Codify your developer VMs!

19 August 2019 | **Software Engineering** | **Torben Knerr**
Reading time: 3 minutes

What is a “Developer VM “?

A developer VM is a virtual machine image that contains the complete development toolchain that is needed to work on a specific project (including compiler toolchains, IDEs, system wide settings, etc…). The aim is to ensure a consistent environment not only between team members of that project (which may have completely different operating systems on their laptops), but also with the build agents in the CI environment.

Why would I need a developer VM anyway?

Some people hate it, others love it. No matter how you put it, it is a sure-fire way to ensure that the whole development team uses a consistent environment and avoids the typical “works on my machine” issues we all know about. Especially for us, executing and delivering a multitude of projects (and engineers joining and leaving these projects more or less frequently), developer VMs have become an indispensable tool. Not only for the consistency aspect, but also for reducing ramp-up times or being able to easily archive the development environment when the project enters the retirement phase.
On regulated projects this is even more relevant, since you need to validate and verify the development toolchain and need to ensure that everyone uses exactly that defined toolchain.

OK, but why should I automate my developer VM?

While we started out to craft these virtual machines manually (and with love) in the early days, we can nowadays easily automate the whole procedure with tools like Vagrant, Chef, Ansible, etc. Applying automation and Infrastructure-as-Code principles to developer VMs not only increases transparency and reproducibility, but also allows for an in-virtual-machine update mechanism to maintain consistency while the toolchain evolves during the lifetime of the project.

Finally, an automated developer VM should exhibit the following properties:

- Updatability – the toolchain can be updated with the push of a button from within the developer VM
- Testability – automated tests are run in the developer VM to ensure that it works as expected
- Installability – the developer VM is distributed as a standard VirtualBox / VMware image
- Adaptability – the development team can easily adapt and improve the developer VM as the toolchain evolves

- Reproducibility – changes to the development environment are transparent, traceable and reproducibly
  How can I automate it?
  The good news is that there are quite a few so-called “Configuration Management” tools out there, which essentially allow you to automate the provisioning of software and configuration on top of an existing operating system (in our case a fresh installation of the operating system of choice, within a virtual machine image).

  The bad news is that you have to choose one. In addition to the configuration management tools that do the provisioning, there is (fortunately) also a multitude of tools for testing the configuration of such a (virtual) machine. Depending on your preference, these are the tools you are likely to end up with:

  Rubyists will like

- Chef – as the configuration management tool
- Foodcritic – for linting their Chef recipes
- InSpec / ServerSpec – for testing the Chef-provisioned systems
  Pythonistas probably prefer

- Ansible – as the configuration management tool
- Ansible Lint – for linting their Ansible roles
- TestInfra – for testing the Ansible-provisioned systems
  It is not all a bed of roses though!
  Two things to highlight:

1. Developer VMs are clearly a compromise between consistency and performance: you save valuable time by avoiding the “works on my machine” issue and speeding up onboarding time for new team members. The price you pay is the runtime overhead for running the virtual machine in a hypervisor rather than working with the tools natively on your operating system of choice.
2. The automation part is sometimes hard (especially on Windows) and is not just a case of software installation but also configuration of settings like networking, firewalls, access rights, etc. These are not things that everyone knows how to automate, and you will have to grow and master these skills as part of your journey.
   I’m still thrilled, sounds like pretty cool stuff! What is the fastest way to get there?
   It may sound like a bit of effort is required to achieve this if you start from scratch, so that is why we have a template / skeleton project along with a tutorial for you. No excuses for doing it manually anymore!

This is what you need to get started:

- A ready-to-use template / skeleton project for a minimal, Linux based developer VM
- An example Java developer VM based on the above (see Pull Requests for a step-by-step guide)
- The accompanying conference talk for creating the example above
- A ready-to-use developer VM to with a toolchain for automating developer VMs