Multi-Máquina

Vagrant puede definir y controlar múltiples máquinas invitadas por archivo Vagrant. Esto se conoce como un entorno "multi-máquina".

Estas máquinas generalmente pueden funcionar juntas o de alguna manera están asociadas entre sí. Aquí hay algunos casos de uso que las personas están utilizando en entornos de máquinas múltiples para hoy:

- Modelado preciso de una topología de producción de varios servidores, como la separación de un servidor web y de base de datos.
- Modelar un sistema distribuido y cómo interactúan entre sí.
- Probar una interfaz, como una API a un componente de servicio.
- Pruebas de desastres: máquinas que mueren, particiones de red, redes lentas, vistas incoherentes del mundo, etc.

Históricamente, la ejecución de entornos complejos como estos se realizó aplanándolos en una sola máquina. El problema con eso es que es un modelo inexacto de la configuración de producción, que puede comportarse de manera muy diferente.

Con la característica de máquinas múltiples de Vagrant, estos entornos se pueden modelar en el contexto de un único entorno Vagrant sin perder ninguno de los beneficios de Vagrant.

Definiendo Máquinas Múltiples

Se definen varias máquinas dentro del mismo proyecto Vagrantfile (/docs/vagrantfile/) usando la `config.vm.define`llamada a método. Esta directiva de configuración es un poco divertida, ya que crea una configuración Vagrant dentro de una configuración. Un ejemplo muestra esto mejor:

```
Vagrant.configure("2") do |config|
  config.vm.provision "shell", inline: "echo Hello"

  config.vm.define "web" do |web|
    web.vm.box = "apache"
  end

  config.vm.define "db" do |db|
    db.vm.box = "mysql"
  end
end
```

Como puede ver, `config.vm.define`toma un bloque con otra variable. Esta variable, como la `web`anterior, es *exactamente* igual que la `config`variable, excepto que cualquier configuración de la variable interna se aplica solo a la máquina que se está definiendo. Por lo tanto, cualquier configuración `web`activada solo afectará a la `web`máquina.

And importantly, you can continue to use the `config` object as well. The configuration object is loaded and merged before the machine-specific configuration, just like other Vagrantfiles within the Vagrantfile load order (/docs/vagrantfile/#load-order).

If you are familiar with programming, this is similar to how languages have different variable scopes.

When using these scopes, order of execution for things such as provisioners becomes important. Vagrant enforces ordering outside-in, in the order listed in the Vagrantfile. For example, with the Vagrantfile below:

```
Vagrant.configure("2") do |config|
  config.vm.provision :shell, inline: "echo A"

  config.vm.define :testing do |test|
    test.vm.provision :shell, inline: "echo B"
  end

  config.vm.provision :shell, inline: "echo C"
end
```

The provisioners in this case will output "A", then "C", then "B". Notice that "B" is last. That is because the ordering is outside-in, in the order of the file.

If you want to apply a slightly different configuration to multiple machines, see this tip (/docs/vagrantfile/tips.html#loop-over-vm-definitions).

Controlling Multiple Machines

The moment more than one machine is defined within a Vagrantfile, the usage of the various `vagrant` commands changes slightly. The change should be mostly intuitive.

Commands that only make sense to target a single machine, such as `vagrant ssh`, now *require* the name of the machine to control. Using the example above, you would say `vagrant ssh web` or `vagrant ssh db`.

Other commands, such as `vagrant up`, operate on *every* machine by default. So if you ran `vagrant up`, Vagrant would bring up both the web and DB machine. You could also optionally be specific and say `vagrant up web` or `vagrant up db`.

Additionally, you can specify a regular expression for matching only certain machines. This is useful in some cases where you specify many similar machines, for example if you are testing a distributed service you may have a `leader` machine as well as a `follower0`, `follower1`, `follower2`, etc. If you want to bring up all the followers but not the leader, you can just do `vagrant up /follower[0-9]/`. If Vagrant sees a machine name within forward slashes, it assumes you are using a regular expression.

Communication Between Machines

In order to facilitate communication within machines in a multi-machine setup, the various networking (/docs/networking/) options should be used. In particular, the private network (/docs/networking/private_network.html) can be used to make a private network between multiple machines and the host.

Especificando una máquina primaria

También puede especificar una *máquina primaria* . La máquina primaria será la máquina predeterminada utilizada cuando no se especifica una máquina específica en un entorno de varias máquinas.

Para especificar una máquina predeterminada, simplemente márquela primaria al definirla. Solo se puede especificar una máquina primaria.

```
config.vm.define "web", primary: true do |web|
  # ...
end
```

Máquinas de inicio automático

De forma predeterminada en un entorno de varias máquinas, `vagrant up`se iniciarán todas las máquinas definidas. La `autostart`configuración le permite decirle a Vagrant que *no* inicie máquinas específicas. Ejemplo:

```
config.vm.define "web"
config.vm.define "db"
config.vm.define "db_follower", autostart: false
```

Cuando se ejecuta `vagrant up`con la configuración anterior, Vagrant iniciará automáticamente las máquinas "web" y "db", pero no iniciará la máquina "db_follower". Puede forzar manualmente la ejecución de la máquina "db_follower" `vagrant up db_follower`.