Step-by-Step Plan for Execution in EXCRUCIATING Detail

Day 1: Server Setup and Basic Infrastructure

Step 1: Create DigitalOcean Account (30 minutes)

Open web browser and navigate to digitalocean.com

Click "Sign Up" in the top right corner

Enter your email address and create a password

Verify your email by clicking the link in the verification email

Log in to your new account

Add a payment method (credit card or PayPal)

Navigate to the Billing section to verify payment method is active

Step 2: Create Droplet (30 minutes)

Click "Create" button in the top navigation

Select "Droplets" from the dropdown menu

Choose "Ubuntu 20.04 (LTS) x64" as the operating system

Select "Basic" plan

Choose the $20/month option (4GB RAM/2 CPUs)

Select a datacenter region closest to you

Under "Authentication", select "Password"

Create a strong password and note it securely

Leave all other options at default

Click "Create Droplet" button

Wait for droplet creation (approximately 1 minute)

Note the IP address of your new droplet

Step 3: Connect to Server (15 minutes)

If on Windows, download and install PuTTY

If on Mac/Linux, open Terminal

For Windows: Open PuTTY and enter your droplet's IP in the "Host Name" field

For Mac/Linux: Type ssh root@your\_droplet\_ip in Terminal

Enter the password you created

Accept any security prompts

Step 4: Update System and Install Dependencies (45 minutes)

Run: apt update

Run: apt upgrade -y

Run: apt install -y python3-pip python3-dev build-essential libssl-dev libffi-dev python3-setuptools python3-venv git nginx

Run: apt install -y postgresql postgresql-contrib

Run: apt install -y redis-server

Run: apt install -y supervisor

Run: apt install -y chromium-browser

Run: apt install -y xvfb

Step 5: Set Up Project Directory (15 minutes)

Run: mkdir -p /opt/twitter\_manager

Run: cd /opt/twitter\_manager

Run: python3 -m venv venv

Run: source venv/bin/activate

Run: pip install wheel

Step 6: Install Python Dependencies (30 minutes)

Run: pip install flask flask-sqlalchemy flask-migrate psycopg2-binary gunicorn selenium requests tweepy beautifulsoup4 pandas numpy schedule apscheduler redis rq cryptography

Run: pip install webdriver-manager fake-useragent

Step 7: Set Up PostgreSQL Database (45 minutes)

Run: sudo -u postgres psql

In PostgreSQL prompt, run: CREATE DATABASE twitter\_manager;

Run: CREATE USER twitter\_user WITH PASSWORD 'secure\_password';

Run: GRANT ALL PRIVILEGES ON DATABASE twitter\_manager TO twitter\_user;

Run: \q to exit PostgreSQL prompt

Test connection: psql -U twitter\_user -d twitter\_manager -h localhost -W

Enter the password when prompted

Day 2: Core Application Development

Step 8: Create Basic Project Structure (30 minutes)

Run: mkdir -p /opt/twitter\_manager/app

Run: mkdir -p /opt/twitter\_manager/app/models

Run: mkdir -p /opt/twitter\_manager/app/controllers

Run: mkdir -p /opt/twitter\_manager/app/services

Run: mkdir -p /opt/twitter\_manager/app/utils

Run: mkdir -p /opt/twitter\_manager/app/templates

Run: mkdir -p /opt/twitter\_manager/app/static

Run: mkdir -p /opt/twitter\_manager/logs

Run: touch /opt/twitter\_manager/app/\_\_init\_\_.py

Run: touch /opt/twitter\_manager/app/models/\_\_init\_\_.py

Run: touch /opt/twitter\_manager/app/controllers/\_\_init\_\_.py

Run: touch /opt/twitter\_manager/app/services/\_\_init\_\_.py

Run: touch /opt/twitter\_manager/app/utils/\_\_init\_\_.py

Step 9: Create Database Models (1 hour)

Create account model: nano /opt/twitter\_manager/app/models/account.py

Paste the Account model code I'll provide

Create content model: nano /opt/twitter\_manager/app/models/content.py

Paste the Content model code I'll provide

Create engagement model: nano /opt/twitter\_manager/app/models/engagement.py

Paste the Engagement model code I'll provide

Create analytics model: nano /opt/twitter\_manager/app/models/analytics.py

Paste the Analytics model code I'll provide

Create rules model: nano /opt/twitter\_manager/app/models/rule.py

Paste the Rule model code I'll provide

Update models/init.py to import all models

Step 10: Create Authentication Service (2 hours)

Create auth service: nano /opt/twitter\_manager/app/services/auth\_service.py

Paste the authentication service code I'll provide

Create credential encryption: nano /opt/twitter\_manager/app/utils/encryption.py

Paste the encryption utility code I'll provide

Create session manager: nano /opt/twitter\_manager/app/services/session\_manager.py

Paste the session management code I'll provide

Create verification handler: nano /opt/twitter\_manager/app/services/verification\_handler.py

Paste the verification handling code I'll provide

Step 11: Create Twitter Interaction Service (2 hours)

Create Twitter service: nano /opt/twitter\_manager/app/services/twitter\_service.py

Paste the Twitter interaction code I'll provide

Create content poster: nano /opt/twitter\_manager/app/services/content\_poster.py

Paste the content posting code I'll provide

Create engagement monitor: nano /opt/twitter\_manager/app/services/engagement\_monitor.py

Paste the engagement monitoring code I'll provide

Create DM handler: nano /opt/twitter\_manager/app/services/dm\_handler.py

Paste the DM handling code I'll provide

Step 12: Create Analytics Service (2 hours)

Create analytics service: nano /opt/twitter\_manager/app/services/analytics\_service.py

Paste the analytics service code I'll provide

Create data collector: nano /opt/twitter\_manager/app/services/data\_collector.py

Paste the data collection code I'll provide

Create metrics calculator: nano /opt/twitter\_manager/app/services/metrics\_calculator.py

Paste the metrics calculation code I'll provide

Create reporting engine: nano /opt/twitter\_manager/app/services/reporting\_engine.py

Paste the reporting engine code I'll provide

Day 3: Bridge Application and Scheduler

Step 13: Create Bridge Application (3 hours)

Create main bridge app: nano /opt/twitter\_manager/app/bridge\_app.py

Paste the bridge application code I'll provide

Create API endpoints: nano /opt/twitter\_manager/app/controllers/api\_controller.py

Paste the API controller code I'll provide

Create instruction parser: nano /opt/twitter\_manager/app/utils/instruction\_parser.py

Paste the instruction parsing code I'll provide

Create action executor: nano /opt/twitter\_manager/app/services/action\_executor.py

Paste the action execution code I'll provide

Step 14: Create Scheduler System (2 hours)

Create scheduler service: nano /opt/twitter\_manager/app/services/scheduler\_service.py

Paste the scheduler service code I'll provide

Create job definitions: nano /opt/twitter\_manager/app/utils/job\_definitions.py

Paste the job definitions code I'll provide

Create task queue: nano /opt/twitter\_manager/app/services/task\_queue.py

Paste the task queue code I'll provide

Create worker process: nano /opt/twitter\_manager/app/services/worker.py

Paste the worker process code I'll provide

Step 15: Create Rule Engine (2 hours)

Create rule engine: nano /opt/twitter\_manager/app/services/rule\_engine.py

Paste the rule engine code I'll provide

Create trigger detector: nano /opt/twitter\_manager/app/services/trigger\_detector.py

Paste the trigger detection code I'll provide

Create action framework: nano /opt/twitter\_manager/app/services/action\_framework.py

Paste the action framework code I'll provide

Create condition evaluator: nano /opt/twitter\_manager/app/utils/condition\_evaluator.py

Paste the condition evaluation code I'll provide

Step 16: Create Web Dashboard (2 hours)

Create Flask app: nano /opt/twitter\_manager/app/\_\_init\_\_.py

Paste the Flask application code I'll provide

Create dashboard controller: nano /opt/twitter\_manager/app/controllers/dashboard\_controller.py

Paste the dashboard controller code I'll provide

Create account controller: nano /opt/twitter\_manager/app/controllers/account\_controller.py

Paste the account controller code I'll provide

Create templates and static files (HTML, CSS, JS)

Create main application entry: nano /opt/twitter\_manager/run.py

Paste the application entry code I'll provide

Day 4: Integration, Testing, and Deployment

Step 17: Set Up Account Configuration (1 hour)

Create account setup script: nano /opt/twitter\_manager/setup\_accounts.py

Paste the account setup code I'll provide

Create account configuration file: nano /opt/twitter\_manager/accounts.json

Paste the account configuration template I'll provide

Fill in the account details for all 25 ecommerce course-seller accounts

Run the setup script: python setup\_accounts.py

Step 18: Create Content Templates (1 hour)

Create template setup script: nano /opt/twitter\_manager/setup\_templates.py

Paste the template setup code I'll provide

Create template configuration file: nano /opt/twitter\_manager/templates.json

Paste the template configuration for ecommerce course-seller accounts

Run the template setup script: python setup\_templates.py

Step 19: Set Up Automation Rules (1 hour)

Create rule setup script: nano /opt/twitter\_manager/setup\_rules.py

Paste the rule setup code I'll provide

Create rule configuration file: nano /opt/twitter\_manager/rules.json

Paste the rule configuration for ecommerce course-seller accounts

Run the rule setup script: python setup\_rules.py

Step 20: Configure System Services (1 hour)

Create Supervisor configuration: nano /etc/supervisor/conf.d/twitter\_manager.conf

Paste the Supervisor configuration I'll provide

Create Nginx configuration: nano /etc/nginx/sites-available/twitter\_manager

Paste the Nginx configuration I'll provide

Enable the Nginx site: ln -s /etc/nginx/sites-available/twitter\_manager /etc/nginx/sites-enabled

Test Nginx configuration: nginx -t

Restart Nginx: systemctl restart nginx

Update Supervisor: supervisorctl reread && supervisorctl update

Step 21: Test Single Account (2 hours)

Create test script: nano /opt/twitter\_manager/test\_account.py

Paste the test script code I'll provide

Run the test script for one account: python test\_account.py --account\_id 1

Monitor logs for any errors: tail -f /opt/twitter\_manager/logs/app.log

Fix any issues that arise

Verify successful posting, engagement, and analytics collection

Step 22: Deploy Full System (2 hours)

Start all system services: supervisorctl start all

Monitor system startup: supervisorctl status

Check application logs: tail -f /opt/twitter\_manager/logs/app.log

Verify all 25 accounts are properly authenticated

Confirm scheduler is running: ps aux | grep scheduler

Test API endpoint: curl http://localhost:5000/api/status

Access web dashboard: http://your\_droplet\_ip:5000

Step 23: Final Configuration and Launch (1 hour)

Set up email notifications: nano /opt/twitter\_manager/app/utils/notifier.py

Paste the notification code I'll provide

Configure backup system: nano /opt/twitter\_manager/backup.sh

Paste the backup script I'll provide

Set up cron job for backups: crontab -e

Add backup schedule: 0 0 \* \* \* /opt/twitter\_manager/backup.sh

Perform final system check: python /opt/twitter\_manager/system\_check.py

Launch the complete system: supervisorctl restart all

Missing Context for Next Chat

When starting the implementation chat, please note:

We're building a hybrid system that uses Manus for strategic control while external infrastructure handles autonomous operations to bypass Manus restrictions.

The system is designed for 25 ecommerce course-seller themed Twitter accounts, not French-themed accounts as previously discussed.

The total budget is under $100/month, with estimated costs of $20-45/month.

The goal is a fully autonomous system that posts content, replies to comments and DMs, analyzes engagement, and implements custom automation rules.

You have no prior command line or programming experience, so all instructions will be provided in excruciating detail.

We're using a DigitalOcean droplet as the server infrastructure.

The system uses a bridge application to connect Manus conversations to the autonomous infrastructure.

All 25 accounts need to be operational by Saturday night.