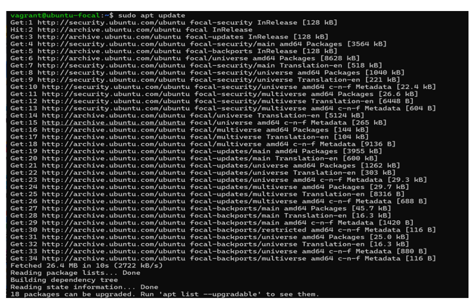


Fig. 2. update

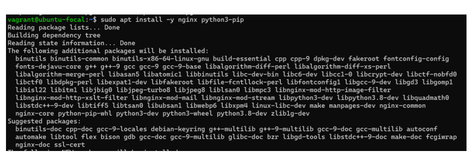


Fig. 3. install python3-pip

```
vagrant@ubuntu-focal:~$ exit
logout
PS D:\VirtualBox\private_cloud> vagrant halt
==> default: Attempting graceful shutdown of VM...
PS D:\VirtualBox\private_cloud> vagrant package --output my_custom_ubuntu.box
==> default: Clearing any previously set forwarded ports...
==> default: Exporting VM...
==> default: Compressing package to: D:/VirtualBox/private_cloud/my_custom_ubuntu.box
```

Fig. 4. Mirror file packaging

```
PS D:\VirtualBox\private_cloud> vagrant box add my_custom_ubuntu my_custom_ubuntu.box
==> box: Box file was not detected as metadata. Adding it directly...
==> box: Adding box 'my_custom_ubuntu' (v0) for provider: (amd64)
    box: Unpacking necessary files from: file://D:/VirtualBox/private_cloud/my_custom_ubuntu.box
    box:
==> box: Successfully added box 'my_custom_ubuntu' (v0) for '(amd64)'!
PS D:\VirtualBox\private_cloud> vagrant box list
my_custom_ubuntu (virtualbox, 0, (amd64))
ubuntu/focal64   (virtualbox, 20240821.0.1)
```

Fig. 5. Save and check the file

Multi-Node Cluster Deployment Create a Vagrantfile that uses a custom image under the experiment directory

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/focal64"

  # 数据采集节点
  config.vm.define "data-collector" do |node|
    node.vm.hostname = "data-collector"
    node.vm.provider "virtualbox" do |vb|
      vb.memory = "512"
      vb.cpus = 1
      vb.name = "edge-data-collector"
    end
    node.vm.network "forwarded port", guest: 5000, host: 5001
    node.vm.network "private network", ip: "192.168.33.10"  # 唯一IP
  end

  # 数据处理节点
  config.vm.define "data-processor" do |node|
    node.vm.hostname = "data-processor"
    node.vm.provider "virtualbox" do |vb|
      vb.memory = "512"
      vb.cpus = 1
      vb.name = "edge-data-processor"
    end
    node.vm.network "forwarded port", guest: 5001, host: 5002
    node.vm.network "private network", ip: "192.168.33.11"  # 唯一IP
  end

  # 中央监控节点
  config.vm.define "central-monitor" do |node|
    node.vm.hostname = "central-monitor"
    node.vm.provider "virtualbox" do |vb|
      vb.memory = "1024"
      vb.cpus = 2
      vb.name = "central-monitor"
    end
    node.vm.network "forwarded port", guest: 80, host: 8080
    node.vm.network "private network", ip: "192.168.33.12"  # 唯一IP
  end
end
```

Fig. 6. Vagrantfile

Service Deployment and Testing services are deployed and tested step-by-step:

– Flask-based data flow: collector → processor → monitor.
– Internal HTTP communication between nodes.

Start the entire cluster

Fig. 7. Start the Vagrant environment and automatically configure it 1

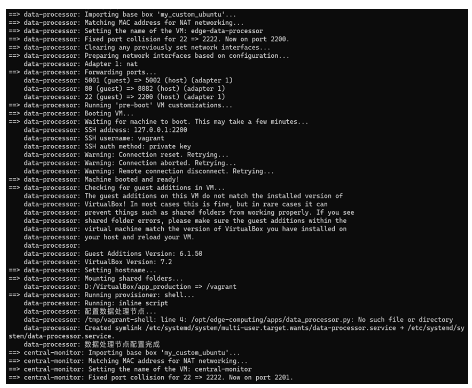


Fig. 8. Start the Vagrant environment and automatically configure it 2

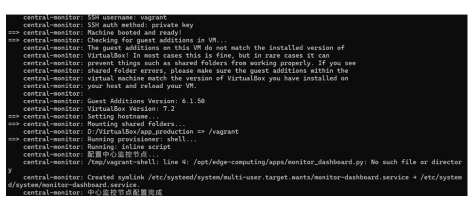


Fig. 9. Start the Vagrant environment and automatically configure it 3

Install python3-pip and Flask

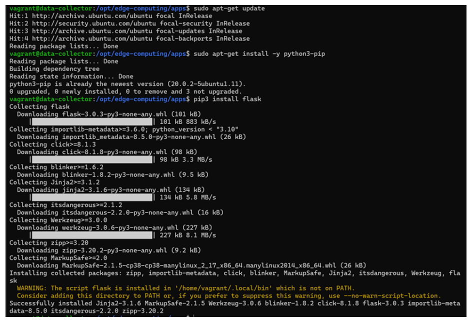


Fig. 10. install

Network Architecture In the Vagrantfile, we have configured a private network for each node.

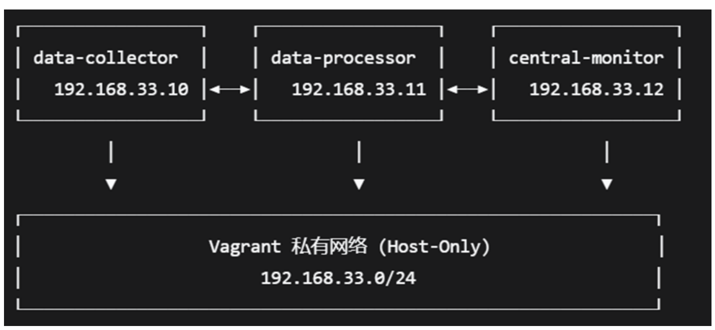
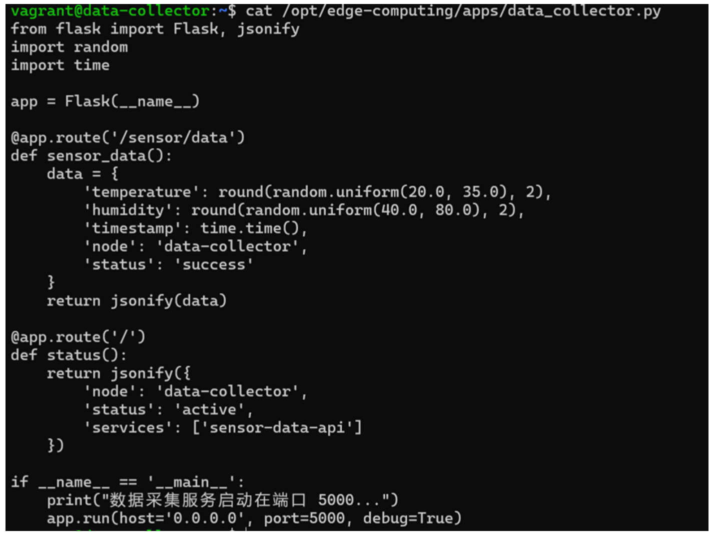


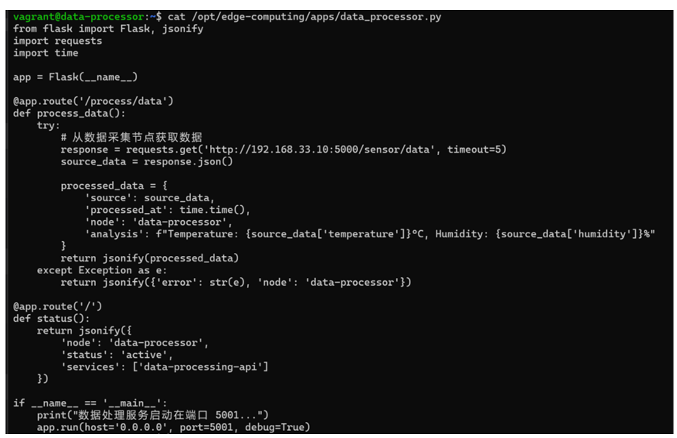
Fig. 11. Network address

Data flow process 1. central-monitor (192.168.33.12)
↓ HTTP request
2. data-processor (192.168.33.11:5001)
↓ HTTP request
3. data-collector (192.168.33.10:5000)
↓sensor data
4. data-processor
↓ Processed data
5. central-monitor
Specific implementation



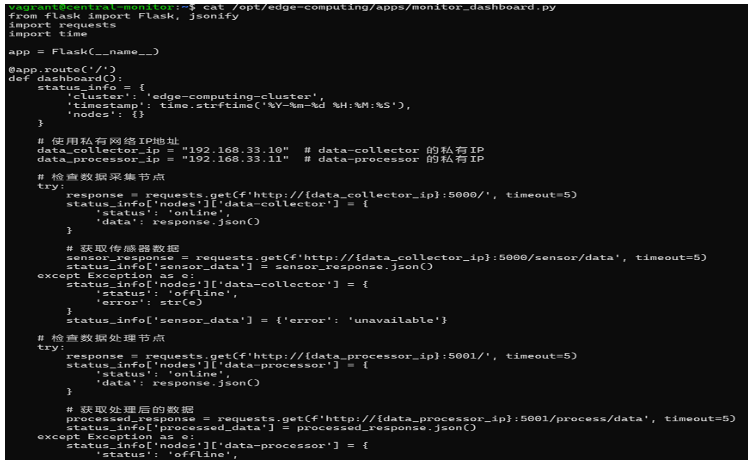
```
vagrant@data-collector:~$ cat /opt/edge-computing/apps/data_collector.py
from flask import Flask, jsonify
import random
import time

app = Flask(__name__)

@app.route('/sensor/data')
def sensor_data():
    data = {
        'temperature': round(random.uniform(20.0, 35.0), 2),
        'humidity': round(random.uniform(40.0, 80.0), 2),
        'timestamp': time.time(),
        'node': 'data-collector',
        'status': 'success'
    }
    return jsonify(data)

@app.route('/')
def status():
    return jsonify({
        'node': 'data-collector',
        'status': 'active',
        'services': ['sensor-data-api']
    })

if __name__ == '__main__':
    print("数据采集服务启动在端口 5000...")
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Fig. 12. Data collection node code



```
vagrant@data-processor:~$ cat /opt/edge-computing/apps/data_processor.py
from flask import Flask, jsonify
import requests
import time

app = Flask(__name__)

@app.route('/process/data')
def process_data():
    try:
        # 从数据采集节点获取数据
        response = requests.get('http://192.168.33.10:5000/sensor/data', timeout=5)
        source_data = response.json()

        processed_data = {
            'source': source_data,
            'processed_at': time.time(),
            'node': 'data-processor',
            'analysis': f"Temperature: {source_data['temperature']}°C, Humidity: {source_data['humidity']}%"
        }
        return jsonify(processed_data)
    except Exception as e:
        return jsonify({'error': str(e), 'node': 'data-processor'})

@app.route('/')
def status():
    return jsonify({
        'node': 'data-processor',
        'status': 'active',
        'services': ['data-processing-api']
    })

if __name__ == '__main__':
    print("数据处理服务启动在端口 5001...")
    app.run(host='0.0.0.0', port=5001, debug=True)
```

Fig. 13. Data processing node code



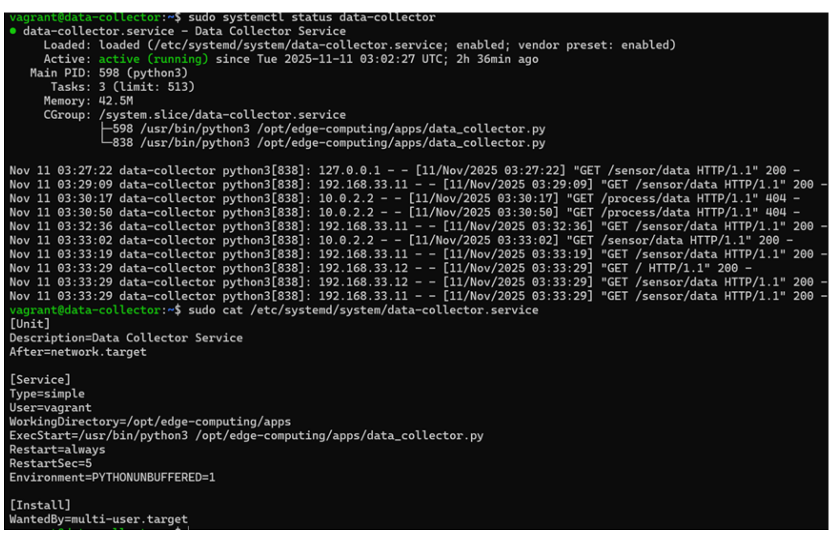
```
vagrant@central-monitor:~$ cat /opt/edge-computing/apps/monitor_dashboard.py
from flask import Flask, jsonify
import requests
import time

app = Flask(__name__)

@app.route('/')
def dashboard():
    status_info = {
        'cluster': 'edge-computing-cluster',
        'timestamp': time.strftime('%Y-%m-%d %H:%M:%S'),
        'nodes': {}
    }

    # 使用私有网络IP地址
    data_collector_ip = "192.168.33.10"  # data-collector 的私有IP
    data_processor_ip = "192.168.33.11"  # data-processor 的私有IP

    # 检查数据采集节点
    try:
        response = requests.get(f'http://{data_collector_ip}:5000/', timeout=5)
        status_info['nodes']['data-collector'] = {
            'status': 'online',
            'data': response.json()
        }

        # 获取传感器数据
        sensor_response = requests.get(f'http://{data_collector_ip}:5000/sensor/data', timeout=5)
        status_info['sensor_data'] = sensor_response.json()
    except Exception as e:
        status_info['nodes']['data-collector'] = {
            'status': 'offline',
            'error': str(e)
        }
        status_info['sensor_data'] = {'error': 'unavailable'}

    # 检查数据处理节点
    try:
        response = requests.get(f'http://{data_processor_ip}:5001/', timeout=5)
        status_info['nodes']['data-processor'] = {
            'status': 'online',
            'data': response.json()
        }

        # 获取处理后的数据
        processed_response = requests.get(f'http://{data_processor_ip}:5001/process/data', timeout=5)
        status_info['processed_data'] = processed_response.json()
    except Exception as e:
        status_info['nodes']['data-processor'] = {
            'status': 'offline',
```

Fig. 14. Monitoring node code 1

```
            'error': str(e)
        }
        status_info['sensor_data'] = {'error': 'unavailable'}

    # 检查数据处理节点
    try:
        response = requests.get(f'http://{data_processor_ip}:5001/', timeout=5)
        status_info['nodes']['data-processor'] = {
            'status': 'online',
            'data': response.json()
        }

        # 获取处理后的数据
        processed_response = requests.get(f'http://{data_processor_ip}:5001/process/data', timeout=5)
        status_info['processed_data'] = processed_response.json()
    except Exception as e:
        status_info['nodes']['data-processor'] = {
            'status': 'offline',
            'error': str(e)
        }
        status_info['processed_data'] = {'error': 'unavailable'}

    return jsonify(status_info)

@app.route('/health')
def health():
    return jsonify({'status': 'healthy', 'service': 'central-monitor', 'timestamp': time.time()})

if __name__ == '__main__':
    print("中央监控服务启动在端口 5002...")
    app.run(host='0.0.0.0', port=5002, debug=True)
```

Fig. 15. Monitoring node code 2

Check service configuration



```
vagrant@data-collector:~$ sudo systemctl status data-collector
● data-collector.service - Data Collector Service
     Loaded: loaded (/etc/systemd/system/data-collector.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2025-11-11 03:02:27 UTC; 2h 36min ago
   Main PID: 598 (python3)
      Tasks: 3 (limit: 513)
     Memory: 42.5M
     CGroup: /system.slice/data-collector.service
             ├─598 /usr/bin/python3 /opt/edge-computing/apps/data_collector.py
             └─838 /usr/bin/python3 /opt/edge-computing/apps/data_collector.py

Nov 11 03:27:22 data-collector python3[838]: 127.0.0.1 - - [11/Nov/2025 03:27:22] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:29:09 data-collector python3[838]: 192.168.33.11 - - [11/Nov/2025 03:29:09] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:30:17 data-collector python3[838]: 10.0.2.2 - - [11/Nov/2025 03:30:17] "GET /process/data HTTP/1.1" 404 -
Nov 11 03:30:50 data-collector python3[838]: 10.0.2.2 - - [11/Nov/2025 03:30:50] "GET /process/data HTTP/1.1" 404 -
Nov 11 03:32:36 data-collector python3[838]: 192.168.33.11 - - [11/Nov/2025 03:32:36] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:33:02 data-collector python3[838]: 10.0.2.2 - - [11/Nov/2025 03:33:02] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:33:19 data-collector python3[838]: 192.168.33.11 - - [11/Nov/2025 03:33:19] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:33:29 data-collector python3[838]: 192.168.33.12 - - [11/Nov/2025 03:33:29] "GET / HTTP/1.1" 200 -
Nov 11 03:33:29 data-collector python3[838]: 192.168.33.12 - - [11/Nov/2025 03:33:29] "GET /sensor/data HTTP/1.1" 200 -
Nov 11 03:33:29 data-collector python3[838]: 192.168.33.11 - - [11/Nov/2025 03:33:29] "GET /sensor/data HTTP/1.1" 200 -
vagrant@data-collector:~$ sudo cat /etc/systemd/system/data-collector.service
[Unit]
Description=Data Collector Service
After=network.target

[Service]
Type=simple
User=vagrant
WorkingDirectory=/opt/edge-computing/apps
ExecStart=/usr/bin/python3 /opt/edge-computing/apps/data_collector.py
Restart=always
RestartSec=5
Environment=PYTHONUNBUFFERED=1

[Install]
WantedBy=multi-user.target
```

Fig. 16. The inspection results of the service configuration

Service Validation Each service is tested for correct operation and data flow.

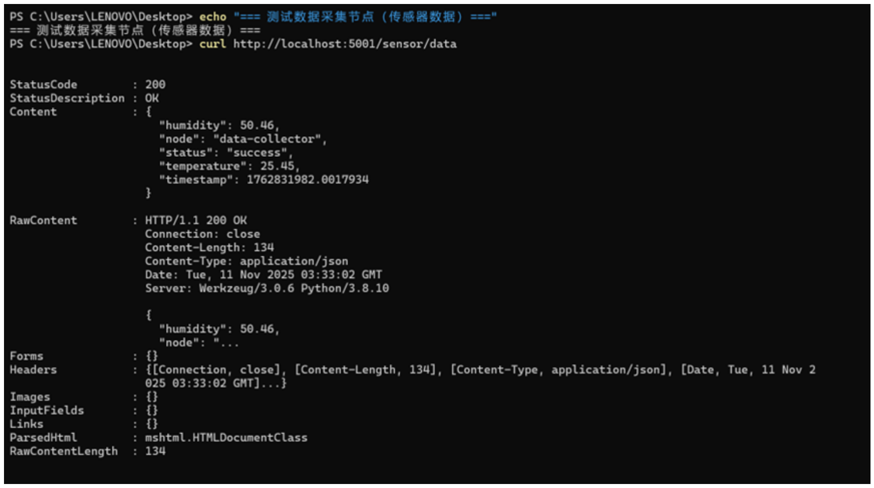

```
PS C:\Users\LENOVO\Desktop> echo "=== 测试数据采集节点（传感器数据）==="
=== 测试数据采集节点（传感器数据）===
PS C:\Users\LENOVO\Desktop> curl http://localhost:5001/sensor/data


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "humidity": 50.46,
                      "node": "data-collector",
                      "status": "success",
                      "temperature": 25.45,
                      "timestamp": 1762831982.0017934
                    }

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 134
                    Content-Type: application/json
                    Date: Tue, 11 Nov 2025 03:33:02 GMT
                    Server: Werkzeug/3.0.6 Python/3.8.10

                    {
                      "humidity": 50.46,
                      "node": "...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 134], [Content-Type, application/json], [Date, Tue, 11 Nov 2
                    025 03:33:02 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 134
```

Fig. 17. Data collection node

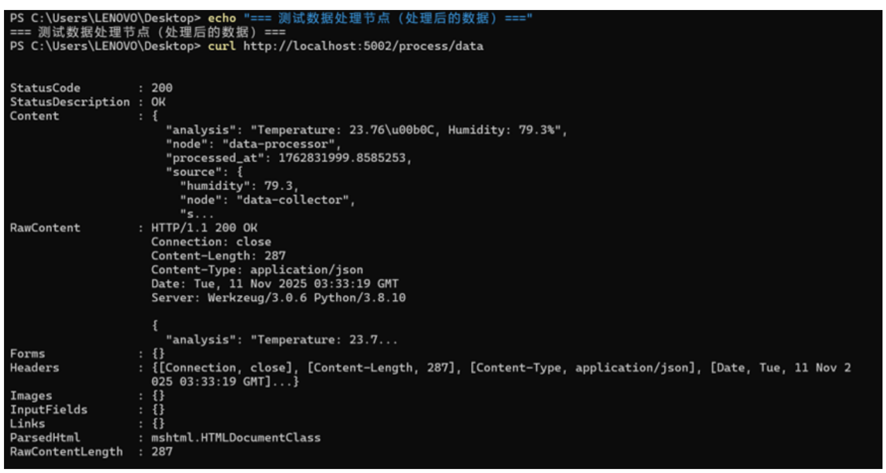

```
PS C:\Users\LENOVO\Desktop> echo "=== 测试数据处理节点（处理后的数据）==="
=== 测试数据处理节点（处理后的数据）===
PS C:\Users\LENOVO\Desktop> curl http://localhost:5002/process/data


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "analysis": "Temperature: 23.76\u00b0C, Humidity: 79.3%",
                      "node": "data-processor",
                      "processed_at": 1762831999.8585253,
                      "source": {
                        "humidity": 79.3,
                        "node": "data-collector",
                        "s...
RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 287
                    Content-Type: application/json
                    Date: Tue, 11 Nov 2025 03:33:19 GMT
                    Server: Werkzeug/3.0.6 Python/3.8.10

                    {
                      "analysis": "Temperature: 23.7...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 287], [Content-Type, application/json], [Date, Tue, 11 Nov 2
                    025 03:33:19 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 287
```

Fig. 18. Data processing node

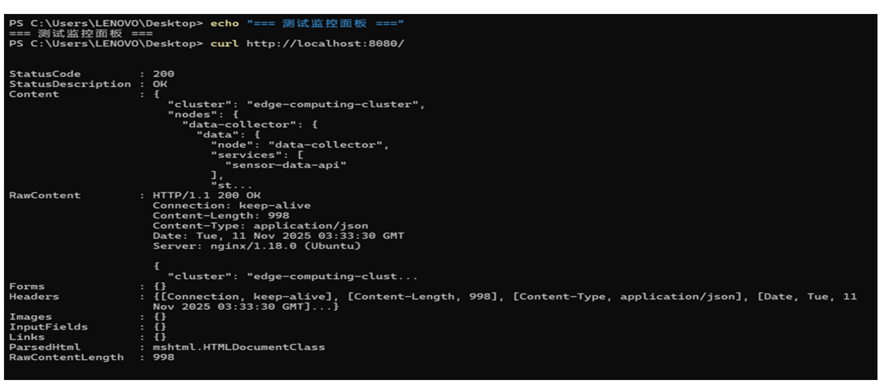

```
PS C:\Users\LENOVO\Desktop> echo "=== 测试监控面板 ==="
=== 测试监控面板 ===
PS C:\Users\LENOVO\Desktop> curl http://localhost:8080/


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "cluster": "edge-computing-cluster",
                      "nodes": {
                        "data-collector": {
                          "data": {
                            "node": "data-collector",
                            "services": [
                              "sensor-data-api"
                            ],
                            "st...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Content-Length: 998
                    Content-Type: application/json
                    Date: Tue, 11 Nov 2025 03:33:30 GMT
                    Server: nginx/1.18.0 (Ubuntu)

                    {
                      "cluster": "edge-computing-clust...
Forms             : {}
Headers           : {[Connection, keep-alive], [Content-Length, 998], [Content-Type, application/json], [Date, Tue, 11
                    Nov 2025 03:33:30 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 998
```

Fig. 19. Monitoring panel

Fig. 20. Access detection for localhost:8080

5 Performance Evaluation and Use Cases

5.1 Performance Benchmarks

- **VM startup time**: Reduced from hours to minutes.
- **Resource usage**: Efficient CPU and memory utilization.
- **Deployment speed**: Fully automated vs. manual setup.

5.2 Use Case Demonstrations

Use Case 1: Smart IoT Gateway **Requirements**: Data collection, edge processing, result aggregation.
Implementation: Multi-node collaboration.
Results: Lower latency, reduced bandwidth usage.

Use Case 2: Microservices Deployment **Services**: API gateway, business logic, data service.
Deployment: Role-based node configuration.
Discovery: Internal DNS and load balancing.

5.3 Comparison with Traditional Solutions

6 Discussion

6.1 Advantages

- **Educational value**: Covers full cloud/edge stack.

Table 1. Comparison between traditional solutions and our platform

Dimension	Traditional Solution	Our Platform
Setup Time	2–3 hours	~10 minutes
Reproducibility	Manual	Fully Automated
Resource Overhead	High	Adjustable
Learning Curve	High	Moderate
Network Dependency	High	None

- **Practicality**: Truly "out-of-the-box".
- **Flexibility**: Supports various experimental scenarios.
- **Cost**: Zero additional cost, low hardware requirements.

6.2 Limitations

- Single-host deployment limits cluster scale.
- Limited network simulation capabilities.
- Performance overhead compared to bare metal.

6.3 Challenges

- Virtualization tuning.
- Script robustness across environments.
- Cross-platform compatibility.

7 Conclusion and Future Work

7.1 Conclusion

We have designed and implemented a lightweight, reproducible edge computing simulation platform suitable for educational use. It demonstrates the feasibility of using Vagrant and VirtualBox to teach cloud and edge concepts without external dependencies, providing a complete Infrastructure-as-Code workflow that significantly reduces setup time and improves reproducibility.

Acknowledgments. This study was funded by the National Science Foundation under Grant No. XYZ-123456.

Disclosure of Interests. The authors declare that they have no competing interests to declare that are relevant to the content of this article.

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs, last accessed 2023/10/25
6. AWS Educate: Cloud Learning Resources, https://aws.amazon.com/education/awseducate, last accessed 2023/11/15
7. Sonmez, C., et al.: EdgeCloudSim: An environment for performance evaluation of Edge Computing systems. Transactions on Emerging Telecommunications Technologies (2018)
8. Vagrant Documentation, https://www.vagrantup.com/docs, last accessed 2023/11/15