amoni

Amoni Project Team

Nov 08, 2022

CONTENTS

Amoni is a command line tool to manage local development tasks for an Anvil application.

Amoni gives you:

- A preconfigured postgresql server running in a docker container
- A preconfigured Anvil application server running in a docker container
- Simple commands to start and stop those servers
- A preconfigured docker container to run your test suite
- Convenient access to the anvil error log
- Your app available in your browser at port 3030 on your local machine
- Your database server available on port 5432 on your local machine
- Commands to fetch an app from a remote repository and add it to your project either as the main app or as a dependency
- Stub files so that your autocompleter can see the tables available in your app

So your workflow becomes:

```
amoni init
cd <my_new_amoni_project_directory>
amoni start

>>> Wait a while the first time while the server images download
>>> Point my browser at http://localhost:3030
>>> Marvel at how simple that was

amoni test

>>> Sigh with relief at those lovely passing tests

amoni stop

>>> Sup coffee (or beverage of choice)
```

CHAPTER

ONE

INSTALLATION

1.1 Pre-Requisites

Amoni is built on top of Docker which needs to be installed on your local machine.

As a minimum, you will need to install the Docker Engine and Docker Compose.

If you prefer, you can install the Docker Desktop App which also includes the Engine and Docker Compose.

1.2 Install Amoni

The recommended installation method is to use pipx:

```
pipx install amoni
```

If you prefer, you can install amoni into any Python environment using:

```
python -m pip install amoni
```

CHAPTER

TWO

GETTING STARTED

2.1 Create a New Amoni Project

In your terminal, navigate to a directory where you would like to create a new amoni project and run:

```
amoni init demo
```

Amoni will now create a new directory named demo and set up the necessary files and directories within it.

2.2 Start Your Servers

Change into your new directory and run:

```
amoni start
```

The first time you run this command, it will take several minutes to complete as it downloads the images for the servers.

Any subsequent times you run the command, the downloads will be unnecessary and the containers will start immediately.

You should see output ending with:

```
Starting anvil app and database servers
Your app is now available at http://localhost:3030
```

In your browser, navigate to that url and you should see your app running.

If there were any errors, you can open the anvil error log which you will find in the 'logs' directory of your project.

2.3 Run Your Tests

You can now run the test suite for your app using:

```
amoni test
```

The test files are in the 'tests' directory of your project and are run using pytest.

2.4 Stop Your Servers

When you are finished, you can stop your anvil and database servers using:

```
amoni stop
```

CHAPTER

THREE

HOWTO...

3.1 Add Dependencies

The *app* directory in your project must contain a folder for the app you wish to run plus a folder for each of its dependencies. The dependencies must also be defined in *app/config.yaml* in order for the app server to find them correctly.

To add a dependency to your project:

```
amoni app add --as-dependency <URL to the app> <Name of the dependency> <ID of the␣
↪dependency>
```

If the URL you provide is at *anvil.works* (perhaps from within the Anvil IDE), you will need to *Configure SSH* in order for it to work. Amoni does not support username/password authentication.

Amoni will clone the repository from the URL you provide and add it as a git submodule to your amoni project. The submodule will be placed in the app folder.

Amoni will also change the settings in *app/config.yaml* so that the app server will know where to find the dependency you've just added.

Finally, amoni will commit the changes you've just made to your project.

3.2 Change the Main App

The *app* directory in your project must contain a folder for the anvil app you want to run locally and *app/config.yaml* must point to that folder so that the app server starts the correct app.

Amoni includes a 'hello_world' app which is the default app when you first create a project.

To change the default to your own anvil app:

```
amoni app add <URL to your app> <Name of your app>
```

If the URL you provide is at *anvil.works* (perhaps from within the Anvil IDE), you will need to *Configure SSH* in order for it to work.

Amoni will clone the repository from the URL you provide and add it as a git submodule to your amoni project. The submodule will be placed in the app folder.

Amoni will also change the settings in *app/config.yaml* so that the app server will now start the app you just installed and it will parse the app's *anvil.yaml* file and add stub entries to *anvil-stubs/tables/app_tables.pyi* (so that your autcompleter will know what tables exist in your app).

Finally, amoni will commit the changes you've just made to your project.

You can now delete the 'hello_world' directory if you wish.

The next time you run 'amoni start', you should see your new app available in your browser.

3.3 Configure SSH

This guide assumes that you have already created an ssh key pair and added your public key to Anvil.

If that's not the case, you can find useful information on how to do those steps at, for example:

- Creating an SSH key pair
- Adding your key to Anvil

In order for Amoni to know how to use your ssh keys to connect to a remote server (e.g. anvil.works), add the following to your SSH config file (possibly *~/.ssh/config*):

```
Host anvil.works
    IdentityFile <path to your ssh key>
```

Depending on the version of openssh you are using, you may also need the following:

```
Host anvil.works
    IdentityFile <path to your ssh key>
    HostkeyAlgorithms +ssh-rsa
    PubkeyAcceptedAlgorithms +ssh-rsa
```

See this forum post for more information.

3.4 Install Server Packages

The default configuration for the anvil app server contains only the server itself.

If you need additional Python packages available on your server you can add them to the 'requirements.txt' file within the 'app' directory of your amoni project.

The next time you run 'amoni start', any packages listed in 'requirements.txt' will be installed and available.

3.5 Use Autocompletion

Autocompleters generally use the libraries installed in your current python environment to provide autocompletion as you type. You should install the *anvil-uplink* package to add Anvil's classes and functions:

```
pip install anvil-uplink
```

In addition, many autocompleters can use Python stub files to understand the options available to you as you type code.

Amoni makes use of this facility to provide autocompletion for the anvil app_tables module where the tables are defined within an app's *anvil.yaml* file. To enable this feature, install the *anvil-stubs* package into your environment:

```
pip install anvil-stubs
```

Note - This package is provided by the anvilistas team and is not an official Anvil tool.

Amoni will then maintain additional stub files in your project's *anvil-stubs* directory. Those are regenerated whenever you add an app or dependency to your project but you can refresh those at any time manually by running:

```
amoni stubs <name of your app>
```

CHAPTER

FOUR

REFERENCE

4.1 API

class amoni.api.**AmoniRemoteCallbacks**(**args: Any*, ***kwargs: Any*)

amoni.api.**add_submodule**(*url: str*, *path: Path*, *name: str*) → None

Add a submodule to the current repository

Parameters

- **url** – The url of the submodule to add
- **path** – The directory where the submodule should be added
- **name** – The name of the submodule

amoni.api.**build_image**(*name: str*) → None

Build a docker image :param name: The name of the image to pull

amoni.api.**build_theme**(*app: str*) → None

Build the theme for the given app

Parameters

app – The name of the app

amoni.api.**generate_table_stubs**(*app: str*, *target: Path = PosixPath('anvil-stubs/tables/app_tables.pyi')*) → None

Generate stub entries for app tables in anvil.yaml

Parameters

- **app** – The name of the app
- **target** – The stub file where the entries should be added

amoni.api.**init**(*directory: Path*, *app: str*) → None

Initialise an amoni project

Parameters

- **directory** – The full path of the amoni project folder to create
- **app** – The name of folder within the 'app' folder which contains the app to be run

amoni.api.**pull_image**(*name: str*) → None

Pull a docker image from the github registry

Parameters
name – The name of the image to pull

amoni.api.**run_service**(*name: str*) → None

Run a given service

Parameters
name – The name of the service to start

amoni.api.**set_app**(*name: str*) → None

Set the app to be run by the anvil app server

Parameters
name – The name of the app

amoni.api.**set_dependency**(*id: str*, *name: str*) → None

Set an app to be a dependency

Parameters

- **id** – The id of the dependency app
- **name** – The name of the dependency app

amoni.api.**start_service**(*name: str*, *detach: bool*) → None

Start a given service

Parameters

- **name** – The name of the service to start
- **detach** – Whether to detach from the service console

amoni.api.**stop_services**() → None

Stop all amoni services

CHAPTER

FIVE

WHY THE NAME?

It's Greek for anvil and the idea for this library was conceived on a sunny veranda on the island of (Corfu).

- genindex
- search

PYTHON MODULE INDEX

INDEX