we set up fully functional virtual machine for basic web development. How do we clean up our development environment?

We can **suspend**, **halt**, or **destroy** the guest machine. Each of these options have pros and cons.

1. **vagrant suspend** : We can suspend the virtual machine by **vagrant suspend** command. The current running state of the machine will be saved and stopped. When we're ready to begin working again, we can just run **vagrant up**, then it will be resumed from where we left off.   
   The main benefit of this method is that it is super fast, usually taking only 5 to 10 seconds to stop and start our work.  
   Suspending virtual machine, however, still eats up our disk space, and requires even more disk space to store all the state of the virtual machine RAM on disk.

vagrant suspend:

**k@laptop:~/my\_vagrant$ vagrant suspend**

**==> default: Saving VM state and suspending execution...**

vagrant up:

**k@laptop:~/my\_vagrant$ vagrant up**

**Bringing machine 'default' up with 'virtualbox' provider...**

**==> default: Checking if box 'ubuntu/trusty32' is up to date...**

**==> default: Resuming suspended VM...**

**==> default: Booting VM...**

**==> default: Waiting for machine to boot. This may take a few minutes...**

**default: SSH address: 127.0.0.1:2222**

**default: SSH username: vagrant**

**default: SSH auth method: private key**

**default: Warning: Connection refused. Retrying...**

**==> default: Machine booted and ready!**

1. **vagrant halt** : We can halt the virtual machine using **vagrant halt** command. It will gracefully shut down the guest operating system and power down the guest machine. We can use **vagrant up** when we're ready to boot it again.  
   The benefit of this method is that it will cleanly shut down our machine, preserving the contents of disk, and allowing it to be cleanly started again.  
   The downside is that it'll take some extra time to start from a cold boot, and the guest machine still consumes disk space.

vagrant halt:

**k@laptop:~/my\_vagrant$ vagrant halt**

**==> default: Attempting graceful shutdown of VM...**

vagrant up:

**k@laptop:~/my\_vagrant$ vagrant up**

**Bringing machine 'default' up with 'virtualbox' provider...**

**==> default: Checking if box 'ubuntu/trusty32' is up to date...**

**==> default: VirtualBox VM is already running.**

**k@laptop:~/my\_vagrant$ vagrant halt**

**==> default: Attempting graceful shutdown of VM...**

**k@laptop:~/my\_vagrant$ vagrant up**

**Bringing machine 'default' up with 'virtualbox' provider...**

**==> default: Checking if box 'ubuntu/trusty32' is up to date...**

**==> default: Clearing any previously set forwarded ports...**

**==> default: Clearing any previously set network interfaces...**

**==> default: Preparing network interfaces based on configuration...**

**default: Adapter 1: nat**

**==> default: Forwarding ports...**

**default: 80 => 4567 (adapter 1)**

**default: 22 => 2222 (adapter 1)**

**==> default: Booting VM...**

**==> default: Waiting for machine to boot. This may take a few minutes...**

**default: SSH address: 127.0.0.1:2222**

**default: SSH username: vagrant**

**default: SSH auth method: private key**

**default: Warning: Connection timeout. Retrying...**

**default: Warning: Connection timeout. Retrying...**

**default: Warning: Connection timeout. Retrying...**

**default: Warning: Connection timeout. Retrying...**

**default: Warning: Remote connection disconnect. Retrying...**

**default: Warning: Remote connection disconnect. Retrying...**

**default: Warning: Remote connection disconnect. Retrying...**

**default: Warning: Remote connection disconnect. Retrying...**

**default: Warning: Remote connection disconnect. Retrying...**

**==> default: Machine booted and ready!**

**==> default: Checking for guest additions in VM...**

**==> default: Mounting shared folders...**

**default: /vagrant => /home/k/my\_vagrant**

**==> default: Machine already provisioned. Run `vagrant provision` or use the `--provision`**

**==> default: to force provisioning. Provisioners marked to run always will still run.**

1. **vagrant destroy** : We can destroy the virtual machine by **vagrant halt** command. It will remove all traces of the guest machine from our system. It'll stop the guest machine, power it down, and remove all of the guest hard disks. Again, when we're ready to work again, just issue a **vagrant up**.   
   The benefit of this is that no trace is left on our machine. The disk space and RAM consumed by the guest machine is reclaimed and our host machine is left clean.   
   The downside is that **vagrant up** to get working again will take some extra time since it has to reimport the machine and reprovision it.
2. Rebuild - vagrant up
3. Since the Vagrant environment is already all configured via the **Vagrantfile**, we simply have to run a **vagrant up** at any time and Vagrant will recreate your work environment.
4. **$ vagrant up**