Red Hat

Automating Container Management with Ansible

Containers and Automation

Objectives

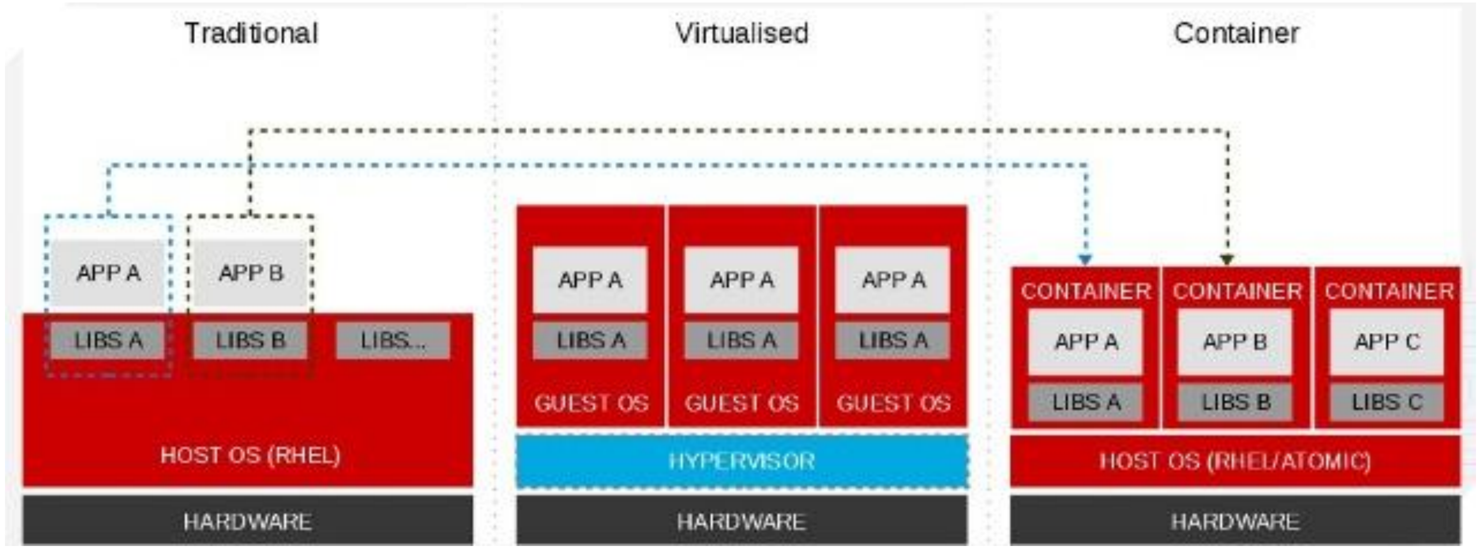
This module provides an overview of what containers are and why they are important, as well as an overview of Ansible automation.

Overview of Containers and Automation

What are Containers?

Containers are images that allow users to package applications, deliver, and execute them in an isolated way from the rest of the system.

They are similar to virtualization (VMs), but differ in that containers share the hosts' kernel instead of creating one. Sharing the kernel greatly reduces the size and runtime overhead compared to virtualization, which in turn makes sharing and running containers much more efficient.



Benefits of Containers

- Isolation is the biggest benefit.
  - Process, network, file, OS, and environment isolation from other containers and the host.

- Low hardware footprint.
  - Sharing the kernel with the host operating system means less CPU and memory overhead compared to virtualization.

- Good performance and rapid deployment.
  - There is no need to install the entire underlying operating system.
  - There is less process overhead since the underlying system calls are direct to the host.

- Reusability and portability.
  - Container images provide a portable format to copy and move containers around.
  - Easily shared because storage needed can be smaller than traditional or virtual architectures.

Container Architecture

From the Linux kernel's perspective, a container is a process with restrictions.

- Namespaces
  - Provide isolation of resources like network interfaces, the process ID list, mount points, IPC resources, and the system's hostname information.
- Control Groups (cgroups)
  - Provide restrictions on how many system resources (such as CPU and memory) that container processes can use.
- SELinux (Security Enhanced Linux)
  - A mandatory access control system that protects host processes from each other.
  - Provides the security that keeps container processes separated.
  - Container processes run as a confined SELinux type that limits access to host resources.

Images and Registries

- Image
    - An image is a file-system bundle that contains all dependencies required to execute a process: files in the file system, installed packages, available resources, running processes, and kernel modules.

- Registry
    - Registries store images for public or private use. Common registries are Quay, Docker Hub, and Red Hat registry.

Automation of Containers

- Allows you to remove manual steps.
- Enables automatic deployments and testing, improved repeatability, and better scaling.
- The lifecycle of a container is easily handled with Ansible modules.
- The contents that make up a container image, are files that can be managed by versioning systems.
- Involving version control improves automation on container and image provisioning.

Demonstration Environment for the Course

- One Linux system (the "control node") will be installed with Ansible.
- You will set up a second Linux system to use Docker to manage containers.

- In this course, you will see demos where Ansible will be used to automate these tasks:

    - Install, configure, and start Docker on the managed host
    - Build container images usable by Docker
    - Retrieve and store container images in a container image repository
    - Start, restart, and stop containers managed by Docker
    - Start or stop a collection of multiple containers on the same host through Docker Compose

Overview of Ansible Automation

Requirements for Running Ansible

- You run Ansible commands on the control node, and it executes and configures on managed hosts.
- Control node requirements
  - A recent Linux distribution, FreeBSD, or macOS.
  - Python 3 (versions 3.5 and higher) or Python 2 (version 2.7) installed.
- Managed host requirements
  - Python 2 (version 2.6 or later) or Python 3 (version 3.5 or later) for most modules
  - A way to communicate with the managed host, typically over SSH, API calls, or WinRM.
  - A way to authenticate access to the managed host.
- See https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Requirements for Installing Ansible

- The Ansible software only needs to be installed on the control node.
- In this course, we will assume that you are using Red Hat Enterprise Linux 7 or CentOS 7.
- For Red Hat Enterprise Linux 7, enable the repository for Ansible:
  - **sudo subscription-manager repos --enable=ansible-2-for-rhel-7-x86_64-rpms**
- For CentOS 7, enable access to the repository for EPEL 7:
  - **sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm**
- To install Ansible on a RHEL or CentOS system that has the right repositories configured, run:
  - **sudo yum install ansible**

Inventory

- An inventory defines a collection of hosts managed by Ansible.
- Hosts can be assigned to groups.
- Groups can be managed collectively.
- Groups can contain child groups.
- Hosts can be members of multiple groups.
- Variables can be set that apply to hosts and groups.

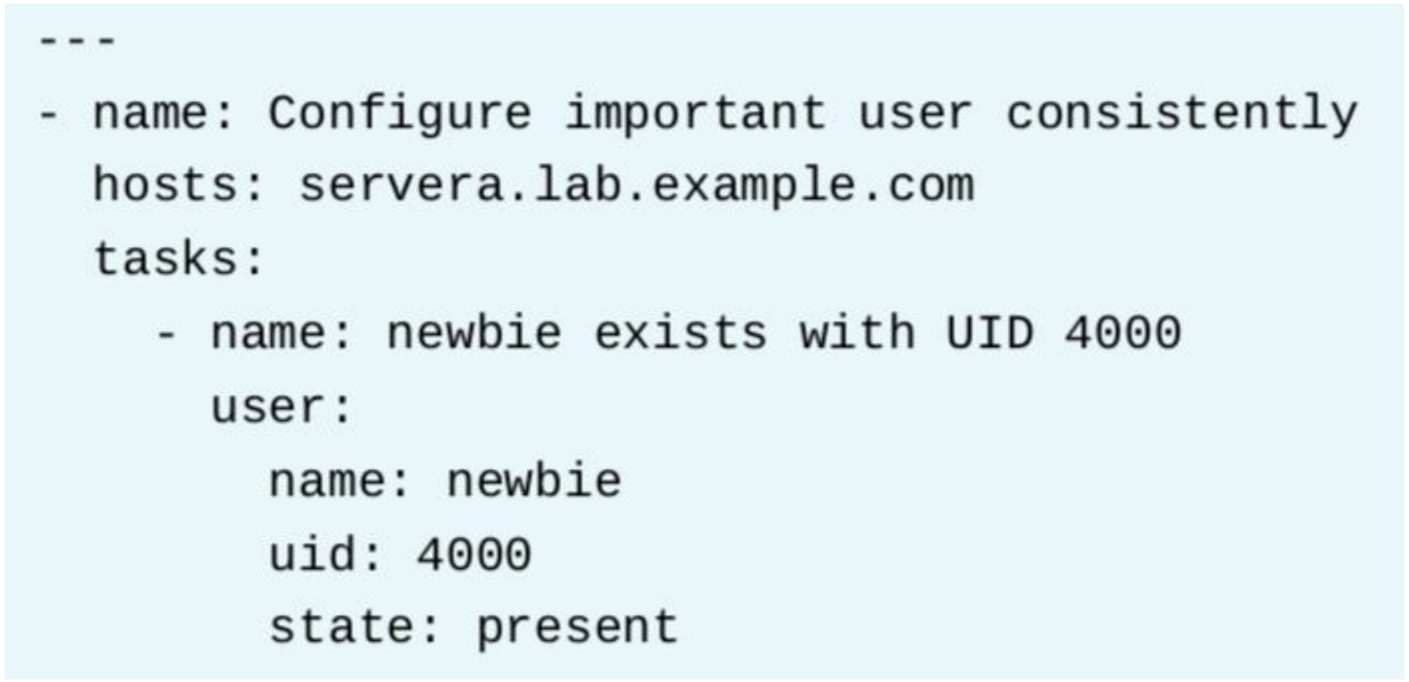
```
[usa]
washington1.example.com
washington2.example.com

[canada]
ontario01.example.com
ontario02.example.com

[north-america:children]
canada
usa
```

Playbooks

Playbooks are easy-to-follow YAML files.

- A playbook is a list of plays that each contain a list of tasks to run on a set of managed hosts.
- A task uses a module to accomplish its purpose.
- Tasks in a play are listed in the order to be run on the managed host.
- Variables are referenced by placing the variable name in double curly braces (**`{{ }}`**).
- Ansible supports iterating a task over a list of items, with the **`loop`** keyword.
- The **`when`** statement is used to run a task when a condition is met.

Sample Playbook



```yaml
---
- name: Configure important user consistently
  hosts: servera.lab.example.com
  tasks:
    - name: newbie exists with UID 4000
      user:
        name: newbie
        uid: 4000
        state: present
```