Deployment tools

If you are using a version control system such as Git, for your project’s code and pushing releases into  
remote repository, you can use Git to deploy code to your production server via the git pull shell  
command instead of uploading files manually. Also, you can write your own shell script to pull new  
repository commits, update vendors, apply migrations, and do more things.

However, there are many tools available for automating the deployment process. In this recipe, we  
consider the tool named Deployer.

Getting ready

Create a new yii2 -app-basic application by using the Composer package manager, as described in the  
official guide at <http://www.yiiframework.com/doc-2.0/guidestart-installation.html>.

How to do it...

If you have a shared remote repository, you can use it for deployment source.

Step 1 - Preparing the remote host

1. Go to your remote host and install Composer and asset - plugin too:  
   global require 'fxp/composer-asset-plugin:~1.1.1'
2. Generate the SSH key via ssh - keygen.
3. Add the ~/. ssh/id\_rsa. pub file content into deployment the SSH keys page of your repository  
   settings on GitHub, Bitbucket, or other repositories storage.
4. Try to clone your repository manually:

git clone git@github.com:user/repo.git

1. Add the Github address and the list of known hosts if the system asks you to do it.

Step 2 - Preparing the localhost

1. Install deploy .phar globally on your local host:

sudo wget <http://deployer.org/deployer.phar>  
sudo mv deployer.phar /usr/local/bin/dep  
sudo chmod +x /usr/local/bin/dep

1. Add the deploy. php file with the deployment configuration:

<?php

require 'recipe/yii2-app-basic.php';

set('shared\_files', [

' config/db.php',

'config/params.php',

'web/index.php ',

' yii',

]);

server('prod', 'site.com', 22) // SSH access to remote server  
->user( ' user' )

// ->password(password) // uncomment for authentication by password  
// ->identityFile() // uncomment for authentication by SSH key

->stage('production')

->env('deploy\_path', '/var/www/project');

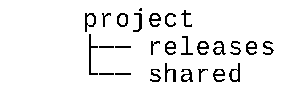
set('repository', 'git@github.com:user/repo.git');

3. Try to prepare remote project directories structure:

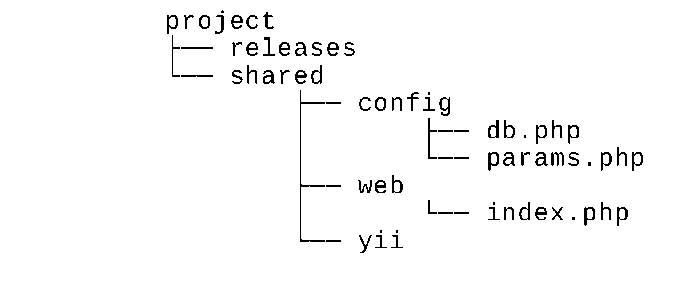
dep deploy:prepare prod

Step 3 - Adding remote configuration

1. Open the server’s /var/www/project directory. It has two subdirectories after the initialization:



2. Create original files with private configurations in a shared directory like this:



The Deployer tool will include these files in every release subdirectory via symbolic links.  
Specify your private configuration in share/config/db. php:

<?php  
return [

'class' => 'yii\db\Connection',

dsn' => 'mysql:host=localhost;dbname=catalog',

'username' => 'root',

'password' => 'root',

' charset' => 'utf8',

];

Also, specify it in share/config/params . php:

<?php  
return [

'adminEmail' => 'admin@example.com',

];

Set the content of share/web/index . php:

<?php

defined('YII\_DEBUG') or define('YII\_DEBUG', false);

defined('YII\_ENV') or define('YII\_ENV', 'prod');

$dir = dirname($\_SERVER['SCRIPT\_FILENAME']);

require($dir . '/../vendor/autoload.php');  
require($dir . '/../vendor/yiisoft/yii2/Yii.php');

$config = require($dir . '/../config/web.php');

(new yii\web\Application($config))->run();

Also, set the content of the share/yii file:

#!/usr/bin/env php  
<?php

defined('YII\_DEBUG') or define('YII\_DEBUG', false);  
defined('YII\_ENV') or define('YII\_ENV', 'prod');

$dir = dirname($\_SERVER['SCRIPT\_FILENAME']);

require($dir . '/vendor/autoload.php');  
require($dir . '/vendor/yiisoft/yii2/Yii.php');

$config = require($dir. '/config/console.php');

$application = new yii\console\Application($config);

$exitCode = $application->run();  
exit($exitCode);

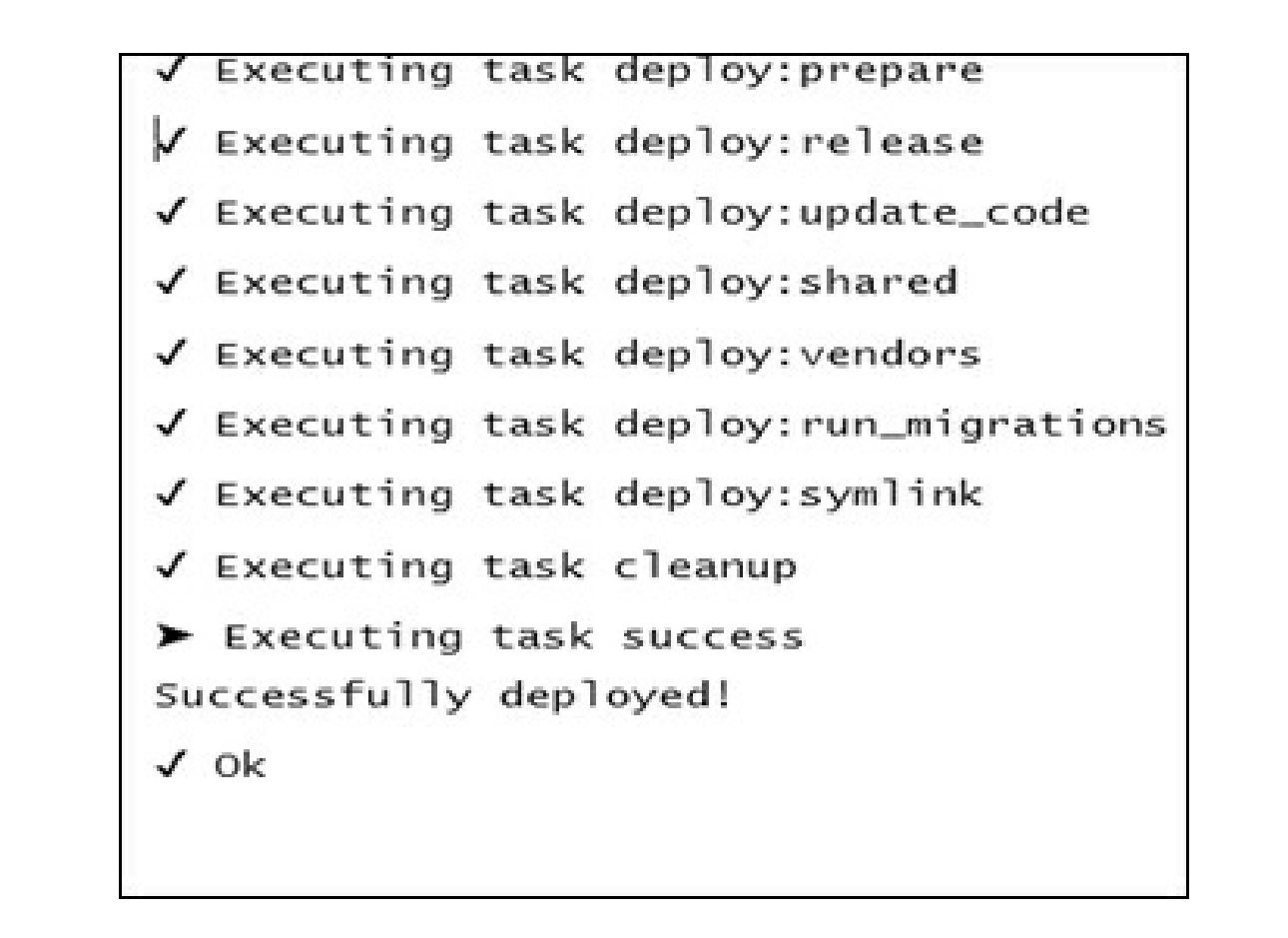
Note

Note: We deliberately use the dirname($\_server [' script\_filename ' ]) code instead of the original  
 dir constant because dir will return incorrect value when the file is included via symbolic link.

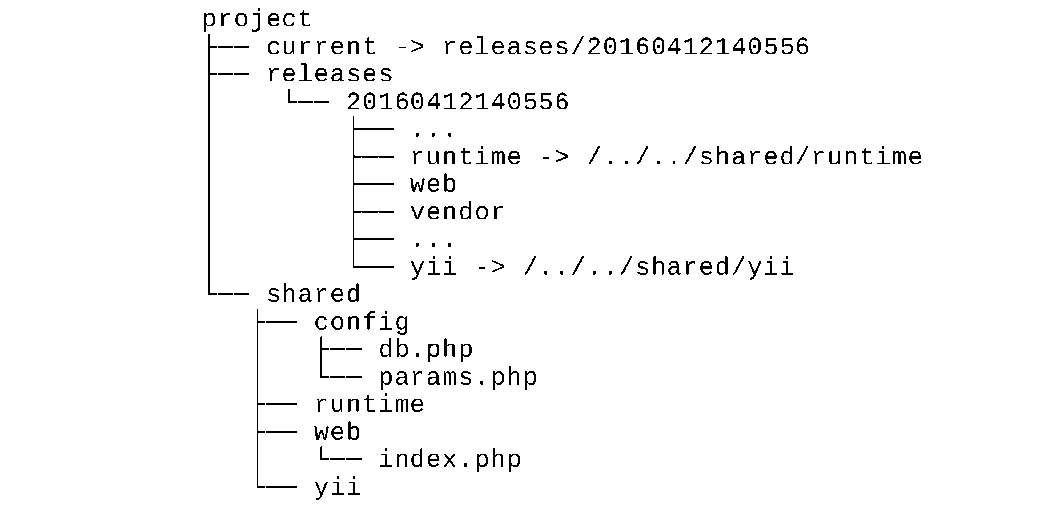
Note: If you use the yii2-app-advanced template you can redeclare only the config/main-local.php  
and config/params-local.php files of every (backend, frontend, console, and common) because  
web/index. php and yii files will be created automatically by the init command.

Step 4 - Trying to deploy

1. Come back to the localhost with the deploy .php file and run the deploy command:  
   dep deploy prod
2. If successful, you will see the deployment report:



3. Deployer created a new release subdirectory on your remote server and added symlinks from your  
project to the shared items and from the current directory to the current release:



1. After all is done, you must set up the DocumentRoot of your server in project/current/web  
   directory.
2. If something goes wrong during the deployment process you can roll back to the previous working  
   release:

dep rollback prod

The current directory will lead to your previous release files.

How it works...

Most of the deployment tools do the same tasks:

* Create a new release subdirectory
* Clone repository files
* Make symlinks from the project to shared directories and to local configuration files
* Install Composer packages
* Apply project migrations
* Switch the symlink from the server’s DocumentRoot path to the current release directory

The Deployer tool has predefined recipes for popular frameworks. You can extend any existing recipe or  
write a new one for your specific case.

See also

* For more information about Deployer, refer to <http://deployer.org/docs>
* [And about creating SSH keys refer to https://git-scm.com/book/en/v2/Git-on-the-Server-](https://git-scm.com/book/en/v2/Git-on-the-Server-Generating-Your-SSH-Public-Key)  
  Generating-Your-SSH-Public-Key