Gunnery Documentation

Release 0.1

Paweł Olejniczak

August 18, 2014

Contents

Gunnery is a multipurpose task execution tool for distributed systems with web-based interface.

CHAPTER 1

Contents

1.1 Overview

Gunnery is a multipurpose task execution tool for distributed systems with web-based interface.

1.1.1 Features

Support for wide variety of tools Thanks to simple design it's possible to integrate with tools like capistrano, ant, phing, fabric, make, or puppet

Designed for distributed systems Handles multi-environment applications with many servers

Usable for deployment, service control, backups Almost any command executed in shell can be turned into Gunnery task

Secure remote execution Certificate based authentication provides secure access to your network

Web-based interface Clear, responsive interface pleases eye and enables usage on mobile devices

User notifications Team members will be notified when tasks are executed

1.1.2 Dependencies

Following list presents all packages required to run project:

```
Django==1.6.5
South==0.8.4
amqp==1.4.5
anyjson==0.3.3
argparse==1.2.1
billiard==3.3.0.17
celery==3.1.7
django-celery==3.1.10
django-crispy-forms==1.4.0
django-extensions==1.3.8
django-guardian==1.2.0
django-pgcrypto==1.1.1
ecdsa==0.11
factory-boy==2.3.1
kombu==3.0.19
paramiko==1.14.0
psycopg2==2.5.3
```

```
pycrypto==2.6.1
pytz==2014.4
six==1.7.2
sqlparse==0.1.11
uWSGI==2.0.5.1
wsgiref==0.1.2
git+https://github.com/Eyjafjallajokull/django-timezone-field.git@develop#egg=django-timezone-field
sphinx
```

1.2 Installation

Instructions on this page will guide you through installation process. You can choose to use puppet tool or setup everything manually.

1.2.1 Provisioning with Puppet

Gunnery repository contains puppet manifests, which can help setting up infrastructure required to run application. Puppet manifests are supported for systems:

- Ubuntu 13.10 (raring)

Below are listed commands which will setup full-stack Gunnery instance on a bare bones server:

```
su
git clone --recurse-submodules https://github.com/Eyjafjallajokull/gunnery.git /var/gunnery
cd /var/gunnery/puppet
cp manifests/hieradata/local.template.yaml manifests/hieradata/local.yaml
vim manifests/hieradata/local.yaml # set secrets
chown 700 manifests/hieradata/local.yaml
bash ./install.sh # ensure puppet 3 is installed
FACTER_environment=production puppet apply manifests/base.pp --hiera_config manifests/hiera.yaml --mo
cd /var/gunnery
export DJANGO_SETTINGS_MODULE="gunnery.settings.production"
source /var/gunnery/virtualenv/production/bin/activate
make build
python manage.py createsuperuser
```

The last step is to edit `/var/gunnery/gunnery/gunnery/settings/production.py` and set the `ALLOWED_HOSTS` directive to the domain (or IP address) that your instance will be running on. Boom, if everything went well you have working application.

1.2.2 Manual Installation

Gunnery may seem like a simple app, but it depends on a few components. This document will guide you through the process of installing all of them. For simplicity's sake, it's assumed that the host machine is Debian-based and that all services are running on a single machine.

```
Nginx
  |
  v
uWSGI
  |
  v
Gunnery <----> Database <----> Celery
```

```
|                                |
+----------> Queue <------------+
```

Setup Database

PostgreSQL is the recommended database for Django projects, although other types may be used as well.

- Postgres installation instructions http://www.postgresql.org/docs/8.0/static/installation.html
- Ubuntu guide https://help.ubuntu.com/community/PostgreSQL

A `gunnery` user without the `createdb` or `superuser` permissions must be created along with a database `gunnery`, which will be owned by the `gunnery` user.

In short:

```
sudo apt-get install postgresql postgresql-contrib
sudo -u postgres psql postgres
\password postgres
sudo -u postgres createuser -D -S -P gunnery
sudo -u postgres createdb -O gunnery gunnery
```

Setup Application

Download the gunnery application by cloning the repository. The recommended path is `/var/gunnery`:

```
sudo git clone --recurse-submodules git@github.com:Eyjafjallajokull/gunnery.git /var/gunnery
sudo cd /var/gunnery
```

Under the `requirements` folder you will find lists of packages required for different environments. To install production packages:

```
pip install -r requirements/production.txt
```

Next, adjust the settings inside `gunnery/settings/production.py` file. In particular, add the domain (or IP address) of your instance to the `ALLOWED_HOSTS` list.

Now we'll need to setup the database we created earlier for gunnery. To do so, we'll need to synchronize the database's schema with gunnery and run any necessary migrations. Then, we create the initial user and prepare the static files.

```
export DJANGO_SETTINGS_MODULE="gunnery.settings.production"
export SECRET_KEY="<insert random string here>"
python manage.py syncdb # synchronize gunnery schema to postgres
python manage.py migrate # run any necessary database schema migrations
python manage.py collectstatic # prepare static files to be served
python manage.py createsuperuser # create the initial user
```

To test that the application is working, you can use Django's built-in HTTP server:

```
python manage.py runserver
```

Optionally you can build html documentation with command:

```
cd /var/gunnery/docs
make htmlembedded
```

Install Messaging Queue

Celery requires a messaging queue for its operation, RabbitMQ being the recommended option. Refer to the Celery documentation for information about using alternatives.

```
sudo apt-get install rabbitmq
```

Configure Celery

Celery was installed in a previous step (`pip install`), it needs to be configured now.

```
# Copy provided init-script for Celery to /etc/init.d
sudo cp /var/gunnery/puppet/modules/component/files/celery.initd /etc/init.d/celeryd
# Copy provided Celery configuration defaults to /etc/default
sudo cp /var/gunnery/puppet/modules/component/templates/celery.default.erb /etc/default/celeryd
# Edit provided default to your satisfaction
sudo vim /etc/default/celeryd
sudo service celeryd start
```

Configure uWSGI

We're going to use uWSGI to manage our Python processes. Just like celery it was installed by pip as a dependency. We need to create init script for it. Copy the example file and adjust variables (search for `<% ... %>`)

```
# Copy example file to /etc/init.d
sudo cp /var/gunnery/puppet/modules/component/templates/uwsgi.erb /etc/init.d/uwsgi
sudo chmod u+x /etc/init.d/uwsgi # Make init script executable
sudo vim /etc/init.d/uwsgi
```

- replace `<%= @log_path %>` with `/var/gunnery/log`
- replace `<%= @run_path %>` with `/var/gunnery/run`
- replace `<%= @virtualenv_path %>` with `/var/gunnery/virtualenv`

Next, setup gunnery-specific configuration:

```
sudo mkdir -p /etc/uwsgi/apps-enabled # Create directory for gunnery uWSGI config
# Copy provided example config to newly created folder
sudo cp /var/gunnery/puppet/modules/component/templates/uwsgi.ini.erb /etc/uwsgi/apps-enabled/gunnery
sudo vim /etc/uwsgi/apps-enabled/gunnery.ini
```

- replace `<%= @app_name %>` with `gunnery`
- replace `<%= @app_path %>` with `/var/gunnery/gunnery`
- replace `<%= @log_path %>` with `/var/gunnery/log`
- replace `<%= @run_path %>` with `/var/gunnery/run`
- replace `<%= @virtualenv_path %>` with `/var/gunnery/virtualenv`
- replace `<%= @environment %>` with `production`

To make sure your config works, try starting the uWSGI service, check the logs for errors, and validate if the socket file exists.

```
sudo service uwsgi start
```

Install Nginx

No magic here. Simply install, copy the provided template, and customize to your needs.

```
sudo apt-get install nginx
sudo cp /var/gunnery/puppet/modules/component/templates/nginx.django.conf.erb /etc/nginx/sites-enable
sudo vim /etc/nginx/sites-enabled/gunnery
sudo service nginx reload
```

1.2.3 Support

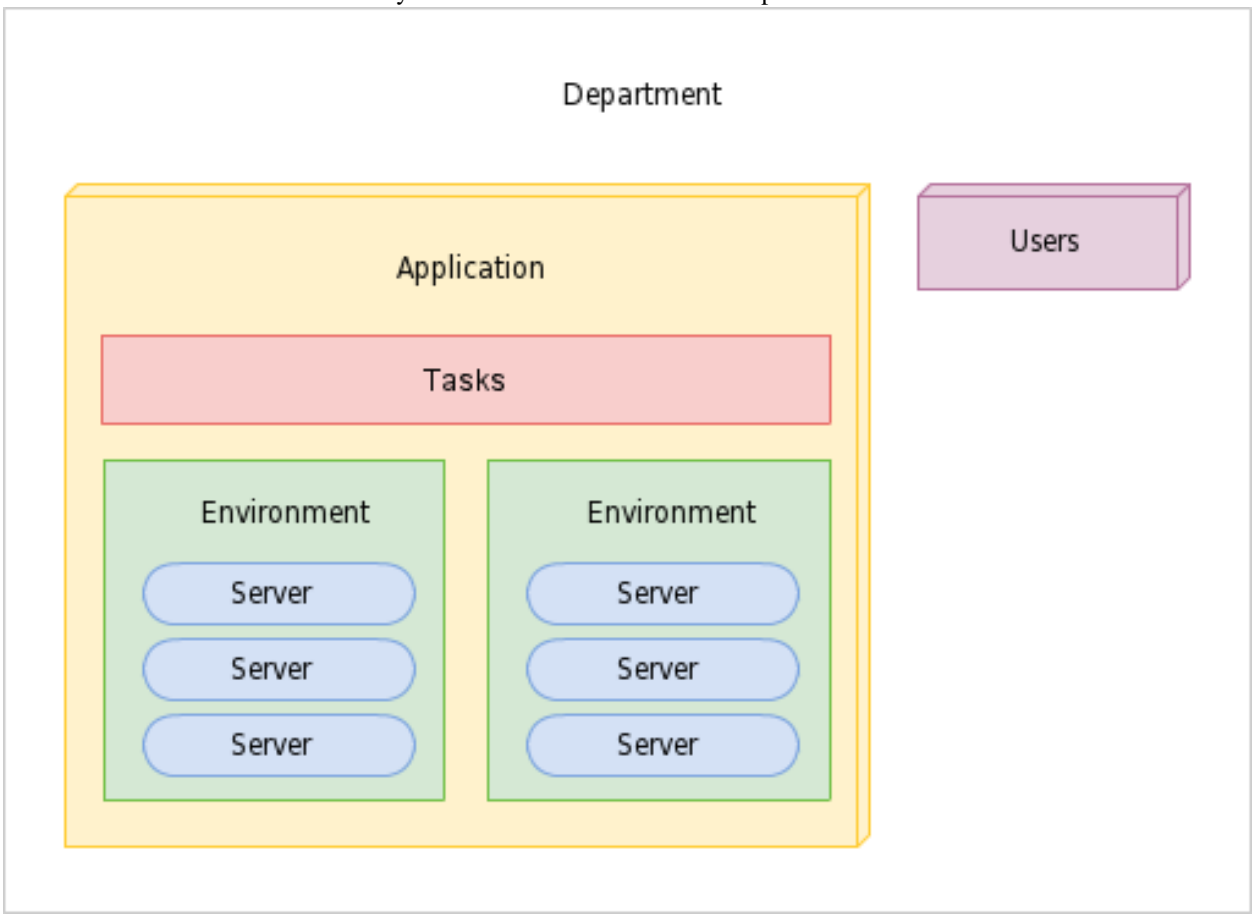
If you run into trouble and can't figure out how to solve it yourself, you can get help via Github issue tracker.

1.3 Usage

This article will guide you through basic usage of Gunnery.

1.3.1 Basic concepts

To better understands how to Gunnery works lets introduce basic concepts.



Department Represents a dev team or company department, can have multiple applications and users assigned.

Application Tasks and server environments.

Task One or more commands which can be executed on any environment within application. Commands can include dynamic or user defined parameters. Every command is only executed on servers with specific role.

Environment Set of related servers, every environment has a different login certificate.

Server Server tagged with specific role eg. db, app, backup.

User Users can only access departments which they are assigned to. Special manager role allows user to change department settings.

1.3.2 First steps

Follow below steps to execute first command.

1. Create application - Go to Settings > Applications > Create - After creating application you will be redirected to the new page, it will contain 2 empty lists of environments and tasks.
2. Create environment - Environment is used to group servers. Create first env by clicking + and entering name eg. test.
3. Create server - fill all required fields - select app role - copy provided SSH public key to target servers' ~/.ssh/authorized_keys file.
4. Create task - enter name of new command eg. process-list - first command eg. ps - assign role app to this command - save
5. Execute task - go to application page - click execute button located next to process-list task - click executed - you will see list of processes on target server

1.4 Cookbook

Following section contains common recipes for integrating Gunnery with external tools.

1.4.1 Capistrano

1. In department settings create *build* role
2. Create application and environment.
3. Create server pointing to your build server (where capistrano is installed).
4. Assign *build* role to this server
5. For every capistrano task - create gunnery task. For example if you have deploy task enter following command in gunnery task form `cd /path/to/capistrano/dir; cap deploy`.

If your capistrano task requires parameter, for example `cap deploy -S branch=master`, define required parameter `branch` in gunnery form and change command to `cap deploy -S branch=${branch}`

Example capistrano tasks:

- `With no parameters`
- `With env and branch parameters`

1.5 Contributing

Gunnery is open source project managed using Git and hosted on Github.

1. Check for open issues or open a fresh issue to start a discussion around a feature idea or a bug.
2. Fork the repository on Github to start making your changes.
3. Write a test which shows that the bug was fixed or that the feature works as expected.
4. Send a pull request and bug the maintainer until it gets merged and published. :)

1.5.1 Setup Vagrant environment

Project repository contains Vagrantfile configuration and Puppet provisioning manifests.

Puppet rules will install and configure, everything you need to start working on gunnery:

- nginx
- uwsgi
- postgresql
- celery
- rabbitmq
- virtualenv
- gunnery application

Before you get started be sure you have installed VirtualBox and Vagrant 1.1+. Start by cloning this repository.

```
git clone --recurse-submodules https://github.com/Eyjafjallajokull/gunnery.git
cd gunnery
vagrant up
```

By now you have working infrastructure for Gunnery application. In the next steps you will create database tables, prepare static files and create first user.

```
vagrant ssh
cd /vagrant/gunnery
python manage.py syncdb
python manage.py migrate
python manage.py collectstatic
python manage.py createsuperuser
```

Gunnery should be now accessible via address http://localhost:8080/.

Source code is mounted inside virtual machine under /vagrant/gunnery. With this setup you can easily edit code, test, commit and create pull requests to official repository.

1.5.2 Run tests

Gunnery uses nose test runner:

```
python manage.py test --settings=gunnery.settings.test
```

Running a specyfic test:

```
python manage.py test --settings=gunnery.settings.test task.tests.test_views:ApplicationTest.test_app
```

To print test coverage report:

```
coverage run --source='.' manage.py test --settings=gunnery.settings.test
coverage report
```

1.5.3 Create pull requests

Please note the following guidelines for contributing:

- Contributed code must be written in the existing style.
- Run the tests before committing your changes. If your changes cause the tests to break, they won't be accepted.
- If you are adding new functionality, you must include basic tests and documentation.

Pull request should be submitted to develop branch.

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*