First, using the fantastic work of Jenny Louthan of Uncorked Studios, implement this directive:

```
Directives.directive('fileModel', ['$parse', function ($parse) {
return {
  restrict: 'A',
  link: function(scope, element, attrs) {
    var model = $parse(attrs.fileModel);
    var modelSetter = model.assign;

    element.bind('change', function(){
      scope.$apply(function(){
        modelSetter(scope, element[0].files[0]);
      });
    });
  }
};
}]);
```

Secondly, write the Angular function to handle the upload:

```
$scope.uploadImage = function (image, event) {

  var url = '/path/to/upload';
  var formData = new FormData();
  formData.append('image', image);

  $http.post(url, formData, {
    transformRequest: angular.identity,
    headers: { 'Content-Type': undefined }
  })
  .success(function(data) {
    //@TODO: something unique here to properly display a success
```

```
message.
  })
  .error(function(data) {
    //@TODO: something unique here for an error message
    alert(data.message);
  });

};
```

Third, write the **action** in Symfony to process the file upload, with the proper route:

path/to/routes.yml

```
image_upload_route:
    path: /path/to/upload
    defaults: { _controller: PathToController:imageUpload }
```

path/to/MyController.php

```
public function imageUploadAction(Request $request)
{
    try {

        $image = $request->files->get('image');

        if(!$image instanceof UploadedFile) {
            throw new InvalidFormException(sprintf('Image must be of
type UploadedFile, %s given.', get_class($image)));
        }

        //$image is now an instance of
SymfonyComponentHttpFoundationFileUploadedFile
```

```
        //@TODO: process your image here.

        $responseData = array(
          'message' => '',
        );

        $response = new Response($responseData);
        $response->headers->set('Content-Type', 'application/json');
        return $response;

    }
    catch(Exception $e) {
        throw $e;
    }
}
```

Now, referencing my previous post on setting up Liip + Gaufrette + AwsS3 I'll assume you've configured and set that up properly. Given that, you'd adjust your `imageUploadAction` as follows.

However, to start, you'll need a FileManager service to properly manage the files to Aws S3 as configured.

app/config/parameters.yml

```
parameters:
    aws_sdk_version: latest
    aws_access_key_id: ***************
    aws_secret_access_key: **************
    aws_s3_region: us-east-1
```

app/config/services.yml

```
myapp.file_manager:
        class: PathToServicesFileManager
        arguments: ["%aws_access_key_id%", "%aws_secret_access_key%",
%aws_s3_region% ]
        calls:
            - [ setContainer, [ @service_container ] ]
```

Services/FileManager.php

```
<?php

namespace MyAppServices;

use SymfonyComponentHttpFoundationFileUploadedFile;
use SymfonyComponentDependencyInjectionContainerAwareInterface;
use SymfonyComponentDependencyInjectionContainerInterface;

use DoctrineORMEntityManager;

use GaufretteFilesystem;
use GaufretteAdapterAwsS3;

use AwsS3S3Client;
use AwsS3ExceptionS3Exception;
use AwsCredentialsCredentials;

/**
 * Class FileManager
 *
 * @package MyAppServices
 */
```

```
class FileManager implements ContainerAwareInterface
{
    private $container;
    private $s3;

    protected $aws_access_key_id;
    protected $aws_secret_access_key;
    protected $aws_s3_region;

    /**
     * @param ContainerInterface $container
     */
    public function setContainer(ContainerInterface $container = null)
    {
        $this->container = $container;
    }

    /**
     * @param $aws_access_key_id
     * @param $aws_secret_access_key
     */
    public function __construct($aws_access_key_id,
$aws_secret_access_key, $aws_s3_region)
    {
        $credentials = new Credentials(
            $aws_access_key_id,
            $aws_secret_access_key
        );

        // Instantiate the S3 client with your AWS credentials
        $s3 = S3Client::factory(array(
```

```
            'credentials' => $credentials,
            'version' => 'latest', //@TODO: not this in production
            'region' => $aws_s3_region
        ));

        $this->s3 = $s3;
    }

    /**
     * @param SymfonyComponentHttpFoundationFileUploadedFile $file
     * @param GaufretteAdapterAwsS3 $adapter
     *
     * @return mixed
     * @throws Exception
     */
    public function upload(UploadedFile $file, AwsS3 $adapter)
    {
        try {

            $filename = sprintf('%s.%s', uniqid(),
$file->getClientOriginalExtension());
            $adapter->setMetadata('Content-Type',
$file->getMimeType());
            $response = $adapter->write($filename,
file_get_contents($file->getPathname()));
            return $filename;

        }
        catch(S3Exception $e) {
            throw $e;
        }
```

```
        catch(Exception $e) {
            throw $e;
        }
    }

}
```

Finally, the whole Action:

```
public function imageUploadAction(Request $request)
{
    try {

        $image = $request->files->get('image');

        if(!$image instanceof UploadedFile) {
            throw new InvalidFormException(sprintf('Image must be of
type UploadedFile, %s given.', get_class($image)));
        }

        $adapter =
$this->container->get('images_filesystem')->getAdapter('images');

        $uploader = $this->container->get('myapp.file_manager');

        $uploader->upload($file, $adapter);

        if(empty($errors)) {
            $data = $image;
            //@TODO: serialize your $data here, this example assumes
you have a serializer service enabled with Symfony
```

```
            $responseData =
$this->container->get('serializer')->serialize($data, 'json');
        }
        else {
            $data = array(
                'message' => implode(',', $errors)
            );
            $responseData = json_encode($data);
        }

        $response = new Response($responseData);
        $response->headers->set('Content-Type', 'application/json');
        return $response;

    }
    catch(Exception $e) {
        throw $e;
    }
}
```