# SOaC Framework - Quick Start Guide

## Overview

This guide will help you get the SOaC Framework Phase 2A application up and running quickly.

## Prerequisites

Choose one of the following options:

### Option 1: Docker (Recommended)

- Docker and Docker Compose installed
- Ports 3000, 8000, and 5432 available

### Option 2: Manual Setup

- Python 3.11+
- Node.js 18+
- PostgreSQL 15+

## Quick Start with Docker (Recommended)

### 1. Start the Application

```
cd /home/ubuntu/code_artifacts/soac-framework
docker compose up --build
```

or if you have docker-compose:

```
docker-compose up --build
```

### 2. Wait for Initialization

The application will:
- Start PostgreSQL database
- Initialize database schema
- Load sample data (users, devices, rules, incidents)
- Start backend API server
- Start frontend development server

This may take 2-3 minutes on first run.

### 3. Access the Application

- **Frontend UI**: http://localhost:3000
- **Backend API**: http://localhost:8000
- **API Docs**: http://localhost:8000/api/docs

### 4. Login

Use one of the default credentials:

**Admin User:**
- Username: `admin`
- Password: `admin123`

**Analyst User:**
- Username: `analyst`
- Password: `analyst123`

### 5. Stop the Application

```
docker compose down
```

To stop and remove all data:

```
docker compose down -v
```

## Manual Setup (Without Docker)

### 1. Set Up PostgreSQL

```
# Install PostgreSQL (Ubuntu/Debian)
sudo apt-get install postgresql postgresql-contrib

# Start PostgreSQL
sudo systemctl start postgresql

# Create database and user
sudo -u postgres psql
```

In PostgreSQL shell:

```
CREATE DATABASE soac_db;
CREATE USER soac_user WITH PASSWORD 'soac_password';
GRANT ALL PRIVILEGES ON DATABASE soac_db TO soac_user;
\q
```

### 2. Set Up Backend

```
# Navigate to backend directory
cd backend

# Create virtual environment
python3 -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate

# Install dependencies
pip install -r requirements.txt

# Initialize database with sample data
python -m app.init_db

# Start backend server
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

The backend will be available at http://localhost:8000

### 3. Set Up Frontend

Open a new terminal:

```
# Navigate to frontend directory
cd frontend

# Install dependencies
npm install

# Start development server
npm run dev
```

The frontend will be available at http://localhost:3000

### 4. Access the Application

- **Frontend**: http://localhost:3000
- **Backend API**: http://localhost:8000
- **API Docs**: http://localhost:8000/api/docs

Login with default credentials (admin/admin123 or analyst/analyst123)

## Verify Installation

### 1. Check Backend Health

```
curl http://localhost:8000/health
```

Expected response:

```
{
  "status": "healthy",
  "timestamp": "2025-11-13T10:00:00.000000"
}
```

### 2. Check API Documentation

Open http://localhost:8000/api/docs in your browser. You should see the interactive Swagger UI with all API endpoints.

### 3. Test Login

```
curl -X POST http://localhost:8000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"admin","password":"admin123"}'
```

### 4. Check Frontend

Open http://localhost:3000 in your browser. You should see the login page.

## Explore the Application

### Dashboard

- View active incidents and investigations
- Monitor device health status
- See playbook execution metrics
- View incidents by severity

### Devices Page

- View all configured devices (6 sample devices pre-loaded)
- Add new devices (Palo Alto, EntraID, SIEM)
- Test device connections
- Edit or delete devices

### Rules Page

- View all detection rules (8 sample rules pre-loaded)
- Create new detection rules
- Edit existing rules
- Enable/disable rules
- Filter by device, severity, or status

## Sample Data Included

The application comes with pre-loaded sample data:

### Devices (6)

1. **PaloAlto NGFW - Main Firewall** (connected)
2. **PaloAlto NGFW - Regional Firewall** (connected)
3. **Microsoft EntraID - Primary Tenant** (connected)
4. **Microsoft EntraID - Dev Tenant** (disconnected)
5. **Elastic SIEM - Production** (connected)
6. **Splunk SIEM - Secondary** (error)

### Detection Rules (8)

- 3 EntraID authentication rules (brute force, password spraying, geo-velocity)
- 3 PaloAlto network rules (C2 beaconing, data exfiltration, malware traffic)
- 2 SIEM correlation rules (suspicious processes, lateral movement)

### Incidents (3)

- Intrusion chain incident (investigating)
- Data exfiltration incident (open)
- Ransomware incident (contained)

### Users (2)

- Admin user (full access)
- Analyst user (analyst role)

## Common Operations

### Add a New Device

1. Navigate to **Devices** page
2. Click **Add Device** button
3. Fill in device details:
   - Name: “My Device”
   - Type: Select device type
   - Configuration: Enter connection details
   - Enable: Check to enable
4. Click **Create**
5. Click the cable icon to test connection

### Add a New Rule

1. Navigate to **Rules** page
2. Click **Add Rule** button
3. Fill in rule details:
   - Rule ID: “DEVICE-001”
   - Device: Select device
   - Name: “My Detection Rule”
   - Description: Rule description
   - Severity: Select severity level
   - Query: Detection query
4. Click **Create**

### Test Device Connection

1. Navigate to **Devices** page
2. Find your device in the list
3. Click the cable icon in the Actions column
4. View connection test result

# Troubleshooting

## Port Already in Use

If you see “port already in use” errors:

```
# Check what's using the ports
sudo lsof -i :3000  # Frontend
sudo lsof -i :8000  # Backend
sudo lsof -i :5432  # PostgreSQL

# Kill the process or use different ports
```

## Database Connection Failed

```
# Check PostgreSQL is running
sudo systemctl status postgresql

# Restart PostgreSQL
sudo systemctl restart postgresql

# Check database logs
sudo tail -f /var/log/postgresql/postgresql-*.log
```

## Backend Not Starting

```
# Check Python version
python --version  # Should be 3.11+

# Reinstall dependencies
pip install -r requirements.txt --force-reinstall

# Check for errors
python -m app.main
```

## Frontend Not Loading

```
# Clear node_modules and reinstall
rm -rf node_modules package-lock.json
npm install

# Check Node version
node --version  # Should be 18+

# Try different port
npm run dev -- --port 3001
```

### Docker Issues

```
# View logs
docker compose logs backend
docker compose logs frontend
docker compose logs postgres

# Restart specific service
docker compose restart backend

# Rebuild from scratch
docker compose down -v
docker compose up --build
```

## Next Steps

### Configuration

- Update `.env` files for your environment
- Change default passwords
- Configure device integrations with real credentials
- Set up CORS for your domain

### Development

- Explore the API documentation at http://localhost:8000/api/docs
- Review the codebase in `backend/` and `frontend/` directories
- Check out `analysis_summary.md` for architecture details
- Read the full `README.md` for detailed documentation

### Production Deployment

- Use production-grade database (managed PostgreSQL)
- Enable HTTPS with SSL certificates
- Use production WSGI server (Gunicorn)
- Set up reverse proxy (nginx)
- Configure secrets management
- Build frontend for production ( `npm run build` )
- Set up monitoring and logging

## Support

For detailed documentation, see the main README.md (README.md) file.

For questions or issues, contact the SOaC Framework Team.

---

**Note**: This localhost refers to localhost of the computer that I'm using to run the application, not your local machine. To access it locally or remotely, you'll need to deploy the application on your own system following the instructions above.

---