Automating Container Management with Ansible

Automating Docker Configuration

Objectives

This module covers the architecture of Docker, basic Docker CLI commands, and how to install Docker with Ansible.

Red Hat

Architecture of a Docker Implementation

Docker Architecture

Docker uses a client-server architecture.

Client

The command-line tool (**`docker`**) is responsible for communicating with a server using a RESTful API to request operations.

Server

This service, which runs as a daemon on an operating system, does the heavy lifting of building, running, and downloading container images.

The daemon can run either on the same system as the docker client or remotely.

Docker Daemon

Docker daemon needs to be started for Docker commands to work. Manually you can run:

```
# systemctl enable --now docker
```

Because the daemon is run as the root user, the Docker commands must also be run as root. Other container runtimes like Podman do not require root to run containers.

Docker CLI

Some of the most common commands are:

Command	Description
docker search <term>	Search an image registry for an image related to the term.
docker pull <image>	Download an image or images from a registry.
docker run <image>	Creates the container from the image.
docker ps	Lists the containers.
docker images	List the downloaded images.

Installing Docker with Ansible

Red Hat

Objectives

- Use Ansible to automate the installation of Docker on a managed host.

Installing Docker with Ansible

- You can write a simple play to install Docker on RHEL 7 automatically.

- You will need to:
  - Make sure that the RHEL 7 managed host is configured to get updates for the right channels.
  - Make sure that the *docker* package is installed.
  - Make sure that the **docker** service is enabled and started.

Configure Updates (RHEL 7)

- At right is the first part of a play to install Docker on RHEL 7 or CentOS 7.
- It installs Docker on managed hosts in the group *docker_hosts*.
- The **redhat_subscription** task makes sure RHEL hosts are registered for updates. Its three variables are assumed to be set elsewhere as inventory variables.
- The **rhsm_repository** task makes sure RHEL 7 hosts enable the correct software channels.

```
- name: Make sure Docker is installed correctly
  hosts: docker_hosts

  tasks:
    - name: Is this a registered RHEL host
      redhat_subscription:
        state: present
        username: "{{ rh_user }}"
        password: "{{ rh_pass }}"
        pool_ids: "{{ pool_ids }}"
      when:
        - ansible_facts['distribution'] == "RedHat"

    - name: Enable RHEL repositories
      rhsm_repository:
        name: "{{ item }}"
        state: enabled
      loop:
        - rhel-7-server-rpms
        - rhel-7-server-extras-rpms
      when:
        - ansible_facts['distribution'] == "RedHat"
        - ansible_facts['distribution_major_version'] == "7"
```

Install Docker

- The rest of the play from the previous slide is shown at right.
- The **yum** task makes sure the latest version of Docker is installed from the RHEL 7 repository
- The **service** task makes sure that the Docker daemon is running and automatically starts at boot.

```
- name: Make sure Docker is up to date
  yum:
    name: docker
    state: latest

- name: Make sure Docker is running
  service:
    name: docker
    state: started
    enabled: yes
```

Validate Installation

After running the playbook with the variables filled in, Docker will be installed, started, and enabled.

Validate the install by running the hello world container.

```
# docker run hello-world
```