Sony Docker Containerization

From INetU Wiki

This article is a work in progress.
Please remove this template from the page when this page is updated.

Contents

Docker

Docker allows you to package an application with all of its dependencies into a standardized unit for software development.

Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run: code, runtime, system tools, system libraries – anything you can install on a server. This guarantees that it will always run the same, regardless of the environment it is running in.

Read more at the Docker (https://docker.com) homepage.

Outline

The basic process has these steps:

1. Prepare the code in gitlab
2. Build the docker image and push to the registry
3. Create if not exists DB

4. Create if not exists OBS
5. Import if exists DB/OBS
6. Run docker container
7. Configure nginx (webserver for devel/stage/prod, edge for prod)
8. Add DNS to cloudflare

Preparing applications

Docker containers are created by first building an image of what the application will look like. We've done most of the work here already in that we have created a Gitlab project in the Sony Music Gitlab (https://gitlab.smehost.net/inetu/wordpress) which contains everything you need to get started. This project is used as the initial codebase for all applications which will be deployed for both the current PaaS offering and the Gen2 IaaS offering that exist today. Once you have the code required to build a docker image, you will use docker build to create a unique image, and push that image up to the SME docker registry.

Getting the code ready

Gen 1

If the site in question is already live at INetU using the Gen1 infrastructure, you will first need to get the site into version control.

More on this later

Gen 2

If you are updating an existing Gen2 site, you will need to merge the latest changes from the inetu/wordpress repo into the project code base (docker/deploy branch).

More on this later

New Sites

For new sites, all you need to do is create a bare repo and merge the contents of the docker/deploy branch of the inetu/wordpress repo. Afterwards, create devel, stage, and prod branches.

More on this later.

Building the image

Prerequisites:

- Code for the site should be in gitlab
- Site should be wordpress (others need more attention)
- sony-build01 deploy key should be active in gitlab

This process will create your image with 2 tags. The first tag is the name of the branch you are building, the second is the actual ref of the current HEAD of the branch. Afterwards, the push command will push all associated tags to the registry.

- Note: Text in ALL_CAPS should be replaced with the appropriate values

1. Log into `sony-build01.inetuhosted.net`
2. Clone the correct branch of the repo (devel/stage/prod) to `/opt/REGION`, or pull from origin if there is already a checkout:

```
/opt/REGION
git clone GIT_URL -b BRANCH
cd HYPHENATED_FQDN
```

3. Check for existing docker images for this site:

```
docker images | egrep "($(basename `pwd`)|IMAGE ID)"
```

4. If there is already a docker image for this site, see if the code is the same using git diff:

```
git diff --quiet @ DOCKER_IMAGE_TAG;RC=$?;if [ $RC -eq 0 ];then ech
```

5. If the code is the DIFFERENT **or** there is no image yet, build the image and tag with the short hashref:

```
docker build -t registry.smehost.net:5000/`pwd | sed 's|/opt/|sme_|
```

6. If the code is the SAME, tag the existing image with the new hash ref also:

```
docker tag DOCKER_IMAGE_ID registry.smehost.net:5000/`pwd | sed 's|
```

7. You should now have an image tagged like this`registry.smehost.net:5000/sme_ca/coldcreekcounty-com:8804723` Now we will tag it with the branch name

```
docker tag -f DOCKER_IMAGE_ID registry.smehost.net:5000/`pwd | sed
```

8. And push the docker image to the registry:

```
docker push registry.smehost.net:5000/`pwd | sed 's|/opt/|sme_|'`
```

Preparing the Content

Since Docker containers are considered to be ephemeral, and the backing storage is not guaranteed to be reused from one run to the next, it's important to leverage Object Storage (OBS) for the uploaded content (media, images, etc) rather than storing them locally with the filesystem. If an application is coming from (or once existed on) OpenShift you can be pretty confident that the required accounts have already been created. If not, you will need to create 3 storage accounts (one per district).

We have created a tool for administration of desman managed assets (called desctl) which makes this very easy. This tool is to be run from the sony-vm-sftp01 server (drop host), and will need to be run once per district. The default behavior assumes devel, so you only need to specify the district (-d) for stage and prod.

```
[ 20:17:50 root@sony-vm-sftp01: ~ ]$ desctl storage create uk/example.cc
Bucket: examplecom-ukdevel
Access Key ID: FFJL1YHE4YQYTASCYQ4Y
Secret Access Key: IDxTHjNNly6SbcVIYJ3wrexcmqXM2fbF2h-iYw==
URL: examplecom-ukdevel.obs.smehost.net
[ 20:18:06 root@sony-vm-sftp01: ~ ]$ desctl storage create -d stage uk/e
Bucket: examplecom-ukstage
Access Key ID: VVR1SNWP6ZKWP0JHDMVQ
Secret Access Key: YXGibxLwKy4H5SmxX6v8SKsHbsmeplkIt6-nCw==
URL: examplecom-ukstage.obs.smehost.net
[ 20:19:23 root@sony-vm-sftp01: ~ ]$ desctl storage create -d prod uk/ex
Bucket: examplecom-ukprod
Access Key ID: YSFEXM6FIAOEAIVGOMLA
Secret Access Key: ufkXJQQyOZvUp0iWhBa1CnbsmUkGpogfAJ1xrQ==
URL: examplecom-ukprod.obs.smehost.net
```

Once all 3 buckets have been created, you can migrate the uploads to the bucket if they aren't already there (unlikely if you created them in the previous step).

Copy wordpress uploads to drop server

For this document, we're going to focus only on devel, but the process is the same for all districts. This command should be run on the nfs server for the appropriate district[s].

```
[root@sony-d-vm-db01 example.com]# FQDN=$(basename $(pwd));REGION=$(base
```

Import to each bucket

For this we will need to jump back over to the sftp server where we can leverage desctl. This tool expects a certain filesystem structure, but it makes uploading and downloading from the OBS cluster consistent for all projects. The example below assumes you only have the devel contents to upload. If you were going to use separate content for each one, you would not need to specify the source (-s) as devel.

```
[ 20:19:23 root@sony-vm-sftp01: ~ ]$ desctl storage import -d devel uk/e
[ 20:19:23 root@sony-vm-sftp01: ~ ]$ desctl storage import -s devel -d s
[ 20:19:23 root@sony-vm-sftp01: ~ ]$ desctl storage import -s devel -d p
```

Each command will output one line per file that exists in the $repo/.inetu/$district/ directory. Files should be stored in this location relative to the docroot and should never be committed to the actual repository. For this reason we've added the .inetu directory to .gitignore.

```
You may also want to make a backup of the database because once you make the switch there will be some
changes to the database after you login, and if any problems crop up it's probably best to have a backout pla
```

Running the Docker container

Docker images built in the previous steps will be used to create new containers on the Gen2 Web Servers. This currently includes the following:

```
---
devel:
  - sony-d-vm-web03.inetuhosted.net
stage:
  - sony-s-vm-web03.inetuhosted.net
  - sony-s-vm-web04.inetuhosted.net
prod:
  - sony-p45press01.inetuhosted.net
  - sony-p45press02.inetuhosted.net
  - sony-p45press03.inetuhosted.net
```

Each time a site is updated, the image will need to be rebuilt, and that new image should be used to run a new version of the existing application.

Running a container is as simple as specifying the image you want to run, setting a restart policy, and defining some environment variables. It's also a good idea to set a name so that you can more easily identify the running application in case you need to troubleshoot something (or remove it in the future). A typical run line for devel would read something along the lines of:

- NOTE: The PORT_NUMBER below is unique to each site and >= 10080. These ports are tracked on SONY Site Status

```
docker run -d -p PORT_NUMBER:80 \
  --name BUCKET_NAME \
  --restart always \
  -e DESMAN_ENV=ENVIRONMENT \
  -e DESMAN_DB_ENV_MYSQL_USER=DB_USER_NAME \
  -e DESMAN_DB_ENV_MYSQL_DATABASE=DB_NAME \
  -e DESMAN_DB_ENV_MYSQL_PASSWORD=DB_PASSWORD \
  -e DESMAN_DB_PORT_3306_TCP_ADDR=DB_HOST_ADDR \
  -e DESMAN_OBS_KEY_ID=OBS_KEY_ID \
  -e DESMAN_OBS_KEY_SECRET=OBS_KEY_SECRET \
  -e DESMAN_OBS_BASE_URL=http://obs.smehost.net \
  -e DESMAN_OBS_EXT_URL=http://cdn.smehost.net \
  registry.smehost.net:5000/sme_REGION/HYPEHNATED_FQDN:ENVIRONMENT
```

the -p option above will bind the host port 10082 to the container port 80. All containers are listening on port 80 internally but will need to be on a unique port. Normally docker will simply find a port and bind to it, but that would make a static nginx configuration difficult since the port may change with each restart (or rebuild). For consistency sake we are going to bind the same port in all 3 districts for a site. This information is being tracked on the SONY Site Status page. Make sure to update it when you create a new site.

The tag name you should use is the REF from the repo when you built the image. This was pushed at the time you pushed your image to the registry. Theoretically you can also use the $DESMAN_ENV for your current branch, but this is dangerous as it's possible for the devel tag to be behind the repo while this is still being done manually. You can list the current tags for a particular repository via the docker registry api:

```
root@sony-build01:~# curl -sL https://registry.smehost.net:5000/v1/repos
{
    "b5c27ef0": "a5e3034eb1e9faafc8a8c1217573f6362962b4e45c028c3909a3fa2
    "devel": "a5e3034eb1e9faafc8a8c1217573f6362962b4e45c028c3909a3fa28c8
}
```

You can then line up the tags with the branch HEAD using git rev-parse HEAD on the current branch (in a working copy).

Configure NginX

Once the application is up and running, you will need to configure Nginx on the web server to begin proxying requests to the container instead of the local instance of php-fpm. This can be done using a simple sed statement along the lines of:

```
sed -e 's/SITENAME/$SITENAME/g;s/TCP_PORT/$PORT/g;s/FQDN/$FQDN/g;s/REGI(
    /etc/nginx/vhost.d/docker.template > /etc/nginx/vhost.d/$FQDN.conf
```

Be sure to replace the variables where appropriate, test the configuration, and reload nginx. From now on all requests that go to that host for the site in question will be directly proxied to the container. You can test this using curl or your browser.

Add DNS to cloudflare

1. Go to the **smehost.net** domain and click **DNS**
2. Add a **CNAME** for **BUCKET_NAME.i** -> **edge-ENV_LETTER.smehost.net**

An example would be: **coldcreekcountycom-cadevel.i.smehost.net. 300 IN CNAME edge-d.smehost.net.**

Categories: Work In Progress | SONY
