i2b2 Production Deployment Guide

This guide provides step-by-step instructions for deploying i2b2 in a production environment using Docker, connected to a PostgreSQL database on a remote production server.

- 1. Setting up Subnets in the Cloud

Ensure appropriate cloud subnets and security groups are configured to allow communication between the application containers and the production PostgreSQL database. Follow your cloud provider's documentation to:

- Create private and public subnets
- Configure routing tables
- Set up security groups to allow traffic on necessary ports (e.g., PostgreSQL default 5432, i2b2 web client port)

- 2. Setting up PostgreSQL Database

Refer to the official i2b2 Data Installation Guide to prepare the PostgreSQL database schemas and users:

i2b2 Data Installation Guide

- 3. Deploying i2b2 Client Containers and Connecting to the Production Database

This section details how to deploy the i2b2 web client and Core Server using Docker, and connect them to the pre-configured PostgreSQL production database.

Step 1: Clone the i2b2 Docker Repository

```
git clone https://github.com/i2b2/i2b2-docker.git
```

Step 2: Navigate into the Cloned Directory

```
cd i2b2-docker
```

Step 3: Checkout the Production Configuration Branch

```
git checkout release-v1.8.1a.0001_prod_db
```

Step 4: Configure the Environment Variables

Edit the `.env` file and update the following fields with your production database configuration:

- `DB_HOST`
- `DB_PORT`
- `DB_USERNAME`
- `DB_PASSWORD`
- `DB_NAME`
- `DB_SCHEMA_NAME`

Step 5: Start the Web Client and Core Server Containers

```
docker-compose up -d i2b2-webclient
```

Step 6: Verify Deployment

Check container logs to verify successful startup:

```
docker-compose logs i2b2-core-server
```

Then access the i2b2 web client using the production URL and port.

- 4. Custom Configuration

Update Database Lookup Tables in `i2b2hive`

Run the following SQL statements to map the i2b2 services to correct schemas:

```
UPDATE crc_db_lookup SET c_db_fullschema = 'i2b2demodata';
UPDATE work_db_lookup SET c_db_fullschema = 'i2b2workdata';
UPDATE ont_db_lookup SET c_db_fullschema = 'i2b2metadata';
```

Update Core Server URL in the Database

In the PM schema, update the `PM_CELL_DATA` table to reflect the production server URL.

Replace any instance of:

```
http://localhost:9090/
```

With:

```
http://$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT/
```

Refer to this SQL file for examples:

pm_access_insert_data.sql

Update the Web Client Configuration

Access the running web client container and modify the configuration files:

```
docker exec -it i2b2-webclient bash
cd /var/www/html/webclient/
```

1. Edit i2b2_config_domains.json

```
vi i2b2_config_domains.json
```

Update:

- "urlProxy": Replace "/~proxy" with "proxy.php"
- "urlCellPM": Replace default URL with "http://$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT/i2b2/services/PMService/'

2. Update proxy.php

```
vi proxy.php
```

Whitelist URL:

replace this URL `http://services.i2b2.org/i2b2/services/PMService/` with following

```
http://$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT/i2b2/services/PMService/
```

Example for quick-start:

```
http://i2b2-wildfly:8080/i2b2/services/PMService/
```

Replace `127.0.0.1:8080` with `$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT`

3. Update Proxy Configuration in-place

```
sed -i 's#127.0.0.1:8080/#$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT/#g'
proxy.php
sed -i
's#http://services.i2b2.org#http://$I2B2_CORE_SERVER_HOST:$I2B2_CORE_SERVER_PORT#
proxy.php
```

This will update the ProxyPass and ProxyPassReverse.

Now restart the apache2 service

```
docker exec -it i2b2-webclient bash -c "service apache2 restart"
```

Deployment Complete

You have successfully deployed i2b2 in a production environment. For any troubleshooting, consult the container logs and official documentation.