Accompany

INSY7315

MO4 & MO5:

Bot Systems Manual

Group Members:

Abdul Baari Davids - ST10267411
Ethan Smyth - ST10255309
Ethan Donaldson - ST10318621
Jacques du Plessis - ST10329686
Joshua Wood (PM) - ST102961671

Table of Contents

1. System Overview

1.1 Purpose

The ZATech Slack Bot v3 is an async event-driven bot that provides:

- **Automated Greetings**: Welcome new members in #introductions
- **Pattern-Based Auto-Responses**: Automated responses to specific message patterns
- **Moderation Logging**: Track message edits and deletions
- **Extensible Plugin System**: Add new features without modifying core code
- **Admin Dashboard**: Web-based configuration interface

1.2 Key Components

Component	Purpose	Technology
FastAPI	HTTP server for admin dashboard	Python async web framework
Slack Bolt	Slack event handling	Official Slack SDK
Socket Mode	Receive Slack events via WebSocket	Persistent connection
Plugin Manager	Dynamic plugin loading & lifecycle	Python module system
Storage	Persist plugin configuration	SQLite/PostgreSQL
Admin Dashboard	Web UI for configuration	Jinja2 templates
Docker	Containerization	Docker + Docker Compose
Nginx	Reverse proxy & TLS termination	Web server

1.3 System Requirements

VPS Specifications (minimum):

- **CPU**: 2 vCPUs
- **RAM**: 2 GB (4 GB recommended for 25,000+ users)
- **Storage**: 20 GB SSD
- **OS**: Ubuntu 22.04 LTS or newer
- **Network**: 1 Gbps, 20 TB/month transfer

Software Requirements:

- Docker 20.10+
- Docker Compose 1.29+
- Git
- Nginx 1.18+
- Python 3.11+ (if running without Docker)

---

2. Installation & Deployment

2.1 Initial VPS Setup

Step 1: Provision VPS

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

# Install required packages
sudo apt install -y docker.io docker-compose nginx git ufw

# Start and enable Docker
sudo systemctl start docker
sudo systemctl enable docker

# Add user to docker group (replace 'ubuntu' with your username)
sudo usermod -aG docker ubuntu
newgrp docker
```

Step 2: Configure Firewall

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

# Enable firewall
sudo ufw enable
sudo ufw status
```

Step 3: Clone Repository

```
# Clone the bot repository
git clone https://github.com/zatech/zatech-bot.git
cd zatech-bot
```

2.2 Slack App Configuration

Step 1: Create Slack App

1. Visit https://api.slack.com/apps
2. Click **Create New App** → **From scratch**
3. Name: `ZATech Bot v3`
4. Workspace: Select your workspace

Step 2: Configure OAuth Scopes

Navigate to **OAuth & Permissions** and add the following **Bot Token Scopes**:

- `app_mentions:read` - Read messages mentioning the bot
- `channels:history` - View messages in public channels
- `channels:read` - View basic channel info
- `chat:write` - Send messages
- `users:read` - View user information
- `reactions:read` - View emoji reactions

Step 3: Enable Socket Mode

1. Navigate to **Socket Mode** in sidebar
2. Enable Socket Mode
3. Generate App-Level Token:
    - Name: `zatech-bot-socket`
    - Scope: `connections:write`
    - Copy token (starts with `xapp-`)

Step 4: Install App to Workspace

1. Navigate to **Install App**
2. Click **Install to Workspace**
3. Authorize the app
4. Copy **Bot User OAuth Token** (starts with `xoxb-`)

2.3 Environment Configuration

Create `.env` file:

```
cp .env.example .env
nano .env
```

Required environment variables:

```
# Slack Tokens (REQUIRED)
SLACK_BOT_TOKEN=xoxb-your-bot-token-here
SLACK_APP_TOKEN=xapp-your-app-token-here

# Database (Choose one)
# For SQLite (development/small deployments):
DATABASE_URL=sqlite+aiosqlite:///./bot.db

# For PostgreSQL (production):
# DATABASE_URL=postgresql+asyncpg://postgres:postgres@db:5432/zatech_bot

# Logging
LOG_LEVEL=INFO

# Server Configuration
HOST=0.0.0.0
PORT=3000

# Plugin Configuration
PLUGIN_PACKAGES=plugins
ENABLED_PLUGINS=  # Leave empty to use plugin defaults

# Optional: OpenAI API (for AI-powered greetings)
OPENAI_API_KEY=sk-your-openai-key-here  # Optional
```

2.4 Docker Deployment

Option A: Docker Compose with PostgreSQL (Recommended)

```
# Start services
docker compose up -d

# View logs
docker compose logs -f app

# Check status
docker compose ps
```

Services started:

- `app`: Bot application (port 3000 internal)
- `db`: PostgreSQL database (port 5432 internal)

Option B: Docker Standalone (SQLite)

```
# Build image
docker build -t zatech-bot:latest .

# Run container
docker run -d \
  --name zatech-bot \
  --env-file .env \
  -p 3000:3000 \
  -v $(pwd)/bot.db:/app/bot.db \
  zatech-bot:latest

# View logs
docker logs -f zatech-bot
```

2.5 Nginx Reverse Proxy Setup

Step 1: Configure Nginx

```
sudo nano /etc/nginx/sites-available/zatech-bot
```

Add configuration:

```
server {
    listen 80;
    server_name bot.zatech.co.za;

    location / {
        return 301 https://hostrequest_uri;
    }
}

server {
    listen 443 ssl http2;
    server_name bot.zatech.co.za;

    # SSL Configuration (use Let's Encrypt)
    ssl_certificate /etc/letsencrypt/live/bot.zatech.co.za/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/bot.zatech.co.za/privkey.pem;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;

    # Security Headers
    add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains" always;
    add_header X-Frame-Options "DENY" always;
    add_header X-Content-Type-Options "nosniff" always;

    # Rate Limiting
    limit_req_zone $binary_remote_addr zone=admin:10m rate=10r/s;

    location / {
        proxy_pass http://localhost:3000;
        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;

        # Rate limiting for admin dashboard
        limit_req zone=admin burst=20 nodelay;
    }

    location /health {
        proxy_pass http://localhost:3000/health;
        access_log off;
    }
}
```

Step 2: Enable Site

```
# Create symbolic link
sudo ln -s /etc/nginx/sites-available/zatech-bot /etc/nginx/sites-enabled/

# Test configuration
sudo nginx -t

# Reload Nginx
sudo systemctl reload nginx
```

Step 3: SSL Certificate (Let's Encrypt)

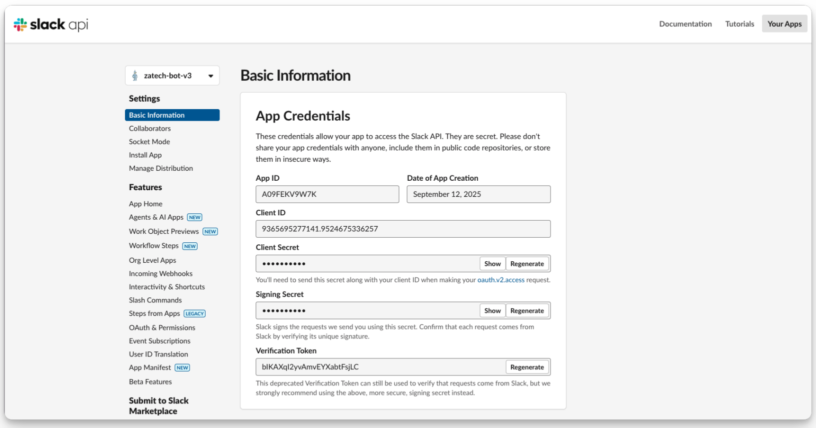
```
# Install Certbot
sudo apt install -y certbot python3-certbot-nginx

# Obtain certificate
sudo certbot --nginx -d bot.zatech.co.za

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

2.6 Slack App Configuration

This section covers how to create, configure, and obtain credentials for your Slack App to enable bot functionality.



Basic Information

App Credentials

These credentials allow your app to access the Slack API. They are secret. Please don't share your app credentials with anyone, include them in public code repositories, or store them in insecure ways.

App ID: A09FEKV9W7K

Date of App Creation: September 12, 2025

Client ID: 9365695277141.9524675336257

Client Secret: ••••••••••

You'll need to send this secret along with your client ID when making your oauth.v2.access request.

Signing Secret: ••••••••••

Slack signs the requests we send you using this secret. Confirm that each request comes from Slack by verifying its unique signature.

Verification Token: bIKAXqI2yvAmvEYXabtFsjLC

This deprecated Verification Token can still be used to verify that requests come from Slack, but we strongly recommend using the above, more secure, signing secret instead.

2.6.1 Creating a Slack App

Prerequisites:

- Slack workspace with admin permissions (or permission to install apps)
- Access to https://api.slack.com/apps

Steps:

1. **Navigate to Slack App Management**
    - Go to https://api.slack.com/apps
    - Click "Create New App"
2. **Choose Creation Method**
    - Select "From scratch" for manual configuration
    - Or select "From an app manifest" if you have a pre-configured YAML/JSON manifest
3. **Basic Information**
    - **App Name**: e.g., "ZATech Bot"
    - **Workspace**: Select your development/production workspace
    - Click "Create App"

2.6.2 Configuring OAuth Scopes

OAuth scopes define what your bot can do in Slack. Navigate to **OAuth & Permissions** in the sidebar.

Required Bot Token Scopes (`xoxb-*`):

Scope	Purpose
`app_mentions:read`	Receive messages that mention the bot
`channels:history`	View messages in public channels
`channels:read`	View basic channel information
`chat:write`	Send messages as the bot
`groups:history`	View messages in private channels (if invited)
`groups:read`	View basic private channel information
`im:history`	View messages in direct messages
`im:read`	View basic DM information
`im:write`	Send direct messages
`reactions:read`	View emoji reactions
`reactions:write`	Add emoji reactions
`users:read`	View user information (names, profiles)
`team:read`	View workspace information

Optional Scopes (depending on features):

- `files:read`: If bot needs to access uploaded files
- `usergroups:read`: If bot needs to work with user groups
- `channels:manage`: If bot creates/archives channels
- `pins:write`: If bot pins messages

2.6.3 Enabling Socket Mode

Socket Mode allows the bot to receive events via WebSocket without exposing a public HTTP endpoint.

Steps:

1. **Navigate to Socket Mode**
   - Sidebar: **Settings** → **Socket Mode**
2. **Enable Socket Mode**
   - Toggle "Enable Socket Mode" to **ON**
3. **Generate App-Level Token**
   - Click "Generate Token and Scopes"
   - **Token Name**: e.g., "Socket Mode Token"
   - **Required Scope**: `connections:write`
   - Click "Generate"
   - **Copy the token** (starts with `xapp-`) - you won't see it again!

2.6.4 Subscribing to Events

Events tell your bot when things happen in Slack. Navigate to **Event Subscriptions**.

Steps:

1. **Enable Events**
    - Toggle "Enable Events" to **ON**
2. **Subscribe to Bot Events**

    Add the events your bot needs to listen for:

Event	Purpose
`app_mention`	Bot is @mentioned in a channel
`message.channels`	Messages posted in public channels
`message.groups`	Messages posted in private channels
`message.im`	Direct messages to the bot
`member_joined_channel`	User joins a channel (for greetings)
`reaction_added`	Emoji reaction added to a message
`reaction_removed`	Emoji reaction removed

3. **Save Changes**
    - Click "Save Changes" at the bottom

2.6.5 Installing the App to Workspace

Steps:

1. **Navigate to Install App**
    - Sidebar: **Settings** → **Install App**
2. **Install to Workspace**
    - Click "Install to Workspace"
    - Review permissions
    - Click "Allow"
3. **Copy Bot User OAuth Token**
    - After installation, you'll see the **Bot User OAuth Token** (starts with `xoxb-`)
    - **Copy this token** - you'll need it for environment variables

2.6.6 Environment Variables Setup

The bot requires two critical tokens to operate. These should be stored in a `.env` file (never committed to version control).

Required Environment Variables:

```
# .env file structure
SLACK_BOT_TOKEN=xoxb-your-bot-token-here
SLACK_APP_TOKEN=xapp-your-app-token-here

# Optional: For AI-powered features
OPENAI_API_KEY=sk-your-openai-key-here

# Database configuration
DATABASE_URL=sqlite:///./bot.db  # Development
# DATABASE_URL=postgresql://user:pass@localhost/dbname  # Production
```

Where to Find Each Token:

1. **SLACK_BOT_TOKEN** (`xoxb-*`)
   - Location: **OAuth & Permissions → Bot User OAuth Token**
   - Format: `xoxb-1234567890-1234567890123-abcdefghijklmnopqrstuvwx`
   - Purpose: Authenticates bot actions (sending messages, reading data)

2. **SLACK_APP_TOKEN** (`xapp-*`)
   - Location: **Basic Information → App-Level Tokens**
   - Format: `xapp-1-ABCDEFGHIJ-1234567890-abcdefghijklmnopqrstuvwxyz1234567890abcdefghijklmnopqrs`
   - Purpose: Authenticates Socket Mode WebSocket connection

3. **OPENAI_API_KEY** (Optional)
   - Get from: https://platform.openai.com/api-keys
   - Format: `sk-proj-...` (varies by key type)
   - Purpose: Powers AI greeting generation in AutoResponder plugin

Security Best Practices:

```
# .gitignore - ALWAYS exclude .env files
.env
.env.local
.env.production
*.env
```

Loading Environment Variables (Python example):

```python
import os
from dotenv import load_dotenv

# Load .env file
load_dotenv()

# Access variables
SLACK_BOT_TOKEN = os.environ.get("SLACK_BOT_TOKEN")
SLACK_APP_TOKEN = os.environ.get("SLACK_APP_TOKEN")

if not SLACK_BOT_TOKEN or not SLACK_APP_TOKEN:
    raise ValueError("Missing required Slack tokens in environment")
```

2.6.7 Verifying Configuration

Checklist:

- ☐ App created in Slack App portal
- ☐ OAuth scopes configured (minimum: `chat:write`, `channels:history`, `app_mentions:read`)
- ☐ Socket Mode enabled with `connections:write` scope
- ☐ App-level token (`xapp-*`) generated and saved
- ☐ Event subscriptions configured (e.g., `message.channels`, `app_mention`)
- ☐ App installed to workspace
- ☐ Bot User OAuth Token (`xoxb-*`) copied
- ☐ Both tokens added to `.env` file
- ☐ `.env` file added to `.gitignore`

Testing Connection:

```python
# Quick test script
from slack_bolt.async_app import AsyncApp
import asyncio

app = AsyncApp(
    token=os.environ.get("SLACK_BOT_TOKEN"),
    app_token=os.environ.get("SLACK_APP_TOKEN")
)

@app.event("app_mention")
async def handle_mention(event, say):
    await say(f"Hello <@{event['user']}>! Bot is working!")

async def main():
    handler = AsyncSocketModeHandler(app, os.environ["SLACK_APP_TOKEN"])
    await handler.start_async()

if __name__ == "__main__":
    asyncio.run(main())
```

If the bot responds to `@botname` mentions in Slack, configuration is successful!

2.6.8 Common Configuration Issues

Issue	**Symptom**	**Solution**
Missing scopes	`missing_scope` API errors	Add required scopes in OAuth & Permissions, reinstall app
Socket Mode disabled	Bot doesn't receive events	Enable Socket Mode and generate app-level token
Wrong token type	`invalid_auth` or `not_authed` errors	Verify using correct token (`xoxb-` for bot, `xapp-` for Socket Mode)
Token not saved	Connection errors on startup	Check `.env` file exists and has correct variable names
Events not subscribed	Bot doesn't respond to messages	Subscribe to required events in Event Subscriptions
Bot not in channel	`channel_not_found` errors	Invite bot to channel: `/invite @botname`

3. Configuration Management

3.1 Environment Variables

Critical Variables (required for operation):

Variable	Purpose	Example	Required
`SLACK_BOT_TOKEN`	Bot authentication	`xoxb-...`	✅ Yes
`SLACK_APP_TOKEN`	Socket Mode connection	`xapp-...`	✅ Yes
`DATABASE_URL`	Database connection	`sqlite+aiosqlite:///./bot.db`	✅ Yes

Optional Variables (for advanced features):

Variable	Purpose	Default	Example
`LOG_LEVEL`	Logging verbosity	`INFO`	`DEBUG`, `WARNING`
`HOST`	Server bind address	`0.0.0.0`	`127.0.0.1`
`PORT`	Server port	`3000`	`8080`
`PLUGIN_PACKAGES`	Plugin directories	`plugins`	`plugins,custom_plugins`
`ENABLED_PLUGINS`	Specific plugins to load	(all)	`autoresponder,modlog`
`OPENAI_API_KEY`	AI-powered greetings	(none)	`sk-...`

3.2 Plugin Configuration

Plugins are configured via the **Admin Dashboard** at `https://bot.zatech.co.za/admin`.

AutoResponder Plugin

Greeting Configuration:

- Navigate to `/admin/tabs/autoresponder_greeter`
- Configure:
  - **Template**: Welcome message (use `{mention}` for user mention)
  - **Channel ID**: Channel to greet in (get from Slack URL)
  - **AI Mode**: Enable OpenAI-powered greetings (requires API key)

Auto-Response Rules:

- Navigate to `/admin/tabs/autoresponder`
- Add rules:
  - **Pattern**: Regular expression (e.g., `(?i)\bhelp\b`)
  - **Response**: Message to send
  - **Enabled**: Toggle on/off

ModLog Plugin

Configuration:

- Navigate to `/admin/tabs/modlog`
- Set **Log Channel ID** (where moderation logs are posted)

3.3 Database Configuration

SQLite (Development/Small Scale)

```
# Connection string
DATABASE_URL=sqlite+aiosqlite:///./bot.db

# Database file location
ls -lh bot.db

# Backup database
```

```
cp bot.db bot.db.backup
```

PostgreSQL (Production)

```
# Connection string format
DATABASE_URL=postgresql+asyncpg://user:password@host:port/database

# Docker Compose example
DATABASE_URL=postgresql+asyncpg://postgres:postgres@db:5432/zatech_bot

# External PostgreSQL example
DATABASE_URL=postgresql+asyncpg://botuser:securepass@postgres.example.com:
5432/zatech
```

---

4. System Operation

4.1 Starting the System

Docker Compose

```
# Start all services
docker compose up -d

# Start only the bot (keep database running)
docker compose up -d app
```

Standalone Docker

```
# Start container
docker start zatech-bot

# Start with logs
docker start zatech-bot && docker logs -f zatech-bot
```

4.2 Stopping the System

Graceful Shutdown

```
# Docker Compose
docker compose down

# Standalone
docker stop zatech-bot
```

Immediate Shutdown (Emergency Only)

```
# Force stop
docker compose kill
```

4.3 Restarting the System

After Configuration Changes

```
# Restart with new environment variables
docker compose down
docker compose up -d

# View logs to verify restart
docker compose logs -f app
```

After Code Updates

```
# Pull latest code
git pull origin main

# Rebuild and restart
docker compose down
docker compose build --no-cache
docker compose up -d
```

4.4 Viewing Logs

Real-Time Logs

```
# All services
docker compose logs -f

# Bot only
docker compose logs -f app

# Database only
docker compose logs -f db
```

Historical Logs

```
# Last 100 lines
docker compose logs app --tail 100

# Logs from specific time
docker compose logs app --since "2025-10-30T10:00:00"
```

Log Levels

Set `LOG_LEVEL` in `.env`:

- `DEBUG`: Verbose logs (event details, plugin execution)
- `INFO`: Normal operations (startup, plugin registration)
- `WARNING`: Potential issues (rate limits, deprecated features)
- `ERROR`: Failures (API errors, database issues)

4.5 Accessing the Admin Dashboard

1. **URL**: `https://bot.zatech.co.za/admin`
2. **Authentication**: Uses Firebase
3. **Features**:
    - Plugin configuration
    - System health status
    - Greeting statistics
    - Auto-response rule management

---

5. Monitoring & Health Checks

5.1 Health Endpoint

5.1 Health Endpoint

HTTP Health Check:

```
# Check application health
curl https://bot.zatech.co.za/health

# Expected response:
{
  "status": "healthy",
  "timestamp": "2025-10-30T12:00:00Z"
}
```

5.2 System Metrics

Docker Container Stats

```
# Real-time resource usage
docker stats zatech-bot

# Expected output:
# CONTAINER    CPU %    MEM USAGE / LIMIT    MEM %
# zatech-bot   5-10%    500MB / 2GB          25%
```

Database Size

```
# SQLite
ls -lh bot.db

# PostgreSQL
docker compose exec db psql -U postgres -d zatech_bot -c "SELECT
pg_size_pretty(pg_database_size('zatech_bot'));"
```

5.3 Monitoring Setup (Free Tier Tools)

UptimeRobot (Uptime Monitoring)

1. Create account at https://uptimerobot.com
2. Add HTTP(S) monitor:
   - **URL**: `https://bot.zatech.co.za/health`
   - **Interval**: 5 minutes
   - **Alert**: Email/SMS when down

Grafana Cloud (Log Aggregation)

```
# Install Promtail for log shipping
wget -qO - https://grafana.com/docs/loki/latest/setup/install/
# Follow Grafana Cloud setup for Loki
```

5.4 Key Metrics to Monitor

Metric	Threshold	Action
CPU Usage	> 80% for 10 min	Investigate high load
RAM Usage	> 90%	Upgrade VPS tier
Disk Usage	> 80%	Clean logs, backup database
Health Check	Fails 3× in 5 min	Restart container
Slack Connection	Disconnected > 2 min	Check tokens, restart

5.5 Alert Configuration

Critical Alerts (Immediate Response)

- Bot container stopped
- Health endpoint returning errors
- Database connection failures
- Slack Socket Mode disconnected

Warning Alerts (Next Business Day)

- CPU/RAM usage > 80%
- Disk usage > 70%
- Slow database queries

6. Backup & Recovery

6.1 Backup Strategy

What to Backup:

1. Database (bot.db or PostgreSQL dump)
2. Environment configuration (.env file)
3. Custom plugin code (if any)
4. Nginx configuration

Backup Frequency:

- **Database**: Daily automated backups
- **Configuration**: After any changes
- **Code**: Version controlled in Git

6.2 Automated Database Backups

SQLite Backup Script

Create `/opt/backups/backup-bot-db.sh`:

```bash
#!/bin/bash
# Backup ZaTech Bot SQLite database

BACKUP_DIR="/opt/backups/zatech-bot"
DATE=$(date +%Y-%m-%d-%H%M)
DB_FILE="/path/to/zatech-bot/bot.db"

# Create backup directory
mkdir -p $BACKUP_DIR

# Backup database
cp DB_FILE $BACKUP_DIR/bot-DATE.db

# Keep only last 30 days
find $BACKUP_DIR -name "bot-*.db" -mtime +30 -delete

echo "Backup completed: bot-$DATE.db"
```

Make executable and schedule:

```
chmod +x /opt/backups/backup-bot-db.sh

# Add to crontab (daily at 2 AM)
crontab -e
# Add line:
0 2 * * * /opt/backups/backup-bot-db.sh >> /var/log/bot-backup.log 2>&1
```

PostgreSQL Backup Script

```
#!/bin/bash
# Backup PostgreSQL database

BACKUP_DIR="/opt/backups/zatech-bot"
DATE=$(date +%Y-%m-%d-%H%M)

mkdir -p $BACKUP_DIR

# Dump database
docker compose exec -T db pg_dump -U postgres zatech_bot | gzip >
BACKUP_DIR/bot-DATE.sql.gz

# Keep only last 30 days
find $BACKUP_DIR -name "bot-*.sql.gz" -mtime +30 -delete

echo "Backup completed: bot-$DATE.sql.gz"
```

6.3 Offsite Backups

Git Repository (Configuration)

```bash
# Backup .env and configs to private Git repo
git init backup-configs
cd backup-configs
cp /path/to/.env .
git add .env
git commit -m "Backup configuration $(date +%Y-%m-%d)"
git push origin main
```

Cloud Storage (Database)

```bash
# Upload to AWS S3
aws s3 cp /opt/backups/zatech-bot/ s3://zatech-backups/bot/ --recursive

# Upload to Backblaze B2
b2 sync /opt/backups/zatech-bot/ b2://zatech-backups/bot/
```

6.4 Disaster Recovery Procedures

Scenario 1: Database Corruption

```bash
# 1. Stop bot
docker compose down

# 2. Restore from latest backup
cp /opt/backups/zatech-bot/bot-2025-10-30-0200.db ./bot.db

# 3. Restart bot
docker compose up -d

# 4. Verify health
curl https://bot.zatech.co.za/health
```

Scenario 2: Complete VPS Failure

```
# 1. Provision new VPS (see Section 2.1)

# 2. Clone repository
git clone https://github.com/zatech/zatech-bot.git
cd zatech-bot

# 3. Restore .env file
# (retrieve from secure backup)

# 4. Restore database
cp /backup/bot.db ./bot.db

# 5. Deploy
docker compose up -d

# 6. Configure Nginx and SSL (see Section 2.5)

# 7. Update DNS to point to new VPS IP
```

Recovery Time Objective (RTO): 30 minutes
Recovery Point Objective (RPO): 24 hours (daily backups)

6.5 Testing Backups

Monthly Backup Test (first Monday of each month):

```
# 1. Create test directory
mkdir /tmp/bot-restore-test

# 2. Restore latest backup
cp /opt/backups/zatech-bot/bot-latest.db /tmp/bot-restore-test/

# 3. Start bot in test mode
docker run --rm -it \
  --env-file .env \
  -v /tmp/bot-restore-test/bot-latest.db:/app/bot.db \
  zatech-bot:latest

# 4. Verify data integrity
# - Check admin dashboard
# - Verify plugin configurations

# 5. Clean up
rm -rf /tmp/bot-restore-test
```

7. Security Procedures

7.1 Access Control

SSH Access

```
# Use SSH keys only (disable password authentication)
sudo nano /etc/ssh/sshd_config
# Set: PasswordAuthentication no
# Set: PubkeyAuthentication yes

# Restart SSH
sudo systemctl restart sshd
```

Admin Dashboard Security

TODO: Implement authentication

Current state: Admin dashboard has no authentication (localhost-only recommended)

Mitigation:

- Keep admin dashboard behind VPN
- Use Nginx IP whitelist
- Only bind to localhost (use SSH tunnel)

Example Nginx IP whitelist:

```
location /admin {
    # Allow only specific IPs
    allow 203.0.113.0/24;  # Office network
    allow 198.51.100.50;   # Admin home IP
    deny all;

    proxy_pass http://localhost:3000/admin;
}
```

7.2 Secrets Management

Slack Tokens

```
# Store in .env file
# NEVER commit .env to Git
echo ".env" >> .gitignore

# Restrict file permissions
chmod 600 .env
chown ubuntu:ubuntu .env
```

Token Rotation

1. Generate new Slack tokens (see Section 2.2)
2. Update `.env` file
3. Restart bot: `docker compose restart app`
4. Revoke old tokens in Slack App settings

OpenAI API Key Security

```
# Store in .env
OPENAI_API_KEY=sk-your-key-here

# Monitor usage at https://platform.openai.com/usage
# Set spending limits to prevent abuse
```

7.3 Security Updates

System Updates

```
# Monthly security updates (first Monday)
sudo apt update
sudo apt upgrade -y
sudo reboot
```

Docker Image Updates

```
# Rebuild with latest base image
docker compose down
docker compose build --no-cache
docker compose up -d
```

Dependency Updates

```
# Update Python dependencies
pip install --upgrade -r requirements.txt

# Check for vulnerabilities
pip-audit
```

7.4 Security Monitoring

Log Analysis

```
# Monitor failed login attempts
sudo grep "Failed password" /var/log/auth.log

# Monitor suspicious Nginx requests
sudo grep "400\|404\|403" /var/log/nginx/error.log
```

Firewall Status

```
# Check UFW status
sudo ufw status verbose

# Review rules
sudo ufw show added
```

7.5 Incident Response

Security Incident Checklist:

1. **Contain**: Isolate affected system (firewall, disconnect)
2. **Identify**: Determine scope and attack vector
3. **Eradicate**: Remove malicious code, rotate secrets
4. **Recover**: Restore from clean backup
5. **Document**: Write incident report
6. **Review**: Update security procedures

8. Troubleshooting

8.1 Common Issues

Issue: Bot Not Responding in Slack

Symptoms:

- Bot doesn't respond to messages
- No greeting in #introductions

Diagnosis:

```
# 1. Check container status
docker compose ps
# Expected: app (healthy)

# 2. Check logs for errors
docker compose logs app --tail 50
# Look for: "Socket Mode handler is running"

# 3. Check Socket Mode connection
# Look for: "Connected to Slack via Socket Mode"
```

Solutions:

```
# Solution 1: Restart bot
docker compose restart app

# Solution 2: Verify Slack tokens
# Check .env file for correct SLACK_BOT_TOKEN and SLACK_APP_TOKEN

# Solution 3: Check bot invitation
# Invite bot to channel: /invite @ZATech Bot v3

# Solution 4: Verify Slack App scopes
# Go to https://api.slack.com/apps → OAuth & Permissions
```

8.2 Diagnostic Commands

```
# System health overview
curl https://bot.zatech.co.za/health

# Container logs (last 100 lines)
docker compose logs app --tail 100

# Real-time logs
docker compose logs -f app

# Container resource usage
docker stats zatech-bot

# Database size
ls -lh bot.db

# Network connectivity to Slack
curl -I https://slack.com

# Check Docker daemon
sudo systemctl status docker

# Check Nginx status
sudo systemctl status nginx

# Check open ports
sudo netstat -tulnp | grep -E ':(80|443|3000)'
```

8.3 Getting Help

Internal Resources:

- Architecture documentation: `bot-arch.md`
- README: `bot-readme.md`
- Running costs: `running-costs-doc.md`

External Resources:

- Slack Bolt Python: https://slack.dev/bolt-python/
- FastAPI Docs: https://fastapi.tiangolo.com/
- Docker Docs: https://docs.docker.com/

Support Channels:

- GitHub Issues: https://github.com/zatech/zatech-bot/issues