Enhanced SICA Installation Guide

Version: 1.0
Date: December 2024
Document Type: Installation Guide
Classification: Public

Table of Contents

System Requirements

Hardware Requirements

Minimum Requirements (Development/Testing)

- **CPU**: 2 cores, 2.0 GHz (x86_64)
- **RAM**: 4 GB
- **Storage**: 10 GB free space
- **Network**: 100 Mbps Ethernet
- **Architecture**: x86_64 (AMD64)

Recommended Requirements (Production)

- **CPU**: 8 cores, 3.0 GHz (x86_64)
- **RAM**: 16 GB
- **Storage**: 100 GB SSD
- **Network**: 1 Gbps Ethernet
- **Architecture**: x86_64 (AMD64)

Enterprise Requirements (High-Scale Production)

- **CPU**: 32 cores, 3.5 GHz (x86_64)
- **RAM**: 64 GB
- **Storage**: 500 GB NVMe SSD
- **Network**: 10 Gbps Ethernet with redundancy
- **Architecture**: x86_64 (AMD64)

Software Requirements

Operating System Support

Linux Distributions (Recommended): - Ubuntu 20.04 LTS, 22.04 LTS, 24.04 LTS - Debian 11 (Bullseye), 12 (Bookworm) - CentOS 8, 9 - Red Hat Enterprise Linux (RHEL) 8, 9 - Rocky Linux 8, 9 - Fedora 36+

Windows Support: - Windows 10 (64-bit) - Windows 11 (64-bit) - Windows Server 2019, 2022

Required Software Dependencies

Core Dependencies: - Python 3.8+ (3.11 recommended) - Node.js 16+ (18 LTS recommended) - SQLite 3.35+ (or PostgreSQL 12+ for production) - Git 2.25+

System Tools: - curl or wget - unzip/tar - systemd (Linux) or Windows Service Manager - Firewall (ufw/firewalld/Windows Firewall)

Optional Dependencies: - Docker 20.10+ (for containerized deployment) - Kubernetes 1.20+ (for orchestrated deployment) - Nginx 1.18+ (for reverse proxy) - Redis 6.0+ (for distributed caching)

Network Requirements

Port Requirements

Default Ports: - **3000**: Web Dashboard (HTTP) - **5000**: API Server (HTTP) - **8080**: Agent Communication - **443**: HTTPS (if SSL configured)

Protocol Monitoring Ports: - **502**: Modbus TCP - **4840**: OPC UA - **20000**: DNP3 - **44818**: Ethernet/IP - **47808**: BACnet

Firewall Configuration

Inbound Rules:

```
# Web Dashboard
Allow TCP 3000 from trusted networks

# API Access
Allow TCP 5000 from application networks

# Agent Communication
Allow TCP 8080 from agent networks

# Protocol Monitoring
Allow TCP 502, 4840, 20000, 44818, 47808 from OT networks
```

Outbound Rules:

```
# Internet access for updates
Allow TCP 80, 443 to any

# DNS resolution
Allow UDP 53 to DNS servers

# NTP synchronization
Allow UDP 123 to NTP servers
```

Installation Steps

Pre-Installation Checklist

Before beginning installation, verify the following:

1. **System Requirements**: Confirm hardware and software requirements are met

2. **Network Access**: Ensure internet connectivity for downloading dependencies
3. **Permissions**: Verify administrative/sudo access
4. **Firewall**: Configure firewall rules as specified
5. **Backup**: Create system backup if installing on existing infrastructure

Linux Installation

Method 1: Automated Installation Script

Step 1: Download Installation Package

```
# Download the latest release
wget https://releases.enhanced-sica.com/v1.0/enhanced-sica-linux-package.tar.gz

# Verify checksum
sha256sum enhanced-sica-linux-package.tar.gz
# Expected: [checksum_value]

# Extract package
tar -xzf enhanced-sica-linux-package.tar.gz
cd enhanced-sica-linux
```

Step 2: Run Installation Script

```
# Make script executable
chmod +x install-linux.sh

# Run installation (requires sudo)
./install-linux.sh
```

Step 3: Verify Installation

```
# Check service status
sudo systemctl status enhanced-sica
sudo systemctl status enhanced-sica-dashboard

# Test CLI
sica status

# Access web dashboard
firefox http://localhost
```

Method 2: Manual Installation

Step 1: Install System Dependencies

Ubuntu/Debian:

```
 # Update package list
sudo apt update

# Install dependencies
sudo apt install -y \
    python3 \
    python3-pip \
    python3-venv \
    nodejs \
    npm \
    curl \
    wget \
    git \
    sqlite3 \
    build-essential \
    systemd \
    nginx \
    ufw
```

CentOS/RHEL:

```
 # Update system
sudo dnf update -y

# Install dependencies
sudo dnf install -y \
    python3 \
    python3-pip \
    nodejs \
    npm \
    curl \
    wget \
    git \
    sqlite \
    gcc \
    gcc-c++ \
    make \
    systemd \
    nginx \
    firewalld
```

Step 2: Create System User

```
# Create Enhanced SICA system user
sudo useradd -r -s /bin/false -d /opt/enhanced-sica sica

# Create directories
sudo mkdir -p /opt/enhanced-sica
sudo mkdir -p /var/log/enhanced-sica
sudo mkdir -p /etc/enhanced-sica
sudo mkdir -p /var/lib/enhanced-sica

# Set ownership
sudo chown -R sica:sica /opt/enhanced-sica
sudo chown -R sica:sica /var/log/enhanced-sica
sudo chown -R sica:sica /var/lib/enhanced-sica
```

Step 3: Install Enhanced SICA

```
# Copy application files
sudo cp -r . /opt/enhanced-sica/
sudo chown -R sica:sica /opt/enhanced-sica

# Set up Python virtual environment
cd /opt/enhanced-sica
sudo -u sica python3 -m venv venv
sudo -u sica ./venv/bin/pip install --upgrade pip
sudo -u sica ./venv/bin/pip install -r requirements.txt
```

Step 4: Build Web Dashboard

```
# Install Node.js dependencies and build
cd /opt/enhanced-sica/src/web/sica-dashboard
sudo -u sica npm install
sudo -u sica npm run build
```

Step 5: Configure System Services

```
# Create systemd service file
sudo tee /etc/systemd/system/enhanced-sica.service > /dev/null <<EOF
[Unit]
Description=Enhanced SICA Cybersecurity Platform
After=network.target
Wants=network.target

[Service]
Type=simple
User=sica
Group=sica
WorkingDirectory=/opt/enhanced-sica
Environment=PYTHONPATH=/opt/enhanced-sica/src
ExecStart=/opt/enhanced-sica/venv/bin/python src/core/enhanced_sica_engine.py
Restart=always
RestartSec=10
StandardOutput=journal
StandardError=journal
SyslogIdentifier=enhanced-sica

[Install]
WantedBy=multi-user.target
EOF

# Enable and start service
sudo systemctl daemon-reload
sudo systemctl enable enhanced-sica.service
sudo systemctl start enhanced-sica.service
```

Windows Installation

Method 1: Automated Installation

Step 1: Download Windows Package

```
# Download from: https://releases.enhanced-sica.com/v1.0/enhanced-sica-windows-package.zip
# Extract to desired location (e.g., C:\Enhanced-SICA\)
```

Step 2: Run Installation

```
# Open Command Prompt as Administrator
cd C:\Enhanced-SICA

# Run installation script
install-windows.bat
```

Step 3: Start Enhanced SICA

```
# Start all services
start-windows.bat

# Access dashboard
start http://localhost:3000
```

Method 2: Manual Windows Installation

Step 1: Install Prerequisites

1. **Python 3.11**:
2. Download from https://python.org/downloads/windows/
3. Check "Add Python to PATH" during installation
4. Verify: `python --version`
5. **Node.js 18 LTS**:
6. Download from https://nodejs.org/en/download/
7. Install with default options
8. Verify: `node --version`
9. **Git for Windows**:
10. Download from https://git-scm.com/download/win
11. Install with default options

Step 2: Install Enhanced SICA

```
# Create installation directory
mkdir C:\Enhanced-SICA
cd C:\Enhanced-SICA

# Extract application files
# (Copy extracted files to this directory)

# Install Python dependencies
python -m pip install --upgrade pip
pip install -r requirements.txt

# Install Node.js dependencies
cd src\web\sica-dashboard
npm install
npm run build
cd ..\..\..
```

Step 3: Configure Windows Service

```
# Install NSSM (Non-Sucking Service Manager)
# Download from: https://nssm.cc/download

# Create service
nssm install "Enhanced SICA"
# Set Application Path: C:\Enhanced-SICA\venv\Scripts\python.exe
# Set Arguments: src\core\enhanced_sica_engine.py
# Set Startup directory: C:\Enhanced-SICA

# Start service
net start "Enhanced SICA"
```

macOS Installation (Development Only)

Step 1: Install Prerequisites

```
# Install Homebrew
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

# Install dependencies
brew install python@3.11 node@18 git sqlite

# Verify installations
python3 --version
node --version
```

Step 2: Install Enhanced SICA

```
# Create installation directory
sudo mkdir -p /opt/enhanced-sica
sudo chown $(whoami):staff /opt/enhanced-sica

# Extract and install
cd /opt/enhanced-sica
# Copy application files here

# Set up Python environment
python3 -m venv venv
source venv/bin/activate
pip install --upgrade pip
pip install -r requirements.txt

# Build web dashboard
cd src/web/sica-dashboard
npm install
npm run build
cd ../../..
```

Step 3: Start Services

```
# Start Enhanced SICA
source venv/bin/activate
python src/core/enhanced_sica_engine.py &

# Start web dashboard
cd src/web/sica-dashboard
npm run preview &

# Access dashboard
open http://localhost:3000
```

Configuration

Initial Configuration

First-Time Setup Wizard

After installation, access the web dashboard to complete initial setup:

1. **Access Setup Wizard**:
2. Open browser to http://localhost:3000
3. Click "Initial Setup" if prompted
4. **Administrator Account**: `Username: admin Password: [create strong password] Email: admin@yourcompany.com`
5. **System Configuration**: `System Name: Enhanced SICA Production Location: Data Center 1 Time Zone: UTC`
6. **Network Configuration**: `Management Interface: eth0 Monitoring Interfaces: eth1, eth2 DNS Servers: 8.8.8.8, 8.8.4.4`
7. **Protocol Selection**: `☑ Modbus TCP (Port 502) ☑ OPC UA (Port 4840) ☑ DNP3 (Port 20000) ☐ Ethernet/IP (Port 44818) ☐ BACnet (Port 47808)`

Configuration Files

Main Configuration File: `/etc/enhanced-sica/config.conf`

```
[core]
debug = false
log_level = INFO
data_dir = /var/lib/enhanced-sica
log_dir = /var/log/enhanced-sica
max_workers = 8

[database]
type = sqlite
path = /var/lib/enhanced-sica/sica.db
# For PostgreSQL:
# type = postgresql
# host = localhost
# port = 5432
# database = enhanced_sica
# username = sica_user
# password = secure_password

[api]
host = 127.0.0.1
port = 5000
cors_origins = ["http://localhost:3000"]
rate_limit = 1000

[dashboard]
host = 0.0.0.0
port = 3000
session_timeout = 3600

[security]
jwt_secret = [generate_random_secret]
encryption_key = [generate_encryption_key]
quantum_enabled = true
mfa_enabled = true

[protocols]
modbus_enabled = true
modbus_port = 502
opcua_enabled = true
opcua_port = 4840
dnp3_enabled = false
dnp3_port = 20000

[agents]
max_agents = 50
deployment_timeout = 300
heartbeat_interval = 30
default_mode = stealth

[ai]
models_enabled = true
prediction_horizon = 72
confidence_threshold = 0.8
training_interval = 24

[quantum]
qkd_enabled = true
post_quantum_crypto = true
key_refresh_interval = 3600

[monitoring]
```

```
metrics_enabled = true
prometheus_port = 9090
health_check_interval = 60
```

Advanced Configuration

Database Configuration

SQLite Configuration (Default):

```
[database]
type = sqlite
path = /var/lib/enhanced-sica/sica.db
journal_mode = WAL
synchronous = NORMAL
cache_size = 10000
temp_store = MEMORY
```

PostgreSQL Configuration (Production):

```
[database]
type = postgresql
host = localhost
port = 5432
database = enhanced_sica
username = sica_user
password = secure_password
pool_size = 20
max_overflow = 30
pool_timeout = 30
pool_recycle = 3600
```

PostgreSQL Setup:

```
-- Create database and user
CREATE DATABASE enhanced_sica;
CREATE USER sica_user WITH PASSWORD 'secure_password';
GRANT ALL PRIVILEGES ON DATABASE enhanced_sica TO sica_user;

-- Connect to enhanced_sica database
\c enhanced_sica

-- Create extensions
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
CREATE EXTENSION IF NOT EXISTS "pg_stat_statements";

-- Grant schema permissions
GRANT ALL ON SCHEMA public TO sica_user;
```

SSL/TLS Configuration

Generate SSL Certificates:

```
 # Self-signed certificate (development)
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout /etc/enhanced-sica/ssl/sica.key \
    -out /etc/enhanced-sica/ssl/sica.crt \
    -subj "/C=US/ST=State/L=City/O=Organization/CN=enhanced-sica.local"

# Let's Encrypt certificate (production)
sudo certbot certonly --standalone -d your-domain.com
```

Nginx SSL Configuration:

```
server {
    listen 443 ssl http2;
    server_name enhanced-sica.local;

    ssl_certificate /etc/enhanced-sica/ssl/sica.crt;
    ssl_certificate_key /etc/enhanced-sica/ssl/sica.key;

    # SSL Security
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;
    ssl_session_cache shared:SSL:10m;
    ssl_session_timeout 10m;

    # 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;
    add_header X-XSS-Protection "1; mode=block" always;

    location / {
        proxy_pass http://127.0.0.1:3000;
        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;
    }

    location /api/ {
        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;
    }
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name enhanced-sica.local;
    return 301 https://$`server_name`$request_uri;
}
```

Logging Configuration

Log Configuration: `/etc/enhanced-sica/logging.conf`

```
[loggers]
keys = root, sica, agents, threats, protocols

[handlers]
keys = console, file, syslog

[formatters]
keys = standard, detailed

[logger_root]
level = INFO
handlers = console, file

[logger_sica]
level = INFO
handlers = file, syslog
qualname = sica
propagate = 0

[logger_agents]
level = DEBUG
handlers = file
qualname = sica.agents
propagate = 0

[logger_threats]
level = INFO
handlers = file, syslog
qualname = sica.threats
propagate = 0

[logger_protocols]
level = INFO
handlers = file
qualname = sica.protocols
propagate = 0

[handler_console]
class = StreamHandler
level = INFO
formatter = standard
args = (sys.stdout,)

[handler_file]
class = handlers.RotatingFileHandler
level = DEBUG
formatter = detailed
args = ('/var/log/enhanced-sica/sica.log', 'a', 10485760, 5)

[handler_syslog]
class = handlers.SysLogHandler
level = WARNING
formatter = standard
args = (('localhost', 514),)

[formatter_standard]
format = %(asctime)s [%(levelname)s] %(name)s: %(message)s

[formatter_detailed]
format = %(asctime)s [%(levelname)s] %(name)s:%(lineno)d: %(message)s
```

Docker Deployment

Docker Installation

Prerequisites

```
 # Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Install Docker Compose
sudo curl -L
"https://github.com/docker/compose/releases/download/v2.20.0/docker-
compose-$`(uname -s)-`$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Add user to docker group
sudo usermod -aG docker $USER
newgrp docker
```

Docker Compose Configuration

docker-compose.yml:

```yaml
version: '3.8'

services:
  enhanced-sica-core:
    build:
      context: .
      dockerfile: docker/Dockerfile.core
    container_name: sica-core
    restart: unless-stopped
    ports:
      - "5000:5000"
    environment:
      - PYTHONPATH=/app/src
      -
DATABASE_URL=postgresql://sica_user:secure_password@postgres:5432/enhanced_sica
      - REDIS_URL=redis://redis:6379/0
    volumes:
      - sica_data:/var/lib/enhanced-sica
      - sica_logs:/var/log/enhanced-sica
      - ./config:/etc/enhanced-sica
    depends_on:
      - postgres
      - redis
    networks:
      - sica_network

  enhanced-sica-dashboard:
    build:
      context: .
      dockerfile: docker/Dockerfile.dashboard
    container_name: sica-dashboard
    restart: unless-stopped
    ports:
      - "3000:3000"
    environment:
      - REACT_APP_API_URL=http://localhost:5000/api/v1
    depends_on:
      - enhanced-sica-core
    networks:
      - sica_network

  postgres:
    image: postgres:15-alpine
    container_name: sica-postgres
    restart: unless-stopped
    environment:
      - POSTGRES_DB=enhanced_sica
      - POSTGRES_USER=sica_user
      - POSTGRES_PASSWORD=secure_password
    volumes:
      - postgres_data:/var/lib/postgresql/data
      - ./docker/init-db.sql:/docker-entrypoint-initdb.d/init-db.sql
    networks:
      - sica_network

  redis:
    image: redis:7-alpine
    container_name: sica-redis
    restart: unless-stopped
    command: redis-server --appendonly yes
    volumes:
```

```
      - redis_data:/data
    networks:
      - sica_network

  nginx:
    image: nginx:alpine
    container_name: sica-nginx
    restart: unless-stopped
    ports:
      - "80:80"
      - "443:443"
    volumes:
      - ./docker/nginx.conf:/etc/nginx/nginx.conf
      - ./ssl:/etc/nginx/ssl
    depends_on:
      - enhanced-sica-core
      - enhanced-sica-dashboard
    networks:
      - sica_network

volumes:
  sica_data:
  sica_logs:
  postgres_data:
  redis_data:

networks:
  sica_network:
    driver: bridge
```

Dockerfile Configuration

docker/Dockerfile.core:

```dockerfile
FROM python:3.11-slim

# Install system dependencies
RUN apt-get update && apt-get install -y \
    gcc \
    g++ \
    make \
    curl \
    && rm -rf /var/lib/apt/lists/*

# Create app user
RUN useradd -r -s /bin/false sica

# Set working directory
WORKDIR /app

# Copy requirements and install Python dependencies
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

# Copy application code
COPY src/ src/
COPY config/ config/

# Create necessary directories
RUN mkdir -p /var/lib/enhanced-sica /var/log/enhanced-sica
RUN chown -R sica:sica /app /var/lib/enhanced-sica /var/log/enhanced-sica

# Switch to app user
USER sica

# Expose port
EXPOSE 5000

# Health check
HEALTHCHECK --interval=30s --timeout=10s --start-period=5s --retries=3 \
    CMD curl -f http://localhost:5000/health || exit 1

# Start application
CMD ["python", "src/core/enhanced_sica_engine.py"]
```

docker/Dockerfile.dashboard:

```dockerfile
FROM node:18-alpine AS builder

WORKDIR /app

# Copy package files
COPY src/web/sica-dashboard/package*.json ./
RUN npm ci --only=production

# Copy source code
COPY src/web/sica-dashboard/ .

# Build application
RUN npm run build

# Production stage
FROM nginx:alpine

# Copy built application
COPY --from=builder /app/dist /usr/share/nginx/html

# Copy nginx configuration
COPY docker/nginx-dashboard.conf /etc/nginx/conf.d/default.conf

# Expose port
EXPOSE 3000

# Start nginx
CMD ["nginx", "-g", "daemon off;"]
```

Docker Deployment Commands

```bash
# Build and start services
docker-compose up -d

# View logs
docker-compose logs -f enhanced-sica-core
docker-compose logs -f enhanced-sica-dashboard

# Scale services
docker-compose up -d --scale enhanced-sica-core=3

# Update services
docker-compose pull
docker-compose up -d

# Backup data
docker run --rm -v sica_data:/data -v $(pwd):/backup alpine tar czf
/backup/sica-backup.tar.gz -C /data .

# Restore data
docker run --rm -v sica_data:/data -v $(pwd):/backup alpine tar xzf
/backup/sica-backup.tar.gz -C /data
```

Kubernetes Setup

Prerequisites

Install Kubernetes Tools

```
# Install kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

# Install Helm
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Verify installations
kubectl version --client
helm version
```

Cluster Setup (Example with kubeadm)

```
# Initialize cluster (master node)
sudo kubeadm init --pod-network-cidr=10.244.0.0/16

# Configure kubectl
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config

# Install CNI plugin (Flannel)
kubectl apply -f
https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-
flannel.yml

# Join worker nodes (run on worker nodes)
sudo kubeadm join [master-ip]:6443 --token [token] --discovery-token-ca-cert-
hash [hash]
```

Kubernetes Manifests

Namespace

```
# namespace.yaml
apiVersion: v1
kind: Namespace
metadata:
  name: enhanced-sica
  labels:
    name: enhanced-sica
```

ConfigMap

```yaml
# configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: sica-config
  namespace: enhanced-sica
data:
  config.conf: |
    [core]
    debug = false
    log_level = INFO
    max_workers = 8

    [database]
    type = postgresql
    host = postgres-service
    port = 5432
    database = enhanced_sica

    [api]
    host = 0.0.0.0
    port = 5000

    [dashboard]
    host = 0.0.0.0
    port = 3000

    [security]
    quantum_enabled = true
    mfa_enabled = true
```

Secrets

```yaml
# secrets.yaml
apiVersion: v1
kind: Secret
metadata:
  name: sica-secrets
  namespace: enhanced-sica
type: Opaque
data:
  database-password: c2VjdXJlX3Bhc3N3b3Jk  # base64 encoded
  jwt-secret: [base64_encoded_jwt_secret]
  encryption-key: [base64_encoded_encryption_key]
```

PostgreSQL Deployment

```yaml
# postgres.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgres
  namespace: enhanced-sica
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgres
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
      - name: postgres
        image: postgres:15-alpine
        env:
        - name: POSTGRES_DB
          value: enhanced_sica
        - name: POSTGRES_USER
          value: sica_user
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: sica-secrets
              key: database-password
        ports:
        - containerPort: 5432
        volumeMounts:
        - name: postgres-storage
          mountPath: /var/lib/postgresql/data
        resources:
          requests:
            memory: "512Mi"
            cpu: "250m"
          limits:
            memory: "1Gi"
            cpu: "500m"
      volumes:
      - name: postgres-storage
        persistentVolumeClaim:
          claimName: postgres-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: postgres-service
  namespace: enhanced-sica
spec:
  selector:
    app: postgres
  ports:
  - port: 5432
    targetPort: 5432
```

Enhanced SICA Core Deployment

```yaml
# sica-core.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: enhanced-sica-core
  namespace: enhanced-sica
spec:
  replicas: 3
  selector:
    matchLabels:
      app: enhanced-sica-core
  template:
    metadata:
      labels:
        app: enhanced-sica-core
    spec:
      containers:
      - name: sica-core
        image: enhanced-sica/core:1.0
        ports:
        - containerPort: 5000
        env:
        - name: PYTHONPATH
          value: "/app/src"
        - name: DATABASE_URL
          value: "postgresql://sica_user:$(DATABASE_PASSWORD)@postgres-
service:5432/enhanced_sica"
        - name: DATABASE_PASSWORD
          valueFrom:
            secretKeyRef:
              name: sica-secrets
              key: database-password
        volumeMounts:
        - name: config-volume
          mountPath: /etc/enhanced-sica
        - name: data-volume
          mountPath: /var/lib/enhanced-sica
        - name: logs-volume
          mountPath: /var/log/enhanced-sica
        resources:
          requests:
            memory: "1Gi"
            cpu: "500m"
          limits:
            memory: "2Gi"
            cpu: "1000m"
        livenessProbe:
          httpGet:
            path: /health
            port: 5000
          initialDelaySeconds: 30
          periodSeconds: 10
        readinessProbe:
          httpGet:
            path: /ready
            port: 5000
          initialDelaySeconds: 5
          periodSeconds: 5
      volumes:
```

```
      - name: config-volume
        configMap:
          name: sica-config
      - name: data-volume
        persistentVolumeClaim:
          claimName: sica-data-pvc
      - name: logs-volume
        persistentVolumeClaim:
          claimName: sica-logs-pvc
---
apiVersion: v1
kind: Service
metadata:
  name: sica-core-service
  namespace: enhanced-sica
spec:
  selector:
    app: enhanced-sica-core
  ports:
  - port: 5000
    targetPort: 5000
  type: ClusterIP
```

Ingress Configuration

```
# ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: sica-ingress
  namespace: enhanced-sica
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    cert-manager.io/cluster-issuer: "letsencrypt-prod"
spec:
  tls:
  - hosts:
    - enhanced-sica.yourdomain.com
    secretName: sica-tls
  rules:
  - host: enhanced-sica.yourdomain.com
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: sica-core-service
            port:
              number: 5000
      - path: /
        pathType: Prefix
        backend:
          service:
            name: sica-dashboard-service
            port:
              number: 3000
```

Persistent Volume Claims

```yaml
# pvc.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: postgres-pvc
  namespace: enhanced-sica
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: fast-ssd
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sica-data-pvc
  namespace: enhanced-sica
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Gi
  storageClassName: fast-ssd
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sica-logs-pvc
  namespace: enhanced-sica
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: standard
```

Deployment Commands

```
 # Create namespace
kubectl apply -f namespace.yaml

# Apply configurations
kubectl apply -f configmap.yaml
kubectl apply -f secrets.yaml
kubectl apply -f pvc.yaml

# Deploy database
kubectl apply -f postgres.yaml

# Wait for database to be ready
kubectl wait --for=condition=ready pod -l app=postgres -n enhanced-sica --
timeout=300s

# Deploy Enhanced SICA
kubectl apply -f sica-core.yaml
kubectl apply -f sica-dashboard.yaml

# Configure ingress
kubectl apply -f ingress.yaml

# Verify deployment
kubectl get pods -n enhanced-sica
kubectl get services -n enhanced-sica
kubectl get ingress -n enhanced-sica
```

Monitoring and Scaling

Horizontal Pod Autoscaler

```yaml
# hpa.yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: sica-core-hpa
  namespace: enhanced-sica
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: enhanced-sica-core
  minReplicas: 3
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 70
  - type: Resource
    resource:
      name: memory
      target:
        type: Utilization
        averageUtilization: 80
```

Monitoring with Prometheus

```yaml
# monitoring.yaml
apiVersion: v1
kind: ServiceMonitor
metadata:
  name: sica-core-monitor
  namespace: enhanced-sica
spec:
  selector:
    matchLabels:
      app: enhanced-sica-core
  endpoints:
  - port: metrics
    interval: 30s
    path: /metrics
```

Verification and Testing

Post-Installation Verification

System Health Check

```
# Check service status
sudo systemctl status enhanced-sica
sudo systemctl status enhanced-sica-dashboard

# Check process status
ps aux | grep python | grep sica
ps aux | grep node | grep sica

# Check port binding
sudo netstat -tlnp | grep :5000
sudo netstat -tlnp | grep :3000

# Check logs
sudo journalctl -u enhanced-sica --no-pager -n 20
tail -f /var/log/enhanced-sica/sica.log
```

Web Interface Test

```
# Test API endpoint
curl -X GET http://localhost:5000/api/v1/health
# Expected response: {"status": "healthy", "timestamp": "..."}

# Test dashboard
curl -I http://localhost:3000
# Expected: HTTP/1.1 200 OK

# Test authentication
curl -X POST http://localhost:5000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username": "admin", "password": "your_password"}'
```

CLI Testing

```
# Test CLI functionality
sica --help
sica status
sica protocols
sica agents

# Test demo mode
sica demo

# Test system health
sica system health-check
```

Functional Testing

Protocol Testing

```
# Test Modbus protocol support
sica protocols test modbus --target 192.168.1.10:502

# Test OPC UA protocol support
sica protocols test opcua --endpoint opc.tcp://192.168.1.20:4840

# List supported protocols
sica protocols list
```

Agent Testing

```
# Deploy test agent
sica agents deploy --type eco_agent --target 192.168.1.0/24 --mode stealth

# Check agent status
sica agents status

# View agent reports
sica agents reports --agent-id agent_001
```

AI System Testing

```
# Test predictive intelligence
sica ai predict --timeframe 24h --data-source network_traffic

# Test explainable AI
sica ai explain --prediction-id pred_123

# Test model status
sica ai models status
```

Performance Testing

Load Testing

```
# Install testing tools
pip install locust

# Create load test script
cat > load_test.py << 'EOF'
from locust import HttpUser, task, between

class SICAUser(HttpUser):
    wait_time = between(1, 3)

    def on_start(self):
        # Login
        response = self.client.post("/api/v1/auth/login", json={
            "username": "admin",
            "password": "your_password"
        })
        self.token = response.json()["access_token"]
        self.client.headers.update({"Authorization": f"Bearer {self.token}"})

    @task(3)
    def get_threats(self):
        self.client.get("/api/v1/threats")

    @task(2)
    def get_agents(self):
        self.client.get("/api/v1/agents")

    @task(1)
    def get_system_status(self):
        self.client.get("/api/v1/system/status")
EOF

# Run load test
locust -f load_test.py --host=http://localhost:5000 --users 50 --spawn-rate 5
```

Stress Testing

```
# CPU stress test
stress --cpu 8 --timeout 60s

# Memory stress test
stress --vm 2 --vm-bytes 1G --timeout 60s

# I/O stress test
stress --io 4 --timeout 60s

# Monitor system during stress
watch -n 1 'sica system metrics'
```

Security Testing

Authentication Testing

```
# Test invalid credentials
curl -X POST http://localhost:5000/api/v1/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username": "admin", "password": "wrong_password"}'
# Expected: 401 Unauthorized

# Test token expiration
# (Use expired token)
curl -X GET http://localhost:5000/api/v1/threats \
  -H "Authorization: Bearer expired_token"
# Expected: 401 Unauthorized
```

SSL/TLS Testing

```
# Test SSL configuration
openssl s_client -connect localhost:443 -servername enhanced-sica.local

# Test SSL strength
nmap --script ssl-enum-ciphers -p 443 localhost

# Test certificate validity
openssl x509 -in /etc/enhanced-sica/ssl/sica.crt -text -noout
```

Troubleshooting Common Issues

Installation Issues

Issue: Python dependencies fail to install

```
# Solution: Update pip and install build tools
python -m pip install --upgrade pip setuptools wheel
sudo apt install python3-dev build-essential  # Ubuntu/Debian
sudo dnf install python3-devel gcc gcc-c++    # CentOS/RHEL
```

Issue: Node.js build fails

```
# Solution: Clear npm cache and reinstall
npm cache clean --force
rm -rf node_modules package-lock.json
npm install
```

Issue: Service fails to start

```
# Check logs for errors
sudo journalctl -u enhanced-sica --no-pager -n 50

# Check configuration
sica config validate

# Check permissions
sudo chown -R sica:sica /opt/enhanced-sica
sudo chown -R sica:sica /var/lib/enhanced-sica
sudo chown -R sica:sica /var/log/enhanced-sica
```

Runtime Issues

Issue: High memory usage

```
# Check memory usage
free -h
ps aux --sort=-%mem | head -10

# Optimize configuration
# Edit /etc/enhanced-sica/config.conf
[core]
max_workers = 4  # Reduce from default
log_level = WARN  # Reduce logging

# Restart service
sudo systemctl restart enhanced-sica
```

Issue: Database connection errors

```
# Check database status
sudo systemctl status postgresql  # For PostgreSQL
sqlite3 /var/lib/enhanced-sica/sica.db ".tables"  # For SQLite

# Test database connection
sica database test-connection

# Repair database if needed
sica database repair
```

Backup and Recovery Testing

Backup Testing

```
# Create backup
sica backup create --output /backup/sica-backup-$(date +%Y%m%d).tar.gz

# Verify backup
tar -tzf /backup/sica-backup-*.tar.gz

# Test backup integrity
sica backup verify --file /backup/sica-backup-*.tar.gz
```

Recovery Testing

```
# Stop services
sudo systemctl stop enhanced-sica enhanced-sica-dashboard

# Simulate data loss
sudo rm -rf /var/lib/enhanced-sica/*

# Restore from backup
sica backup restore --file /backup/sica-backup-*.tar.gz

# Start services
sudo systemctl start enhanced-sica enhanced-sica-dashboard

# Verify restoration
sica status
```

Document Information: - **Version**: 1.0 - **Last Updated**: December 2024 - **Document Owner**: Enhanced SICA Development Team - **Classification**: Public - **Distribution**: Unlimited