How To Use Dokku Plugins to Access Additional Functionality

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: *http://faq.asphosthelpdesk.com/article.php?id=219*

Introduction

Dokku is an application deployment solution that can be used to easily launch code into a production environment. The result is that your application environment will be configured automatically and the necessary software will be installed and implemented.

Dokku adheres to the philosophy that application resources should be handled as separate components from the actual application. This creates a flexible relationship where resources or main application code can be swapped out easily without much additional configuration. Dokku handles outside resources through a plugin system.

In this guide, we will discuss how to utilize Dokku's plugin system to extend the functionality of the project and link your application with the backends that it needs. We will also look into some other plugins that can help you monitor and manage your applications.

Prerequisites

This article assumes that you have a working configuration of Dokku installed using our Dokku one-click installation. You should complete at least up through Step Two in this tutorial to complete your Dokku configuration.

What Kind of Plugins are Available?

Dokku is built to be readily extensible and has a plugin system that is easy to hook into. In fact, many of the operations that some would consider core functionality are handled by plugins in Dokku. This allows the main application to remain slim and modular.

The main thing most people will want out of the plugin system is database access. Through community plugins, Dokku provides access to many popular database systems including MariaDB, PostgreSQL, MongoDB, and Redis. You can find a full list of database
plugins.

Beyond database access, there are plugins for multiple process managers. Dokku runs processes through a "Procfile", which lists the process types that must be run and the commands that are needed to execute them. By default, Dokku only runs "web" processes.

To run other types of processes (like workers), you can use a process manager, which runs the Procfile to execute other process types. There are multiple ones to choose from, but they all basically do the same thing. The current selection includes circus, shoreman, and supervisord. On top starting additional process types, some of these also have the ability to restart crashed applications.

Additionally, there are some other interesting plugins that may help your application run smoother. For instance, there is a plugin called user-env-compile which will provide your build environment with environmental variables that are usually not set until the deploy stage.

General Plugin Instructions

Most plugins are installed and configured using the following general format. They all must be installed into Dokku's plugin directory. Change to that directory now:

```
cd /var/lib/dokku/plugins
```

After that, you download the plugin code with a simple git clone:

```
git clone git_repository optional_alternative_plugin_name
```

After that, you can install the plugins and configure them to your environment with the following command:

```
dokku plugins-install
```

This will grab all of the necessary dependencies using `apt-get` and configure your environment. It will then install any plugins that have not been installed.

You can access Dokku's help system by typing:

```
dokku help
```

Dokku's help system is dynamically generated to include the help files packaged with plugins. As a result, if you install the PostgreSQL plugin for instance, you will get new entries describing its usage:

```
. . .
postgresql:create <app>     Create a PostgreSQL container
postgresql:delete <app>     Delete specified PostgreSQL container
postgresql:info <app>       Display database informations
postgresql:link <app> <db>  Link an app to a PostgreSQL database
postgresql:logs <app>       Display last logs from PostgreSQL container
. . .
```

This means that no matter how many plugins you install, your documentation will still all be in one place. It is always a good idea to run `dokku help` after installing a new plugin.

Example Plugin Workflow

To give you a good idea of how you would work with one of these plugins in a deployment situation, we will go over an example procedure. We will assume that our application is using a PostgreSQL database.

One Time Setup

Before using the PostgreSQL plugin for the first time, we need to download the appropriate files from GitHub. First, change into the plugins directory:

```
cd /var/lib/dokku/plugins
```

Next, go to the project's page to see how they recommend installing it. Some plugins clone their files into a directory with a different name than the GitHub name. The PostgreSQL plugin's page uses the name "postgresql" after the clone command:

```
git clone https://github.com/Kloadut/dokku-pg-plugin postgresql
```

Install the plugin components and their dependencies:

```
dokku plugins-install
```

Application-Specific Steps

After you have installed the plugin, you can use it with every application you want to deploy. Your process will largely depend on if you've already deployed your application or not, as well as the recommended steps for your specific plugin.

For instance, the PostgreSQL plugin allows you to create the database instance prior to deployment. However the MongoDB plugin requires you to deploy your application, create the database, and then repush the application.

The PostgreSQL plugin is pretty flexible. It can handle either situation, but it's one less step if you create the database first.

Creating the Database First

If you are creating the database first and want it to be automatically linked to your application on deployment, you need to select the same name for your database that you plan on using for your application. The general syntax for PostgreSQL is:

```
dokku postgresql:create app_name
```

Dokku will then link any application pushed with the name `app_name` to the database instance.

When you create the database, you will usually receive some information about the connection parameters. In our example, the connection parameters would look something like:

```
-----> PostgreSQL container created: postgresql/app_name
       Host: 111.222.333.444
       User: 'root'
       Password: 'ClUzB4nRhEWgjiVt'
       Database: 'db'
       Public port: 49187
```

If you need to hard code the connection parameters in your app (rarely recommended), you can use this information. You could also use this information to connect to the database on the local machine through the usual database interfaces (psql in this case), and pre-populate data.

In most cases, you will never need to use this information, because upon deployment, Dokku sets an environmental variable called `DATABASE_URL`, which contains a connection string with the appropriate parameters. You can access this variable and use it in your program to maintain flexibility.

If you do need to access this information, you can always recall it by typing:

```
dokku postgresql:info app_name
```

Now that we have a database created for our application, we would push our application to Dokku as normal using git. In the output that follows, towards the end, you should be able to see that your application is linked with your database correctly:

```
remote: -----> App app_name linked to postgresql/app_name database
remote:        DATABASE_URL=postgres://root:ClUzB4nRhEWgjiVt@111.222.333.444/db
```

As you can see, the `DATABASE_URL` variable has been set using the database information we discussed earlier.

If you are experiencing database issues, most plugins maintain their own logs:

```
dokku postgresql:logs app_name
```

Deploying the App First

If you wish to deploy an application with a database with a different name than your application, or if you forgot to deploy the database before deploying the application, you can always link the database afterward.

In this case, you can do one of two things. If you simply forgot to create the database, you can simply issue the normal create database command with a matching name:

```
dokku postgresql:create app_name
```

Afterwards, you can re-push your application because the `DATABASE_URL` variable is created during application deployment.

However, you can also just link it. This is also necessary if you would like to use a different name for you database, you can create it and then link it:

```
dokku postgresql:create database_name
(By this point, your application must be deployed)
dokku postgresql:link app_name database_name
```

This will link the database to the application so that your program will function correctly.

Conclusion

Dokku is designed to take advantage of modular features. This is helps keep the main development efforts focused and provides easy ways to extend the core model. Learning how to take advantage of plugins is essential to get the most functionality out of the software.

If you find that Dokku is missing some functionality that you need, consider making a plugin yourself. If you create a plugin for Dokku, you can edit the Dokku plugins page to feature your plugin.