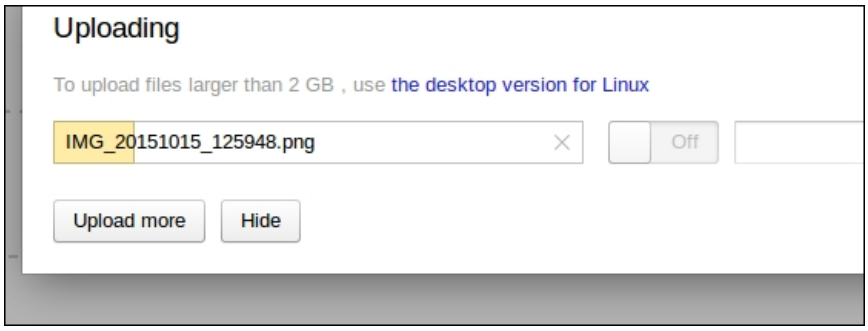
Showing a file upload progress bar

Uploading files from the web browser to the server is a rather common feature of modern web applications. Any number of CMS or publishing systems allows users to upload images to include these with their textual content, as shown in the following image:



Here is an example of a web upload in progress. The basic idea behind one of the algorithms to implement the progress bar is to initiate a POST request in an IFrame and then poll some well-known URL for the progress counter. Modern browsers allow us to get the progress information right on the client's side; this is a part of XMLHttpRequest Level 2 and was standardized about 3 years ago. There are a lot of older web applications that still rely on the older methods.

The described method only works if your client-side posts to your server-side with the same speed that the user actually sees in their interface. The problem is Nginx that buffers the long POST and then quickly and efficiently pushes it to the server-side code. The progress-poller process will not be able to get any progress until the very last moment when suddenly the entirety of the upload process happens in an instant.

There are several solutions to it. A dirty workaround is to process the uploads that you want to show progress for outside of Nginx. That is, have a backend server that is directly connected to the Internet, POST all your files to it, and get your progress from it.

A much better solution is to spend some resources and reimplement the progress bar part of the interface to use progress events available in modern browsers. The JavaScript (with jQuery + jQuery Form plugin) code will look as simple as this:

Copy

```
    var status = $('#status');

    $('form').ajaxForm({
        beforeSend: function() {
            status.empty();
            var percentVal = '0%';
            percent.html(percentVal);
            bar.width(percentVal);
        },
        uploadProgress: function(event, position, total, percentComplete) {
            var percentVal = percentComplete + '%';
            percent.html(percentVal);
            bar.width(percentVal);
        },
        complete: function(xhr) {
            status.html(xhr.responseText);
        }
    });
});
```

A somewhat strange, middle-ground solution would be to use the `nginx_uploadprogress` module, which provides its own progress reporting endpoint. The example configuration will look like this:

Copy

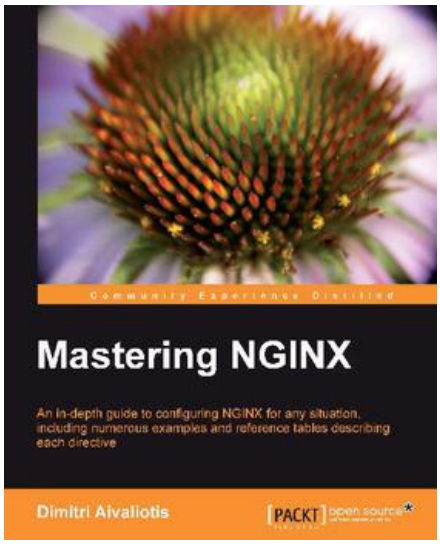
```
        location / {
            proxy_pass http://backend;
            track_uploads proxied 30s;
        }

        location ^~ /progress {
            report_uploads proxied;
        }
```

The client side will have to mark all the POSTs to the `/` location with a special header or GET parameter `X-Progress-ID`, which may also be used to get the progress of that particular upload via the `/progress` resource.

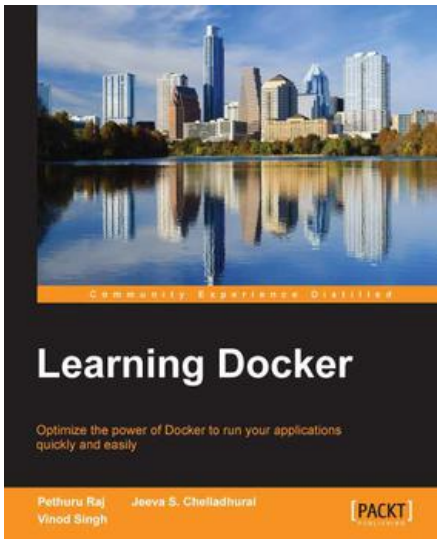
Next Section (/mapt/book/networking-and-servers/9781785288654/5/ch05lvl1sec29/solving-the-problem-of-an-idle-u

---

Related titles



(/mapt/book/networking_and_servers/9781849517447)



(/mapt/book/virtualization_and_cloud/9781784397937)