Docker

Docker is a very popular software containerization platform. While it's not difficult to use, it can be a little confusing for folks who are less familiar with containerization.

We'll attempt to walk you through a Docker setup here, but please see the Docker documentation for a more in-depth understanding of Docker fundamentals. If you don't know the implications of the steps you're taking here, you could potentially lose access to your data.

> **A NOTE FOR N00BS:**
>
> This Docker image uses some advanced concepts such as linking, storage volumes, and multiple images, so it may not be a good choice for learning Docker, however if you're willing to put some time into researching the concepts and background of Docker, we'll be happy to help you as best we can.

The easiest way to get started with Snipe-IT on Docker is to use the Docker image we push to Docker Hub:

Shell

```
docker pull snipe/snipe-it
```

If you'd rather grab a specific version of Snipe-IT, use the tag for the release you'd like to use. For example, to use the v3.4.0 release of Snipe-IT, you would use:

Shell

```
docker pull snipe/snipe-it:v3.4.0
```

> **Options Have Changed From v2.x!**
>
> Please note some of the options have changed, and some new options are now required!

To handle the various settings for your containers, you'll create an `env-file`, which is a simple text file with some variables and values that your Snipe-IT `.env` file needs, separated by `=` signs.

You can call this file anything you want, and put it wherever you like, just remember what you called it and where you put it, since you'll need to reference this file when you run your Docker containers, using the `--env-file=<your_docker_env_file_name>` argument. You'll see examples of this in the commands further down the page as you initialize your Docker container.

For the purposes of this documentation, we'll call our file `my_env_file`.

Your `my_env_file` is *not* the same `.env` file Snipe-IT uses directly, but the values you put in it are used to generate the `.env file` that Snipe-IT does use, so what you put in here matters.

Your `my_env_file` should look like this:

my_env_file

```
# Mysql Parameters
MYSQL_ROOT_PASSWORD=YOUR_SUPER_SECRET_PASSWORD
MYSQL_DATABASE=snipeit
MYSQL_USER=snipeit
MYSQL_PASSWORD=YOUR_snipeit_USER_PASSWORD

# Email Parameters
# - the hostname/IP address of your mailserver
MAIL_PORT_587_TCP_ADDR=smtp.whatever.com
#the port for the mailserver (probably 587, could be another)
MAIL_PORT_587_TCP_PORT=587
# the default from address, and from name for emails
MAIL_ENV_FROM_ADDR=youremail@yourdomain.com
MAIL_ENV_FROM_NAME=Your Full Email Name
# - pick 'tls' for SMTP-over-SSL, 'tcp' for unencrypted
MAIL_ENV_ENCRYPTION=tcp
# SMTP username and password
MAIL_ENV_USERNAME=your_email_username
MAIL_ENV_PASSWORD=your_email_password

# Snipe-IT Settings
APP_ENV=production
```

First get a MySQL container running. MySQL 5.6 or earlier are easiest.

Shell

```
docker run --name snipe-mysql --env-file=my_env_file -d -p $(docker-machine ip b2d)::33
```

If your `my_env_file` contains a `MYSQL_USER`, `MYSQL_PASSWORD`, and a `MYSQL_DATABASE`, then the MySQL container will automatically create that database and grant permissions to access it. If you're using an external database, or an already existing container, you'll have to create a database for Snipe-IT to use, and grant a user access to it - and put those credentials in your `my_env_file`.

> **WARNING:**
>
> Newer MySQL containers (5.7 and later, or MariaDB) may run in strict-mode by default, and the initial migrations and application setup will fail in strict mode. If you want to use one of those versions, you need to disable strict mode first!

That should set you up with your database to use. (You can also specify environment variables on the command-line instead of the env-file, but that can get very clunky very quickly; see `docker run --help` for details)

Using a Standalone Database (not a container)

If you're using a separate, standalone database, you can specify the hostname as `MYSQL_PORT_3306_TCP_ADDR`. If your database runs on a different port, you can specify that with `MYSQL_PORT_3306_TCP_PORT`. You don't need to specify the `MYSQL_ROOT_PASSWORD`.

Thus, your config should look as follows:

my_standalone_env_file

```
# Mysql Parameters
MYSQL_PORT_3306_TCP_ADDR=XXX.XXX.XXX.XXX
MYSQL_PORT_3306_TCP_PORT=3306
```

> **NOTE:**
>
> If your Email solution requires its own container, start that container or service. Make sure to expose port 587 for mail submission, and use `--link mail:...`.

Start your Snipe-IT container

First, we'll need to generate an app key. If you already have a Snipe-IT container running, make note of the `APP_KEY`. In older versions of Snipe-IT, it was stored in `app/config/production/app.php`.

Otherwise, you can easily generate a new one, just run the container:

Shell

```
docker run --rm snipe/snipe-it
```

The output should look like:

Text

```
Please re-run this container with an environment variable $APP_KEY
An example APP_KEY you could use is:
base64:D5oGA+zhFSVA3VwuoZoQ21RAcwBtJv/RGiqOcZ7BUvI=
```

Add that `APP_KEY` to your docker env-file (we left a placeholder for it in your starting docker env file, above). **Make sure to include the `base64:` prefix if it is given!**

Next, determine how you want your instance of Snipe-IT to store files. We recommend, for production use, mounting a volume at `/var/lib/snipeit` with directory permissions set to 755 to store user-uploaded files. This way you can upgrade containers without losing your user's file uploads.

You'll also need to decide which port number to 'expose' for the HTTP and HTTPS endpoints. Once you've selected one, make sure to set your `APP_URL` variable in your Docker env-file to reference it.

Finally, decide whether or not you want to have your Snipe-IT container manage SSL for you, or not.

SSL disabled and no file storage volume.

Start your Snipe-IT container:

Shell

```
docker run -d -p YOUR_PORT_NUMBER:80 --name="snipeit" --link snipe-mysql:mysql --env-fi
```

SSL enabled and/or using a file storage volume

Start your Snipe-IT container - but make sure you can "mount" your local copies of your SSL key and SSL certificate onto the container.

Create a directory to store snipe-it data. If you want to use SSL, create a subdirectory called "ssl" within that directory, and put your certificate in it as `snipeit-ssl.crt` and your key in it as `snipeit-ssl.key`.

Shell

```
docker run -d -p YOUR_PORT_NUMBER:80 -p YOUR_SSL_PORT:443 --name="snipeit" --link snipe
```

Email, Management, Access

If you have a separate container running for email, you will also want a `--link` setting for email as well.

That's It!

You can now initialize the application and database by pointing your web browser to: `http://your_ip:YOUR_PORT`, which will automatically redirect you to the setup screen.

For Development

You can build the snipe-it image using the `Dockerfile` at the root directory of Snipe-IT by doing this:

Shell

```
docker build -t snipe-it .
```

Shell

```
docker run -d -v /Path/To/My/snipe-it/checkout:/var/www/html -p $(docker-machine ip b2d
```

Then your local changes to the code will be reflected. You will have to re-run `composer install` -

Shell

```
docker exec -i -t snipeit composer install
```

And you may still need to generate the key with -

Shell

```
docker exec -i -t snipeit php artisan key:generate - -env=production
```

While you're developing, you may need to occasionally run:

Shell

```
docker exec snipeit composer dump
```

To fix the autoloading cache (if, for example, your class names change, or you add new ones...)