Bitnami Documentation For PHP Frameworks

Some Bitnami stacks include PHP frameworks to simply development of PHP Web applications. Frameworks include CakePHP, CodeIgniter, Symfony, Laravel, Smarty and Zend Framework.

CakePHP

Overview

The CakePHP framework is installed in the *frameworks/cakephp* directory of the installation directory. This folder includes an example application. Application files are in the *app/* directory and public files, such as HTML pages, CSS and JavaScript files, images and other media assets are stored in the *app/webroot* directory.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

```
Include "/opt/bitnami/frameworks/cakephp/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

You can now verify that the example application is working by visiting its URL using your browser at *http://SERVER-IP/cakephp*.

Here is an example of what you might see:

Configuration

Before using the example application, here are a few important points to consider:

1. To secure your application, modify the encryption keys in the *app/Config/core.php* file. Ideally, use a key that's 32 characters or longer in length.

   ```
   Configure::write('Security.salt', '');
   Configure::write('Security.cipherSeed', '');
   ```

   On Linux, you can use a command like *pwgen 32* to generate a 32-character random key. On Windows, you can use a tool like PWGen.

2. If your application will use a database, edit the database settings at *app/Config/database.php*.

   ```
   public $default = array(
           'datasource' => 'Database/Mysql',
           'persistent' => false,
           'host' => 'localhost',
           'port' => '3306',
           'login' => 'user',
           'password' => 'password',
           'database' => 'database_name',
           'prefix' => '',
           //'unix_socket' => '/opt/bitnami/mysql/tmp/mysql.sock',
   ```

```
        //'encoding' => 'utf8',
);
```

> NOTE: If you are using an operating system that supports sockets, such as Linux or Mac OS X, you can optionally specify the *unix_socket* parameter in the above configuration array instead of the *host* and *port* parameters.

MySQL support is already available by default. If you plan to use PostgreSQL, enable the *php_pdo_pgsql* extension in the */opt/bitnami/php/etc/php.ini* file.

```
extension=php_pdo_pgsql
```

3. To move the CakePHP example application such that it is available at the root URL of the server (without the */cakephp* URL suffix), follow these steps:
    - Edit the */opt/bitnami/frameworks/cakephp/conf/httpd-prefix.conf* file so that it looks like this:

      ```
      DocumentRoot "/opt/bitnami/frameworks/cakephp/app/webroot/"
      #Alias /cakephp/ "/opt/bitnami/frameworks/cakephp/app/webroot/"
      #Alias /cakephp "/opt/bitnami/frameworks/cakephp/app/webroot"
      Include "/opt/bitnami/frameworks/cakephp/conf/httpd-app.conf"
      ```

    - Edit the */opt/bitnami/frameworks/cakephp/app/webroot/.htaccess* file so that the *RewriteBase* directive is set to the root URL:

      ```
      RewriteBase /
      ```

    - Restart the Apache server:

      ```
      $ sudo /opt/bitnami/ctlscript.sh restart apache
      ```

    You should now be able to access the example application at the root URL of your server.

Upgrading To CakePHP 3.X

If your Bitnami stack currently uses CakePHP 2.x and you would like to upgrade to CakePHP 3.x, follow these steps:

> NOTE: The steps below assume that you have already activated the CakePHP framework and example application.

- Log in to your server console.
- Back up the current version of CakePHP:

  ```
  $ cd /opt/bitnami/frameworks
  $ sudo mv cakephp cakephp.old
  ```

- Download and install the latest version of CakePHP (3.0.10 at the time of writing):

  ```
  $ cd /opt/bitnami/frameworks
  $ sudo wget https://github.com/cakephp/cakephp/archive/3.0.10.tar.gz
  $ sudo tar -xzvf 3.0.10.tar.gz
  $ sudo mv cakephp-3.0.10 cakephp
  ```

- Create a demo application:

  ```
  $ cd /opt/bitnami/frameworks/cakephp
  $ sudo composer self-update
  $ sudo composer create-project --prefer-dist cakephp/app app
  ```

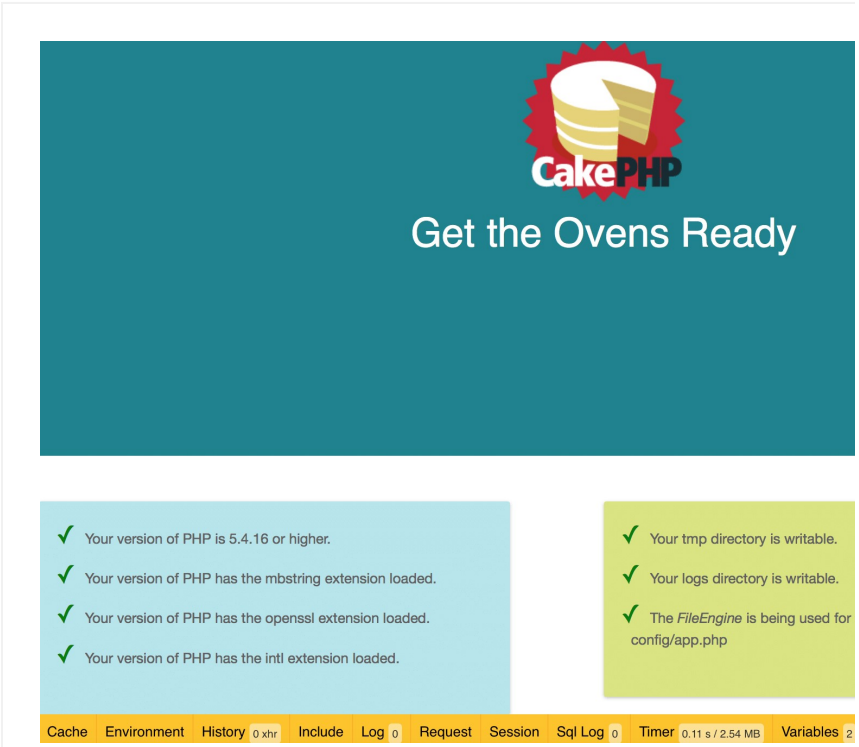
- Copy the necessary configuration files from the previous CakePHP installation:

```
$ sudo cp -R /opt/bitnami/frameworks/cakephp.old/conf /opt/bitnami/framework
$ sudo cp /opt/bitnami/frameworks/cakephp.old/app/webroot/.htaccess /opt/bit
```

- Change file ownerships of the CakePHP directory:

```
$ cd /opt/bitnami/frameworks/
$ sudo chown -R bitnami:root cakephp
```

You should now be able to access the example application and verify that it is using CakePHP 3.x, as shown below:



More Information

Learn more about developing applications with CakePHP at http://book.cakephp.org/.

CodeIgniter

Overview

The CodeIgniter framework is installed in the *frameworks/codeigniter* directory in the installation directory. This directory includes an example application. Application configuration files are in the *conf/* directory and public files, such as HTML pages, CSS and JavaScript files, images and other media assets are stored in the *htdocs/* directory.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

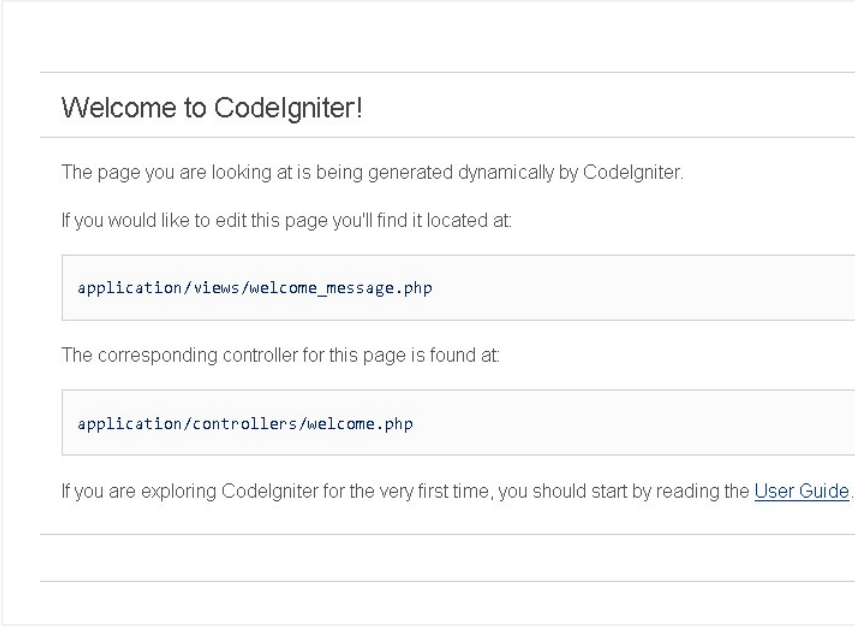
```
Include "/opt/bitnami/frameworks/codeigniter/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

You can now verify that the example application is working by visiting its URL using your browser at *http://SERVER-IP/codeigniter*.

Here is an example of what you might see:



Configuration

Before using the example application, here are a few important points to consider:

1. To secure your application session, modify the encryption keys in the *application/config/config.php* file. Ideally, use a key that's 32 characters or longer in length.

   ```
   $config['encryption_key'] = '';
   ```

   On Linux, you can use a command like *pwgen 32* to generate a 32-character random key. On Windows, you can use a tool like PWGen.

2. If your application will use a database, edit the database settings at *app/Config/database.php*.

   ```
   $db['default']['hostname'] = 'localhost';
   $db['default']['username'] = 'user';
   $db['default']['password'] = 'pass';
   $db['default']['database'] = 'database_name';
   $db['default']['dbdriver'] = 'mysql';
   $db['default']['port'] = 3306;
   $db['default']['dbprefix'] = '';
   ```

   MySQL support is already available by default. If you plan to use PostgreSQL, enable the *php_pdo_pgsql* extension in the */opt/bitnami/php/etc/php.ini* file.

   ```
   extension=php_pdo_pgsql
   ```

3. To move the CodeIgniter example application such that it is available at the root URL of the server (without the */codeigniter* URL suffix), follow these steps:

    - Edit the */opt/bitnami/frameworks/codeigniter/conf/httpd-prefix.conf* file so that it looks like this:

      ```
      DocumentRoot "/opt/bitnami/frameworks/codeigniter/htdocs"
      #Alias /codeigniter/ "/opt/bitnami/frameworks/codeigniter/htdocs/"
      #Alias /codeigniter "/opt/bitnami/frameworks/codeigniter/htdocs"
      Include "/opt/bitnami/frameworks/codeigniter/conf/httpd-app.conf"
      ```

    - Edit the */opt/bitnami/frameworks/codeigniter/conf/httpd-app.conf * file replace the *AllowOverride None* directive with the *AllowOverride All* directive:

      ```
      AllowOverride All
      ```

    - Restart the Apache server:

      ```
      $ sudo /opt/bitnami/ctlscript.sh restart apache
      ```

    You should now be able to access the example application at the root URL of your server.

Upgrading To CodeIgniter 3.X

If your Bitnami stack currently uses CodeIgniter 2.x and you would like to upgrade to CodeIgniter 3.x, follow these steps:

> NOTE: The steps below assume that you have already activated the CodeIgniter framework and example application.

- Log in to your server console.
- Back up the current version of CodeIgniter:

  ```
  $ cd /opt/bitnami/frameworks
  $ sudo mv codeigniter codeigniter.old
  ```

- Download and install the latest version of CodeIgniter (3.0.0 at the time of writing):

  ```
  $ cd /opt/bitnami/frameworks
  $ sudo wget https://github.com/bcit-ci/CodeIgniter/archive/3.0.0.zip
  $ sudo unzip 3.0.0.zip
  $ sudo mv CodeIgniter-3.0.0 codeigniter
  ```

- Create a demo application:

  ```
  $ cd /opt/bitnami/frameworks/codeigniter
  $ sudo composer self-update
  $ sudo composer create-project --prefer-dist codeigniter/app app
  ```

- Copy the necessary configuration files from the previous CodeIgniter installation:

  ```
  $ sudo cp -R /opt/bitnami/frameworks/codeigniter.old/conf /opt/bitnami/frame
  $ cd /opt/bitnami/frameworks/codeigniter
  $ sudo mkdir htdocs
  $ sudo mv index.php user_guide htdocs
  ```

- Modify the *index.php* file:

  ```
  $ cd /opt/bitnami/frameworks/codeigniter/htdocs
  $ sudo vi index.php
  ```

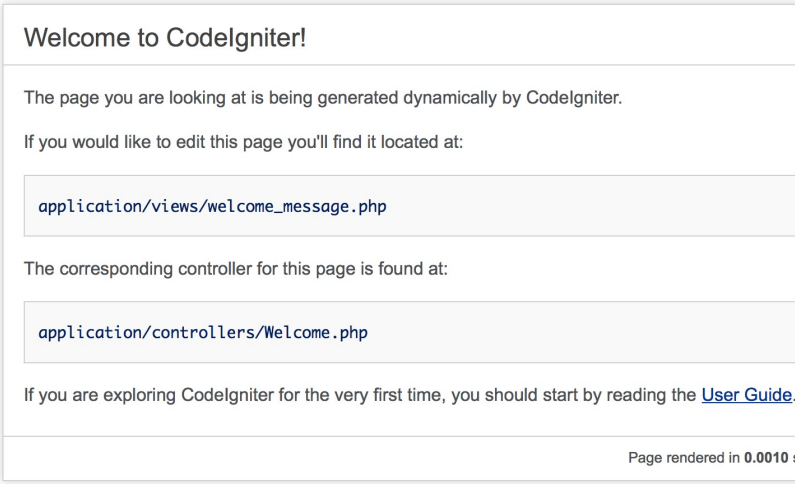
- Within the file, update the values of the *$system_path* and *$application_folder* variables so they look like this.

```
<?php
$system_path = '/opt/bitnami/frameworks/codeigniter/system';
$application_folder = '/opt/bitnami/frameworks/codeigniter/application';
```

- Change file ownerships of the CodeIgniter directory:

```
$ cd /opt/bitnami/frameworks
$ sudo chown -R bitnami:root codeigniter
```

You should now be able to access the example application and verify that it is using CodeIgniter 3.x, as shown below:



More Information

Learn more about developing applications with CodeIgniter at http://codeigniter.com/user_guide.

Laravel

Overview

The Laravel framework is installed in the *frameworks/laravel* directory in the installation directory. This directory includes an example application. Application configuration files are in the *conf/* directory and public files, such as HTML pages, CSS and JavaScript files, images and other media assets are stored in the *public/* directory.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

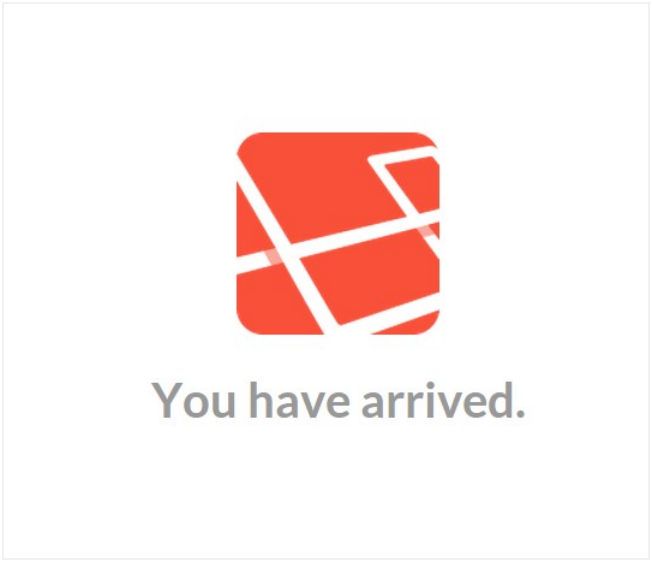
```
Include "/opt/bitnami/frameworks/laravel/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

You can now verify that the example application is working by visiting its URL using your browser at *http://SERVER-IP/laravel*.

Here is an example of what you might see (Laravel 4.x):



Here is an example of what you might see (Laravel 5.x):

Configuration

Before using the example application, here are a few important points to consider:

1. To start a new project, edit the file at */opt/bitnami/frameworks/laravel/app/routes.php* (Laravel 4.x) or */opt/bitnami/frameworks/laravel/app/Http/routes.php* (Laravel 5.x) and add a new route:

   ```
   Route::get('my-first-route', function()
   {
       return 'Hello World!';
   });
   ```

   This will create the application route */my-first-route*. To see this route in action, append this route to the application URL and visit it in your browser, such as *http://SERVER-IP/laravel/index.php/my-first-route*. If all is working correctly, you will see the output "Hello World!".

2. If your application will use a database, edit the database settings in the *app/config/database.php* (Laravel 4.x) or *config/database.php* (Laravel 5.x) file.

```
return array(
        'connections' => array(
                'mysql' => array(
                        'driver'    => 'mysql',
                        'host'      => 'localhost',
                        'database'  => 'database_name',
                        'username'  => 'user',
                        'password'  => 'pass',
                        'charset'   => 'utf8',
                        'collation' => 'utf8_unicode_ci',
                        'prefix'    => '',
                ),
    )
);
```

   MySQL support is already available by default. If you plan to use PostgreSQL, enable the *php_pdo_pgsql* extension in the */opt/bitnami/php/etc/php.ini* file.

```
extension=php_pdo_pgsql
```

3. To move the Laravel application such that it is available at the root URL of the server (without the */laravel* URL suffix), follow these steps:
   - Edit the */opt/bitnami/frameworks/laravel/conf/httpd-prefix.conf* file so that it looks like this:

     ```
     DocumentRoot "/opt/bitnami/frameworks/laravel/public"
     #Alias /laravel/ "/opt/bitnami/frameworks/laravel/public/"
     #Alias /laravel "/opt/bitnami/frameworks/laravel/public"
     Include "/opt/bitnami/frameworks/laravel/conf/httpd-app.conf"
     ```

   - Edit the */opt/bitnami/frameworks/laravel/conf/httpd-app.conf* file and replace the *AllowOverride None* directive with the *AllowOverride All* directive:

     ```
     AllowOverride All
     ```

   - Restart the Apache server:

     ```
     $ sudo /opt/bitnami/ctlscript.sh restart apache
     ```

   You should now be able to access the example application at the root URL of your server.

Key Regeneration

A random key of 32 characters is generated during the installation. You can change it later to another random key.

```
$ cd /opt/bitnami/frameworks/laravel
$ sudo php artisan key:generate
```

These commands will generate a new key and print it in the terminal. Copy the 32-character key and then edit the */opt/bitnami/frameworks/laravel/config/app.php* file. You should see a line like this:

```
'key' => env('APP_KEY', 'PasteYourKeyHere')
```

Just paste your generated key and save the file.

Upgrading To Laravel 5.1

Follow the steps below to upgrade to Laravel 5.1:

1. Remove the *composer.lock* file in the */opt/bitnami/frameworks/laravel* directory:

   ```
   $ rm /opt/bitnami/frameworks/laravel/composer.lock
   ```

2. Update *composer.json* with the latest version:

   ```
   ...
   "laravel/framework": "5.1.*"
   ```

3. Create a *cache/* directory under */opt/bitnami/frameworks/laravel/bootstrap/*:

   ```
   $ mkdir -p /opt/bitnami/frameworks/laravel/bootstrap/cache/
   ```

4. Edit the *$compiledPath* variable in the */opt/bitnami/frameworks/laravel/bootstrap/autoload.php* file as below:

   ```
   $compiledPath = __DIR__.'/cache/compiled.php';
   ```

5. Create the file */opt/bitnami/frameworks/laravel/bootstrap/cache/.gitignore* with this content:

   ```
   !.gitignore
   ```

6. Set *bitnami* as the owner and give writable permission to */opt/bitnami/frameworks/laravel/bootstrap/cache*:

   ```
   $ sudo chmod 775 /opt/bitnami/frameworks/laravel/bootstrap/cache
   $ sudo chown bitnami:root /opt/bitnami/frameworks/laravel/bootstrap/cache
   ```

7. Add *Illuminate\Broadcasting\BroadcastServiceProvider* to the *providers* array in the */opt/bitnami/frameworks/laravel/config/app.php* file:

   ```
   ...
   'Illuminate\Broadcasting\BroadcastServiceProvider',
   ...
   ```

   Note the comma (,) at the end of the line.

8. Modify the */opt/bitnami/frameworks/laravel/app/Http/Controllers/Auth/AuthController.php* header from

   ```
   use App\Http\Controllers\Controller;
   use Illuminate\Contracts\Auth\Guard;
   use Illuminate\Contracts\Auth\Registrar;
   use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;
   ```

   to

   ```
   use App\User;
   use Validator;
   use App\Http\Controllers\Controller;
   use Illuminate\Foundation\Auth\AuthenticatesAndRegistersUsers;
   ```

9. Modify */opt/bitnami/frameworks/laravel/app/Http/Controllers/Auth/AuthController.php* constructor from

```
/**
 * Create a new authentication controller instance. *
 * @param \Illuminate\Contracts\Auth\Guard $auth
 * @param \Illuminate\Contracts\Auth\Registrar $registrar
 * @return void */

public function __construct(Guard $auth, Registrar $registrar) { $this->aut
        $this->middleware('guest', ['except' => 'getLogout']);
}
```

to

```
/**
 * Create a new authentication controller instance. *
 * @return void */

public function __construct() {
  $this->middleware('guest', ['except' => 'getLogout']);
}
```

10. Add in the same file just after the method above:

```
/**
 * Get a validator for an incoming registration request. *
 * @param array $data
 * @return \Illuminate\Contracts\Validation\Validator
*/

protected function validator(array $data) {
  return Validator::make($data, [ 'name' => 'required|max:255', 'email' => '
}

/**
 * Create a new user instance after a valid registration. *
 * @param array $data
 * @return User
*/
protected function create(array $data) {
  return User::create([ 'name' => $data['name'], 'email' => $data['email'],
}
```

11. Remove the */opt/bitnami/frameworks/laravel/app/Services* directory.

```
$ rm -rf /opt/bitnami/frameworks/laravel/app/Services
```

12. Modify the */opt/bitnami/frameworks/laravel/app/Http/Controllers/Auth/PasswordController.php* file from

```
/**
 * Create a new password controller instance.
public function __construct(Guard $auth, PasswordBroker $passwords) { $this-
        $this->middleware('guest');
}
```

to

```
/**
* Create a new password controller instance.
*
* @return void
*/
public function __construct()
{
  $this->middleware('guest');
}
```

13. Modify the */opt/bitnami/frameworks/laravel/app/Http/routes.php* file from

```
Route::get('/', 'WelcomeController@index');

Route::get('home', 'HomeController@index');

Route::controllers([
    'auth' => 'Auth\AuthController',
    'password' => 'Auth\PasswordController',
]);
```

to

```
Route::get('/', function () {
  return view('welcome');
});
```

14. Remove the */opt/bitnami/frameworks/laravel/vendor* directory:

```
$ rm -rf /opt/bitnami/frameworks/laravel/vendor
```

15. Include the *baseUrl* class variable in the */opt/bitnami/frameworks/laravel/tests/TestCase.php* file:

```
class TestCase extends Illuminate\Foundation\Testing\TestCase {

/**
* The base URL to use while testing the application.
*
* @var string
*/
protected $baseUrl = 'http://localhost';
...
```

16. Delete the */opt/bitnami/frameworks/laravel/storage/framework/compiled.php* file:

```
$ sudo rm /opt/bitnami/frameworks/laravel/storage/framework/compiled.php
```

17. Run *composer update* in the */opt/bitnami/framework/laravel* directory:

```
$ composer update
```

18. Check the version of Laravel:

```
$ php artisan --version
```

You can also make these optional changes if you wish.

More Information

Learn more about developing applications with Laravel at http://laravel.com/docs/.

Smarty

Overview

The Smarty framework is installed in the *frameworks/smarty* directory of the installation directory. This directory includes two example applications, located in the *sample/* and *demo/* directories respectively.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

```
Include "/opt/bitnami/frameworks/smarty/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

You can now verify that the example application is working by visiting its URL using your browser at *http://SERVER-IP/smarty*.

Here is an example of what you might see:

To enable the second example application, follow these steps:

- Edit the Apache configuration file at */opt/bitnami/frameworks/smarty/conf/httpd-prefix.conf* and modify it to point to the *demo/* directory, as shown below:

  ```
  Alias /smarty/ "/opt/bitnami/frameworks/smarty/demo/"
  Alias /smarty "/opt/bitnami/frameworks/smarty/demo"
  Include "/opt/bitnami/frameworks/smarty/conf/httpd-app.conf"
  ```

- Edit the */opt/bitnami/frameworks/smarty/conf/httpd-app.conf* file and modify the ** directive to reflect the new path:

  ```
  <Directory "/opt/bitnami/frameworks/smarty/demo">
  ...
  </Directory>
  ```

- You should also modify the permissions and ownership of the *demo/* directory so that it is writable by the Web server user:

```
$ cd /opt/bitnami/frameworks/smarty
$ sudo chown -R bitnami:daemon demo
$ sudo chmod -R 775 demo
```

- Restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

You can now verify that the second example application is working by visiting the same URL as before. Here is an example of what you should see.

```
Title: Welcome To Smarty!

The current date and time is 2015-01-05 02:12:41

The value of global assigned variable $SCRIPT_NAME is /smarty/index.php

Example of accessing server environment variable SERVER_NAME: new-server-0a99cb.bitns

The value of {$Name} is Fred Irving Johnathan Bradley Peppergill

variable modifier example of {$Name|upper}

FRED IRVING JOHNATHAN BRADLEY PEPPERGILL


An example of a section loop:

        1 * John Doe
        2 * Mary Smith
        3 . James Johnson
        4 . Henry Case

An example of section looped key values:

        phone: 1

        fax: 2

        cell: 3

        phone: 555-4444
```

Upgrading Smarty 3.X

NOTE: The steps below assume that you have already activated the Smarty example application.

If you would like to upgrade to the latest Smarty version, follow these steps:

- Log in to your server console.
- Back up the current version of Smarty:

```
$ cd /opt/bitnami/frameworks
$ sudo mv smarty smarty.old
```

- Download and install the latest version of Smarty (3.1.27 at the time of writing):

```
$ cd /opt/bitnami/frameworks
$ sudo wget https://github.com/smarty-php/smarty/archive/v3.1.27.zip
$ sudo unzip v3.1.27.zip
$ sudo mv smarty-3.1.27 smarty
```

- Copy the necessary configuration files from the previous Smarty installation:

```
$ sudo cp -R /opt/bitnami/frameworks/smarty.old/conf /opt/bitnami/frameworks
$ sudo cp -R /opt/bitnami/frameworks/smarty.old/sample /opt/bitnami/framewor
```

- Change file ownerships of the Smarty directory:

```
$ cd /opt/bitnami/frameworks/
$ sudo chown -R bitnami:root smarty
$ cd /opt/bitnami/frameworks/smarty
$ sudo chown -R bitnami:daemon sample
$ sudo chmod -R 775 sample
```

You should now be able to access the sample application, as shown below:

More Information

Learn more about developing applications with Smarty at http://www.smarty.net/quick_install.

Symfony

Overview

The Symfony framework is installed in the *frameworks/symfony* directory in the installation directory. This directory includes an example application. Application code is in the *app/* directory, application configuration files are in the *conf/* directory and public files, such as HTML pages, CSS and JavaScript files, images and other media assets are stored in the *web/* directory.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

```
Include "/opt/bitnami/frameworks/symfony/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

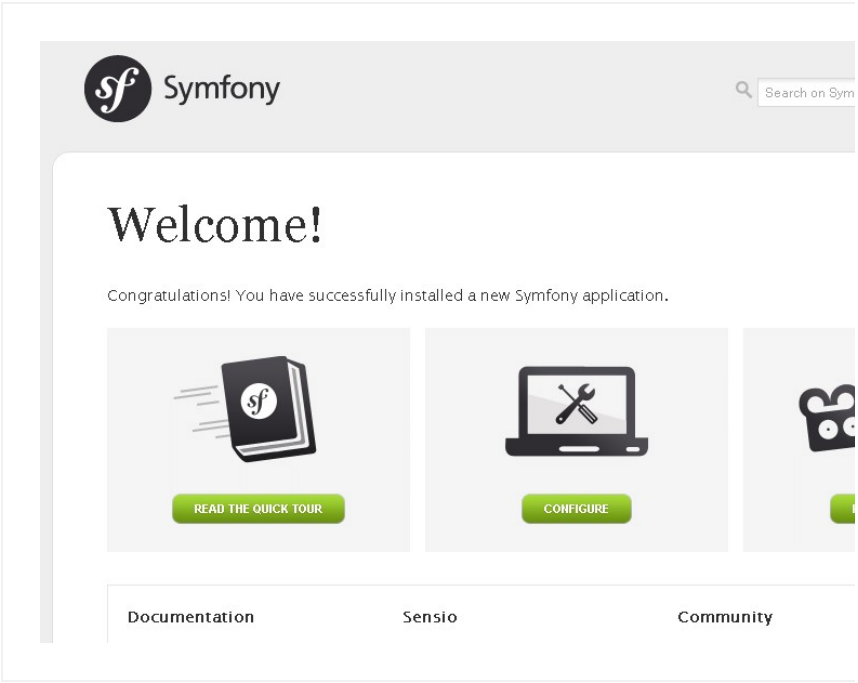
The Symfony application is configured by default to only allow requests originating from *localhost* for security reasons. As a result, you will see a message like *You are not allowed to access this file. Check app_dev.php for more information* when you try to access the example application on these platform.

To allow access to the example application from hosts other than *localhost*, edit the file */opt/bitnami/frameworks/symfony/web/app_dev.php* and within it, disable the IP address check at the top of the file by commenting out the relevant lines. After modification, the relevant section of the file should look like this:

```
// This check prevents access to debug front controllers that are deployed by ac
// Feel free to remove this, extend it, or make something more sophisticated.
/*
if (isset($_SERVER['HTTP_CLIENT_IP'])
    || isset($_SERVER['HTTP_X_FORWARDED_FOR'])
    || !(in_array(@$_SERVER['REMOTE_ADDR'], array('127.0.0.1', 'fe80::1', '::1')
) {
    header('HTTP/1.0 403 Forbidden');
    exit('You are not allowed to access this file. Check '.basename(__FILE__).'
}
*/
```

You can now verify that the example application is working by visiting its URL using your browser at *http://SERVER-IP/symfony/app_dev.php*.

Here is an example of what you might see:



Configuration

Before using the example application, here are a few important points to consider:

1. If your application will use a database, edit the database settings in the *app/config/parameters.yml* file.

   ```
   parameters:
     database_driver: pdo_mysql
     database_host: 127.0.0.1
     database_port: 3306
     database_name: database_name
     database_user: user
     database_password: pass
   ```

   MySQL support is already available by default. If you plan to use PostgreSQL, enable the *php_pdo_pgsql* extension in the */opt/bitnami/php/etc/php.ini* file.

```
extension=php_pdo_pgsql
```

2. To move the Symfony application such that it is available at the root URL of the server (without the */symfony* URL suffix), follow these steps:
    - Edit the */opt/bitnami/frameworks/symfony/conf/httpd-prefix.conf* file so that it looks like this:

      ```
      DocumentRoot "/opt/bitnami/frameworks/symfony/web/"
      #Alias /symfony/ "/opt/bitnami/frameworks/symfony/web/"
      #Alias /symfony "/opt/bitnami/frameworks/symfony/web"
      Include "/opt/bitnami/frameworks/symfony/conf/httpd-app.conf"
      ```

    - Edit the */opt/bitnami/frameworks/symfony/conf/httpd-app.conf* file and replace the *AllowOverride None* directive with the *AllowOverride All* directive:

      ```
      AllowOverride All
      ```

    - Restart the Apache server:

      ```
      $ sudo /opt/bitnami/ctlscript.sh restart apache
      ```

    You should now be able to access the example application at the root URL of your server.

Upgrading Symfony 2.X

> NOTE: The steps below assume that you have already activated the Symfony example application.

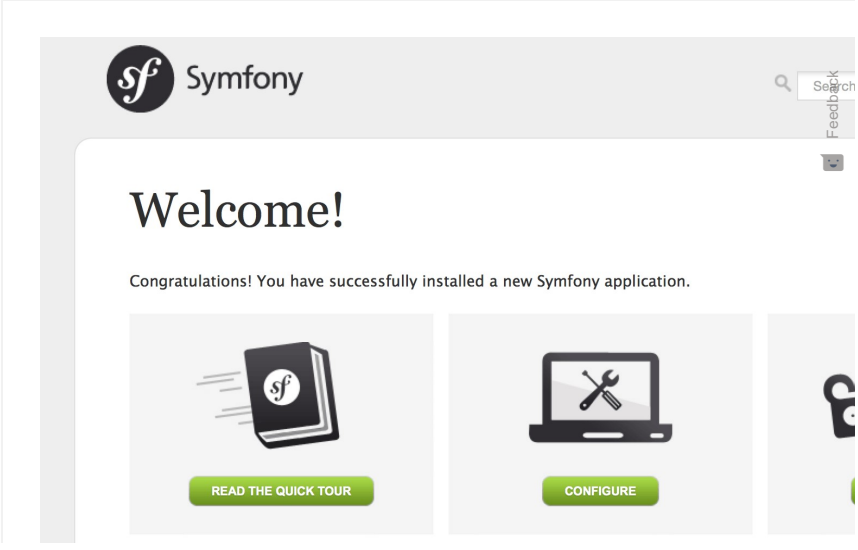
- Log in to your server console.
- Update the framework by executing the following commands:

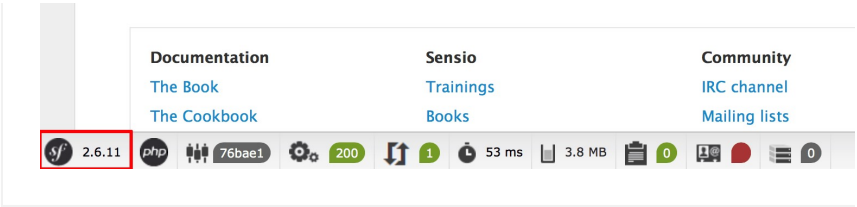
  ```
  $ cd /opt/bitnami/frameworks/symfony
  $ sudo composer self-update
  $ sudo composer update
  ```

- Change the permissions of the *app/cache* directory so that it is writable by the Web server:

  ```
  $ cd /opt/bitnami/frameworks/symfony/app
  $ sudo chown -R bitnami:daemon cache
  $ sudo chmod -R 775 cache
  ```

You should now be able to access the example application and verify that it is using the latest version of Symfony, as shown below:





More Information

Learn more about developing applications with Symfony at http://symfony.com/doc/current/.

Zend Framework 2

Overview

The Zend Framework is installed in the *frameworks/zendframework* directory in the installation directory. This directory includes an example application. Application configuration files are in the *config/* directory, application modules are in the *module/* directory and public files, such as HTML pages, CSS and JavaScript files, images and other media assets are stored in the *public/* directory.

Activation And Testing

To enable the example application, edit the Apache configuration file at */opt/bitnami/apache2/conf/bitnami/bitnami-apps-prefix.conf* and uncomment the following line

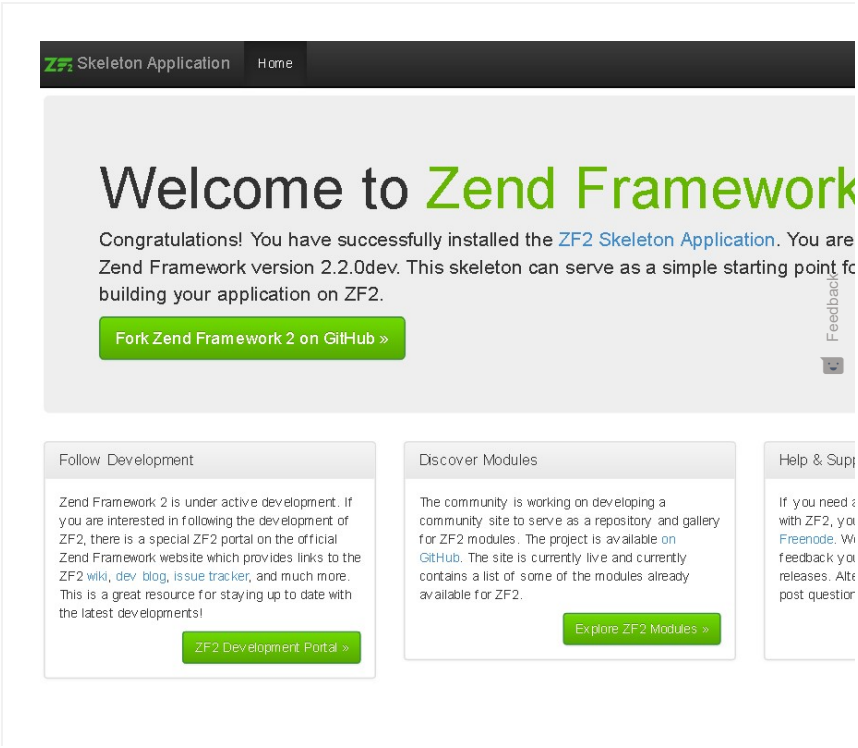
```
Include "/opt/bitnami/frameworks/zendframework/conf/httpd-prefix.conf"
```

Then, restart the Apache server.

```
$ sudo /opt/bitnami/ctlscript.sh restart apache
```

Access the example application via your browser at *http://SERVER-IP/zendframework*.

Here is an example of what you might see:



Configuration

Before using the example application, here are a few important points to consider:

1. If your application will use a database, MySQL support is already available by default. If you plan to use PostgreSQL, enable the *php_pdo_pgsql* extension in the */opt/bitnami/php/etc/php.ini* file.

   ```
   extension=php_pdo_pgsql
   ```

2. To move the Zend Framework application such that it is available at the root URL of the server (without the */zendframework* URL suffix), follow these steps:

   - Edit the */opt/bitnami/frameworks/zendframework/conf/httpd-prefix.conf* file so that it looks like this:

     ```
     DocumentRoot "/opt/bitnami/frameworks/zendframework/public"
     #Alias /zendframework/ "/opt/bitnami/frameworks/zendframework/public/
     #Alias /zendframework "/opt/bitnami/frameworks/zendframework/public"
     Include "/opt/bitnami/frameworks/zendframework/conf/httpd-app.conf"
     ```

   - Edit */opt/bitnami/frameworks/zendframework/public/.htaccess* and comment the *RewriteBase* line:

     ```
     # RewriteBase /zendframework
     ```

   - Edit the */opt/bitnami/frameworks/zendframework/conf/httpd-app.conf* file and replace the *AllowOverride None* directive with the *AllowOverride All* directive:

     ```
     AllowOverride All
     ```

   - Restart the Apache server:

     ```
     $ sudo /opt/bitnami/ctlscript.sh restart apache
     ```

   You should now be able to access the example application at the root URL of your server.

Upgrading Zend Framework 2.X

> NOTE: The steps below assume that you have already activated the Zend Framework example application.

- Log in to your server console.
- Change to the Zend Framework directory and edit the *composer.json* file:

  ```
  $ cd /opt/bitnami/frameworks/zendframework
  $ sudo vi composer.json
  ```

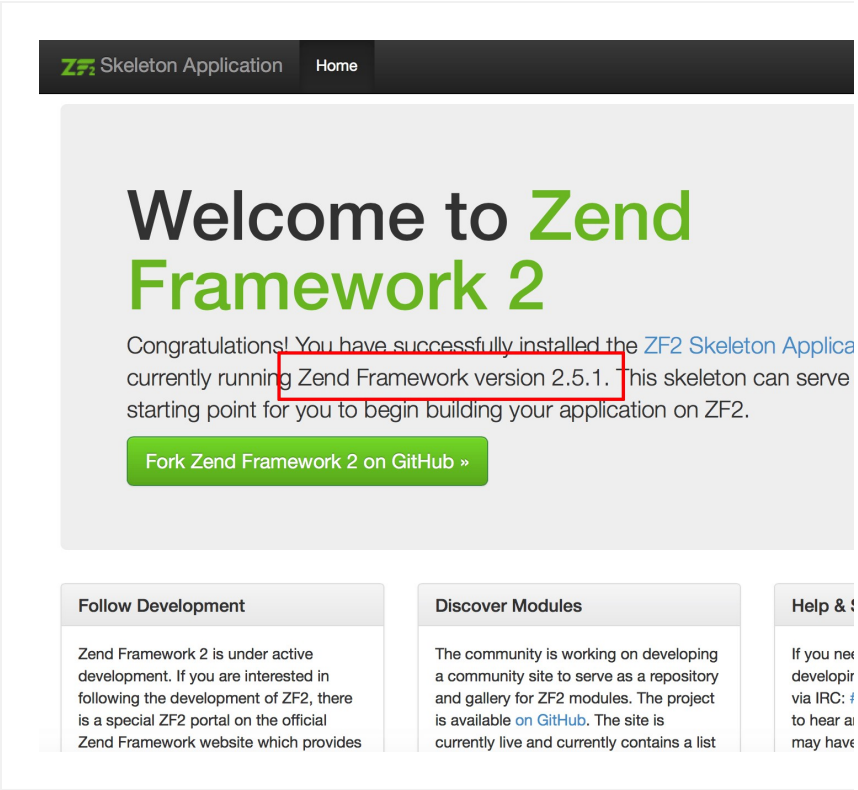
- Within the *composer.json* file, update the version of the *zendframework/zendframework* package to the latest available version. Save your changes. For example:

  ```
  {
    ...
    "require": {
      "php": ">=5.3.3",
      "zendframework/zendframework": "2.5.*"
    }
  }
  ```

- Update to the specified version of the framework by executing the following commands:

```
$ cd /opt/bitnami/frameworks/zendframework
$ sudo composer self-update
$ sudo composer update
```

You should now be able to access the example application and verify that it is using the specified version of Zend Framework 2.x, as shown below:



More Information

Learn more about developing applications with the Zend Framework at http://framework.zend.com/learn/.

Bitnami Documentation

FAQs

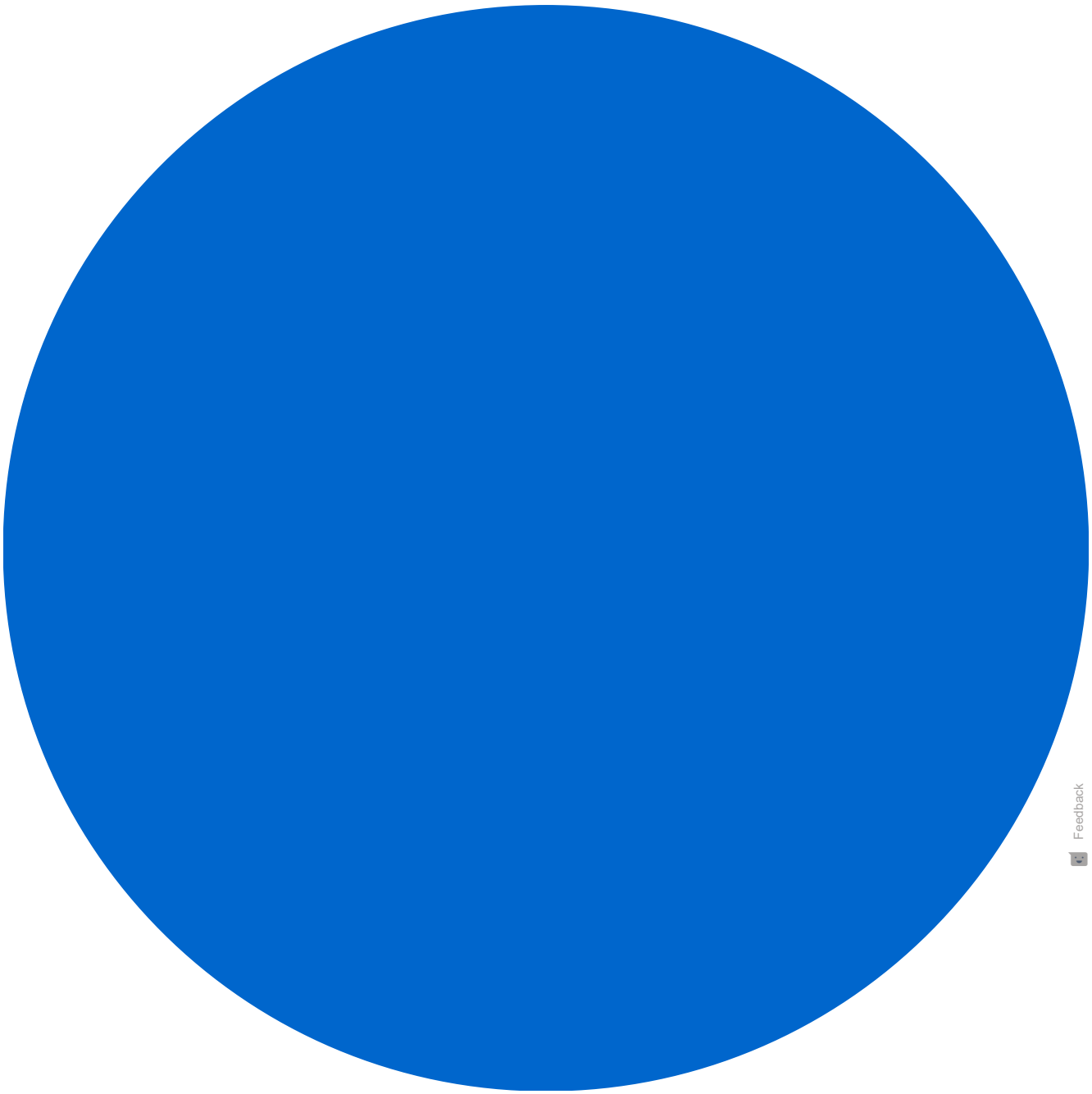
- How to find application credentials?
- How to connect to the server through SSH?
- How to upload files to the server with SFTP?
- How to open the server ports for remote access?
- How to configure your application to use a third-party SMTP service for outgoing email?
- How to block a suspicious IP address?

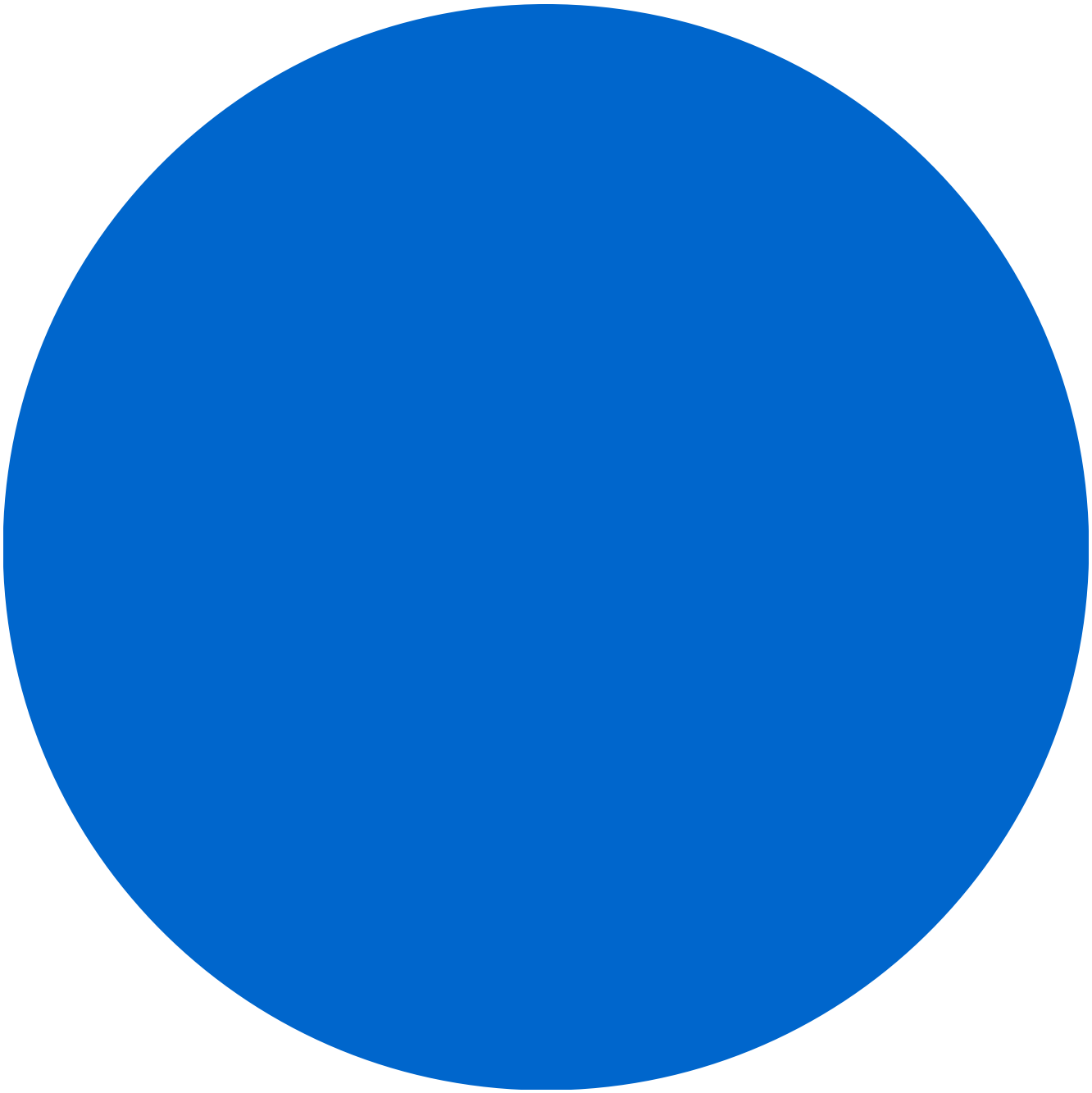
Platform Documentation

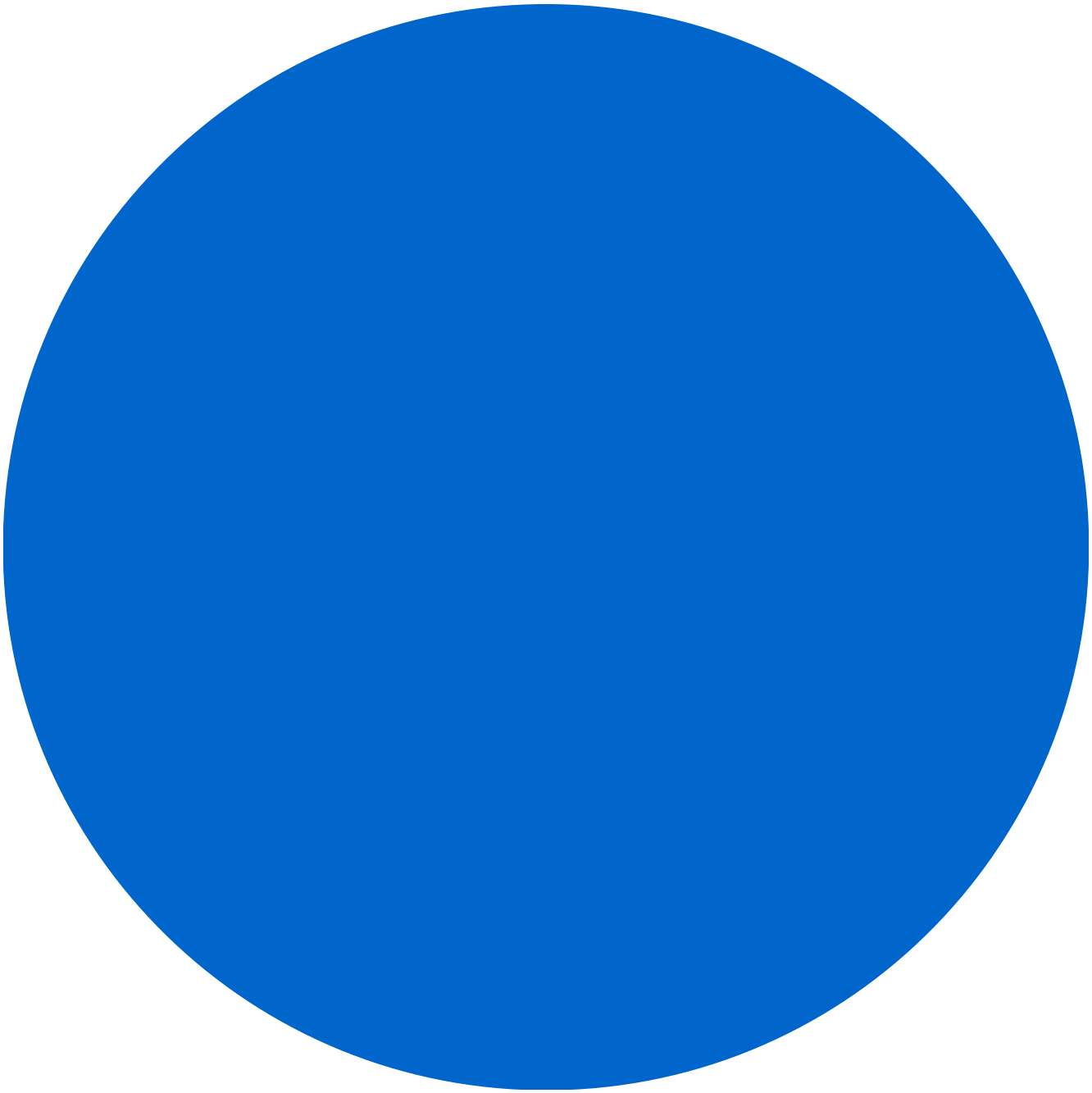
- Google Cloud Platform
- AWS Cloud
- Oracle Cloud Infrastructure Classic
- Microsoft Azure
- Bitnami Cloud Hosting
- CenturyLink Cloud
- 1&1 Cloud Platform
- Huawei Cloud
- Windows / Linux / MacOS
- Virtual Machines
- Containers
- Kubernetes

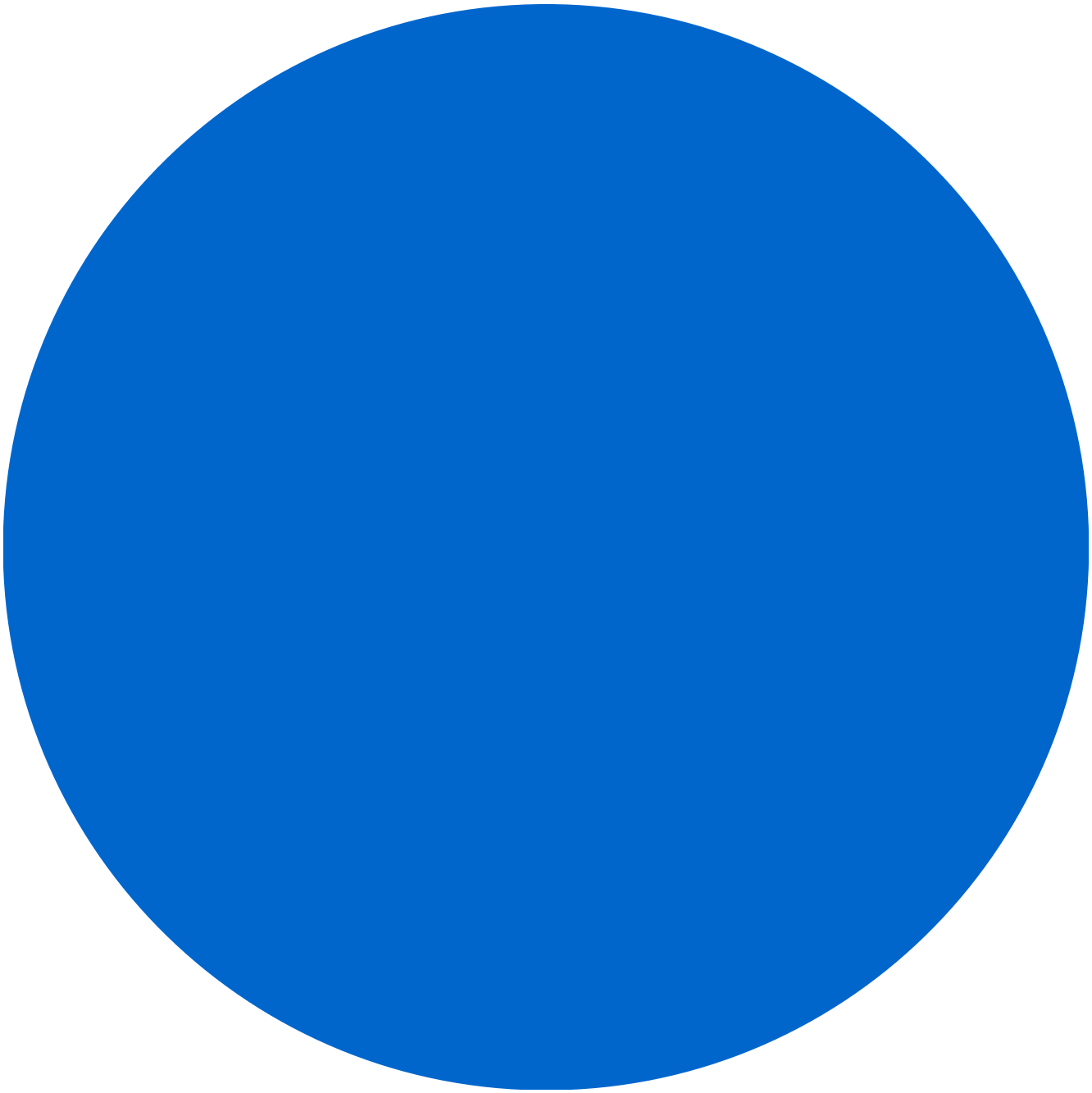
General Documentation

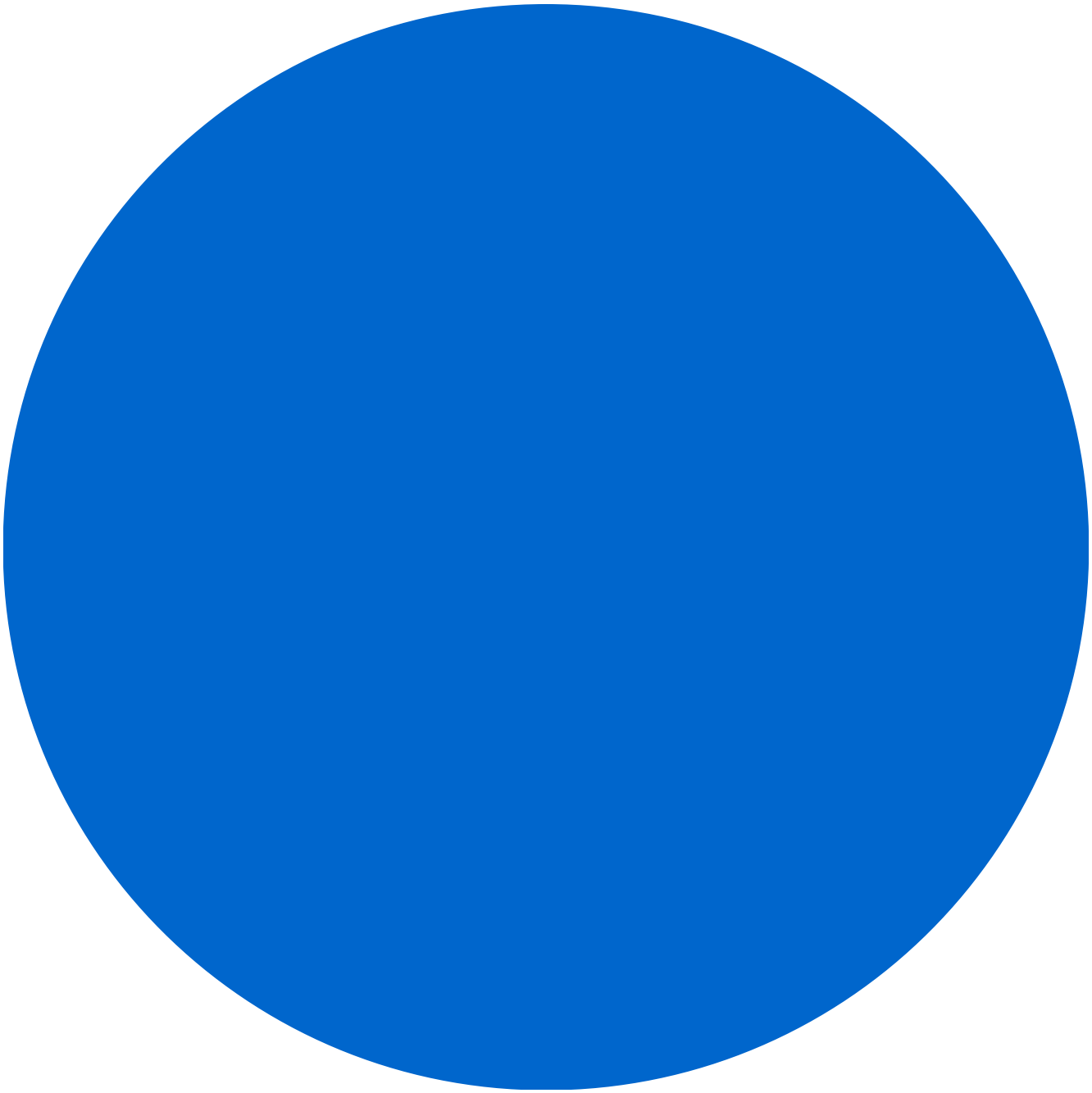
- Bitnami Application Stacks
- Bitnami Infrastructure Stacks
- How-To Guides
- Bitnami Components
- Security Notices











Products

- Application Catalog
- Stacksmith

Solutions

- Application Packaging
- Cloud Migration
- Kubernetes

Partners

- Cloud
- Software
- Technology
- SI / Resellers

Company

- About Us
- Team
- Careers
- Customers
- Contact
- Blogs
- Press
- Events
- Press Releases
- Logos

Legal

- Customer Agreement
- Terms of Use
- Privacy
- Trademark

Support

- Community
- Helpdesk
- Webinars
- Training