Deploying to Dokku

Deploy tutorial

Once Dokku has been configured with at least one user, applications can be deployed via a `git push` command. To quickly see Dokku deployment in action, you can use the Heroku Ruby on Rails example app.

```
# from your local machine
# SSH access to github must be enabled on this host
git clone git@github.com:heroku/ruby-rails-sample.git
```

Create the app

Create the application on the Dokku host. You will need to ssh onto the host to run this command.

```
# on the Dokku host
dokku apps:create ruby-rails-sample
```

Create the backing services

When you create a new app, Dokku by default *does not* provide any datastores such as MySQL or PostgreSQL. You will need to install plugins to handle that, but fortunately Dokku has official plugins for common datastores. Our sample app requires a PostgreSQL service:

```
# on the Dokku host
# install the postgres plugin
# plugin installation requires root, hence the user change
sudo dokku plugin:install https://github.com/dokku/dokku-postgres.git

# create a postgres service with the name rails-database
dokku postgres:create rails-database
```

> Each service may take a few moments to create.

Linking backing services to applications

Once the service creation is complete, set the `POSTGRES_URL` environment variable by linking the service.

```
# on the Dokku host
# each official datastore offers a `link` method to link a service to any application
dokku postgres:link rails-database ruby-rails-sample
```

> You can link a single service to multiple applications or use one service per application.

Deploy the app

Now you can deploy the `ruby-rails-sample` app to your Dokku server. All you have to do is add a remote to name the app. Applications are created on-the-fly on the Dokku server.

```
# from your local machine
# the remote username *must* be dokku or pushes will fail
cd ruby-rails-sample
git remote add dokku dokku@dokku.me:ruby-rails-sample
git push dokku master
```

> Note: Some tools may not support the short-upstream syntax referenced above, and you may need to prefix the upstream with the scheme `ssh://` like so: `ssh://dokku@dokku.me:ruby-rails-sample` Please see the Git documentation for more details.

```
Counting objects: 231, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (162/162), done.
Writing objects: 100% (231/231), 36.96 KiB | 0 bytes/s, done.
Total 231 (delta 93), reused 147 (delta 53)
-----> Cleaning up...
-----> Building ruby-rails-sample from herokuish...
-----> Adding BUILD_ENV to build environment...
-----> Ruby app detected
-----> Compiling Ruby/Rails
-----> Using Ruby version: ruby-2.2.1
-----> Installing dependencies using 1.9.7
       Running: bundle install --without development:test --path vendor/bundle --binstubs
vendor/bundle/bin -j4 --deployment
       Fetching gem metadata from https://rubygems.org/...........
       Fetching version metadata from https://rubygems.org/...
       Fetching dependency metadata from https://rubygems.org/..
       Using rake 10.4.2

...

=====> Application deployed:
       http://ruby-rails-sample.dokku.me
```

When the deploy finishes, the application's URL will be shown as seen above.

Dokku supports deploying applications via Heroku buildpacks with Herokuish or using a project's dockerfile.

Skipping deployment

If you only want to rebuild and tag a container, you can skip the deployment phase by setting `$DOKKU_SKIP_DEPLOY` to `true` by running:

```
# on the Dokku host
dokku config:set ruby-rails-sample DOKKU_SKIP_DEPLOY=true
```

Re-Deploying / restarting

If you need to re-deploy (or restart) your app:

```
# on the Dokku host
dokku ps:rebuild ruby-rails-sample
```

See the process scaling documentation for more information.

Deploying with private git submodules

Dokku uses git locally (i.e. not a docker image) to build its own copy of your app repo, including submodules. This is done as the `dokku` user. Therefore, in order to deploy private git submodules, you'll need to drop your deploy key in `/home/dokku/.ssh/` and potentially add github.com (or your VCS host key) into `/home/dokku/.ssh/known_hosts`. The following test should help confirm you've done it correctly.

```
# on the Dokku host
su - dokku
ssh-keyscan -t rsa github.com >> ~/.ssh/known_hosts
ssh -T git@github.com
```

Note that if the buildpack or dockerfile build process require ssh key access for other reasons, the above may not always apply.

Deploying to subdomains

The name of remote repository is used as the name of application to be deployed, as for example above:

```
# from your local machine
# the remote username *must* be dokku or pushes will fail
git remote add dokku dokku@dokku.me:ruby-rails-sample
git push dokku master
```

```
remote: -----> Application deployed:
remote:        http://ruby-rails-sample.dokku.me
```

You can also specify fully qualified names, say `app.dokku.me` , as

```
# from your local machine
# the remote username *must* be dokku or pushes will fail
git remote add dokku dokku@dokku.me:app.dokku.me
git push dokku master
```

```
remote: -----> Application deployed:
remote:        http://app.dokku.me
```

This is in particular useful, then you want to deploy to root domain, as

```
# from your local machine
# the remote username *must* be dokku or pushes will fail
git remote add dokku dokku@dokku.me:dokku.me
git push dokku master
```

```
... deployment ...

remote: -----> Application deployed:
remote:        http://dokku.me
```

Dokku/Docker Container Management Compatibility

Dokku is, at its core, a docker container manager. Thus, it does not necessarily play well with other out-of-band processes interacting with the docker daemon. One thing to note as in issue #1220, dokku executes a cleanup function prior to every deployment.

As of 0.5.x, this function removes all containers with the label `dokku` where the status is either `dead` or `exited`, as well as all `dangling` images. Previous versions would remove `dead` or `exited` containers, regardless of their label.

Adding deploy users

See the user management documentation.

Default vhost

See the nginx documentation.

Deploying non-master branch

See the git documentation.

Dockerfile deployment

See the dockerfile documentation.

Image tagging

See the image tagging documentation.

Specifying a custom buildpack

See the buildpack documentation.

Removing a deployed app

See the application management documentation.

Renaming a deployed app

See the application management documentation.

Zero downtime deploy

See the zero-downtime deploy documentation.