Remla Extension Proposal

Team 17

June 2025

1 Problem

Our release engineering shortcoming has to do with the robustness of vagrant. Over the duration of this course vagrant has given us several issues and complexities. Those issues being:

- Timeout of VM startup.
  It regurarly occurs that the startup takes such a long time that the program just times out.

- VM's getting stuck in startup.
  VM's have quite a significant chance to get stuck when starting up. This may also contribute to the previously mentioned problem. Only when opening virtual box and manually inspecting the VM's they start operating normally again. It only behaves normally when you analyse it, when you are not looking it gets stuck quite often.

- Complications using windows file paths.
  Vagrant defaultly creates a .vagrant folder on the host pc at the filepath where the up command was used. Then it creates a simple key pair that is used to connect to the virtual machine easily. When connecting vagrant with ansible this could lead to an error, because ansible has protection against using ssh keys that are in a folder with certain posix permissions enabled. Of course in vagrant we can chmod after creating the vm's but when vagrant is used in a windows filepath chmod does not work. This means we need to find a workaround to still be able to use vagrant with ansible in windows filepaths. Our team decided to create a script that generates a new ssh key, copies it to a new folder in your linux home filepath (so it can be used with WSL), chmods the keypair and then copies the public key to the vm on startup. Next we had to disable the default automatic key generation in the .vagrant folder otherwise ansible would still return an error. This seems very complicated and should be way simpeler in our eyes. A thread on stackoverflow contains people with the same struggles. User Majid alDosari validly reports "(why is is ansible's business to check ssh checks??)" which exactly describes our thoughts when it comes to this problem.

All these issues make it quite annoying to work with vagrant and ansible. It costs a lot of time starting the virtual machines and requires manual inspection to make sure the vms are not stuck. Furthermore when one vm fails or times out just running ”vagrant up” just makes it timeout again. The only way to try again is to do vagrant destroy and delete the images from virtual box and delete the .vagrant folder. Only then you can try to start it again and hope it will this time run all vms without problems.

A possible extension that could fix these shortcomings is the use of Canonicals multipass vm system which can create virtual machines quicker and easier and can be provisioned automatically or manually in a system that is similar to amazon webservices. This makes it easy to integrate and understand for students taking the Release engineering for machine learning applications course. To create vms it only takes some seconds. Furthermore it is OS-agnostic, meaning it will not have OS dependent functionality like vagrant's connection with ansible using the unprotected private key. There exists great documentation which can be understood by any level of developer. It can work with hyper-V, virtualBox, QEMU and linux' virtual machine system. To test the robustness of using canonicals multipass we can record the success rate of starting and provisioning vm's and record their startup and provisioning time. Furthermore we can test on different operating systems like Windows, MacOS, Linux, but also on WSL with windows filepaths. This way we can more accurately decide if it is a good idea to upgrade from vagrant to multipass.