Overview:

There are eight(8) hosts that we are working with:

* docker-build - This server is used to build Docker containers. The container images are then saved off this host.
* docker01 and docker02 and …. - These hosts are used to run containers. Containers from build are uploaded and made available to all of these hosts.

All these hosts in the environment have Docker and the Docker SDK for Python installed. The SDK is a requirement for using the Ansible Docker modules.

Workflow:

The first step in this workflow is building an actual container image. A simple Ansible playbook can then be used to build a container image based on a given Dockerfile.

The build playbook creates a build directory on the "build" server, copies the Dockerfile to this directory, and then builds the container using the docker\_image Ansible module. You can log into the build server after the playbook completes to see that the container image has been successfully built and is available to launch a container.

Container images will be saved to a tarball using the docker save command. Tarballs provide a convenient way to "export" your container images. The Ansible Docker image module also includes support to export an image to a tar file.

Next playbook first archives the image using the docker\_image module and then fetches the file from the remote server and places it into the local directory. After successfully running the playbook, you have a tar file in your local directory that contains the contents of the image.

With the image now downloaded to your local system, you can again use the Ansible Docker image module to upload the tarball to all of your Docker hosts and import it. Once imported, the container image is available to launch containers. You can then launch a container on one of the hosts ie. http://34.76.206.87:8080/

## [Ansible Basic Concepts](https://scotch.io/tutorials/getting-started-with-ansible" \l "toc-ansible-basic-concepts)

* **Inventory –** Ansible works against multiple systems/hosts at the same time, these hosts are selected from an inventory file ( by default in etc/ansible/hosts ). Through inventory files, you can specify meaningful groups of hosts that Ansible will provision. You also can specify group variables or host variables that will help to control how Ansible interacts with remote hosts and they will be available later in playbooks. [webservers] for example is a name of a group, and any hosts listed under it will be part of that group. Ansible does not need to be installed on managed machines. Local inventory file can be created and used with “-i name” parameter.
* **Modules –** discrete units of code which can be used in a playbook task to be executed on a target machine. Usually grouped into ready made collections available from ansible falaxy.
* **Control node –** a machine that has ansible installed. Used to send out tasks and commands to managed machines. Requires python installation. Can not be installed on windows.
* **Collections** - Collections are a distribution format for Ansible content that can include playbooks, roles, modules, and plugins. You can install and use collections through Ansible Galaxy.
* **Tasks** - The units of action in Ansible. Used to execute actions on target machines when a play is run. They are sequential, running one after another and waiting for the execution of the previous one.
* **Playbooks -** Ordered lists of tasks, saved so you can run those tasks in that order repeatedly. Playbooks can include variables as well as tasks. Playbooks are written in YAML and are easy to read, write, share and understand.
* **Roles** - Roles provide a framework for fully independent, or interdependent collections of variables, tasks, files, templates, and modules. In Ansible, the role is the primary mechanism for breaking a playbook into multiple files. This simplifies writing complex playbooks, and it makes them easier to reuse. The breaking of playbook allows you to logically break the playbook into reusable components.

Installation

Ansible is best installed via pip, python’s package manager. Before install, it is best to ensure python is installed and python3 is referenced by the main python variable, same for pip. Instructions can be found at: <https://www.vultr.com/docs/update-python3-on-debian/>.

Once pip and python are installed, referenced and up to date, then install ansible to the system with

**sudo python -m pip install ansible .**

This should install ansible and create default configs at **etc/ansible.** The hosts file which is the default inventory, and the ansible.cfg, which is the default ansible config and can be used to setup ansible configuration settings ( such as **private\_key\_file** = /home/user/.ssh/id\_rsa , to tell ansible the particular private key to use ).

After installation, proceed to create a target ansible folder which will contain the playbook files. A suggested directory structure can be found at [https://docs.ansible.com/ansible/2.8/user\_guide/playbooks\_best\_practices.html#directory-layout](https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html" \l "directory-layout)

Some notes:

* group\_vars – this folder is where the variables for different groups will be kept at. Create an all.yml file here to list key-value pairs which can be called upon inside of ansible tasks as “{{var}}”.
* Roles: - inside of the roles folder, custom roles can be fined to be reused for different setups.

Ready made collections exist for working with different tools or setups. For example, a docker collection provides modules to execute docker-related commands in a clear and faster way. Download from ansible galaxy with <https://galaxy.ansible.com/community/docker> . Other collections can be found on ansible galaxy and depend on use case.

Ansible galaxy roles example :

ansible-galaxy install geerlingguy.php

ansible-galaxy install geerlingguy.composer

ansible-galaxy install geerlingguy.drupal

ansible-galaxy install geerlingguy.apache