Automated Phone Answering System - Deployment Guide

Table of Contents

Prerequisites

Required Accounts and Services

- **Vonage Voice API Account** with active application and phone number
- **Google Cloud Platform Account** with Calendar API enabled
- **Domain name** (for production deployment with HTTPS)
- **Server or VPS** with public IP address
- **SSL Certificate** (recommended: Let's Encrypt)

Technical Requirements

- Python 3.8 or higher
- Git
- Web server (Nginx recommended for production)
- Process manager (systemd, PM2, or supervisor)
- Reverse proxy capability

System Requirements

Minimum Hardware

- **CPU**: 2 cores
- **RAM**: 4GB
- **Storage**: 10GB available space
- **Network**: Stable internet connection with low latency

Recommended Hardware

- **CPU**: 4 cores
- **RAM**: 8GB
- **Storage**: 20GB available space
- **Network**: High-speed internet with redundant connectivity

Initial Setup

1. Clone Repository

```
# Clone the repository
git clone <your-repo-url>
cd auto_call_system

# Verify all files are present
ls -la
```

2. Create Python Virtual Environment

```
# Create virtual environment
python3 -m venv venv

# Activate virtual environment
source venv/bin/activate

# Upgrade pip
pip install --upgrade pip
```

3. Install Dependencies

```
# Install required Python packages
pip install -r requirements.txt

# Verify installation
python -c "import flask, vonage, google.auth; print('All dependencies installed suc-
cessfully')"
```

Service Configuration

1. Vonage Voice API Setup

Step 1: Create Vonage Application

1. Log in to Vonage API Dashboard (https://dashboard.nexmo.com)
2. Navigate to “Your Applications”
3. Click “Create a new application”
4. Configure application:
   - **Name**: `Auto Call System`
   - **Capabilities**: Enable “Voice”
   - **Answer URL**: `https://yourdomain.com/webhooks/answer`
   - **Event URL**: `https://yourdomain.com/webhooks/events`
5. Download the private key file and save as `private.key`

Step 2: Purchase Phone Number

1. In Vonage Dashboard, go to “Numbers” → “Buy Numbers”
2. Select your country and purchase a number with Voice capability
3. Link the number to your application

Step 3: Configure Application Settings

```
# Place the private key in the project directory
cp /path/to/downloaded/private.key ./private.key
chmod 600 private.key
```

2. Google Calendar API Setup

Step 1: Create Google Cloud Project

1. Go to Google Cloud Console (https://console.cloud.google.com)
2. Create new project or select existing one
3. Enable Calendar API:
   - Navigate to "APIs & Services" → "Library"
   - Search for "Google Calendar API"
   - Click "Enable"

Step 2: Create Service Account

1. Go to "APIs & Services" → "Credentials"
2. Click "Create Credentials" → "Service Account"
3. Fill details:
   - **Name**: `auto-call-system-calendar`
   - **Description**: `Service account for automated call system calendar access`
4. Grant roles:
   - `Calendar API Editor` (if custom role exists)
   - Or `Editor` role for broader access

Step 3: Generate and Download Credentials

1. Click on created service account
2. Go to "Keys" tab
3. Click "Add Key" → "Create new key"
4. Choose JSON format
5. Download and save as `credentials.json`

Step 4: Share Calendar with Service Account

1. Open Google Calendar
2. Create a new calendar or use existing one
3. Go to calendar settings → "Share with specific people"
4. Add service account email (from credentials.json)
5. Grant "Make changes and manage sharing" permission

3. Environment Configuration

Create Environment File

```
# Copy example environment file
cp .env.example .env

# Edit with your actual values
nano .env
```

Configure Environment Variables

```
# Vonage Voice API Configuration
VONAGE_API_KEY=your_actual_api_key
VONAGE_API_SECRET=your_actual_secret
VONAGE_APPLICATION_ID=your_application_uuid
VONAGE_PRIVATE_KEY_PATH=./private.key
VONAGE_PHONE_NUMBER=+1234567890

# Staff Configuration (for escalations)
STAFF_PHONE_NUMBER=+15551234567

# Google Calendar Configuration
GOOGLE_CALENDAR_ID=calendar_id@group.calendar.google.com
GOOGLE_CREDENTIALS_PATH=./credentials.json
GOOGLE_TOKEN_PATH=./token.json

# Application Configuration
FLASK_ENV=production
PORT=5000
BASE_URL=https://yourdomain.com

# Optional Configuration
HOLD_MUSIC_URL=https://yourdomain.com/static/hold-music.mp3
ESCALATION_LOG_FILE=/var/log/auto-call-system/escalations.log
CALLBACK_LOG_FILE=/var/log/auto-call-system/callbacks.log

# Facility Configuration
FACILITY_TIMEZONE=America/New_York
BUSINESS_HOURS_START=9
BUSINESS_HOURS_END=21
```

Deployment Steps

1. Production Server Setup

Install System Dependencies (Ubuntu/Debian)

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install required packages
sudo apt install -y python3 python3-pip python3-venv nginx supervisor git

# Install certbot for SSL (optional but recommended)
sudo apt install -y certbot python3-certbot-nginx
```

2. Application Deployment

Upload Application Files

```
# Transfer files to server (example using scp)
scp -r auto_call_system/ user@your-server:/opt/
ssh user@your-server

# Set proper ownership and permissions
sudo chown -R www-data:www-data /opt/auto_call_system
sudo chmod 755 /opt/auto_call_system
sudo chmod 600 /opt/auto_call_system/.env
sudo chmod 600 /opt/auto_call_system/private.key
sudo chmod 600 /opt/auto_call_system/credentials.json
```

Setup Python Environment on Server

```
cd /opt/auto_call_system

# Create virtual environment
python3 -m venv venv

# Install dependencies
venv/bin/pip install -r requirements.txt
```

3. Process Management Setup

Create Systemd Service

```
sudo nano /etc/systemd/system/auto-call-system.service
```

Service Configuration

```
[Unit]
Description=Automated Phone Answering System
After=network.target

[Service]
Type=simple
User=www-data
Group=www-data
WorkingDirectory=/opt/auto_call_system
Environment=PATH=/opt/auto_call_system/venv/bin
ExecStart=/opt/auto_call_system/venv/bin/python app.py
Restart=always
RestartSec=10

# Logging
StandardOutput=journal
StandardError=journal
SyslogIdentifier=auto-call-system

# Security
NoNewPrivileges=true
ProtectSystem=strict
ProtectHome=true
ReadWritePaths=/opt/auto_call_system /var/log/auto-call-system

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

Enable and Start Service

```
# Reload systemd
sudo systemctl daemon-reload

# Enable service to start on boot
sudo systemctl enable auto-call-system

# Start service
sudo systemctl start auto-call-system

# Check status
sudo systemctl status auto-call-system
```

4. Reverse Proxy Setup (Nginx)

Create Nginx Configuration

```
sudo nano /etc/nginx/sites-available/auto-call-system
```

Nginx Configuration

```
server {
    listen 80;
    server_name yourdomain.com;

    # Redirect HTTP to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl http2;
    server_name yourdomain.com;

    # SSL Configuration (after obtaining certificate)
    ssl_certificate /etc/letsencrypt/live/yourdomain.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/yourdomain.com/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-
AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;

    # Security headers
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";

    # Main application
    location / {
        proxy_pass http://127.0.0.1:5000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        # Timeouts for long-running calls
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Static files (if any)
    location /static/ {
        alias /opt/auto_call_system/static/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # Health check endpoint
    location /health {
        proxy_pass http://127.0.0.1:5000/health;
        access_log off;
    }
}
```

Enable Site and Reload Nginx

```
# Enable site
sudo ln -s /etc/nginx/sites-available/auto-call-system /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# Reload nginx
sudo systemctl reload nginx
```

Post-Deployment Verification

1. Service Health Checks

Check Application Status

```
# Check systemd service
sudo systemctl status auto-call-system

# Check application logs
sudo journalctl -u auto-call-system -f

# Check nginx status
sudo systemctl status nginx

# Verify port binding
sudo netstat -tlpn | grep :5000
```

Test HTTP Endpoints

```
# Test application health
curl -I http://localhost:5000/health

# Test webhook endpoint (should return 405 Method Not Allowed for GET)
curl -I https://yourdomain.com/webhooks/answer
```

2. Integration Testing

Test Vonage Integration

1. Make a test call to your Vonage number
2. Verify call is answered with greeting message
3. Check application logs for webhook requests
4. Test speech recognition by speaking test phrases

Test Google Calendar Integration

```
# Test calendar access from application directory
cd /opt/auto_call_system
venv/bin/python -c "
from calendar_helper import CalendarHelper
helper = CalendarHelper()
print('Calendar integration:', 'OK' if helper.check_availability() else 'FAILED')
"
```

3. End-to-End Testing

Complete Call Flow Test

1. **Make test call** to your Vonage number
2. **Listen to greeting** - should hear welcome message
3. **Test pricing inquiry** - ask “How much does it cost?”
4. **Test availability check** - ask “Are you available tomorrow?”
5. **Test escalation** - ask about “payment issues”
6. **Verify logging** - check logs for all interactions

SSL/HTTPS Setup

Using Let’s Encrypt (Recommended)

Obtain SSL Certificate

```
# Stop nginx temporarily
sudo systemctl stop nginx

# Obtain certificate
sudo certbot certonly --standalone -d yourdomain.com

# Start nginx
sudo systemctl start nginx

# Setup auto-renewal
sudo certbot renew --dry-run
```

Update Vonage Webhook URLs

1. Go to Vonage API Dashboard
2. Edit your application
3. Update URLs to use HTTPS:
   - Answer URL: `https://yourdomain.com/webhooks/answer`
   - Event URL: `https://yourdomain.com/webhooks/events`

Production Considerations

1. Security Hardening

Firewall Configuration

```
# Enable UFW
sudo ufw enable

# Allow SSH, HTTP, and HTTPS
sudo ufw allow 22/tcp
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# Deny all other incoming traffic
sudo ufw default deny incoming
sudo ufw default allow outgoing
```

File Permissions Audit

```
# Ensure sensitive files have proper permissions
sudo chmod 600 /opt/auto_call_system/.env
sudo chmod 600 /opt/auto_call_system/private.key
sudo chmod 600 /opt/auto_call_system/credentials.json
```

2. Monitoring and Logging

Log Rotation Setup

```
sudo nano /etc/logrotate.d/auto-call-system
```

```
/var/log/auto-call-system/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 www-data www-data
    postrotate
        systemctl reload auto-call-system > /dev/null 2>&1 || true
    endscript
}
```

Create Log Directories

```
sudo mkdir -p /var/log/auto-call-system
sudo chown www-data:www-data /var/log/auto-call-system
```

3. Performance Optimization

Application-Level Caching

- Consider implementing Redis for session storage
- Cache frequently accessed pricing data
- Implement request rate limiting

System-Level Optimization

```
# Increase file descriptor limits
sudo nano /etc/security/limits.conf
```

Add:

```
www-data soft nofile 65536
www-data hard nofile 65536
```

4. Backup Strategy

Automated Backup Script

```
#!/bin/bash
# /opt/scripts/backup-auto-call-system.sh

BACKUP_DIR="/opt/backups/auto-call-system"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p $BACKUP_DIR

# Backup application files
tar -czf "$BACKUP_DIR/app_$DATE.tar.gz" \
    --exclude="venv" \
    --exclude="__pycache__" \
    --exclude="*.pyc" \
    /opt/auto_call_system/

# Backup logs
tar -czf "$BACKUP_DIR/logs_$DATE.tar.gz" /var/log/auto-call-system/

# Keep only last 30 days of backups
find $BACKUP_DIR -name "*.tar.gz" -mtime +30 -delete

echo "Backup completed: $DATE"
```

Schedule Backups

```
# Add to crontab
sudo crontab -e
```

Add:

```
# Backup auto-call-system daily at 2 AM
0 2 * * * /opt/scripts/backup-auto-call-system.sh
```

Next Steps

After completing deployment:

1. **Review Configuration Guide** - Customize responses and pricing
2. **Read Testing Guide** - Perform comprehensive testing
3. **Setup Monitoring** - Implement health checks and alerting
4. **Review Maintenance Guide** - Plan for ongoing maintenance

---

Deployment Complete! 🎉

Your automated phone answering system should now be running and ready to handle calls. Monitor the logs and perform regular testing to ensure optimal performance.