



Deploy Laravel su Netsons — Guida Completa



1. Pulizia del Progetto

Prepara l'ambiente di produzione eseguendo i seguenti comandi:

```
php artisan cache:clear
php artisan config:clear
php artisan route:clear
php artisan view:clear
npm run build
```



Il progetto è ora pronto per la distribuzione.



2. Compressione e Caricamento

1. Comprimi l'intero progetto in un file **.zip**
2. Carica il file nella directory:

```
home/utente/nome_progetto
```

3. Estrai il contenuto nella stessa directory.
-



3. Organizzazione delle Cartelle

Crea la directory:

```
home/utente/public_html/nome_progetto
```

Sposta i file da:

```
home/utente/nome_progetto/public
```

a:

```
home/utente/public_html/nome_progetto
```

Nella cartella **public** originale lascia solo:

```
build/
└─ manifest.json
```



4. Modifica del file [index.php](#)



Percorso:

```
home/utente/public_html/nome_progetto/index.php
```

Sostituisci il contenuto con:

```
<?php

use Illuminate\Foundation\Application;
use Illuminate\Http\Request;


define('LARAVEL_START', microtime(true));

// Determine if the application is in maintenance mode...
if (file_exists($maintenance =
'/home/apghciha/nome_progetto/storage/framework/maintenance.php')) {
    require $maintenance;
}

// Register the Composer autoloader...
require '/home/apghciha/nome_progetto/vendor/autoload.php';

// Bootstrap Laravel and handle the request...
/** @var Application $app */
$app = require_once '/home/apghciha/nome_progetto/bootstrap/app.php';

$app->handleRequest(Request::capture());
```

 Ricorda di sostituire **apghciha** e **nome_progetto** con i tuoi dati reali.

Aggiorna anche il file **.htaccess** in **public_html**:

```
RewriteCond %{REQUEST_URI} !^/nome_progetto(/|$)
```

5. Configurazione del Database

Modifica il file **.env** con i tuoi dati:

```
DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=nome_database
DB_USERNAME=nome_utente
DB_PASSWORD=la_tua_password
```

Caricare e visualizzare immagini usando Cloudflare R2

In questo modo si evita l'uso di `storage:link`, che su Netsons non funziona.

Una volta registrati su Cloudflare (gratis fino a 10 GB/mese) e creato il bucket, recupera i token API e inseriscili nel file **.env**:

```
AWS_ACCESS_KEY_ID=72f2e5e58***2fb539b271b6203824e
AWS_SECRET_ACCESS_KEY=2205***ecb6b33071e90112b3babe68b1a4e57434f1eb8210dbda
71859a7fed
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=travel
AWS_ENDPOINT=https://de2ec9bf7***9af3ad6efa6701f4014.r2.cloudflarestorage.c
om
AWS_USE_PATH_STYLE_ENDPOINT=false
AWS_URL=https://pub-8c639c***6d4562ae8cae88c755b89c.r2.dev

VITE_APP_NAME="${APP_NAME}"
```



Modifica del Controller per salvare le immagini

```
public function store(Request $request)
{
    $validated = $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'required|string',
        'price' => 'required|numeric|min:0',
        'duration_days' => 'required|integer|min:1',
        'images' => 'required|array|min:1',
        'images.*' => 'image|mimes:jpeg,png,jpg,gif,webp|max:5000',
    ]);

    // Crea lo slug per la cartella
    $slug = \Illuminate\Support\Str::slug($request->title);
    $folder = "journeys/{$slug}";

    $uploadedPaths = [];

    // Caricamento immagini su S3
    if ($request->hasFile('images')) {
        foreach ($request->file('images') as $file) {
            $path = $file->store($folder, 's3');
            $uploadedPaths[] = $path;
        }
    }

    // Usa la prima immagine come copertina
    $coverUrl = \Illuminate\Support\Facades\Storage::disk('s3')->url($uploadedPaths[0]);

    $journey = Journey::create([
        'title' => $request->title,
        'description' => $request->description,
        'price' => $request->price,
        'duration_days' => $request->duration_days,
        'image' => $coverUrl,
    ]);

    // Salva tutte le immagini nella tabella correlata
    foreach ($uploadedPaths as $path) {
        \App\Models\JourneyImage::create([
            'journey_id' => $journey->id,
            'path' => $path,
        ]);
    }

    return redirect()->back()->with('success', 'Viaggio creato con successo! Ora puoi inserirne un altro.');
```



Installazione librerie

```
composer require league/flysystem-async-aws-s3 async-aws/s3
```



Creazione Adapter personalizzato

Crea il file:

```
app/FileSystem/AsyncAwsFilesystemAdapter.php
```

Contenuto:

```
<?php

namespace App\FileSystem;

use Illuminate\Filesystem\FilesystemAdapter;

class AsyncAwsFilesystemAdapter extends FilesystemAdapter
{
    /**
     * Get the URL for the file at the given path.
     *
     * @param string $path
     * @return string
     *
     * @throws \RuntimeException
     */
    public function url($path)
    {
        if (isset($this->config['url'])) {
            return $this->concatPathToUrl($this->config['url'], $path);
        }

        return parent::url($path);
    }
}
```



Registrazione del driver personalizzato

Modifica **App/Providers/**[AppServiceProvider.php](#):

```
use App\FileSystem\AsyncAwsFilesystemAdapter;
use AsyncAws\S3\S3Client;
use League\Flysystem\Filesystem;
use AsyncAws\S3\Flysystem\AsyncAwsS3Adapter;

public function boot(): void
{
    Storage::extend('s3', function ($app, $config) {
        $client = new S3Client([
            'region' => $config['region'],
            'accessKeyId' => $config['key'],
            'accessKeySecret' => $config['secret'],
            'endpoint' => $config['endpoint'] ?? null,
            'pathStyleEndpoint' => $config['use_path_style_endpoint'] ??
false,
        ]);

        $adapter = new AsyncAwsS3Adapter($client, $config['bucket']);

        return new \App\FileSystem\AsyncAwsFilesystemAdapter(
            new Filesystem($adapter, ['url' => $config['url']]),
            $adapter,
            $config
        );
    });
}
```