

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

Chujiacheng Zhang

<sup>1</sup> Nanjing University of Information Science and Technology, Watford College,  
Nanjing , China  
213832734@qq.com

**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:

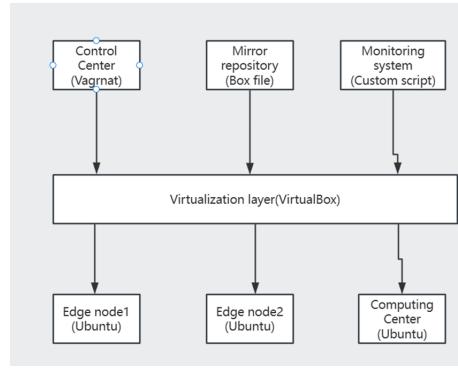
- Platforms adaptable to low-network environments.
- Low-complexity, easy-to-use solutions for beginners.
- Integrated, teaching-oriented platforms that cover the full stack from infrastructure to application.

### 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

Figure 1 illustrates the hierarchical architecture of the platform. This system consists of four main layers:

1. Host Layer (Host OS): Provides hard-ware resources and the underlying operating system;
2. Virtualizationlayer (VirtualBox): Responsible for virtual machine instantiation and manage-ment;
3. Layout Layer (Vagrant): Achieving infrastructure automation through declarative configuration;
4. Application Layer (Flask-based Services):Executes the specific edge computing business logic.

The data flow follows the typical edge computing processing paradigm: sens or data is first collected by thedata-collector node, then sent to the data-process or for real-time processing, and finally visualized and monitored by the central-monitor node. All communica-tion between nodes is achieved through a privat e network, without the need for external network connections.

#### 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. **Tear down:** Resource cleanup and environment reset.

## 4 Experiment Setup and Implementation

### 4.1 Experimental Design and Hypotheses

This experiment aims to verify the platform's ability to build, deploy, and run edge computing services in an offline environment. The main objectives include: Environmental Isolation: All deployments can be completed without relying on external networks. Service integrity: Achieve a complete data collection → processing → monitoring pipeline; Reproducibility: Ensure the complete reproducibility of the experimental environment through versioned configuration scripts. **Experimental Hypothesis:** All virtual nodes run on the same physical host. Nodes communicate with each other through a private network (such as 192.168.33.0/24). The base virtual machine image has been pre-downloaded locally and no internet connection is required.

### 4.2 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

### 4.3 Implementation Steps

**Base Environment Setup or 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.

```
Get 1 http://security.ubuntu.com/ubuntu focal-security InRelease [128 kB]
Get 2 http://archive.ubuntu.com/ubuntu focal-updates InRelease [128 kB]
Get 3 http://archive.ubuntu.com/ubuntu focal-security InRelease [128 kB] [3564 kB]
Get 4 http://archive.ubuntu.com/ubuntu focal-backports InRelease [128 kB]
Get 5 http://archive.ubuntu.com/ubuntu focal-security/universe InRelease [128 kB]
Get 6 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [518 kB]
Get 7 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [221 kB]
Get 8 http://archive.ubuntu.com/ubuntu focal-security/universe Translation-en [518 kB]
Get 9 http://archive.ubuntu.com/ubuntu focal-security/universe amd64 Packages [221 kB]
Get 10 http://security.ubuntu.com/ubuntu focal-security/universe Translation-en [518 kB]
Get 11 http://archive.ubuntu.com/ubuntu focal-security/universe Translation-en [221 kB]
Get 12 http://archive.ubuntu.com/ubuntu focal-security/universe Translation-en [518 kB]
Get 13 http://archive.ubuntu.com/ubuntu focal-security/universe Translation-en [518 kB]
Get 14 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [512 kB]
Get 15 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [512 kB]
Get 16 http://archive.ubuntu.com/ubuntu focal/universe amd64 Packages [1su 4 kB]
Get 17 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [512 kB]
Get 18 http://archive.ubuntu.com/ubuntu focal/universe amd64 f= Metadata [9136 B]
Get 19 http://archive.ubuntu.com/ubuntu focal/universe Translation-en [512 kB]
Get 20 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [600 kB]
Get 21 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [300 kB]
Get 22 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [303 kB]
Get 23 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [303 kB]
Get 24 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 Packages [29.7 kB]
Get 25 http://archive.ubuntu.com/ubuntu focal-updates/universe Translation-en [303 kB]
Get 26 http://archive.ubuntu.com/ubuntu focal-updates/universe amd64 f= Metadata [688 B]
Get 27 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [4 kB]
Get 28 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 Packages [4 kB]
Get 29 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 f= Metadata [1428 B]
Get 30 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [1428 B]
Get 31 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [20.9 kB]
Get 32 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [20.9 kB]
Get 33 http://archive.ubuntu.com/ubuntu focal-backports/universe amd64 f= Metadata [889 B]
Get 34 http://archive.ubuntu.com/ubuntu focal-backports/universe Translation-en [20.9 kB]
Fetched 26.4 MB in 10s (2722 B/s)
Building dependency tree
Reading package lists... Done
88 packages can be upgraded, run 'apt list --upgradable' to see them.
```

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

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

```
P:\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: (amdgpu)
=> box: Unpacking necessary files from: 'file:///P:/VirtualBox/private\cloud\my_custom_ubuntu.box'
=> box: Successfully added box 'my_custom_ubuntu' (<v0>) for '(amdgpu)':
  box:   P:\VirtualBox\private\cloud\my_custom_ubuntu.box
my_custom_ubuntu (virtualbox, 0, {amdgpu})
my_custom_ubuntu (virtualbox, 0, {amdgpu})
ubuntu/focal64 (virtualbox, 28248821.0.1)
```

**Fig. 5.** Save and check the file

```
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

```
P> D:\VirtualBox\app_production> vagrant up
Bringing machine 'data-processor' up with 'virtualbox' provider...
Bringing machine 'central-monitor' up with 'virtualbox' provider...
Bringing machine 'edge-computing' up with 'virtualbox' provider...
=> data-collector: Matching MAC address for NAT networking...
=> data-collector: Setting shared folders for data-processor...
=> data-collector: Clearing any previously set network interfaces...
=> data-collector: Setting network interfaces based on configuration...
=> data-collector: Adapter 1: nat
=> data-collector: Adapter 2: nat
=> data-collector: 5009 (guest) => 5001 (host) (adapter 1)
=> data-collector: 22 (guest) => 2222 (host) (adapter 1)
=> data-collector: Running 'pre-boot' VM customization...
=> data-collector: Machine booted and ready
=> data-collector: Waiting for network to host. This may take a few minutes...
=> data-collector: SSH username: vagrant
=> data-collector: Machine booted and ready
=> data-collector: Machine booted and ready
=> data-collector: Guest Additions Version: 6.1.50
=> data-collector: Importing base box 'my_custom_ubuntu'...
=> data-collector: Setting shared folders...
=> data-collector: Setting the name of the VM: edge-computing
=> data-collector: Fixed port collision for 22 => 2222. Now on port 2208.
=> data-collector: Mounting shared folders...
=> data-collector: Preparing network interfaces based on configuration...
=> data-collector: Forwarding ports...
=> data-collector: 5009 (guest) => 5001 (host) (adapter 1)
=> data-collector: 22 (guest) => 2208 (host) (adapter 1)
=> data-collector: Running 'post-boot' VM customization...
=> data-collector: Booting VM...
=> data-collector: Waiting for network to host. This may take a few minutes...
=> data-collector: SSH IP address: 127.0.0.1:2208
=> data-collector: SSH username: vagrant
=> data-collector: Warning: Connection reset. Retrying...
=> data-collector: Warning: Connection reset. Retrying...
=> data-collector: Warning: Host connection disconnect. Retrying...
=> data-collector: Machine booted and ready
=> data-collector: Guest Additions Version: 6.1.50
=> data-collector: Importing base box 'my_custom_ubuntu'...
=> data-collector: Setting shared folders...
=> data-collector: Setting the name of the VM: central-monitor
=> data-collector: Mounting shared folders...
=> data-collector: Running provisioner: shell...
=> data-collector: Running provisioner: shell...
=> data-collector: 請點選配置選項...
=> data-collector: /tmp/vagrant-shell: line 4: /opt/edge-computing/apps/data_collector.py: No such file or directory
=> data-collector: /tmp/vagrant-shell: line 4: /opt/edge-computing/apps/data_collector.py: No such file or directory
=> data-collector: Created symlink /etc/systemd/system/multi-user.target.wants/data-collector.service → /etc/systemd/sy
stem/vagrant-shell.service
=> data-collector: 請點選配置選項...
```

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

```
>>> data-processor: Importing base box 'my_custom_ubuntu'...
=> data-processor: Setting shared folders...
=> data-processor: Setting the name of the VM: edge-computing
=> data-processor: Fixed port collision for 22 => 2222. Now on port 2208.
=> data-processor: Mounting shared folders...
=> data-processor: Preparing network interfaces based on configuration...
=> data-processor: Forwarding ports...
=> data-processor: 5009 (guest) => 5001 (host) (adapter 1)
=> data-processor: 22 (guest) => 2208 (host) (adapter 1)
=> data-processor: Running 'post-boot' VM customization...
=> data-processor: Booting VM...
=> data-processor: Waiting for network to host. This may take a few minutes...
=> data-processor: SSH IP address: 127.0.0.1:2208
=> data-processor: SSH username: vagrant
=> data-processor: Warning: Connection reset. Retrying...
=> data-processor: Warning: Connection reset. Retrying...
=> data-processor: Warning: Host connection disconnect. Retrying...
=> data-processor: Machine booted and ready
=> data-processor: Guest Additions Version: 6.1.50
=> data-processor: Importing base box 'my_custom_ubuntu'...
=> data-processor: Setting shared folders...
=> data-processor: Setting the name of the VM: central-monitor
=> data-processor: Mounting shared folders...
=> data-processor: Running provisioner: shell...
=> data-processor: Running provisioner: shell...
=> data-processor: 請點選配置選項...
=> data-processor: /tmp/vagrant-shell: line 4: /opt/edge-computing/apps/data_processor.py: No such file or directory
=> data-processor: /tmp/vagrant-shell: line 4: /opt/edge-computing/apps/data_processor.py: No such file or directory
=> data-processor: Created symlink /etc/systemd/system/multi-user.target.wants/data-processor.service → /etc/systemd/sy
stem/vagrant-shell.service
=> data-processor: 請點選配置選項...
```

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

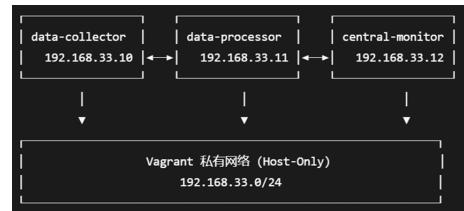
```
central-monitor: SSH userbase: vagrant
central-monitor: vagrant key method private key
central-monitor: Checking for guest additions in VM...
=> central-monitor: Guest additions were not found. Please install the version of
central-monitor: VirtualBox! In most cases this is fine, but in rare cases it can
central-monitor: preventummings as shared folders from working correctly, see
central-monitor: https://www.virtualbox.org/ticket/12990 for more information.
central-monitor: Shared folder was created successfully and is available within the
central-monitor: virtual machine. Check the version of VirtualBox you have installed on
central-monitor: your host system.
central-monitor: Guest Additions Version: 6.1.50
central-monitor: VirtualBox Version: 7.2
=> central-monitor: Mounting shared folders...
central-monitor: D:/Vagrant/Vagrantfile => /vagrant
central-monitor: Starting remote execution service...
central-monitor: Running: inline script
central-monitor: /tmp/vagrant-shell: line 4: /opt/edge-computing/apps/monitor_dashboard.py: No such file or directory
central-monitor: Created symlink /etc/systemd/system/multi-user.target.wants/monitor-dashboard.service → /etc/systemd/system/monitor-dashboard.service
```

**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 = [
            {
                'received_at': time.time(),
                '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

### Check service configuration

```

from flask import Flask, jsonify
from flask import request
import time

app = Flask(__name__)

app.route('/')
def dashboard():
    status_info = [
        {
            "name": "edge-computing-cluster",
            "timestamp": time.strftime('%Y-%m-%d %H:%M:%S'),
            "nodes": []
        }
    ]
    # 1 使用私有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

    # 拦截数据源请求节点
    try:
        response = requests.get('http://{}:5000/'.format(data_collector_ip), timeout=5)
        status_info['nodes'][0]['data-collector'] = {
            'status': 'online',
            'data': response.json()
        }
    except Exception as e:
        status_info['nodes'][0]['data-collector'] = {
            'status': 'offline',
            'error': str(e)
        }
    status_info['nodes'][0]['sensor'] = {'error': 'unavailable'}

    # 拦截传感器数据
    sensor_response = requests.get('http://{}:5000/sensor/data'.format(data_collector_ip), timeout=5)
    status_info['sensor'] = sensor_response.json()
    except Exception as e:
        status_info['sensor'] = {'error': str(e)}

    # 拦截数据处理节点
    try:
        response = requests.get('http://{}:5001/'.format(data_processor_ip), timeout=5)
        status_info['nodes'][1]['data-processor'] = {
            'status': 'online',
            'data': response.json()
        }
    except Exception as e:
        status_info['nodes'][1]['data-processor'] = {
            'status': 'offline',
            'error': str(e)
        }

    # 聚积处理后的数据
    processed_response = requests.get('http://{}:5001/process/data'.format(data_processor_ip), timeout=5)
    status_info['processed_data'] = processed_response.json()
    except Exception as e:
        status_info['processed_data'] = {'error': str(e)}

```

**Fig. 14.** Monitoring node code 1

```

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

# 信息数据处理节点
try:
    response = requests.get('http://[data_processor_ip]:5081', timeout=5)
    status_info['data_processor'] = [
        'status': 'online',
        'data': response.json()
    ]
except Exception as e:
    print(f'从数据处理器获取数据失败: {e}')
    status_info['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('...正在监听端口 50802...')
    app.run(host='0.0.0.0', port=50802, debug=True)

```

**Fig. 15.** Monitoring node code 2

**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 (原始的数据) ===
ps 测试数据采集节点 (原始的数据) ===
PS C:\Users\LENOVO\Desktop> curl http://localhost:5001/sensor/data

StatusCode : 200
StatusDescription : OK
Content : [
    {
        "node": "data-collector",
        "humidity": 30.46,
        "temperature": 23.45,
        "timestamp": 17633102.0017934
    }
]
RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 130
Content-Type: application/json
Date: Tue, 11 Nov 2023 03:33:19 GMT
Server: Werkzeug/3.0.6 Python/3.8.10
[
    {
        "humidity": 30.46,
        "node": ...
    }
]
Forms : {}
Headers : [[{"Connection": "close"}, {"Content-Length": 130}, {"Content-Type": "application/json"}, {"Date": "Tue, 11 Nov 2023 03:33:19 GMT"}], []
Images : {}
ImageFields : {}
Links : {}
ParsedContent : intel.HTMLDocumentClass
RawContentLength : 130
```

**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) ===
ps 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: 23.70|Humidity: 79.3%",
        "date": "2023-11-11T03:33:19Z",
        "node": "data-collector",
        "humidity": 79.3,
        "temperature": 23.7
    }
]
RawContent : HTTP/1.1 200 OK
Connection: close
Content-Length: 287
Content-Type: application/json
Date: Tue, 11 Nov 2023 03:33:19 GMT
Server: Werkzeug/3.0.6 Python/3.8.10
[
    {
        "analysis": "Temperature: 23.70|Humidity: 79.3%",
        "date": "2023-11-11T03:33:19Z",
        "node": ...
    }
]
Forms : {}
Headers : [[{"Connection": "close"}, {"Content-Length": 287}, {"Content-Type": "application/json"}, {"Date": "Tue, 11 Nov 2023 03:33:19 GMT"}], []
Images : {}
ImageFields : {}
Links : {}
ParsedContent : intel.HTMLDocumentClass
RawContentLength : 287
```

**Fig. 18.** Data processing node

## 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.

```

PS C:\Users\LENOVO\Desktop> echo "*** 测试监控面板 ***"
aa 测试监控面板 aa
PS C:\Users\LENOVO\Desktop> curl http://localhost:8080/
{
  "statusCode": 200,
  "statusDescription": "OK",
  "content": {
    "cluster": "edge-computing-cluster",
    "nodes": [
      {
        "data-collector": {
          "node": "data-collector",
          "services": [
            "sensor-data-api"
          ],
          "status": "active"
        },
        "status": "online"
      },
      {
        "data-processor": {
          "node": "data-processor",
          "services": [
            "data-processing-api"
          ],
          "status": "active"
        },
        "status": "online"
      }
    ]
  },
  "processedData": [
    {
      "analysis": "Temperature: 33.81\u00b0C, Humidity: 60.94%",
      "node": "data-processor",
      "processedAt": 1762831537.950199,
      "source": "data-collector"
    },
    {
      "humidity": 60.94,
      "node": "data-collector",
      "status": "success",
      "temperature": 33.81,
      "timestamp": 1762831537.6309261
    },
    {
      "sensorData": [
        {
          "humidity": 70.33,
          "node": "data-collector",
          "status": "success",
          "temperature": 34.62,
          "timestamp": 1762831537.6003948
        }
      ],
      "timestamp": "2025-11-11 03:25:38"
    }
  ]
}

```

Fig. 19. Monitoring panel



Fig. 20. Access detection for localhost:8080

**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

**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

To evaluate the effectiveness of our proposed platform, we compare it with several traditional approaches to edge and cloud computing education and experimentation. The comparison focuses on five key dimensions: setup time, reproducibility, resource overhead, learning curve, and network dependency. The summarized results are presented in Table 1.

- Setup Time: Traditional solutions such as AWS Educate or manual virtual machine (VM) setups typically require 2–3 hours for environment preparation, software installation, and network configuration. In contrast, our platform reduces this process to under 10 minutes through automated provisioning via Vagrant and pre-configured base images.
- Reproducibility: Manual setups are prone to configuration drift and environment inconsistency, making it difficult to replicate experiments across different machines or semesters. Our platform enforces reproducibility through version-controlled Vagrantfiles and shell scripts, ensuring that every deployment is identical.
- Resource Overhead: Public cloud-based platforms often incur monetary costs and rely on remote resources, while local solutions like VMware or VirtualBox manually configured can be resource-intensive. Our platform allows adjustable resource allocation (CPU, memory, disk) and runs entirely locally, minimizing both financial and hardware overhead.
- Learning Curve: Tools like Kubernetes and Docker, while powerful, introduce significant complexity for beginners. Our platform abstracts much of the infrastructure complexity through declarative configuration, lowering the barrier to entry while still exposing learners to essential IaC and orchestration concepts.
- Network Dependency: Many educational platforms require persistent internet access for cloud resource provisioning or container image downloads.

Our platform operates fully offline once the base image is cached, making it suitable for low-connectivity or isolated lab environments.

In summary, our platform provides a balanced trade-off between educational depth and practical usability, offering a self-contained, reproducible, and low-cost alternative to traditional cloud-based or containerized solutions.

## 6 Discussion

### 6.1 Advantages

- **Educational value:** Covers full cloud/edge stack.
- **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. C. Sonmez et al., EdgeCloudSim: An environment for performance evaluation of Edge Computing systems, *Trans. Emerg. Telecommun. Technol.*, 2018.
2. H. Gupta et al., iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments, *Softw. Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.
3. AWS Educate, Cloud learning resources, Online . Available:<https://aws.amazon.com/education/awseducate>
4. Microsoft, Azure for students, Online . Available:<https://azure.microsoft.com/en-us/free/students/>
5. HashiCorp, Vagrant documentation, Online . Available:<https://www.vagrantup.com/docs>
6. Oracle, VirtualBox user manual, Online . Available:<https://www.virtualbox.org/wiki/Documentation>
7. Docker, Docker documentation, Online . Available:<https://docs.docker.com/>
8. kubernetes, kubernetes documentation, Online . Available:<https://kubernetes.io/docs/home/>
9. W. Shi et al., Edge computing: Vision and challenges, *IEEE Internet Things J.*, v ol. 3, no. 5, pp. 637–646, Oct. 2016.
- 10.M. Satyanarayanan, The emergence of edge computing, *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 201