Article
Evgeny Shvarov · Mar 14, 2019 5m read

Using Docker with your InterSystems IRIS development repository

Hi Community!

I think everyone keeps the source code of the project in the repository nowadays: Github, GitLab, bitbucket, etc. Same for InterSystems IRIS projects check any on Open Exchange.

What do we do every time when start or continue working with a certain repository with InterSystems Data Platform?

We need a local InterSystems IRIS machine, have the environment for the project set up and the source code imported.

So every developer performs the following:

1. Check out the code from repo
2. Install/Run local IRIS installation
3. Create a new namespace/database for a project
4. Import the code into this new namespace
5. Setup all the rest environment
6. Start/continue coding the project

If you dockerize your repository this steps line could be shortened to this 3 steps:

1. Check out the code from repo
2. Run docker-compose build
3. Start/continue coding the project

Profit - no any hands-on for 3-4-5 steps which could take minutes and bring head ache sometime.

You can dockerize (almost) any your InterSystems repo with a few following steps. Let ' s go!

How to dockerize the repo and what does this mean?

Basically, the idea is to have docker installed in your machine which will build the code and environment into a container which will then run in docker and will work in the way developer introduced at a first place. No any "What is the OS version?", "What else did you have on this IRIS installation?".

It's every time a clean page (or a clean IRIS container) which we use to setup environment (namespaces, databases, web-apps, users/roles) and import code into a clean just created database.

Will this "dockerize" procedure harm greatly your current repo?

No. It will need to add 2-3 new files in the root of the repo and following a few rules which you can setup on your own.

Prerequisites

Download and Install docker.

Download and install IRIS docker image. In this example, I will use full InterSystems IRIS preview: iris:2019.1.0S.111.0 which you can download from WRC-preview., see the details.

If you work with the instance which needs a key place the iris.key in the place you will use all the time. I put it on my Mac into Home directory.

Dockerising the repo

To dockerise your repo you need to add three files into the root folder of your repo.

Here is the example of dockerized repo - ISC-DEV project, which helps to import/export source code from IRIS database. This repo has additional Dockerfile, docker-compose.yml and installer.cls I will describe below.

First is Dockerfile, which will be used by docker-compose build command

Dockerfile

This Dockerfile copies installer.cls and the source code from /cls folder of repo into /src foler into the container

It also runs some config settings, which give admin user %All role, infinite password ' SYS ', introduces OS level authorization and runs the %Installer.

What ' s i%Installer?

```
Class App.Installer

{


XData MyInstall [ XMLNamespace = INSTALLER ]

{

<Manifest>

  <Default Name="NAMESPACE" Value="ISCDEV"/>

  <Default Name="DBNAME" Value="ISCDEV"/>

  <Default Name="APPPATH" Dir="/opt/app/" />

  <Default Name="SOURCESPATH" Dir="${APPPATH}src" />

  <Default Name="RESOURCE" Value="%DB_${DBNAME}" />


  <Namespace Name="${NAMESPACE}" Code="${DBNAME}-CODE" Data="${DBNAME}-DATA" Create="
yes" Ensemble="0">
```

```
    <Configuration>

      <Database Name="${DBNAME}-CODE" Dir="${APPPATH}${DBNAME}-CODE" Create="yes" Res
ource="${RESOURCE}"/>

      <Database Name="${DBNAME}-DATA" Dir="${APPPATH}${DBNAME}-DATA" Create="yes" Res
ource="${RESOURCE}"/>

    </Configuration>

    <Import File="${SOURCESPATH}" Recurse="1"/>

  </Namespace>



</Manifest>

}



ClassMethod setup(ByRef pVars, pLogLevel As %Integer = 3, pInstaller As %Installer.In
staller, pLogger As %Installer.AbstractLogger) As %Status [ CodeMode = objectgenerato
r, Internal ]

{

  Return ##class(%Installer.Manifest).%Generate(%compiledclass, %code, "MyInstall")

}



}
```

It creates the namespace/database ISCDEV and imports the code from source folder -/src.

Next is docker-compose.yml file, which will be used when we run the container with docker-compose up command.

```
version: '2.4'

services:

  iris:

    build: .

    restart: always

    ports:

      - 52773:52773

    volumes:
```

```
    - ~/iris.key:/usr/irissys/mgr/iris.key
```

This config will tell docker on what port we will expect IRIS working on our host. First (52773) is a host, second is container ' s internal port of a container (52773)

in volumes section docker-compose.yml provides access to an iris key on you machine inside the container in the place, where IRIS is looking for it:

```
- ~/iris.key:/usr/irissys/mgr/iris.key
```

To start coding with this repo you do the following:

1. Clone/git pull the repo into any local directory.

2. Open the terminal in this directory and run

```
user# docker-compose build
```

this will build the container.

3. Run the IRIS container with your project

```
user# docker-compose up -d
```

Open your favorite IDE, connect to the server on localhost://52773 and develop your success with InterSystems IRIS Data Platforms ;)

You can use this 3 files to dockerize your repository. Just put the right name for source code in Dockerfile, the right namespace(s) in Installer.cls and place for iris.key in docker-compose.yml and use benefits of Docker containers in your day-to-day development with InterSystems IRIS.

#Best Practices #Containerization #Development Environment #Docker #InterSystems IRIS

Source URL:https://community.intersystems.com/post/using-docker-your-intersystems-iris-development-repository