CommCare Sync Ansible

Dimagi

Aug 11, 2022

CONTENTS:

Documentation for the commcare-sync-ansible project, which contains tools for setting up and deploying commcare-sync.

We use the Ansible automation framework to provide one-click set up and management of the commcare-sync application a server.

This documentation site also includes a guide for system administrators

CHAPTER

ONE

WHAT'S INSTALLED

The following are installed by default.

The above will all be configured, and after install, commcare-sync should be properly set up.

Note that some of these can be enabled/disabled based on variables. For example, setting `superset_enabled: no` in your environment will prevent Superset from being installed.

CHAPTER

TWO

GETTING STARTED

To get started, follow the guides below to install and/or develop CommCare Sync.

- *Set up local development environment*
- *Set up CommCare Sync for production*

CHAPTER

THREE

INSTALL COMMCARE SYNC FOR PRODUCTION

This page outlines the process you need to follow in order to set up a CommCare Sync instance on a production machine.

3.1 Choose a location for the control

While it is possible to run the ansible playbooks from anywhere, it is recommended to run them *on the server you are setting up*. This will ensure consistency and also streamline the install process.

These instructions assume a set up where the ansible control and playbooks are run on the server being set up.

3.2 Install Ansible

From the Ansible Installation Guide, run the following on your control machine.

```
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

3.3 Install CommCare Sync

3.3.1 Set up user accounts

On the server, create an account for the ansible user to use. This can be done with the following command, answering the prompts.

```
sudo adduser ansible
```

3.3.2 Clone the repository

```
git clone https://github.com/dimagi/commcare-sync-ansible.git
```

3.3.3 Prepare your environment

Initialize Inventory Folder

Production environments live in the `inventories/` folder. First create a new inventory folder for your environment (we'll use `myproject` as an example). This is where your project-specific configuration will live.

```
mkdir -p inventories/myproject
cp -r inventories/example/* inventories/myproject
```

Update Inventory Files

Edit the following files with your project-specific changes:

- `inventories/myproject/hosts.yml`
  - `vars.env`: your environment name
  - `hosts.local1.ansible_user`: username of your ansible user
  - `hosts.local1.ansible_host`: hostname/public IP where your ansible user lives
- `inventories/myproject/group_vars/commcare_sync/vars.yml`
  - `public_host`: your commcare-sync hostname
  - `superset_public_host`: your superset hostname
  - `superset_enabled`: specifies whether Superset should be installed

Ansible Vault

Production environments should use Ansible Vault to manage secrets. That page has lots of details about editing and using files with Vault.

The example environment includes a vault file which you should remove:

```
rm inventories/myproject/group_vars/commcare_sync/vault.yml
```

Initial Vault Setup

The following one-time setup is used to generate keys / files for Ansible Vault.

Generate Vault Key

```
openssl rand -base64 2048 > ~/myproject-ansible-vault
```

Create Vault Vars File

```
ansible-vault create --vault-password-file ~/myproject-ansible-vault ./inventories/
↪myproject/group_vars/commcare_sync/vault.yml
```

Add your secrets here, e.g.

```
# My Project Vault File

vault_default_db_password: <secret1>
vault_django_secret_key: <secret2>
vault_mapbox_api_key: <secret3>
vault_django_secret_key: <secret4>
```

(You can generate a good random key from a command line:)

```
$ python3 -c 'import string
import secrets
chars = string.ascii_letters + string.digits
key = "".join(secrets.choice(chars) for x in range(64))
print(key)'
```

You run the following to edit the file later:

```
ansible-vault edit --vault-password-file ~/myproject-ansible-vault ./inventories/
↪myproject/group_vars/commcare_sync/vault.yml
```

SSH access

Assuming you are running on AWS, *Copy the AWS private key to `~/myproject.pem` on your local machine.*

You may also need to change permissions on it.

```
chmod 400 ~/myproject.pem
```

Test it's working:

```
ssh -i ~/myproject.pem ubuntu@my.server.ip
```

Set hostname

On the remote server

```
sudo hostnamectl set-hostname myproject-server
```

3.3.4 Run the installation scripts

```
ansible-galaxy install -r requirements.yml
ansible-playbook -i inventories/myproject commcare_sync.yml --vault-password-file ~/
↪myproject-ansible-vault -vv
```

3.3.5 Settting up HTTPS

HTTPS set up is currently not supported by this tool. To set up SSL, login to your machine and install certbot:

```
sudo apt install certbot python3-certbot-nginx
```

Then run:

```
sudo certbot --nginx
```

and follow the prompts.

You'll need to repeat this process for each site (e.g. commcare-sync and superset). You may also need to open up port 443 on AWS or your firewall.

After setting up HTTPS you should set `ssl_enabled=yes` and `superset_ssl_enabled=yes` in your `vars.yml` file, otherwise running a full `ansible-playbook` will undo the changes!

You can set `ssl_enabled=yes` and `superset_ssl_enabled=yes` to prevent this from happening after enabling SSL support.

CHAPTER

FOUR

MAINTENANCE

This shows you how to deploy CommCare Sync in steady state as well as some other useful tasks.

4.1 Steady-State Deploy

For existing environments you should get the relevant `myproject-ansible-vault` and `myproject.pem` files from a project team member and jump straight to deployment.

To deploy, run the following *from your local machine*.

```
ansible-playbook -i inventories/myproject commcare_sync.yml --limit myserver --vault-
↪password-file ~/myproject-ansible-vault -vv --tags=deploy
```

You can also modify the fabric example in the app repository to deploy.

4.2 Other tasks

Some other things you might want to do on production.

4.2.1 Setting up passwordless SSH

Create `.ssh` directory in the user's home and make sure to set the permissions to 755.

```
mkdir ~/.ssh
chmod 755 ~/.ssh
```

Add an `authorized_keys` file and make sure to set permissions to 700.

```
touch ~/.ssh/authorized_keys
chmod 700 ~/.ssh/authorized_keys
```

4.2.2 Working with Superset

In order to run any superset native commands (for example `superset db upgrade`) you must enter the superset environment and *manually run the postactivate script*.

```
source ~/www/.virtualenvs/superset/bin/activate
source ~/www/.virtualenvs/superset/bin/postactivate
```

CHAPTER

FIVE

DEVELOPMENT

Development for this tool is set up to run on a Vagrant VM. To develop and test locally, follow the instructions below.

5.1 Install Ansible

From the Ansible Installation Guide, run the following on your control machine.

```
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

5.2 Set up Vagrant

Follow this guide

5.3 Deploy to local VM

From your local machine

```
vagrant up --provision
```

This should download and deploy the latest commcare-sync code to a local Vagrant VM.

If everything is successful you can load http://192.168.11.10/ in a browser and get going!

CHAPTER

SIX

MIGRATING A SERVER

Here are the approximate steps to migrate a server from one environment to another.

1. Stand up a new server with ansible. For ease of migration, it is recommended to use the same secrets as the previous environment unless there is any reason to believe there was a compromise/breach.
2. Back up the commcare-sync, superset, and data export tool databases to pgdump files.
3. Transfer the pgdump files to the new server.
4. Restore the commcare-sync and superset databases to the new server.
5. Copy the export config files from the old server to the new server.
6. Test.

Backup commands

(These commands are run on the old server.)

```
pg_dump -U commcare_sync -h localhost -p 5432 commcare_sync > ~/sync-db-backup.pgdump
pg_dump -U commcare_sync -h localhost -p 5432 superset > ~/superset-db-backup.pgdump
# add any other DBs created where the data might actually be housed
```

Copy commands

(These commands are run on the new server.)

```
scp oldserver.dimagi.com:sync-db-backup.pgdump ./
scp oldserver.dimagi.com:supeset-db-backup.pgdump ./
# other DBs here
```

Restore commands

(These commands are run on the new server.)

First you have to delete and recreate the databases that were created by ansible:

```
sudo -u postgres dropdb commcare_sync
sudo -u postgres dropdb superset
createdb -U commcare_sync -h localhost -p 5432 commcare_sync
createdb -U commcare_sync -h localhost -p 5432 superset
# need to create other DBs here
```

If you get errors about active connections when dropping databases try stopping all processes with `supervisorctl stop all` and killing any other active connections in a `psql` shell with something like the below:

```
SELECT pg_terminate_backend(pg_stat_activity.pid) FROM pg_stat_activity WHERE pg_stat_
↪activity.datname = 'superset' AND pid <> pg_backend_pid();
```

Next you can restore them individually:

```
psql -U commcare_sync -h localhost -p 5432 commcare_sync < sync-db-backup.pgdump
psql -U commcare_sync -h localhost -p 5432 superset < superset-db-backup.pgdump
# restore other DBs here
```

Copying config files

On the old server, in the `~www/commcare-sync/code_root` folder:

```
tar -zcf ~/commcare-sync-media.tar.gz media/
```

On the new server, in the `~www/commcare-sync/code_root` folder:

```
tar -xzf ~/commcare-sync-media.tar.gz
```

CHAPTER

SEVEN

SYSTEM ADMINISTRATION

7.1 Philosophy

CommCare Sync deployments use an "infrastructure as code" philosophy which means all configuration is documented in code files. See the sections below to access everything needed to configure the system.

7.2 Services

The complete list of services is available in the roles/commcare_sync/tasks/main.yml file.

The most important ones are summarized on the *what's installed page*.

7.3 Common Tasks

Some of the common tasks needed to manage an environment.

7.3.1 Enabling/Disabling Public Sign Ups

Sign ups are currently configured via the Django `ACCOUNT_ADAPTER` setting. To enable anyone to sign up, you should set it to `EmailAsUsernameAdapter`. To disable public account creation, set it to `NoNewUsersAccountAdapter`. This behavior is controlled by the `django_allow_public_signups` Ansible variable.

If public sign ups are disabled, then only superusers can create new accounts, via the Django admin UI or command line.

7.3.2 Deploying Changes

You may wish to deploy updates to the server, for example to pull the latest changes from the CommCare Sync code.

The command to deploy updates is:

```
ansible-playbook -i inventories/yourapp commcare_sync.yml --vault-password-file /path/
↪to/password/file -vv --tags=deploy
```

Changes can also be deployed from the CommCare Sync codebase itself by following the instructions in the README.

7.3.3 Accessing the Virtual Environment

To access the virtual environments for commcare sync and superset, run the following commands:

```
source <venv>/bin/activate
source <venv>/bin/postactivate
```

On most servers the commcare-sync is at `~/www/.virtualenvs/commcare-sync` and the superset is at `~/www/.virtualenvs/superset`.

The second command is required to set the project-specific environment variables to the correct values.

7.3.4 Other Useful commands

7.3.5 Checking for Stuck Exports

The logs for running exports don't show up in the UI until they complete. The easiest way to see if an export is still running or if it is stuck/was killed is to login to the server and just run a command to see what "commcare-export" processes are running. E.g.

```
ps -ef | grep commcare-export
```

7.4 Database Management

Because CommCare Sync does not provide direct access to the underlying databases, it is common to have to perform database management tasks, for example, creating new databases, or dropping existing databases or tables.

Database management can be done by logging into the server and getting a postgres shell:

```
psql -U [postgres user] -h localhost -p 5432
```

You will need the postgres username and password from the secrets file, or by finding it on the server e.g. in the django commcare_sync/local.py file. You can also run `./manage.py dbshell` in the CommCare Sync virtual environment.

CHAPTER

EIGHT

DIMAGI-MANAGED ENVIRONMENTS

Dimagi maintains several production commcare-sync environments by making use of a submodule through which the different configurations are managed. This page outlines the details on working with this submodule.

8.1 Initializing the submodule

Dimagi inventories can be accessed by running `git submodule update --init` after cloning the repository. This requires access to the `commcare-inventories` repository on Github. The config files will be loaded in the `/inventories/dimagi` folder.

To work on a Dimagi-managed production environment, all the production instructions are the same, but the root path everywhere must be changed from `inventories/` to `inventories/dimagi/`. Additionally, changes will need to be pushed to the separate `commcare-inventories` private repository, and then committed to the main repository by updating the submodule reference.

8.2 Using the commcare-inventories submodule in production

When referencing the submodule in production environments (on a server), you should use deploy keys.

Follow the instructions there, making sure to run `ssh-keygen` on the server you want to have access.

After adding a deploy key to the `commcare-inventories` repository you should be able to update the submodule as normal.