

Ubersmith and Docker

Architecture

What technology is in use?

Ubersmith is using [Docker](#) to run our software. Everything from our database backend to our search engine is run inside Docker containers.

Filesystem Locations

Where is everything?

The `/usr/local/ubersmith` directory has the following subdirectories:

- `/app/`
 - `/custom/`

The `custom` directory holds custom files, such as service, order, and device modules. Any file in this directory is copied to the webroot of the `web` container when the container starts or restarts. The subfolders `include/{service,order,device}_modules`, serve as a guide for the locations of custom module class files. The subfolder `plugins` is used for the storage of custom plugins.

- `/conf/`
 - `/cron/`

This directory is empty by default. Any files in this directory override the contents of `/etc/cron.d` in the `cron` container. This is used for rare situations when the default Ubersmith cron tasks need to be overridden.

- `/httpd/`

This directory includes the `sites-enabled` folder, which in turn includes the Apache virtual host configuration(s) for Ubersmith. If multiple brands need to be supported, additional virtual host definitions are created in this directory.

- `/mysql/`

This directory contains `ubersmith.cnf` which configures and/or overrides specific MySQL variables. Variables are updated when the `db` container restarts.

- `/ssl/`

This directory contains `.key` and `.pem` files which should be replaced by the Apache key and certificate used for the Ubersmith instance. The `web` and `mail` containers need to be restarted when these files are updated. These file names are important; services with unexpected file names will fail to start or will fail with TLS disabled. To include a CA bundle, first place the leaf certificate in the `.pem` file, then add the CA bundle. Restart the `web` and `mail` containers after modifying certificates.

- `/logs/`

This directory contains the Apache Access and Error in the `ubersmith` directory. As of Ubersmith 4.1.6, logs for containers are redirected to syslog, and can be found in `/var/log/ubersmith`.

Where are my custom files?

After adding your custom files to the `/app/custom` folder, destroy and recreate the Ubersmith containers so they can be added to the system. This is only necessary when adding new files; changes to existing files will be reflected immediately.

```
# cd /usr/local/ubersmith
# docker-compose rm -sf
# ./ubersmith_start.sh
```

On this page:

On this page:

- [Architecture](#)
 - [What technology is in use?](#)
- [Filesystem Locations](#)
 - [Where is everything?](#)
 - [Where are my custom files?](#)
- [The Containers](#)
- [Docker Compose](#)
 - [Starting Ubersmith with Docker Compose](#)
 - [Stopping Ubersmith with Docker Compose](#)
 - [Restarting Individual Container\(s\) with Docker Compose](#)
 - [Checking Status with Docker Compose and Expected Output](#)
 - [Configuring an External Database Server](#)
 - [Making a Backup](#)
- [Troubleshooting](#)
 - [Troubleshooting Container Startup](#)
 - [Restarting Docker](#)
- [Related Topics](#)

The Containers

When containers are running, they are prefixed with `ubersmith_` and suffixed with `_1`. For example, the running `web` container is named `'ubersmith_web_1'`. When using `docker-compose`, the containers can be references by their short names, for example: `docker-compose start web`.

Container Name	Function	Healthcheck	Listens On
Core			
mail	Runs the postfix MTA; handles incoming mail directed at the Support Manager and outgoing mail from Ubersmith.	postfix status	Public: 0.0.0.0:25/tcp Private: 25/tcp
cron	Runs crond; executes Ubersmith's polling cron task, and the daily invoicing task.	supervisorctl status cron	<i>None</i>
web	Runs Apache; serves the Ubersmith HTTP user interface.	curl https://localhost/version.ini	Public: 0.0.0.0:80/tcp, 0.0.0.0:443/tcp Private: 80/tcp, 443/tcp
php	Runs php-fpm; renders PHP code into HTML for use by the web container.	cgi-fcgi -bind -connect 127.0.0.1:9000	Private: 9000/tcp
db	Runs Percona Server (MySQL drop-in replacement).	pt-table-checksum	Public: 127.0.0.1:3306/tcp Private: 3306/tcp
solr	Runs Apache Solr.	Solr PING request	Private: 8983/tcp
backup	Executes Percona xtrabackup and stores a backup in the <code>backups</code> Docker volume. This container is run on demand.	<i>None</i>	<i>None</i>
rwhois	Runs the Ubersmith RWhois server. Disabled by default.	nc localhost 4321	Public: 0.0.0.0:4321/tcp
redis	Runs Redis; used for session handling and advisory locking.	redis-cli -h localhost ping	Private: 6379/tcp
pmm	Runs Percona Monitoring and Management. Disabled by default.	<i>None</i>	Public: 0.0.0.0:8443/tcp
Appliance			
app_web	Runs Apache; serves the Ubersmith Appliance interface.	wget https://localhost/version.ini	Public: 0.0.0.0:8080/tcp, 0.0.0.0:8443/tcp Private: 80/tcp, 443/tcp
app_cron	Runs crond; executes the Appliance's polling tasks.	supervisorctl status cron	<i>None</i>
app_db	Runs Percona Server (MySQL drop-in replacement).	pt-table-checksum	Public: 127.0.0.1:3307/tcp Private: 3306/tcp

Docker Compose

Ubersmith uses the Docker Compose utility to manage using many containers. There are two files in `/usr/local/ubersmith`, `docker-compose.yml` and `docker-compose.override.yml`. Any custom changes specific to your deployment should be made in the `override` file, as the main `docker-compose.yml` may be updated when Ubersmith is upgraded. The commands below assume Ubersmith has been installed in `/usr/local/ubersmith`; if Ubersmith has been installed in a different directory, the option `-p ubersmith` will need to be passed to the following `docker-compose` commands:

Starting Ubersmith with Docker Compose

```
# cd /usr/local/ubersmith
# docker-compose up -d
```

Stopping Ubersmith with Docker Compose

```
# cd /usr/local/ubersmith
# docker-compose stop
```

Restarting Individual Container(s) with Docker Compose

```
# cd /usr/local/ubersmith
# docker-compose restart web mail
```

Checking Status with Docker Compose and Expected Output

Note: The backup container is only used to generate backups, and is not expect to have an Up status.

```
# cd /usr/local/ubersmith
# docker-compose ps
```

Name	Ports	Command	State
ubersmith_cron_1 (healthy)		/usr/bin/supervisord -c /e ...	Up
ubersmith_db_1 (healthy)	127.0.0.1:3306->3306/tcp	docker-entrypoint.sh mysqld	Up
ubersmith_mail_1 (healthy)	0.0.0.0:25->8025/tcp	/entrypoint.sh	Up
ubersmith_php_1 (healthy)	9000/tcp	/usr/sbin/php-fpm5.6 -F	Up
ubersmith_redis_1 (healthy)	6379/tcp	docker-entrypoint.sh redis ...	Up
ubersmith_rwhois_1 4321->4321/tcp		/usr/sbin/xinetd -pidfile ...	Up (healthy)
ubersmith_solr_1 (healthy)	8983/tcp	/bin/bash -c /opt/solr/bin ...	Up
ubersmith_web_1 sh 80->80/tcp		/ubersmith. Up (healthy)	0.0.0.0:443->443/tcp, 0.0.0.0:

Configuring an External Database Server

Ubersmith's containers use environment variables specified in `docker-compose.yml` and `docker-compose.override.yml` to set the details for the database connection to use. By default, Ubersmith expects to connect to the default database container (db). By modifying the environment variables and recreating the Ubersmith containers, you can use a different database host. In the `environment` section of the web service in `docker-compose.override.yml`, set the following values:

environment:

```
MYSQL_USER: your_database_username

MYSQL_DATABASE: your_database_name

DATABASE_HOST: database.yourdomain.com
```

With these values configured, recreate the Ubersmith containers:

```
# cd /usr/local/ubersmith
# docker-compose rm -sf
# ./ubersmith_start.sh
```

Making a Backup

Starting the `backup` container runs Percona xtrabackup and takes a live database snapshot. The output of the process is displayed to the terminal. The backup is stored uncompressed in the `/usr/local/ubersmith/backup` directory.

```
# cd /usr/local/ubersmith
# docker-compose up backup
```

The database container is configured to listen on the loopback interface (127.0.0.1:3306) of the host machine by default. The database credentials can be retrieved from `/usr/local/ubersmith/docker-compose.yml`.

These access credentials can be used to make backups using other utilities (ex. `mysqldump`) or third party applications.

Troubleshooting

Troubleshooting Container Startup

If a container won't start, contact Ubersmith support for assistance. Also, the following are steps to troubleshoot the issue.

For example, if the `web` container isn't starting because of a problem in the Apache configuration in `/usr/local/ubersmith/conf/httpd`:

```
# cd /usr/local/ubersmith
# docker-compose up web
(output follows, possibly with an error message indicating the issue)
(container exits)
```

Once the issue is resolved, start the container normally using `up -d`:

```
# cd /usr/local/ubersmith
# docker-compose up -d web
```

Note: The command above only brings up the `web` container; for this example it is assumed that everything else is running normally.

Restarting Docker

When all else fails, restarting Docker completely may resolve networking and/or firewall issues:

```
# service docker restart
```

The above recreates the firewall rules Docker needs for both outside world and inter-container communication. Ubersmith's containers are configured to start automatically when Docker does.

Related Topics

[Implementation Page](#)