

Remla Extension Proposal

Team 17

June 2025

1 Problem

Our release engineering flaw has to do with the robustness of Vagrant. During the duration of this course, vagrant has given us several issues and complexities. Those issues are

- VM start timeout. It typically occurs that the startup takes such a long time that the program just time 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 problem mentioned above. Only when opening virtual box and manually inspecting the VM's they start operating normally again. It only behaves normally when you analyse it, and when you are not looking, it gets stuck quite often.
- Complications using Windows file paths. By default Vagrant creates a `.vagrant` folder on the host PC at the file path 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 file path `chmod` does not work. This means we need to find a workaround to still be able to use vagrant with Ansible in windows file paths. Our team decided to create a script that generates a new ssh key, copies it to a new folder in your Linux home file path (so it can be used with WSL), `chmod` 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 simpler in our eyes. A thread on Stackoverflow contains people with the same struggles. User Majid alDosari validly reports "(why is is [sic] Ansible's business to check ssh checks [sic] ??)" 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 to start 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 Web Services. 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 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, Virtual-Box, 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 file paths. This way, we can more accurately decide if it is a good idea to upgrade from vagrant to Multipass.

2 sources

- <https://documentation.ubuntu.com/multipass/en/latest/>
- <https://stackoverflow.com/questions/29021246/ssh-fails-due-to-key-file-permissions-when-i-try-to-provision-a-vagrant-vm-with>
- <https://developer.hashicorp.com/vagrant>
- <https://docs.ansible.com/>
- <https://en.wikipedia.org/wiki/POSIX>
- <https://canonical.com/multipass>
- <https://www.microsoft.com/en-us/windows>
- <https://en.wikipedia.org/wiki/MacOS>
- <https://www.linux.org/>
- <https://en.wikipedia.org/wiki/Hypervisor>
- <https://www.virtualbox.org/>
- <https://www.qemu.org/>

- <https://aws.amazon.com/>