…or: How I made **Puppetdev**.

I don't like making things twice. You could call that „innovative" or „efficient". Basically, it's just laziness.

For Puppetdev, which is built using **Packer**, I needed base images, so I wouldn't have to cope with ISO urls, kickstart files and such. Luckily, the wonderful guys at **boxcutter** provide Packer templates for various operating systems. (I used the **Ubuntu** and **Debian** ones)

They also provide some space for customization, which sadly wasn't big enough for Puppetdev: I needed to provide multiple, additional scripts, feed them with user variables, additional files in the guest vm during the build process and provide files to be included into the vagrant box.

As Packer basically uses a bunch of JSON files to build and configure machines, I came up with the idea of a process, that modifies those JSON files and calls Packer with the modified versions.

A somewhat standardized format for modifying JSON files is called **JSONPatch**.
JSONPatch enables you to add, remove, move and change JSON properties and array members. It's actually quite cool stuff.

Sadly, Packer currently doesn't support JSONPatch (**I've opened a ticket for that**), so I needed to wrap some build environment around it. I'm quite comfortable with **Grunt** and could quickly put up some Grunt tasks that will take my patch files and apply them one at a time, thus building a modified version of boxcutter's original Packer template.

That way, I could simply inject the boxcutter repositories as submodules in my git repository and build my environment around it.

The rest was simply placing files in various directories to structure the work flow.

In the end, I had the following structure:

- *base.debian* => submodule of boxcutter/debian
- *base.ubuntu* => submodule of boxcutter/ubuntu
- *lib* => Special Grunt task modules
  - packerbuild.js => A grunt task, that applies the JSONPatches and calls Packer to build the boxes
  - packertest.js => Builds up a vagrant machine out of the generated boxes and runs

Serverspec tests on that

- *local* => A place for local modifications for people building up on Puppetdev
- *patches* => The JSONPatch files
- *scripts* => Custom scripts, that run during the build process
- *test* => A test environment for running the Serverspec files
- *vagrant.includes* => Files, that are included in the final Vagrant box
- *vagrantfiles* => Used Vagrantfile-templates
- *vars* => Variable files, that are provided to Packer when running packer build

I also implemented tests of the generated boxes using the wonderful **Serverspec** test framework. These tests are applied using a **Vagrant** machine, that is created inside the test folder. The Serverspec tests are run using the **vagrant-serverspec plugin**. After the tests complete, the machine is destroyed and everything is cleaned up.