MICRO FOCUS

IDM Container deployment – tips and tricks

TTP November 2021

Rodrigo Gomez

Agenda

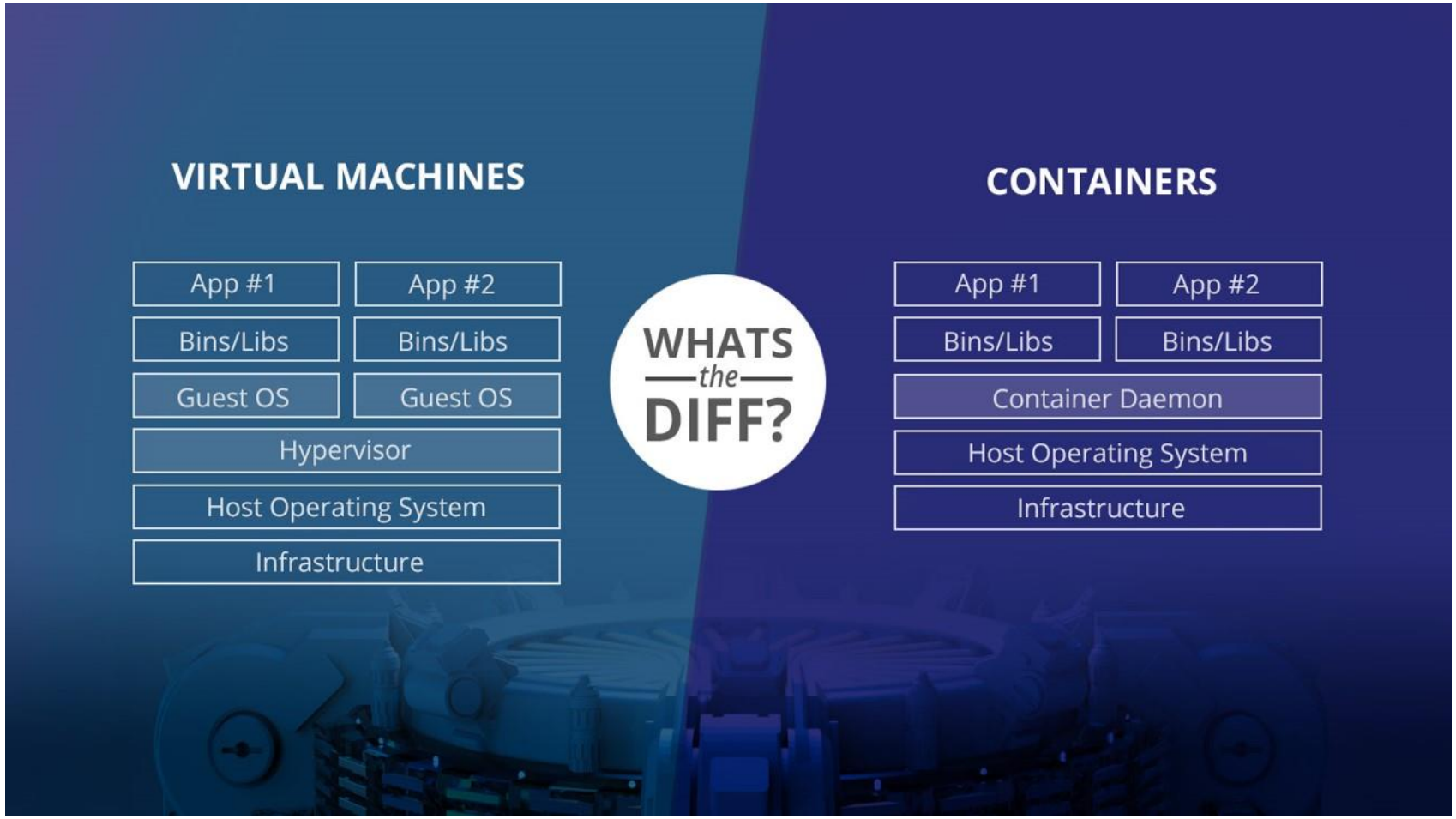
- Introduction to Docker
- Identity Manager component overview
- IDM Container installation approaches
- Demo – Single server walk-through
- Demo: Running the ansible script

Introduction to Docker

What is Docker ?

- Docker is a platform designed to create, deploy, and run applications by using Containers. Containers allow us to package up an application on a Base OS with all its dependencies, such as libraries and other packages, and ship it all out as one package.
- Docker performs OS-level virtualization with legacy Unix kernel features like chroot, cgroups and namespaces.

Virtualization vs Containerization



Why Docker?

- **High Portability:** Any application running in containers can be deployed easily to any Docker supported operating systems and hardware platforms.
- **Ease of deployment:** Containers allow applications to be more rapidly deployed, upgraded or even scaled through Orchestration tools.
- **Consistency:** There will be no impact on the functionality of the containers regardless of where they are deployed, as the base Operating System of the container remains same.

No more of "Works on my machine but not on your machine"

Docker Components

Docker Engine:

- Docker daemon (dockerd) – runs on the Docker Host machine and manages Containers. Listens for requests from Docker CLI.

Image and Dockerfile:

A Docker Image is a lightweight, standalone, multi-layered, immutable, executable package of software that has everything you need to run an application. It is unusable unless started as a Container. Multiple containers can be spawned out of the same image.

Registry:

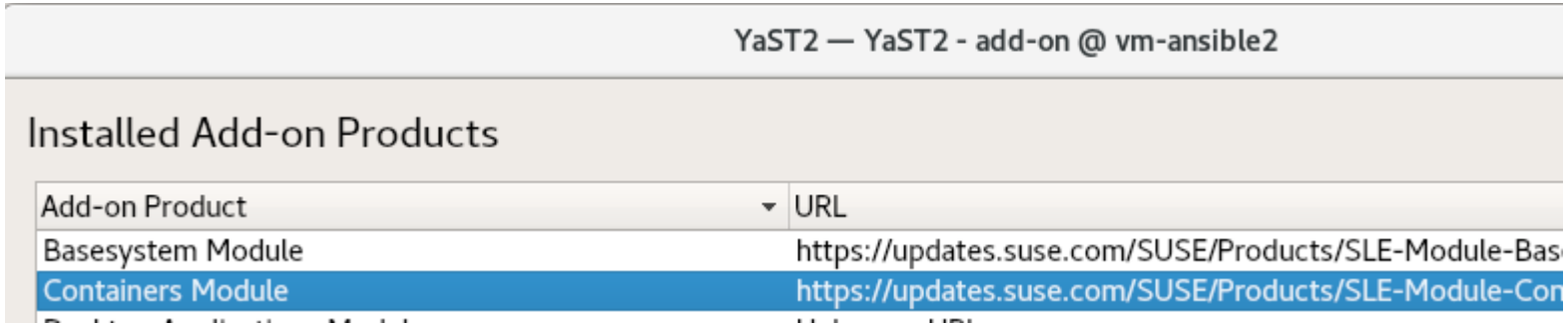
- Public Registry – accessible to all Docker users. Example: Docker Hub
- Private Registry – a company specific Image repository.

Containers:

An instance of an image is a Container. Docker adds an additional R/W layer called ‘container layer’ on top of the Image layers which will hold Container specific configuration and data. This layer is retained even when the Container is stopped, but not after the container is removed.

Installing Docker on SLES servers

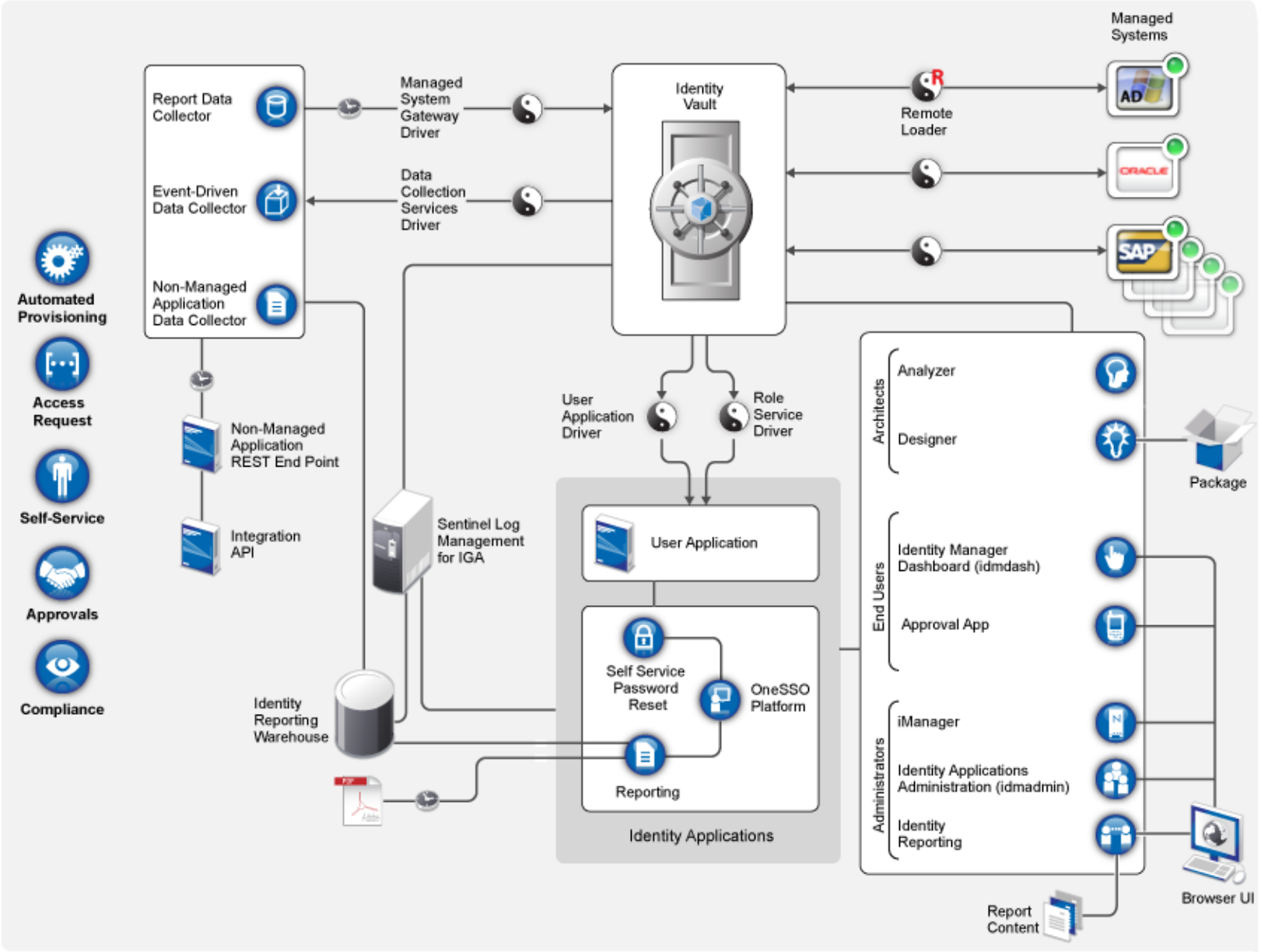
- Add the “Containers Module” as Add-on product



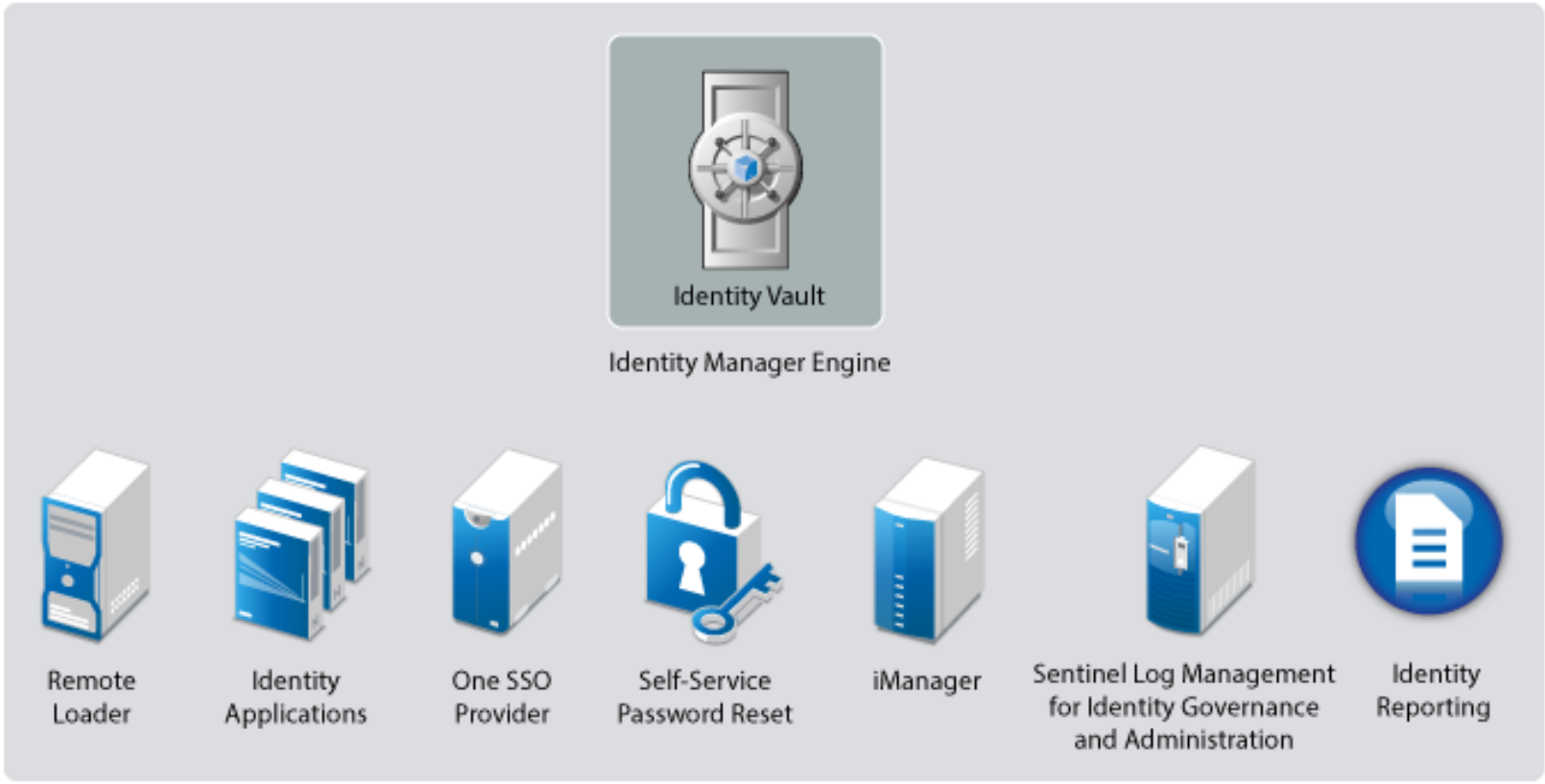
- Install the “docker” package, which will install Docker version 19.0.3, the minimum required for IDM deployment.
- From 4.8.4, the minimum version required is 20.10.6

Identity Manager components

Functional overview



Main components overview



IDM Docker images

- Identity_Manager_4.8_Containers.tar.gz

```
 476034048 Oct 31 08:19 IDM_48_activemq.tar.gz
 403267072 Oct 31 08:29 IDM_48_fanoutagent.tar.gz
 470553088 Oct 31 08:24 IDM_48_formrenderer.tar.gz
2043589120 Oct 31 07:57 IDM_48_identityapplication.tar.gz
1398136320 Oct 31 08:14 IDM_48_identityengine.tar.gz
1997169664 Oct 31 08:07 IDM_48_identityreporting.tar.gz
 228174336 Oct 31 08:40 IDM_48_identityutils.tar.gz
 275106816 Oct 31 08:44 IDM_48_idm_conf_generator.tar.gz
 944158208 Oct 10 16:26 IDM_48_iManager320.tar
 691390464 Oct 31 07:45 IDM_48_osp.tar.gz
  41698816 Oct 31 08:44 IDM_48_postgres.tar.gz
 657768960 Oct 31 08:38 IDM_48_remoteloader.tar.gz
 234422433 Oct  8 19:41 IDM_48_sspr.tar.gz
```

The Docker images are available for the following Identity Manager components:

- Identity Manager Engine
- Remote Loader
- iManager
- One SSO Provider (OSP)
- Fanout Agent
- ActiveMQ
- PostgreSQL (Redistribution)
- Identity Applications
- Self Service Password Reset (SSPR)
- Form Renderer
- Identity Reporting

IDM Container installation approaches

Container documentation

- Go to the IDM documentation page – section “Identity Manager Containers”
  - Deployment Guide 4.8 -> Part of the Setup Guide for Linux
  - Deployment Guide 4.8.1/4.8.2 -> Independent document specific to containers
  - Deployment Guide 4.8.3/4.8.4 -> Part of the 4.8.3/4.8.4: Installation and Upgrade Guide

- Message on support:

  We are shipping a preview version of Docker Container-based deployment with Identity Manager 4.8 for customers to use and provide feedback. Customers willing to deploy containers in a production environment will be supported only with a Professional Services engagement.

Container documentation Installation methods

Prepare your environment:

- Configure your volume data – mapping to /config
- Prepare your network access
- Create silent.properties file

Two different deployment scenarios:

- Single Server deployment
- Distributed Server deployment

Script assistance:

- Single server – bash-based script – community
- Distributed setup – ansible based script (introduced with 4.8.3)

Single host installation – shell script approach

- Shell script executes the necessary docker commands to perform the installation of key components
- It uses host networking -> Each container binds to the local interface
- A cleanup script is also provided, to be able to clean up the images.
- Detailed steps on:

  https://community.microfocus.com/t5/Identity-Manager-Tips/How-to-automate-the-deployment-of-Identity-Manager-Containers-on/ta-p/2765236

Single host installation – plan port usage

- Host networking doesn’t expose specific ports, but there can’t be port conflicts between different containers:

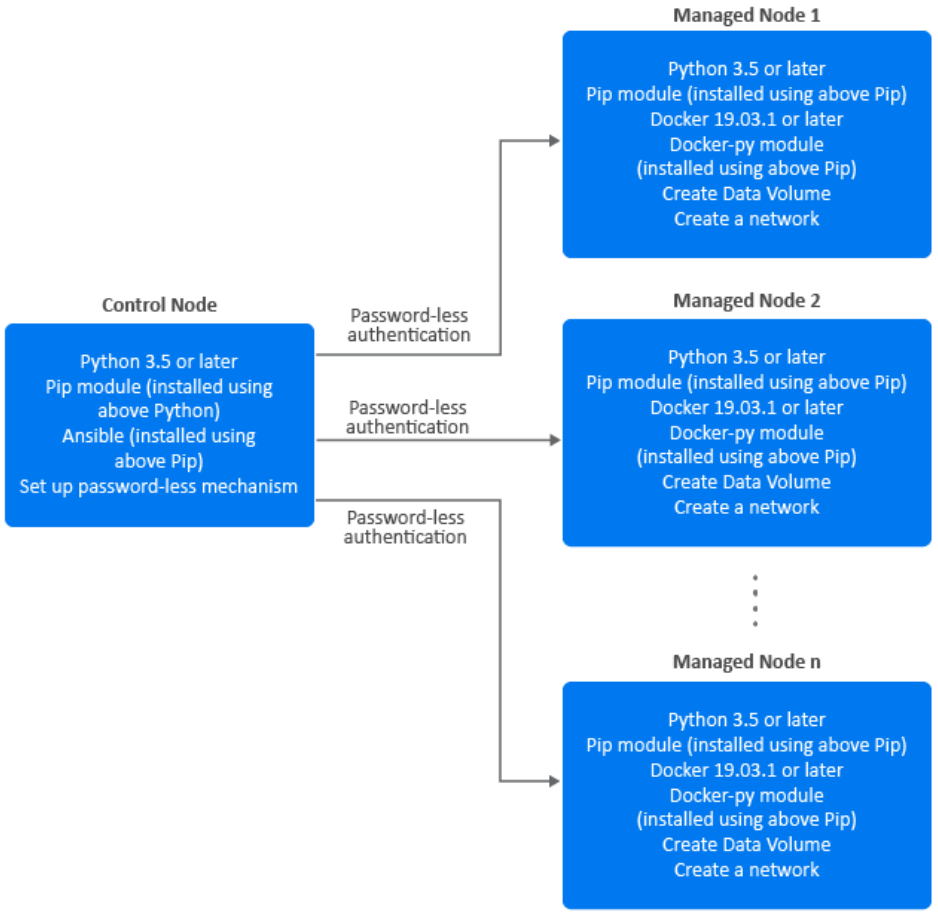
Remote Loader	8090
iManager	8743
iMonitor	8030
OSP	8543
Identity Applications	18543
Identity Reporting	28543
Form Renderer	8600
ActiveMQ	•8161 •61616
PostgreSQL	5432
SSPR	8443 -> SSPR container runs only on 8443 port.

Demo – Single server walk-through

Deploying IDM Containers Using Ansible

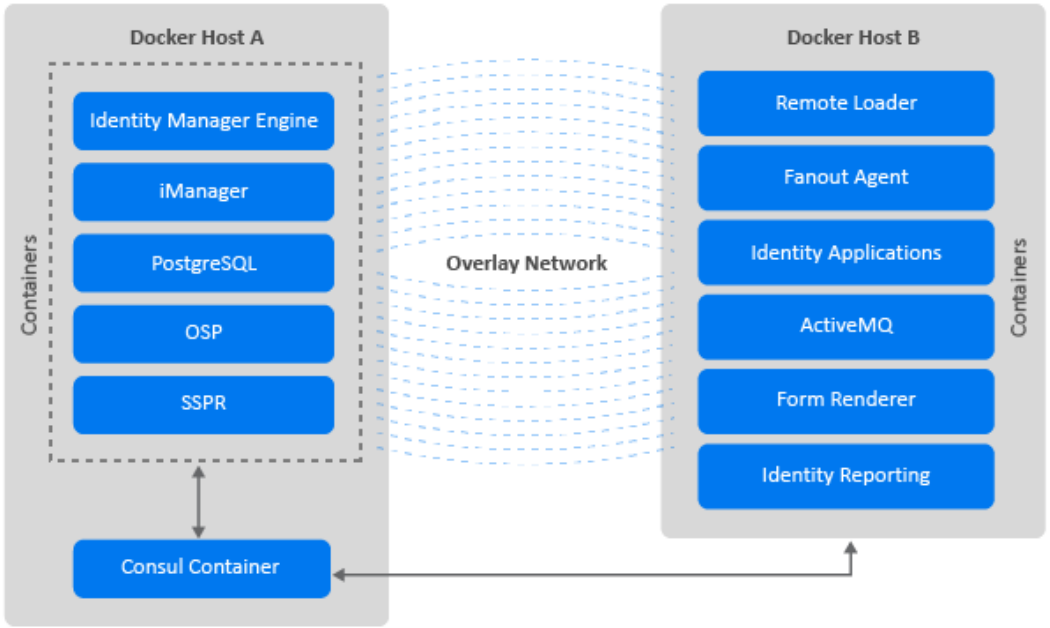
- Ansible is a system of configuration management written in Python programming language which uses a declarative markup language to describe configurations. It's used for automation of configuration and OS setup
- Officially supported since 4.8.3:
  - https://www.netiq.com/documentation/identity-manager-48/idm_installing_updating_483/data/deploying-identity-manager-containers-using-ansible.html
- This release only supports a fresh deployment of containers using Ansible.
- The container tar ball contains two folders, one with the ansible scripts and config files and another with the actual container images

Preparing your ansible nodes



- Identify two or more servers for Ansible-based container deployment
- One of the servers is called Ansible Control Node (control node) and the remaining servers are called Managed Nodes (managed nodes)
- The software requirements for each differ slightly.
- After preparing the nodes, configure a shared storage and an overlay network

Setting up an Overlay network



- Deploy a “Consul” container that creates a virtual network between the different hosts
- Command “docker info”:
  - Cluster Store: consul://192.168.184.226:8500
  - Cluster Advertise: 192.168.184.227:2375
- With 4.8.3 we ship a DNS container as well to facilitate the administration of IP addresses
- Otherwise, each container must have an appropriate /etc/hosts file

Creating the setup.csv file

- The setup.csv file is the input file used by Ansible. It's located in the ansible/input folder.
- Parameters:
  - **Component:** Indicates the container that you want to deploy. For example, `engine`.
  - **Deploy:** Indicates whether you want to deploy the container. The supported values are **yes** and **no**.
  - **DockerHost:** Indicates the Docker host where the container will be deployed. In other words, this can be any of the managed nodes you have identified for your deployment. For example, `DockerHostA`
  - **IP Address:** Indicates the IP Address of the Docker host where the container will be deployed. For example, `192.168.0.15`
  - **ContainerName:** Indicates the name of the container. For example, `engine-container`.
  - **ContainerHostname:** Indicates the host name of the Docker hosts or server where the container will be deployed. NetIQ recommends that you specify the hostname in the FQDN format. For example, `identityengine.example.com`.
  - **ExposedPorts:** Indicates the ports that you want to expose for the container to listen on. For example, 636.
  - **FileMounting:** Indicates the path for any custom files such as `ojdbc.jar`. For example, `/opt/novell/eDirectory/lib/dirxml/classes/ojdbc.jar`.
  - **SharedVolume:** Indicates the shared volume that you want the containers to use for data persistence. For example, `/data`.

Sample setup.csv file

Component	Deploy	DockerHost	IPAddress	ContainerName	ContainerHostname	ExposedPorts	FileMounting	SharedVolume
dns	yes	vm-ansible1	192.168.0.22	dns-container	idmdns.example.com			/data
engine	yes	vm-ansible1	192.168.0.2	engine-container	identityengine.example.com	636 389		/data
remoteloader	yes	vm-ansible1	192.168.0.3	remoteloader-container	remoteloader.example.com	8090		/data
fanoutagent	no	vm-ansible1	192.168.0.4	fanoutagent-container	fanoutagent.example.com			/data
iManager	yes	vm-ansible1	192.168.0.5	iman-container	imanager.example.com	8745		/data
osp	yes	vm-ansible2	192.168.0.6	osp-container	osp.example.com	8543		/data
postgres	yes	vm-ansible1	192.168.0.7	postgresql-container	postgresql.example.com	5432		/data
identityapps	yes	vm-ansible2	192.168.0.8	idapps-container	identityapps.example.com	18543		/data
formrenderer	yes	vm-ansible2	192.168.0.9	fr-container	formrenderer.example.com	8600		/data
activemq	yes	vm-ansible2	192.168.0.10	amq-container	activemq.example.com	8161 61616		/data
rpt	no	vm-ansible1	192.168.0.11	rpt-container	identityreporting.example.com	28543		/data
sspr	yes	vm-ansible2	192.168.0.12	sspr-container	sspr.example.com	8443		/data

Upgrading containers

Updating containers

- Updating a container is normally a process of stopping the running instance, loading the new one and replacing it with the new image
- Since each container's data is stored outside of the container, there is no data loss in this process.
- Some containers, like Postgres, require some extra steps when performing an upgrade of versions

- Challenge: How to preserve specific RPMs and java files installed on an IDM engine

Handling RPM Updates and Third-Party Files

- New feature introduced with 4.8.4
- Ability to preserve driver versions after upgrade of containers:
  - Add manually a folder called “mountfiles“
  - Copy files of type *.jar, *.so and rpms.
    - The rpms in that folder will be updated forcefully on startup of the container
    - The `.so` and `.jar` files are automatically soft linked to the `/opt/novell/eDirectory/lib64/nds-modules/` and `/opt/novell/eDirectory/lib/dirxml/classes/` directories respectively

Conclusion

Conclusion

- Keep in mind that a Consulting agreement is needed to deploy in production
- Start simple – single host solution is the easiest
- Leverage scripts to reduce complexity and human error

Q & A