Deployment Service - Developer Guide | Spree Commerce

The Spree Deployment Service is a free tool that allows you to configure and deploy the Spree Software Stack on any dedicated cloud, virtual private or physical server.

The service can be configured to create single and multi-server configurations on a wide range of Linux based operating systems, and supports deploying demo Spree applications or generating Capistrano recipes for deploying your own application.

Understanding server roles

The Deployment Service supports a wide range of configuration topologies by allowing you to choose individual roles for each server within your deployment.

Each role installs and configures different software packages on the target server, and a server can have one or more roles applied.

Available roles currently include:

- **Application Server** - Installs and configures Nginx, Ruby, and more. Every deployment requires at least one Application Server, but can have as many as required.
- **Database Server** - Installs and configures MySQL, Ruby. Only one Database Server can be configured for a deployment.
- **Utility Server** - Installs and configures Ruby. A deployment can have as many Utility Servers as needed, but a Utility Server role cannot be shared with any other role. Utility Servers generally run Delayed Job workers, Solr indexes, Redis or Memcache services but these applications must be configured manually and will not be installed by the Deployment Service.

- **Load Balancer** - Can only be applied to an Application Server and configures Nginx to load balance Rails requests to all applications servers unicorn back ends. The Load Balancer will handle all requests for static assets (stylesheets, JavaScript files, images, etc).

Every server will get some basic configuration, such as Rails Environment environment variable, placeholder directories for your Spree application, automatically generated database.yml and Procfile files, and more.

Typical server role combinations

For a **Single Server** deployment, you would select **Application** and **Database** as the two roles for your single server.

For **Multi Server** deployments, you would generally select the following:

- At least one dedicated **Application Server**.
- One dedicated **Database Server**.
- One application server can be designated as the **Load Balancer** if you have more that one application server (and you are not using a third party load balancing service).
- Any number of **Utility Servers** as required by your application.

Using the deployment service

The deployment service can be accessed by via the Stores tab under My Account.

Each Store can have one or more “deployments” configured (for production, staging or qa environments), and each deployment can have one or more servers.

To create a deployment:

1. Add a new Store or click on an existing store, and choose the **“Add Deployment Service”** link.

Each **deployment** has a number of global parameters which affects all servers configured within that environment including:

- **Operating System** - The Deployment Service currently supports Ubuntu 12.04 LTS.
- **Rails Environment** - Sets the server wide rails environment, default ‘production’.

- **Ruby Version** - Select the preferred Ruby version you would like installed.
- **Spree Version** - If you choose to install a sample application (see below), the Spree version selected here will be the one used by the sample app. If you choose to deploy your own custom application,\ the Spree Version affects the Capistrano Recipe that gets generated.
- **What would you like deployed** - Choose "A Sample Application" if you do not have a Spree application ready to deploy, or "My own custom Spree application" if you do.
- **Git URL for your application** - Only available when deploying your own application and this value is only used to complete your generated Capistrano Recipe.

2. Once you've configured the global environments details, click "Add" to save the configuration.

Now you need to provide details on each **server** that's included in the environment:

- **Fully Qualified Domain Name** - This should be something like `app1.test.com` or `db1.test.com`, it does not need to be registered or resolvable it is purely used for naming purposes.
- **IP Address** - It's important to supply the correct IP address as it's referenced in several locations within the configuration and supplying an invalid value will result in problems with the configuration. You can use internal / private addresses for your servers as the only requirement is that all servers can communicate with each other using the addresses supplied.
- **Unicorn Workers** - This value is only required for servers with the **Application Server** role, and controls how many worker processes the unicorn master process will fork on startup. Setting this value correctly is important, see the Deployment Options Guide for details on memory requirements / workers.
- **Roles** - Select the required roles for the server, see the **Understanding Server Roles** section above for more details.

3. When you've added the details for all the servers in your deployment, you should then change the **Configuration Status** from "New" to "Complete".

This will upload your server configurations to our deployment server, which may take a few moments. Refresh the page occasionally until you see each servers **Status** change from "Waiting for update from configuration server." to "Ready to Deploy".

4. Now each server will have a **Initialize Configuration** command that must be copied and executed on the relevant server.

This needs to be done as root user, and can take several minutes to complete.

It's important to be aware that while all the **Initialize Configurations** commands appear similar they are server specific so be sure to run the correct

command on each server.

The **Initialize Configuration** command should only be run on servers that are intended to be dedicated Spree servers, it reconfigures large parts of the server to fulfill it's assigned roles and many affect any other applications, configuration or data already present on a server.

5. When the initialize configuration command completes the server will report it's result and you should see each server's Status eventually update to **"Reported as 'changed'"**.

If an error occurs during the build the status might not update or it may update as **"Reported as 'failed'"**, in either event you should attempt to run the **Update Configuration** command listed.

6. Finally when all servers have reported as either **'changed'** or **'unchanged'** the final step is to run the **Update configuration** command to set your site specific database passwords.

The command is generally something like:

```
$ FACTER_db_pass=YOUR_DB_PASSWORD puppet agent –test
```

It's important to substitute YOUR_DB_PASSWORD with the actual password you require.

7. Congratulations you now have a fully configured environment ready for use.

If you choose to deploy a "Sample Application" then you should be able to browser to your Application Servers IP address to view the store running.

If you are deploying your own application, please refer to the Deploying your application on Ubuntu, Deploying your application on Heroku, or Deploying your application on Ninefold Guides for more.

Application Process

The Spree Deployment Service uses Foreman & Upstart to define and manage all the processes required by your application.

Using Foreman

All application processes are managed using Foreman and a default Procfile is generated for you in `/data/spree/shared/config/Procfile`

By default this includes a single Unicorn master process, the unicorn workers are configured via the Deployment Server UI.

Default Procfile

Default Procfile contents:

```
$ web: bundle exec unicorn_rails -c /data/app_name/shared/config/unicorn.rb -p $PORT -E ENV
```

The provided Capistrano deployment script symlinks the Procfile into location after each deploy.

Capistrano Procfile Snippet:

```
namespace :deploy do
  desc "Symlink shared configs and folders on each release."
  task :symlink_shared do
    run "ln -nfs #/config/database.yml #/config/database.yml"
    run "ln -nfs #{shared_path}/config/Procfile #{release_path}/Procfile"
  end
end

after 'deploy:update_code', 'deploy:symlink_shared'
```

It’s important to remember to not edit the Procfile above directly, any changes to this file will be reset automatically by our configuration management system. See the “Customizing Processes” section below for more details on how to correctly change this file.

Upstart

After each deployment of your application the Procfile is exported to Upstart compatible configuration files which the operating system uses to manage your application’s processes.

Upstart is responsible for starting, stopping and restarting the processes and will detect if a process has stopped for some reason and automatically restart it. Upstart does not monitor for memory size growth, and will not restart leaky processes automatically.

Using Upstart you control the processes of application in a number of ways:

All Processes

You can use following commands to start, stop or restart all the processes in your application at once:

```
$ sudo stop spree
$ sudo start spree
$ sudo restart spree
```

Pro

Processes by type

To control all processes of a specific type you can use:

```
$ sudo start spree-web
$ sudo stop spree-worker
$ sudo restart spree-solr
```

Individual Processes

You can control individual processes by prepending the process number at the end:

```
$ sudo start spree-web-1
$ sudo stop spree-worker-3
$ sudo restart spree-solr-2
```

The process number is the concurrency number specified in the foreman export, and not the PID.

Customizing Processes

To add additional processes such as Delayed Job workers to your application, complete the following steps:

- Copy the contents of the default Procfile and save it in the root our application.
- Make any additions or changes you require.
- Remove the Procfile symlink link line from your config/deploy.rb file.
- Commit your changes and deploy the application.
- When your application is deployed the new processes will be automatically started.

Sample Procfile containing Delayed Job workers:

```
$ web: bundle exec unicorn_rails -c /data/app_name/shared/config/unicorn.rb -p $PORT -E production
$ worker: bundle exec rake jobs:work
```

Changing process concurrency

By default the provided Capistrano script will export a single process for each entry in Procfile, for some process types you want to increase the number

of processes started, for example Delayed Job workers. You can customize the number of process exported for each process type by editing the foreman:export task in the Capistrano script.

Custom Capistrano `deploy.rb` - creating 3 worker processes:

```
namespace :foreman do
  desc "Export the Procfile to Ubuntu's upstart scripts"
  task :export, :roles => :app do
    run "cd #{current_path} && bundle exec foreman export upstart /etc/init -a #{application} -c worker=3 -u spree"
  end
  …
end
```

The `-c` (or `--concurrency`) switch accepts the process type and an integer for the number of processes to start up. You specify mulitple types by comma separating as follows: `processname=2,processname=4`

It's important to not export more than one :web process when using the default unicorn configuration, as this refers to the master process and not the workers, the worker count can be configured via the Deployment's Service UI.

Configuration & Customization

Configuration Management

All servers configured with the Spree Deployment Service are configured as clients to our public Puppet server. Puppet is a ruby based configuration management system (similar to Chef), and using it allows us to centrally control and update all configurations to ensure your server is always running the best possible configuration.

Using Puppet also restricts how you can edit or change certain key configuration files, we provide several extension points when you can make your changes to files while still receiving the benefits of Puppet for other files.

Puppet is not configured (or intended) to be run as a client daemon on your server, you should only run the update commands on an as needed basis.

Server Configurations

All servers are currently configured with the following basic elements:

- Ruby - compiled from source using ruby-build, installed system wide
- Puppet client 2.7.x
- Git

- Git
- Foreman - process management

Depending on the role assigned to a server it may get some or all of the following:

Application Servers:

- Unicorn - rack application server
- Nginx - web server / unicorn front end

Database Server:

- MySQL - database server

File Locations

Every server contains a single directory that houses the application and all serves as a location to configure several key aspects of the servers software stack:

- `/data/` - top level directory that contains all application specific data.
- `/data/spree/` - houses the application itself, serves as the deploy destination in Capistrano scripts.
- `/data/spree/current/` - the current live / production version of the application code (symlink to the relevant releases subdirectory).
- `/data/spree/releases/` - contains current and last 5 deployed releases of the application's code.
- `/data/spree/shared/` - directories and files that are common among all deployed versions of an application (like assets, configuration files, etc).
- `/data/spree/shared/config/` - contains several application specific configuration files including:
  - `database.yml` - Automatically generated configuration file for Rails / ActiveRecord, contains all the details required to connect the associated applications database. This file is sym-linked into place as part of the Capistrano deploy process.
  - `Procfile` - Automatically generated process configuration files, for more see the Application Processes article. This file is sym-linked into place as part of the Capistrano deploy process.
  - `unicorn.rb` - Automatically generated unicorn configuration file, reference by the Procfile above when starting the application server processes.
- `/etc/ssl` - You can upload your SSL certificate files into this directory and the will be automatically used by Puppet in your application Nginx configuration file. See the Requesting and Configuring SSL Guide for more details

configuration file. See the Requesting and Configuring SSL Guide for more details.

- **Tutorials**
  - Getting Started
  - Extensions
  - Deface Overrides
- **Source Code**
  - About
  - Navigating
  - Getting Help
  - Contributing
- **The Core**
  - Addresses
  - Adjustments
  - Calculators
  - Inventory
  - Orders
  - Payments
  - Preferences
  - Products
  - Promotions
  - Shipments
  - Taxation
- **Customization**
  - Authentication
  - Internationalization (i18n)
  - View
  - Asset
  - Logic
  - Use S3

- Use S3
- Checkout

- **Deployment**
  - Ninefold
  - Heroku
  - Shelly Cloud
  - Ansible Ubuntu Deployment
  - Manual Ubuntu Deployment
  - Deployment Options
  - Deployment Service
  - Deployment Tips
  - Requesting/Configuring SSL

- **Advanced Topics**
  - Search Engine Optimization
  - Developer Tips
  - Migrating to Spree
  - Security
  - Testing Spree Applications

- **Upgrade Guides**
  - 0.60.x to 0.70.x
  - 0.70.x to 1.0.x
  - 1.0.x to 1.1.x
  - 1.1.x to 1.2.x
  - 1.2.x to 1.3.x
  - 1.3.x to 2.0.x
  - 2.0.x to 2.1.x
  - 2.1.x to 2.2.x
  - 2.2.x to 2.3.x

Product

- Platform

- Platform
- Hub
- Support
- Privacy Policy
- Terms of Service

Developers

- Overview
- Documenation
- Community
- License

Partners

- Pro Services
- Training
- Partnership Program
- Payments

About Spree Commerce

Spree Commerce is an automated enterprise solution built specifically for ecommerce. We effectively manage your operations so you can focus on serving your customers and growing your business.

Take it for a test drive

You can even create your own personal store.

Try The Demo

-
-
-
-