Creating Vagrant development machines for MariaDB

*How To and Best Practices*

Federico Razzoli
$ whoami

Hi, I’m Federico Razzoli from Vettabase Ltd

Database consultant, open source supporter, long time MariaDB and MySQL user,

I love abandonware

- vettabase.com
- Federico-Razzoli.com
MariaDB KB

Vettabase is contributing contents to the MariaDB KnowledgeBase:

**Automated MariaDB Deployment and Administration**

Examples

Examples from this talk are from:

github.com/VettaBase/vagrant-mariadb-examples

- It is a working example
- Provisioning with Shell or Ansible, check the README
- Check the helper tools
Good Practices

WARNING: I have opinions
Development machines principles

● They should be as similar to production as possible
  ○ "Works on my laptop" should mean "works in production"
  ○ Software packages and versions should be the same

● But they should cost much less than production
  ○ Setting up servers identical to production for each employee is unreasonable
  ○ Usually, a single VM with a server and the DB is enough
  ○ Using MS's from staging is fine, as long as tests won't destroy all data

● And they shouldn't stay in the way
  ○ On new employee onboarding, it's fine to use some time to setup dev VMs properly
  ○ During a normal workday, it's not
Vagrant Machines

- VMs or containers? Choose what you use in production
- Use one machine
  - Until there are reasons to use more
MariaDB for Development

Recommendations for MariaDB development instances:

- Same version as production (10.5.x)
  - Note for MySQL users: MySQL doesn't use semantic versioning anymore!
- Same variables that affect queries
- Extra tools / views / settings for easy debugging and to identify performance problems before they reach production
Identical variables

- sql_mode
- old_mode
- max_allowed_packet
- character_set_*
- collation_*
- tx_isolation
- lower_case_table_names
- innodb_strict_mode
- updatable_views_with_limit
Extra settings for troubleshooting

This is a checklist. Find examples in the repository.

- Log all queries into the Slow Log
- SQL Error Log
- userstat = 1
- performance_schema = 1
- Informational views
- pt-duplicate-key-checker from Percona Toolkit
Vagrantfiles
Vi / Emacs options

# -*- mode: ruby -*-
# vi: set ft=ruby :
Vagrantfile Structure

BOX = ENV["BOX"] || "ubuntu/bionic64"
Vagrant.require_version ">= 2.2.14"

Vagrant.configure("2") do |config|
  # set Vagrant options
  config.vm.box = BOX
  ...

  config.vm.provider "virtualbox" do |vb|
    # set provider-level options
    ...
  end
  config.vm.provision "vmware_fusion"

  config.vm.provision :shell, path: "bootstrap.sh"
end
Provider: virtualbox

cfg.vm.provider "virtualbox" do |vb|
  vbcustomize ["modifyvm", :id, "name", "lamp"]
  vbcustomize ["modifyvm", :id, "--memory", 1024 * 4]
  vbcustomize ["modifyvm", :id, "--cpuhotplug", "on"]
  vbcustomize ["modifyvm", :id, "--cpus", "2"]
  vbcustomize ["modifyvm", :id, "--vram", "4"]
end
Provider: virtualbox

```ruby
config.vm.provider "virtualbox" do |vb|
  vb.customize %q{
    "modifyvm", :id, "name", "lamp"
  }
  vb.customize %q{
    "modifyvm", :id, "--memory", 1024 * 4
  }
  vb.customize %q{
    "modifyvm", :id, "--cpuhotplug", "on"
  }
  vb.customize %q{
    "modifyvm", :id, "--cpus", "2"
  }
  vb.customize %q{
    "modifyvm", :id, "--vram", "4"
  }
end
```
Provider: virtualbox
Provider: **virtualbox**

```ruby
config.vm.provider "virtualbox" do |vb|
  vb.customize [:modifyvm, :id, "name", "lamp"]
  vb.customize [:modifyvm, :id, "--memory", 1024 * 4]
  vb.customize [:modifyvm, :id, "--cpuhotplug", "on"]
  vb.customize [:modifyvm, :id, "--cpus", "2"]
  vb.customize [:modifyvm, :id, "--vram", "4"]
end
```
Provider: virtualbox

```ruby
config.vm.provider "virtualbox" do |vb|
  vb.customize ["modifyvm", :id, "name", "lamp"]
  vb.customize ["modifyvm", :id, "--memory", 1024 * 4]
  vb.customize ["modifyvm", :id, "--cpuhotplug", "on"]
  vb.customize ["modifyvm", :id, "--cpus", "2"]
  vb.customize ["modifyvm", :id, "--vram", "4"]
end

VBoxManage modifyvm "VM name" --plugcpu 1
VBoxManage modifyvm "VM name" --unplugcpu 1
```
Provider: virtualbox

```ruby
config.vm.provider "virtualbox" do |vb|
    vb.customize ["modifyvm", :id, "name", "lamp"]
    vb.customize ["modifyvm", :id, "--memory", 1024 * 4]
    vb.customize ["modifyvm", :id, "--cpuhotplug", "on"]
    vb.customize ["modifyvm", :id, "--cpus", "2"]
    vb.customize ["modifyvm", :id, "--vram", "4"]
end
```
Provider: virtualbox

```ruby
config.vm.provider "virtualbox" do |vb|
  vb.customize ["modifyvm", :id, "name", "lamp"]
  vb.customize ["modifyvm", :id, "--memory", 1024 * 4]
  vb.customize ["modifyvm", :id, "--cpuhotplug", "on"]
  vb.customize ["modifyvm", :id, "--cpus", 2]
  vb.customize ["modifyvm", :id, "--vram", 4]
end
```
Provisioners

Vagrant.configure("2") do |config|

  ...

  config.vm.provision :shell, path: "bootstrap.sh"

  # OR

  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "mariadb.yml"
  end
end
"Uploading" files to the VM

- name: Upload my.cnf
copy:
  src: ./files/my.cnf
dest: /etc/mysql/conf.d/
Summary

1. Start with a single machine
2. Prefer VMs over containers
3. MariaDB variables that affect queries = production
4. Use variables in Vagrantfile
5. Start with Vagrant.require_version
6. Learn some Ruby
7. Configure the provisioner (hw resources)
8. Use automation tools
Thanks for attending!

github.com/Vettabase/vagrant-mariadb-examples

THANK YOU :)

VettaBase ltd.