Integration Guide - Multi-Cloud Manager

This guide covers how to integrate the Multi-Cloud Manager droplet with other systems and droplets in the Full Potential mesh.

Table of Contents

---

1. Registry Integration

Overview

The Multi-Cloud Manager integrates with the Registry using JWT authentication for secure communication.

Authentication Flow

```mermaid
sequenceDiagram
    participant MCM as Multi-Cloud Manager
    participant Registry as Registry

    MCM->>Registry: POST /auth/token (with Registry Key)
    Registry-->>MCM: JWT Token (HS256)

    MCM->>Registry: POST /registry/register (with JWT)
    Registry-->>MCM: Registration Confirmed

    loop Every 30 seconds
        MCM->>Registry: POST /registry/heartbeat (with JWT)
        Registry-->>MCM: OK
    end
```

Registration Payload

json

```json
{
  "id": "drop4.fullpotential.ai",
  "host": "drop4.fullpotential.ai",
  "droplet_id": "drop4.fullpotential.ai",
  "ip": "0.0.0.0",
  "status": "active",
  "metadata": {
    "numeric_id": 4,
    "name": "Multi-Cloud Manager",
    "steward": "Hassan",
    "version": "1.0.0",
    "udc_version": "1.0",
    "capabilities": [
      "multi-cloud-management",
      "digitalocean",
      "hetzner",
      "vultr"
    ]
  }
}
```

Heartbeat Payload

json

```json
{
  "id": "drop4.fullpotential.ai",
  "host": "drop4.fullpotential.ai",
  "droplet_id": "drop4.fullpotential.ai",
  "status": "healthy",
  "load": 0.15,
  "metadata": {
    "cpu_percent": 12.5,
    "memory_mb": 5394.10,
    "requests_last_minute": 45,
    "errors_last_minute": 0,
    "uptime_seconds": 3600
  }
}
```

---

2. Authentication

For External Clients

Simple Bearer Token (Development/Testing):

```bash
curl -H "Authorization: Bearer secretkey_7f8b4e2a9d034b5cb7219d6f81e3d2c1" \
  http://localhost:8010/logs
```

For Inter-Droplet Communication

JWT Token (Production):

```bash
# Step 1: Get JWT from Registry
TOKEN=$(curl -s -X POST \
  "https://drop18.fullpotential.ai/auth/token?droplet_id=YOUR_DROPLET" \
  -H "X-Registry-Key: YOUR_REGISTRY_KEY" \
  | jq -r '.token')

# Step 2: Call Multi-Cloud Manager
curl -H "Authorization: Bearer $TOKEN" \
  http://drop4.fullpotential.ai/do/list
```

JWKS Verification

When other droplets call this droplet, tokens are verified using:

1. Extract `kid` (key ID) from token header
2. Fetch public keys from Registry JWKS endpoint
3. Verify token signature with matching public key
4. Validate issuer, audience, and expiration

Fallback: If JWKS unavailable, falls back to simple API token validation.

---

3. UDC Message Protocol

Sending a Message to This Droplet

Endpoint: `POST /message`

Payload:

```json
{
  "trace_id": "uuid-v4-here",
  "source": 5,
  "target": 4,
  "message_type": "command",
  "payload": {
    "action": "list_cloud_resources",
    "provider": "digitalocean"
  },
  "timestamp": "2025-11-13T08:00:00Z"
}
```

Response:

```json
{
  "trace_id": "uuid-v4-here",
  "response": "executing",
  "data": {
    "accepted": true
  }
}
```

Sending a Message FROM This Droplet

Endpoint: `POST /send`

Payload:

```json
```

```
{
  "target_droplet_id": 2,
  "message_type": "event",
  "payload": {
    "event": "cloud_resource_created",
    "provider": "digitalocean",
    "resource_id": "12345",
    "resource_type": "droplet"
  }
}
```

Process:

1. Looks up target droplet in Registry
2. Gets target endpoint URL
3. Sends UDC message to target's `/message` endpoint
4. Returns confirmation with trace ID

---

4. Cloud Provider Setup

DigitalOcean

Get API Token:

1. Go to https://cloud.digitalocean.com/account/api/tokens
2. Generate new token with read/write access
3. Add to `.env`:

```bash
DO_TOKEN=dop_v1_your_token_here
```

Test Connection:

```bash
curl -H "Authorization: Bearer YOUR_API_TOKEN" \
  http://localhost:8010/do/list
```

Hetzner

Get API Token:

1. Go to https://console.hetzner.cloud
2. Select project → Security → API Tokens
3. Generate token with read/write access
4. Add to `.env`:

```bash
HETZNER_TOKEN=your_hetzner_token_here
```

Test Connection:

```bash
curl -H "Authorization: Bearer YOUR_API_TOKEN" \
  http://localhost:8010/hetzner/list
```

Vultr

Get API Token:

1. Go to https://my.vultr.com/settings/#settingsapi
2. Generate new API key
3. Add to `.env`:

```bash
VULTR_TOKEN=your_vultr_token_here
```

Test Connection:

```bash
curl -H "Authorization: Bearer YOUR_API_TOKEN" \
  http://localhost:8010/vultr/list
```

---

5. Calling This Droplet

From Another Droplet

python

```python
import httpx
import asyncio

async def call_multi_cloud_manager():
    # Assume you have a JWT token from Registry
    token = "your_jwt_token"

    async with httpx.AsyncClient() as client:
        # List all DigitalOcean droplets
        response = await client.get(
            "https://drop4.fullpotential.ai/do/list",
            headers={"Authorization": f"Bearer {token}"}
        )

        if response.status_code == 200:
            data = response.json()
            print(f"Found {data['count']} droplets")
            return data['droplets']
        else:
            print(f"Error: {response.status_code}")
            return None

# Run
asyncio.run(call_multi_cloud_manager())
```

From Dashboard/Frontend

javascript

```javascript
// JavaScript example
async function listCloudResources(provider) {
  const token = localStorage.getItem('api_token');

  const response = await fetch(
    `https://drop4.fullpotential.ai/${provider}/list`,
    {
      headers: {
        'Authorization': `Bearer ${token}`
      }
    }
  );

  if (response.ok) {
    const data = await response.json();
    return data;
  } else {
    console.error('Failed to fetch resources');
    return null;
  }
}

// Use
listCloudResources('do').then(data => {
  console.log('DigitalOcean resources:', data);
});
```

Using the Multi-Cloud Endpoint

```bash
bash

# Get all resources from all providers
curl -H "Authorization: Bearer YOUR_TOKEN" \
  http://drop4.fullpotential.ai/multi/list
```

Response:

```json
json
```

```
{
  "do": [
    {"id": "123", "name": "web-server-1", "status": "active"},
    {"id": "456", "name": "db-server-1", "status": "active"}
  ],
  "hetzner": [
    {"id": "789", "name": "app-server-1", "status": "running"}
  ],
  "vultr": [
    {"id": "321", "name": "cache-server-1", "status": "active"}
  ]
}
```

---

6. Error Handling

Standard Error Response

```json
{
  "detail": "Error message here"
}
```

Common Error Codes

Code	Meaning	Solution
401	Unauthorized	Check your Bearer token
403	Forbidden	Token lacks required scope
404	Not Found	Endpoint or resource doesn't exist
422	Validation Error	Check request payload format
500	Internal Error	Check logs, may be provider API issue
503	Service Unavailable	Cloud provider not configured

Retry Logic

For transient errors (500, 503), implement exponential backoff:

```python
```

```
import time

async def retry_request(func, max_retries=3):
    for attempt in range(max_retries):
        try:
            return await func()
        except Exception as e:
            if attempt == max_retries - 1:
                raise
            wait_time = 2 ** attempt  # 1s, 2s, 4s
            await asyncio.sleep(wait_time)
```

Provider-Specific Errors

DigitalOcean:

- Rate limit: 5000 requests/hour
- Error: `429 Too Many Requests`
- Solution: Implement rate limiting

Hetzner:

- Rate limit: Varies by endpoint
- Error: `429 Too Many Requests`
- Solution: Cache results where possible

Vultr:

- Rate limit: 30 requests/second
- Error: `503 Service Unavailable`
- Solution: Add delays between bulk operations

Environment Variables Reference

```bash
```

```
# Application
API_TOKEN=your_api_token_here
PORT=8010

# Droplet Identity
DROPLET_ID=4
DROPLET_NAME=Multi-Cloud Manager
DROPLET_DOMAIN=drop4.fullpotential.ai
STEWARD=YourName

# Registry Integration
REGISTRY_BASE_URL=https://drop18.fullpotential.ai
REGISTRY_KEY=your_registry_key_here
HEARTBEAT_INTERVAL=30

# JWT Configuration (Incoming Auth)
JWT_ISSUER=https://drop18.fullpotential.ai
JWKS_URL=https://drop18.fullpotential.ai/.well-known/jwks.json
JWT_AUDIENCE=fullpotential-mesh
JWT_ALGORITHM=RS256

# Cloud Providers
DO_TOKEN=your_digitalocean_token
HETZNER_TOKEN=your_hetzner_token
VULTR_TOKEN=your_vultr_token
```

Integration Checklist

Before going live:

- [ ] Registry key configured
- [ ] JWT authentication tested
- [ ] All cloud provider tokens added
- [ ] Health endpoint returns 200
- [ ] Heartbeat running (check logs)
- [ ] Test message sending/receiving
- [ ] Error handling verified
- [ ] Monitoring/logging confirmed
- [ ] Documentation reviewed
- [ ] Load testing completed

Support

For integration issues:

- Check logs: `docker logs -f do-multi`
- Review troubleshooting guide
- Test with `/health` and `/capabilities` endpoints
- Verify environment variables are set correctly

Contact: @Hassan (Steward)