RED HAT CONSULTING

redhat

Red Hat Ansible Tower MOD Tzayad Lab Engagement journal

Red Hat Automation for OCP 3.11 multiple clusters deployment

PREPARED FOR: MOD - Eyal David , Guy Aloni

TABLE OF CONTENTS

- TABLE OF CONTENTS
- PREFACE
    - CONFIDENTIALITY, COPYRIGHT AND DISCLAIMER
    - ABOUT THIS DOCUMENT
    - AUDIENCE
    - ADDITIONAL BACKGROUND AND RELATED DOCUMENTS
    - TERMINOLOGY
    - REVISIONS
- PREPARATION
    - Staffing
    - Engagement Approach
- ENGAGEMENT DETAILS
    - Prerequisite Verification
- SOLUTION ARCHITECTURE
    - Architecture
    - Hardware specifications
        - Hardware Detail
        - Network Details
        - Storage Details
        - Supporting Infrastructure
        - Supporting Infrastructure
- ENGAGEMENT CLOSURE AND RECOMMENDATIONS.
    - Knowledge Transfer
    - Testing
    - Engagement Observations
    - Training and Certification
        - Red Hat Certified System Administrator (RHCSA)
            - RHCSA Rapid Track course with exam (RH200)
        - Red Hat OpenStack Administration
- APPENDICES
    - Appendix A: RHOS Accelerator Test Matrix and Test result
    - Appendix B: Installing and configuring Red Hat OpenStack Platform 7.
    - Appendix C: Post-deployment configuration.

I. PREFACE

CONFIDENTIALITY, COPYRIGHT AND DISCLAIMER

This is a Customer-facing document between Red Hat, Inc. and MOD:Mamram (“Client”).
Copyright © 2015 Red Hat, Inc. All Rights Reserved. No part of the work covered by the copyright herein may be reproduced or used in any form or by any means – graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems – without permission in writing from Red Hat except as is required to share this information as provided with the aforementioned confidential parties.

This document is not a quote and does not include any binding commitments by Red Hat.

ABOUT THIS DOCUMENT

This is a confidential document between Red Hat, Inc. and MOD: Mamram detailing a Red Hat OCP 3.11 Deployment process Automation by usage of Red Hat Ansible Tower in Disconnected environment and Integration with Central Monitoring/Metrics system (Prometheus/Grafana)

AUDIENCE

The intended audience for this document is any person involved in the review, decision making, sign off or implementation of the proposed solution and Red Hat Consulting engagement to deliver.

ADDITIONAL BACKGROUND AND RELATED DOCUMENTS

This document also references additional information that can be found on Red Hat‘s documentation site at https://access.redhat.com/knowledge/docs/ and specifically at https://access.redhat.com/documentation/en-us/openshift_container_platform/3.11/. &

https://docs.ansible.com/ansible-tower/3.4.2/html/installandreference/index.html &

https://docs.ansible.com/

Documents specific to products covered in this solution include the following.

- Getting Started Guide
- Installation and Configuration Guide

The following Reference Architecture as in-depth use case studies that may be useful and are available at https://access.redhat.com/site/articles/reference-architecture :

- Deploying and Managing OpenShift 3.9 on VMware vSphere
- Automate Red Hat OpenShift Container Platform Deployment on HPE ProLiant Servers with Ansible

TERMINOLOGY

The table below provides a glossary of the terms and acronyms used within this document.

Term	Definition
RHEL	Red Hat Enterprise Linux.
vSphere	VMware Enterprise Virtualization Management appliance
ESXi	VMware Enterprise Virtualization Host OS
Ansible	Red Hat Automation tool - agentless system configuration tool used for different products deployment and OCP among them
Ansible Tower	Red Hat Ansible orchestration product
OCP	Red Hat OpenShift Container Platform
Disconnected environment	Environment without access to the internet - require preparation of Offline Repository and Offline Container Registry
Offline Repository	Web Server in disconnected environment for RPMs repository hosting
Offline Container Registry	Asset server with Docker distribution installed in disconnected environment to host container images for OpenStack deployment
Inventory	A collection of hosts against which Jobs may be launched.
Host	A system managed by Tower, which may include a physical, virtual, cloud-based server, or other device. Typically an operating system instance. Hosts are contained in Inventory. Sometimes referred to as a “node”.
Group	A set of hosts in Ansible that can be addressed as a set, of which many may exist within a single Inventory.
Job Template	The combination of an Ansible playbook and the set of parameters required to launch it.
Workflow Job Template	A set consisting of any combination of job templates, project syncs, and inventory syncs, linked together in order to execute them as a single unit.
Ad Hoc	Refers to running Ansible to perform some quick command, using /usr/bin/ansible, rather than the orchestration language, which is /usr/bin/ansible-playbook. An example of an ad hoc command might be rebooting 50 machines in your infrastructure. Anything you can do ad hoc can be accomplished by writing a Playbook, and Playbooks can also glue lots of other operations together.
Plays	A playbook is a list of plays. A play is minimally a mapping between a set of hosts selected by a host specifier (usually chosen by groups, but sometimes by hostname globs) and the tasks which run on those hosts to define the role that those systems will perform. There can be one or many plays in a playbook.
Playbook	An Ansible playbook. Refer to http://docs.ansible.com/ for more information.
YAML	Ansible and Tower use YAML to define playbook configuration languages and also variable files. YAML has a minimum of syntax, is very clean, and is easy for people to skim. It is a good data format for configuration files and humans, but is also machine readable. YAML is fairly popular in the dynamic language community and the format has libraries available for serialization in many languages (Python, Perl, Ruby, etc.).
Project	A logical collection of Ansible playbooks, represented in Tower.
Sudo	Ansible does not require root logins and, since it is daemonless, does not require root level daemons (which can be a security concern in sensitive environments). Ansible can log in and perform many operations wrapped in a sudo command, and can work with both password-less and password-based

	sudo. Some operations that do not normally work with sudo (like scp file transfer) can be achieved with Ansible's copy, template, and fetch modules while running in sudo mode.

Table I-a: Terminology

REVISIONS

Name	Version	Date	Comments
Anoel Yakoubov, Matan Carmeli	1.0	24.04.2019	Initial release to customer.

Table I-b: Revisions

1. PREPARATION

This section details prerequisites and preparation efforts that Client was to complete prior to the engagement. Where applicable, the onsite consultant has validated the provided data and made for any discrepancies or additional detail.

POC Requirements defined by MOD Tzayad

1.1 Staffing

The following persons were identified by the Client to support the engagement:

Role	Purpose	Client Assignment	Contact Information
Architect	Architecture & DOD	Eyal David	eyaldavid86@gmail.com
Consultant	Project Delivery	Matan Carmeli	matan.carmeli7@gmail.com

Table 1-1: Client staff/contact information.

Role	Purpose	Red Hat Assignment	Contact Information
Project Manager	Project Management	Etay Nir	etnir@redhat.com
Sr. Cloud Architect	Architecture & Supervision	Orgad Kimchi	okimchi@redhat.com
Consultant	Project Delivery	Anoel Yakoubov	anoel@redhat.com
Consultant	Project Delivery	Guy Rakover	grakover@redhat.com

Table 1-2: Red Hat staff/contact information.

1.2 Engagement Approach

This section details Red Hat's strategy to assist MOD: Tzayad with meeting their goals for this engagement.The following table outlines the focus of each phase.

Purpose	Details
VMware vSphere Virtualization environment deployment	● Install and Configure ESXi 6.7 as a basis to hosted engine vSphere infrastructure on two physical server ● Configure RAID 10 and VMFS as storage backend for vSphere ● Install and Configure hosted engine VM appliance on top of ESXi by deploying Windows 2016 VM and from this VM vSphere appliance ● Configure vSphere (Data Center, Cluster, Storage and Network Infrastructure) ● Install and configure all relevant VMs with RHEL 7.6 for common usage
Offline Repository and Offline Container Registry deployment	● Offline Repository Installation and configuration on one of VMs to enable Disconnected OCP 3.11.98 deployment ● Offline Container Registry Installation and configuration on one of VMs to enable Disconnected OCP 3.11.98 deployment ● Configure DNS on this server for OCP environments
Ansible Tower Deployment	● Deployment and configuration of Red Hat Ansible Tower on one of VMs
Central Grafana Server Deployment and Configuration	● Installing Grafana on one of VMs. ● Configuring Grafana with Data Source - Prometheus
Ansible Tower Configurations	● Projects, Inventories, Groups, Hosts definitions ● Connectivity to Github where all relevant Ansible playbooks and roles resides. ● Creation of Job templates and Workflow templates
Ansible Tower Templates	● Deploy VMware - Job template to create all relevant VMs for OCP deployment ● Destroy VMware - Job template to destroy all relevant VMs if condition will be raised ● Pre Install - Job template to install different utilities on each VM according preinstall.yml playbook ● OCP prerequisites - Job template to run all OCP prerequisites by using openshift-ansible prerequisites.yml playbook ● OCP Deploy Cluster - Job template to run all OCP deployment by using openshift-ansible deploy_cluster.yml playbook ● Post Jobs - Job template to run all OCP post-deployment configurations, such Grafana deployment etc.. ● OCP Deploy Service Catalog - Job template to deploy Service Catalog ● VMware and OCP full deployment - Workflow template that orchestrates all Job templates in certain order and conditioning

Table 1-3: Focus per phase.

2. ENGAGEMENT DETAILS

The engagement focused on the installation and configuration of one (1) environment running VMware Virtualization Platform, on top of which completed Deployment of Ansible Tower Platform and Automation of at least two OCP 3.11 environments Deployment, each with its LB in Disconnected environment. In addition, VMDK used as persistent storage backend for each OCP cluster and each cluster was with its own Metrics system (Prometheus/Grafana) and propagation of metrics to the Central Metrics environment.
Optional: GlusterFS usage as persistent storage backend for at least one OCP cluster.

Prerequisite Verification and validation

Prior to commencing the configuration of the lab environments, the following pre-requisites items were validated by our team.

Criteria	Validated
Systems are in the Red Hat Hardware Compatibility List for RHEL 7.6.	Validated
RHEL 7.6 Server for x86_64 entitlements are available for all systems	Validated
VMware entitlements are available for all systems.	Validated
OCP 3.11 entitlements are available for all systems.	Validated
Compute nodes have enough CPU cores for the expected workload with the required overcommit ratio	Validated
Physical nodes for ESXi have CPU with VT-x or AMD-V	Validated
Available RAM corresponds to the expected workload and overcommit ratio	Validated
Nodes have at least two 1Gb network interfaces	Validated
Network topology satisfies requirements for the selected implementation of Ansible Tower and OCP Networking Services	Validated
Physical nodes for ESXi have at least 1TB local disk space	Validated
RAID controller memory is flash-backed or battery-backed	Validated

Table 2-1: VMware virtualization nodes hardware verification.

3. SOLUTION ARCHITECTURE

The engagement focused on the installation and configuration of one (1) environment running VMware Virtualization Platform, on top of it Ansible Tower Platform and Automation of at least two OCP environments Deployment, each with its LB in Disconnected environment. In addition VMDK used as persistent storage backend for each OCP cluster and each cluster was with its own Metrics system (Prometheus/Grafana) and propagation of metrics to the Central Metrics environment.
Below the VMware vSphere details, Ansible Tower Details, Offline Registry and Repository Details

- VMware vSphere 6.7 environment
  - One Cluster
    - Andromeda01 & Andromeda02 physical Servers with ESXi 6.7 version
    - Andromeda02 - 4 Disks with 1.2 TB each one configured in RAID10 with total Storage 2.4TB defined as Datastore1 in vSphere
    - Andromeda01 - 2 Disks with 1.2 TB each one configured in RAID1 with total Storage 1.2TB defined as Datastore2 in vSphere, 2 Disks with 278 GB each one configured in RAID0 with total Storage 550 GB defined as Datastore3 in vSphere
    - HostedEngine VM appliance installed on top of ESXi 6.7 with DNS name of vSphere Manager: https://vm-76-242.sales.lab.tlv.redhat.com
    - 3 additional VMs:
      1 ocprepo (Offline Repository + Offline Container Registry + DNS)
      1 grafana (Central Grafana Server)
      1 anstower (Ansible Tower server)

- vSphere DataCenter details:
  - Datacenter name: Datasenter 1
  - Cluster name: Cluster 1
  - VM Folder names:
    - Common
    - OCP1
    - OCP2

- Red Hat Ansible Tower environment:
  - One node: VM name: anstower
    - Ansible Tower 3.4.3
    - Ansible 2.7.9

- Offline Repository + Offline Registry + DNS
  - Enabled Relevant Repos for OCP 3.11.98
  - Docker Distribution installed and all Images loaded, tagged and pushed for OCP 3.11.98
  - DNS installed and configured for ocp1.zayadtest.com and ocp2.zayadtest.com

- Central Grafana:
  - 1 VM - Grafana - with Grafana installed on it, version 6.1.4 and Prometheus as Data Source

3.1 Architecture

Here is a logical diagram of different parts of the implemented Ansible Tower Architecture.

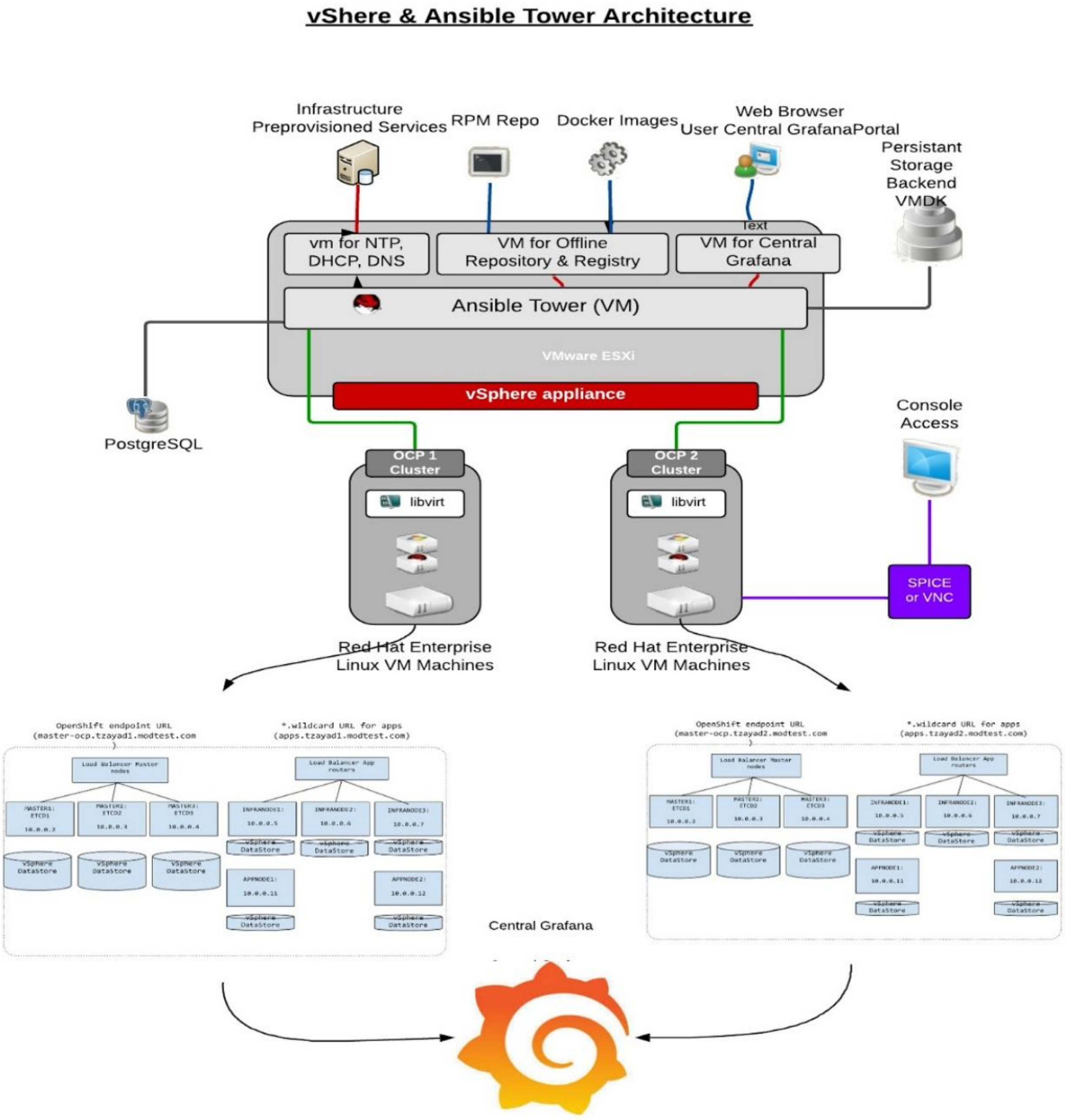
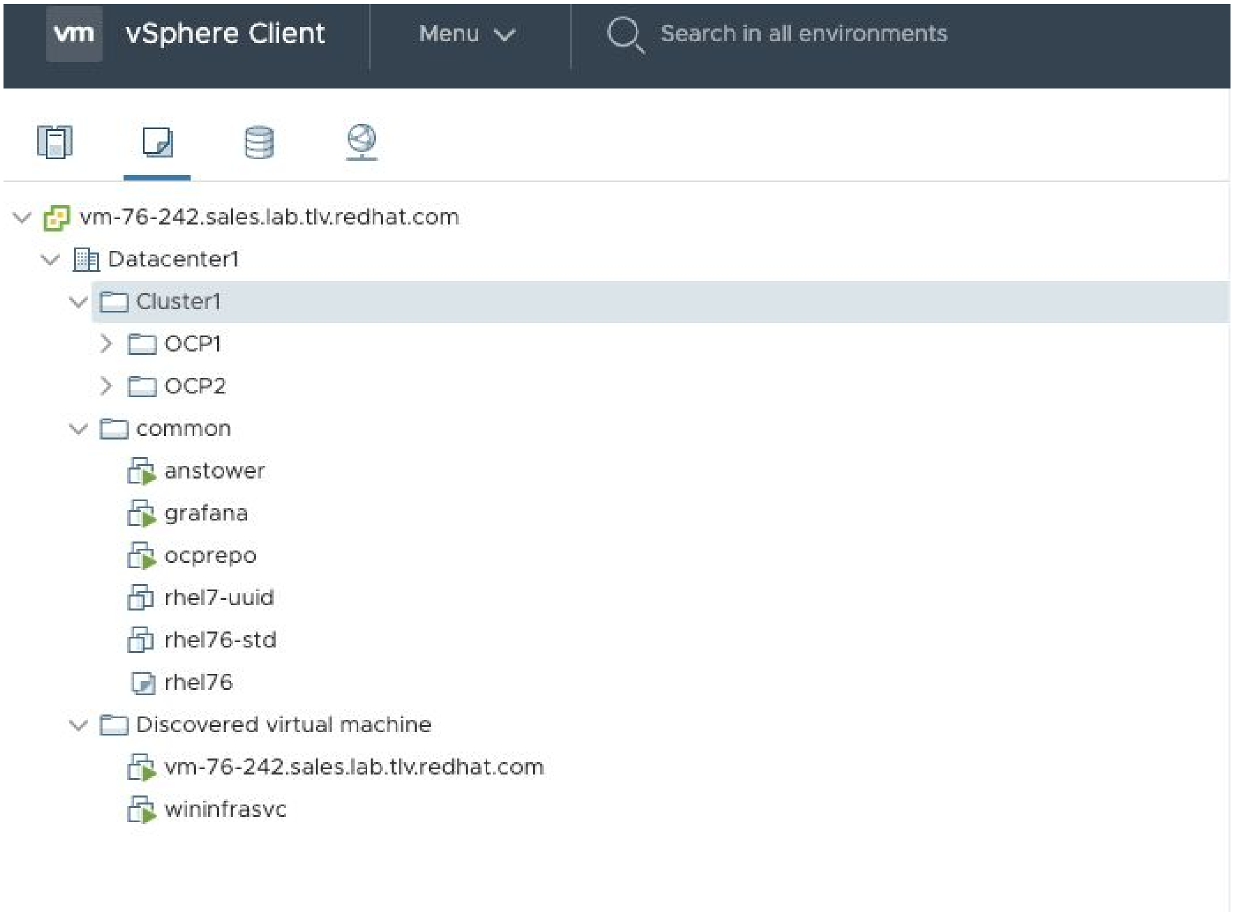
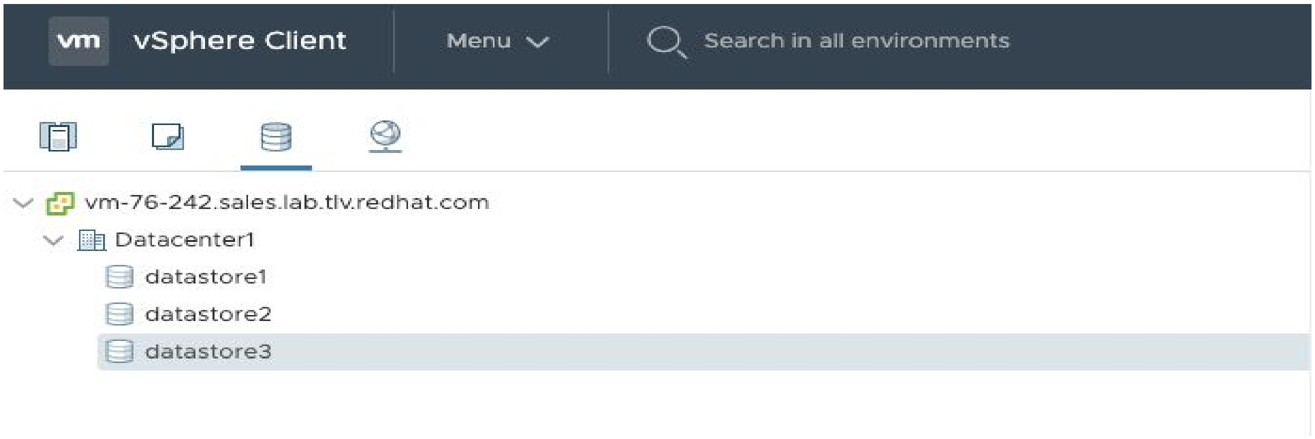

vShere & Ansible Tower Architecture
Infrastructure Preprovisioned Services
RPM Repo
Docker Images
Web Browser User Central GrafanaPortal
Persistant Storage Backend VMDK
Text
vm for NTP, DHCP, DNS
VM for Offline Repository & Registry
VM for Central Grafana
Ansible Tower (VM)
VMware ESXi
vSphere appliance
PostgreSQL
Console Access
OCP 1 Cluster
libvirt
OCP 2 Cluster
libvirt
SPICE or VNC
Red Hat Enterprise Linux VM Machines
Red Hat Enterprise Linux VM Machines
OpenShift endpoint URL (master-ocp.tzayad1.modtest.com)
*.wildcard URL for apps (apps.tzayad1.modtest.com)
Load Balancer Master nodes
Load Balancer App routers
MASTER1: ETCD1 10.0.0.2
MASTER2: ETCD2 10.0.0.3
MASTER3: ETCD3 10.0.0.4
INFRANODE1: 10.0.0.5
INFRANODE2: 10.0.0.6
INFRANODE3: 10.0.0.7
vSphere DataStore
APPNODE1: 10.0.0.11
APPNODE2: 10.0.0.12
OpenShift endpoint URL (master-ocp.tzayad2.modtest.com)
*.wildcard URL for apps (apps.tzayad2.modtest.com)
Central Grafana

Image 3-2: https://vm-76-242.sales.lab.tlv.redhat.com- list of VMs.



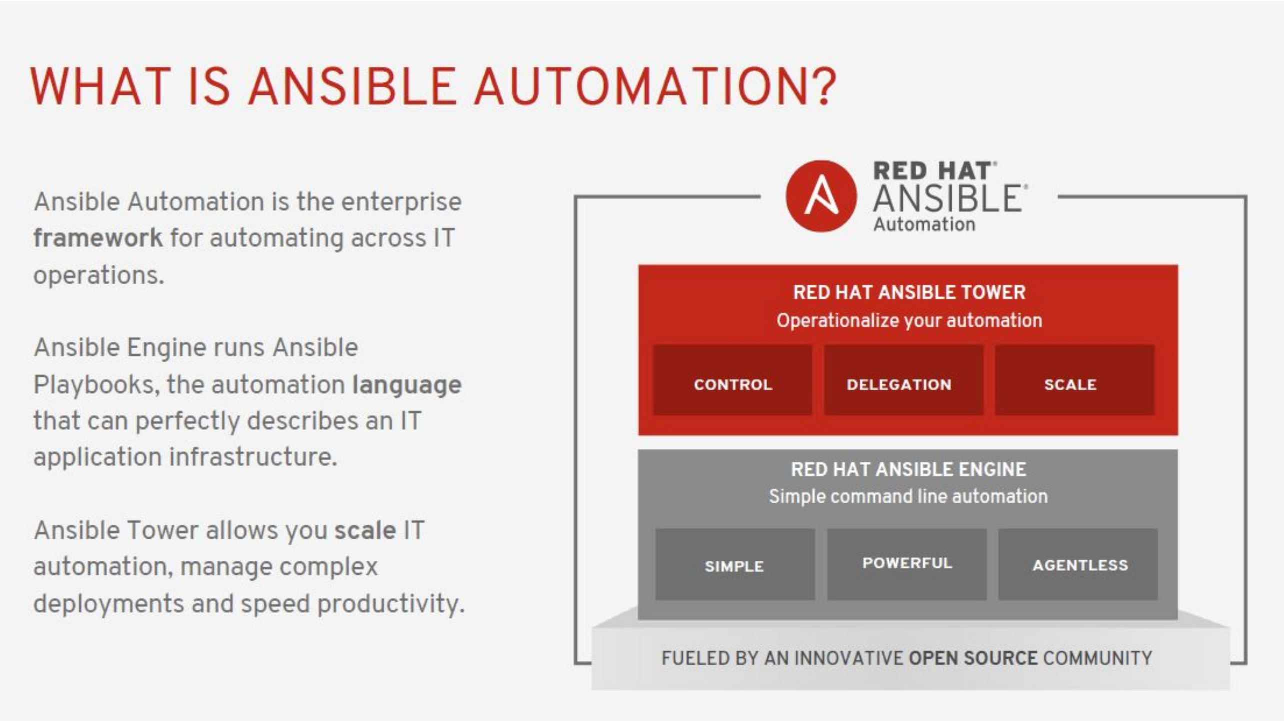
- vm-76-242.sales.lab.tlv.redhat.com
  - Datacenter1
    - Cluster1
      - OCP1
      - OCP2
    - common
      - anstower
      - grafana
      - ocprepo
      - rhel7-uuid
      - rhel76-std
      - rhel76
    - Discovered virtual machine
      - vm-76-242.sales.lab.tlv.redhat.com
      - wininfrasvc

Image 3-3: Logical Diagram of Datastores



- vm-76-242.sales.lab.tlv.redhat.com
  - Datacenter1
    - datastore1
    - datastore2
    - datastore3

Ansible and Ansible Tower - Technical Overview



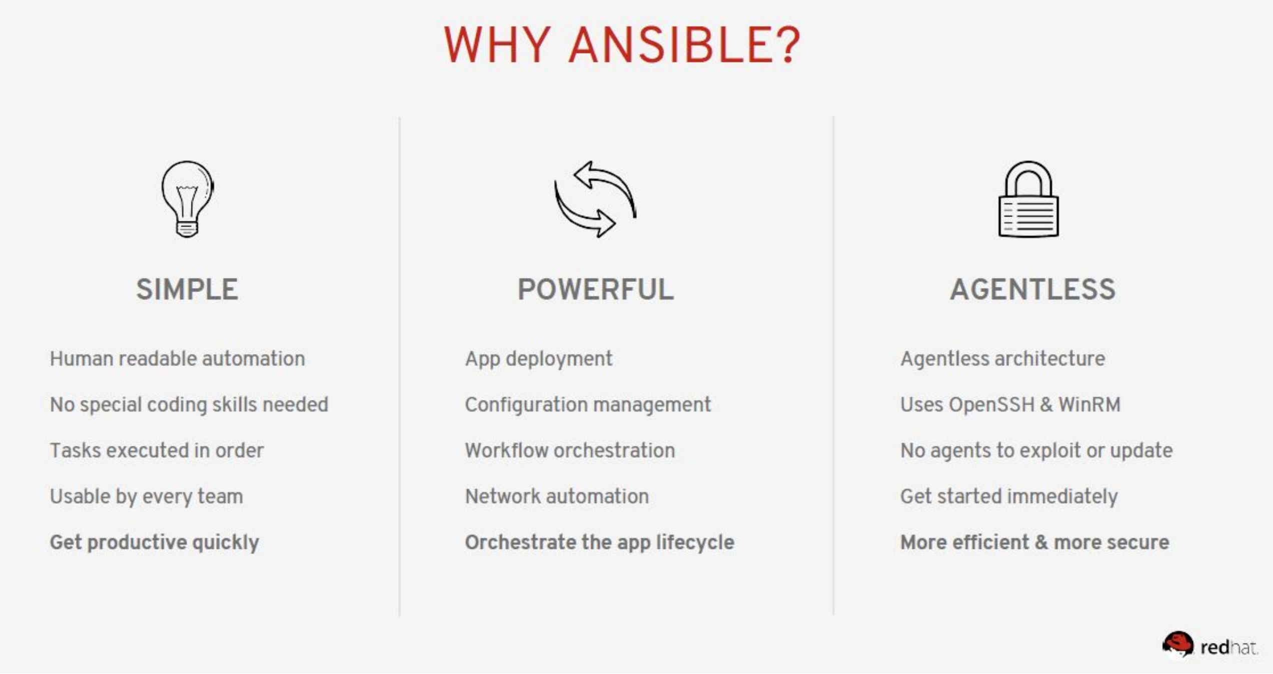
WHAT IS ANSIBLE AUTOMATION?

Ansible Automation is the enterprise **framework** for automating across IT operations.

Ansible Engine runs Ansible Playbooks, the automation **language** that can perfectly describes an IT application infrastructure.

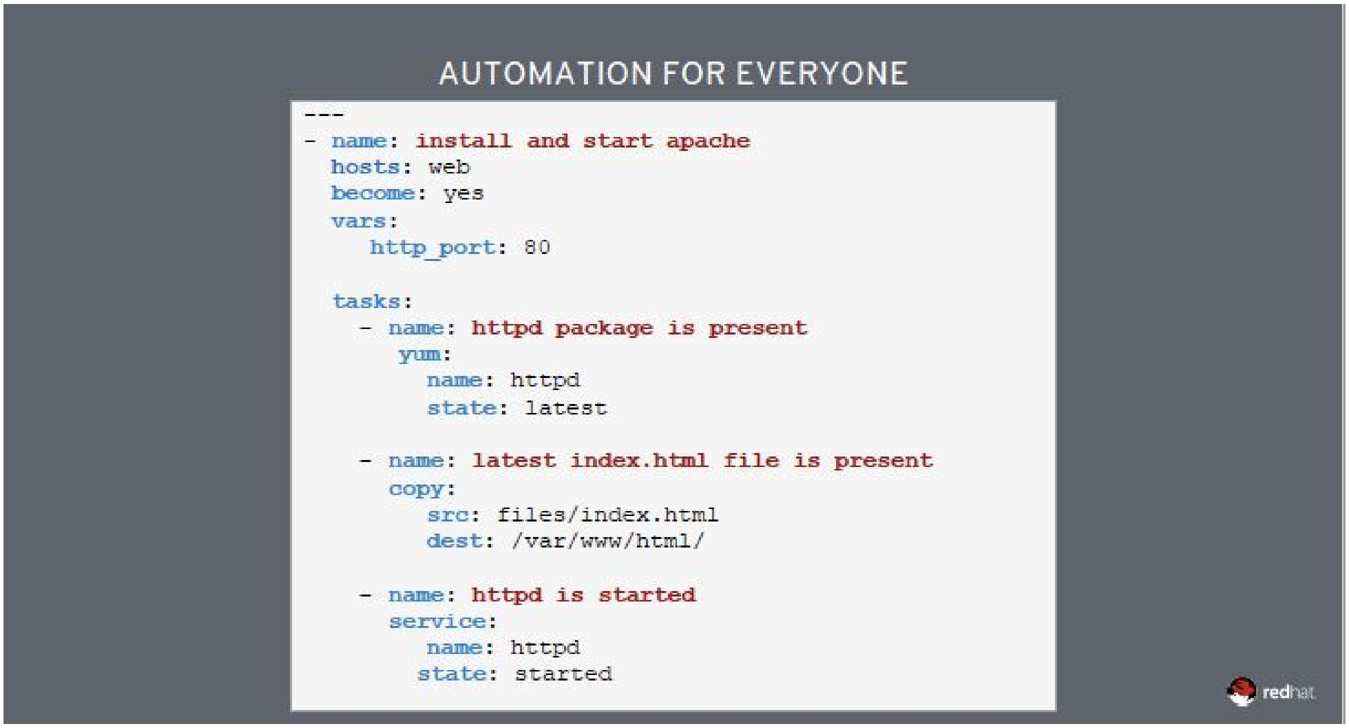
Ansible Tower allows you **scale** IT automation, manage complex deployments and speed productivity.

RED HAT ANSIBLE Automation
RED HAT ANSIBLE TOWER
Operationalize your automation
CONTROL
DELEGATION
SCALE
RED HAT ANSIBLE ENGINE
Simple command line automation
SIMPLE
POWERFUL
AGENTLESS
FUELED BY AN INNOVATIVE OPEN SOURCE COMMUNITY



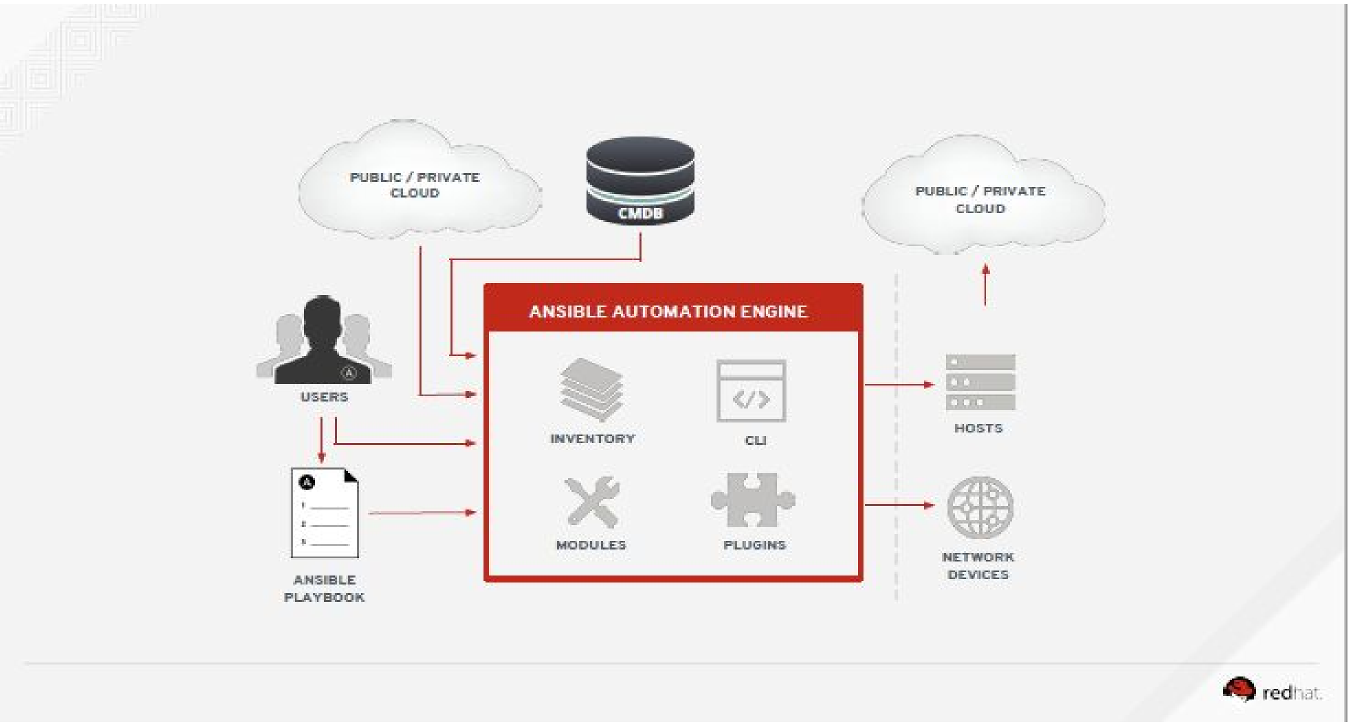
WHY ANSIBLE?

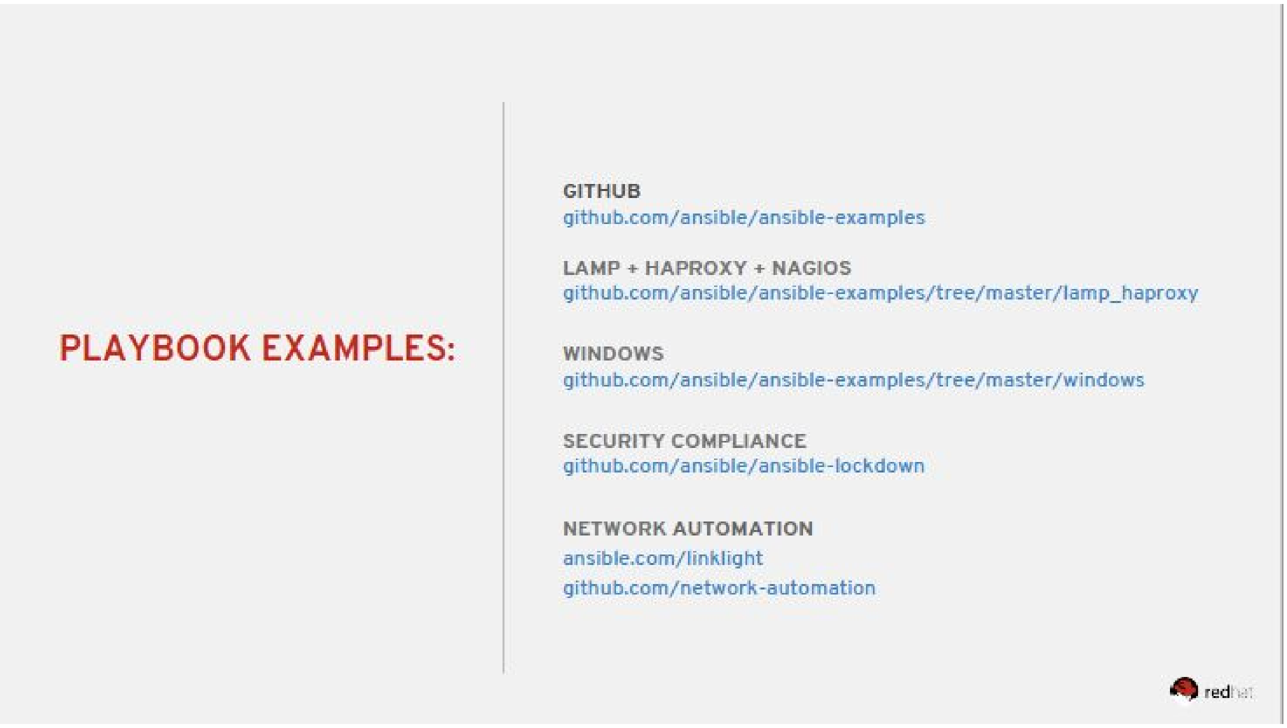
SIMPLE	POWERFUL	AGENTLESS
Human readable automation	App deployment	Agentless architecture
No special coding skills needed	Configuration management	Uses OpenSSH & WinRM
Tasks executed in order	Workflow orchestration	No agents to exploit or update
Usable by every team	Network automation	Get started immediately
Get productive quickly	**Orchestrate the app lifecycle**	**More efficient & more secure**



AUTOMATION FOR EVERYONE

```
---
- name: install and start apache
  hosts: web
  become: yes
  vars:
     http_port: 80

  tasks:
    - name: httpd package is present
      yum:
        name: httpd
        state: latest

    - name: latest index.html file is present
      copy:
        src: files/index.html
        dest: /var/www/html/

    - name: httpd is started
      service:
        name: httpd
       state: started
```





PLAYBOOK EXAMPLES:

GITHUB
github.com/ansible/ansible-examples

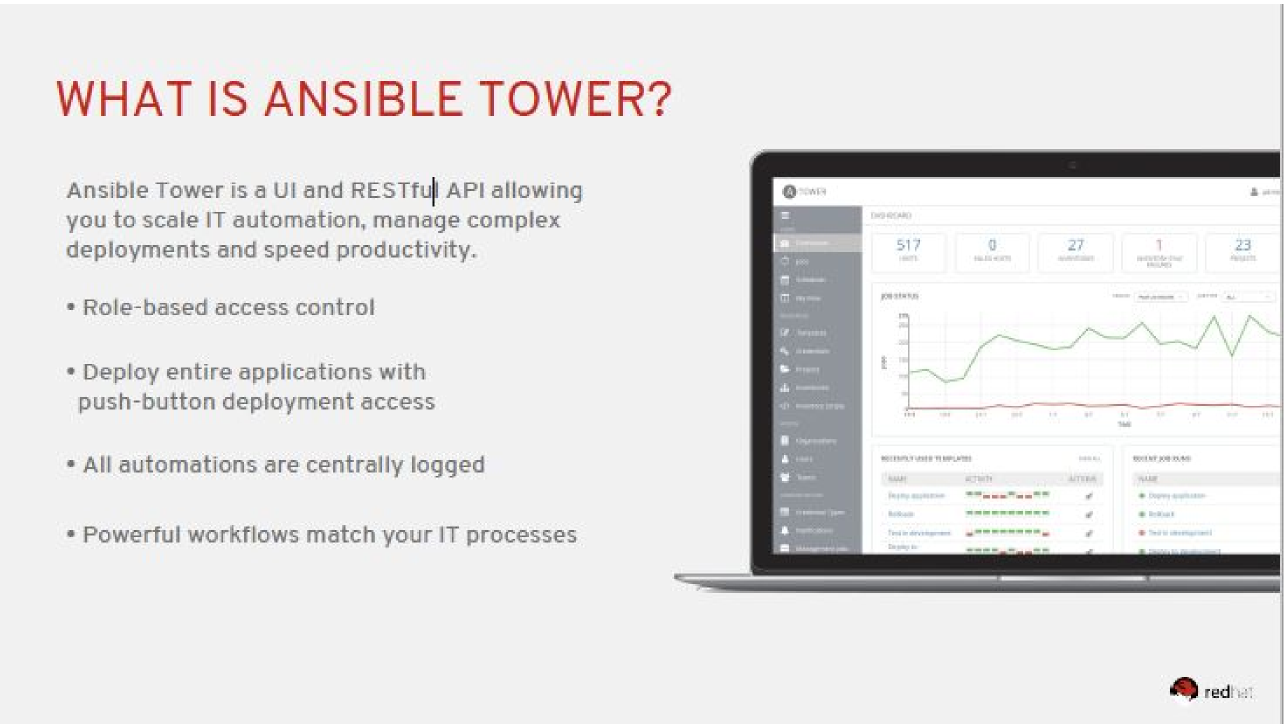
LAMP + HAPROXY + NAGIOS
github.com/ansible/ansible-examples/tree/master/lamp_haproxy

WINDOWS
github.com/ansible/ansible-examples/tree/master/windows

SECURITY COMPLIANCE
github.com/ansible/ansible-lockdown

NETWORK AUTOMATION
ansible.com/linklight
github.com/network-automation

redhat

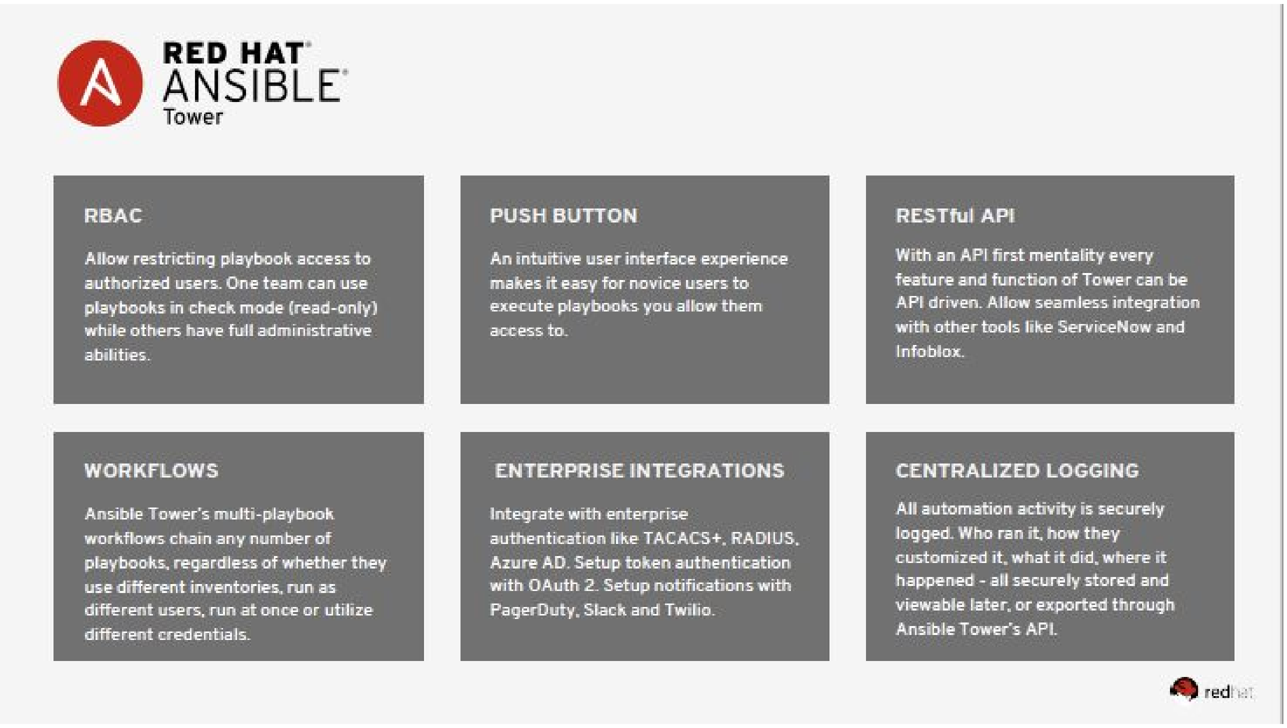


WHAT IS ANSIBLE TOWER?

Ansible Tower is a UI and RESTful API allowing you to scale IT automation, manage complex deployments and speed productivity.

- Role-based access control
- Deploy entire applications with push-button deployment access
- All automations are centrally logged
- Powerful workflows match your IT processes

redhat



RED HAT ANSIBLE Tower

RBAC

Allow restricting playbook access to authorized users. One team can use playbooks in check mode (read-only) while others have full administrative abilities.

PUSH BUTTON

An intuitive user interface experience makes it easy for novice users to execute playbooks you allow them access to.

RESTful API

With an API first mentality every feature and function of Tower can be API driven. Allow seamless integration with other tools like ServiceNow and Infoblox.

WORKFLOWS

Ansible Tower's multi-playbook workflows chain any number of playbooks, regardless of whether they use different inventories, run as different users, run at once or utilize different credentials.

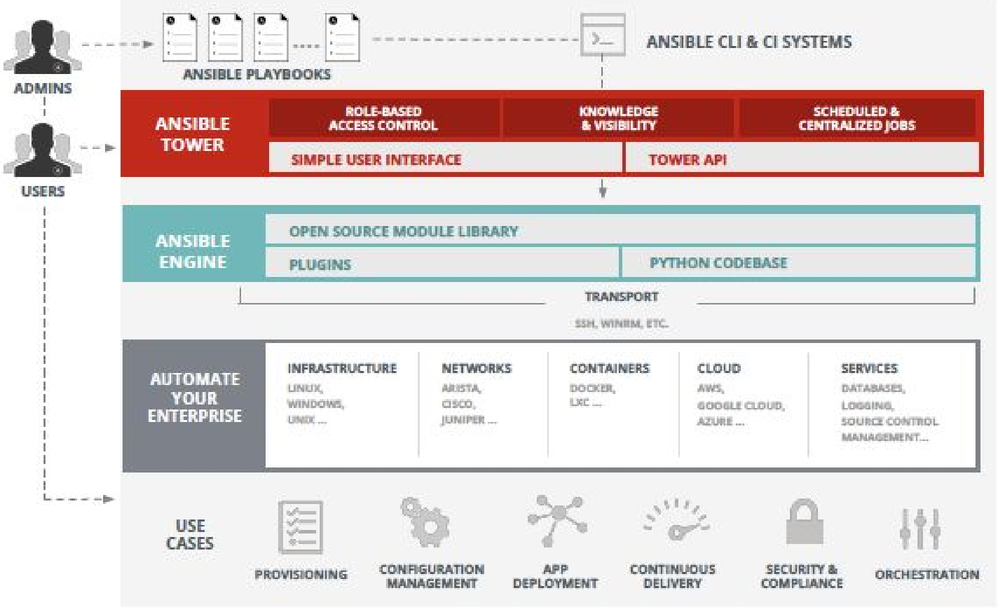
ENTERPRISE INTEGRATIONS

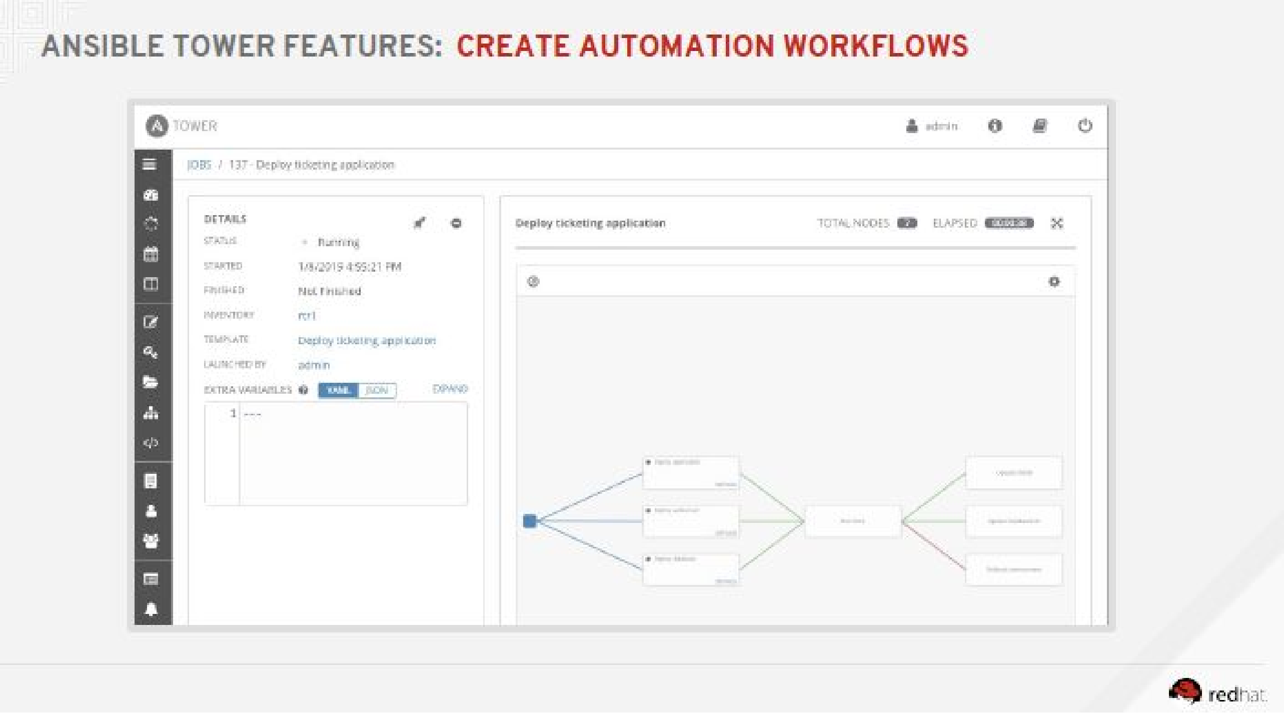
Integrate with enterprise authentication like TACACS+, RADIUS, Azure AD. Setup token authentication with OAuth 2. Setup notifications with PagerDuty, Slack and Twilio.

CENTRALIZED LOGGING

All automation activity is securely logged. Who ran it, how they customized it, what it did, where it happened - all securely stored and viewable later, or exported through Ansible Tower's API.

redhat





ANSIBLE TOWER FEATURES: CREATE AUTOMATION WORKFLOWS

JOBS / 137 - Deploy ticketing application

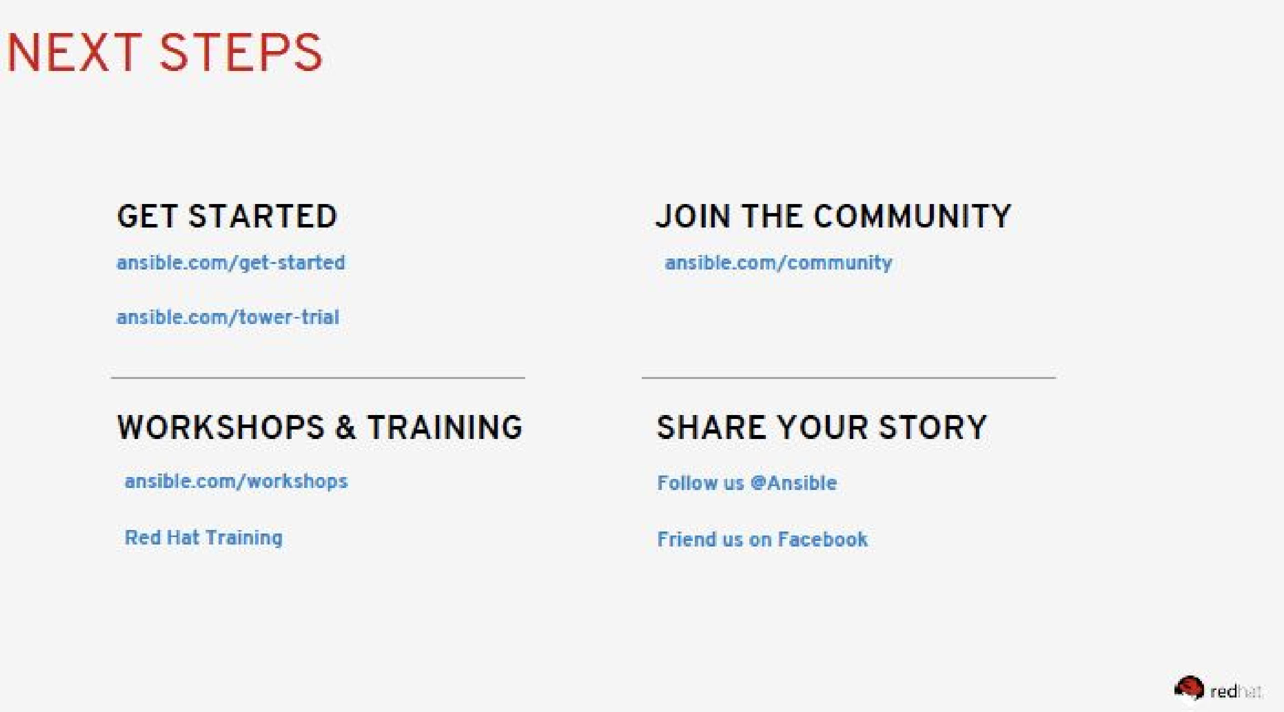
DETAILS

STATUS	Running
STARTED	1/8/2019 4:55:21 PM
FINISHED	Not Finished
INVENTORY	rcrl
TEMPLATE	Deploy ticketing application
LAUNCHED BY	admin

EXTRA VARIABLES YAML JSON EXPAND

```
1 ---
```

Deploy ticketing application TOTAL NODES ELAPSED



NEXT STEPS

GET STARTED

ansible.com/get-started

ansible.com/tower-trial

JOIN THE COMMUNITY

ansible.com/community

WORKSHOPS & TRAINING

ansible.com/workshops

Red Hat Training

SHARE YOUR STORY

Follow us @Ansible

Friend us on Facebook

3.2 Hardware specifications

A. Hardware Detail

Can be found in the Ansible Tower Tzayad Raanana worksheet April 2019

Ansible Tower Tzayad Raanana worksheet April 2019

Table 3.2-1: Hardware details.

All the nodes in the cluster are Dell PowerEdge R430 with the next components:

Dell PowerEdge R430 * 1 (for vmware vsphere 6.7)

- 2x Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz(Base) 8-core (16-thread) CPUs.
- 128GB RAM.
- 1x Integrated 1Gb quad port adapter.
- 1x Integrated RAID Controller
- RAID 1 1.2 TB *2 volume for esxi and the VMs that on top of him (Andromeda01)
- RAID 0 278 GB *2 volume for DataStore3 for pv (Andromeda01)

Dell PowerEdge R430 * 1 (for vmware vsphere 6.7)

- 2x Intel(R) Xeon(R) CPU E5-2609 v4 @ 1.70GHz(Base) 8-core (16-thread) CPUs.
- 128GB RAM.
- 1x Integrated 1Gb quad port adapter.
- 1x Integrated RAID Controller
- RAID 10 1.2 TB *4 volume for esxi and the VMs that on top of him(Andromeda2) and for pv

B. Network Details

Network	Network	Netmask	Gateway	VLAN
Vm network	10.35.76.x	255.255.255.128	10.35.76.254	461

Table 3-2: Network details.

C. Storage Details

Boot devices for physical hardware are provided by locally attached hardware RAID storage.

Usage	Size	Type	Hostname	Additional
DataStore1	2.4 TB	scsi	Andromeda2 (not shared)	For VMs and pv
DataStore2	1.2 TB	scsi	Andromeda1 (not shared)	For VMs
DataStore3	577 GB	scsi	Andromeda1 (not shared)	For pv

Table 3-3: Storage details.

D. Supporting Infrastructure

- NTP Servers: 10.46.0.31
- DNS Servers:
  - 10.46.0.31
  - Search domain
    - sales.lab.tlv.redhat.com , zayadtest.com

E. Supporting Infrastructure

Here is the list of Software repositories that used on the install.

vmware vsphere 6.7 Infrastructure

Esxi 6.7 ISO, VMware-VCSA-all-6.7.0-11726888.iso, RHEL 7.6 ISO and Windows server 2016 ISO

Openshift 3.11 repositories

Channel	Repository Name
Red Hat Enterprise Linux 7 Server (RPMS)	rhel-7-server-rpms
Red Hat Enterprise Linux 7 Server - Extra (RPMs)	rhel-7-server-extras-rpms
Red Hat Enterprise ose 3.11 (RPMs)	rhel-7-server-ose-3.11-rpms
Red Hat Ansible Engine 2.6 RPMs	rhel-7-server-ansible-2.6-rpms

Table 3-4: Required repositories.

4. ENGAGEMENT PHASES IN DETAILS

4.1 VMware vsphere 6.7 deployment

In this engagement the decision was to deploy VMware vsphere 6.7 as virtualization platform. This chapter outlines how to deploy VMware vsphere 6.7 across two servers. Deployment performed on servers Andromeda01,Andoremeda02 - Management IP address 10.35.76.129, 10.35.76.130 according the relevant installation guide that can be found in the next link:

Deploying_VMware_esxi_6.7

Initially we deployed vcenter server appliance 6.7 on a windows server 2016 vm that we created on the esxi that we already installed. The relevant installation guide that can be found in the next link:

Deploy vcenter server appliance 6.7
The names of virtual machines and their roles can be found in the next link

Ansible Tower Tzayad Raanana worksheet April 2019

4.2 Offline Repository deployment

The following procedure is for preparing your own offline deployer. The procedure is based on the following document:
Enabling_the_ose_3.11_Repositories_standalone_install

Setup when using online server (We used GCP instance already provisioned for this purpose)

Goal:
Obtaining required software packages and images

Before you install OpenShift Container Platform in your disconnected environment, obtain the required images and components and store them in your repository.

> You must obtain the required images and software components on a system with the same architecture as the cluster that is in your disconnected environment
>
> For this purpose Virtual Instance was created on Red Hat PS GCP environment , (instance name *ocp-offline-repo-registry)*, you can ask permission to use it if you need (send request to Anoel with your id_rsa.public key)

Obtaining OpenShift Container Platform packages

On the RHEL 7 server with an internet connection, sync the repositories:

1. To ensure that the packages are not deleted after you sync the repository, import the GPG key:

```
$ rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. If `subscription-manager` component not installed please install it:

```
$ sudo yum install subscription-manager
```

3. Register the server with the Red Hat Customer Portal. You must use the credentials that are associated with the account that has access to the OpenShift Container Platform subscriptions:

```
$ sudo subscription-manager register
```

4. Pull the latest subscription data from RHSM:

   ```
   $ sudo subscription-manager refresh
   ```

5. Attach a pool ID for a subscription that provides OpenShift Container Platform:

   ```
   $ sudo subscription-manager attach --pool=<pool_id>
   $ sudo subscription-manager repos --disable="*"
   ```

6. Enable only the repositories required by OpenShift Container Platform 3.11:

   ```
   $ sudo subscription-manager repos \
             --enable="rhel-7-server-rpms" \
             --enable="rhel-7-server-extras-rpms" \
            --enable="rhel-7-server-ose-3.11-rpms" \
            --enable="rhel-7-server-ansible-2.6-rpms"
   ```

7. Install required packages:

   ```
   $ sudo yum -y install yum-utils createrepo docker mlocate screen git
   $ sudo updatedb
   ```

   The `yum-utils` package provides the **`reposync`** utility, which lets you mirror yum repositories, and you can use the `createrepo` package to create a usable `yum` repository from a directory.

8. Make a directory to store the software in the server’s storage or to a USB drive or other external device:

   ```
   $ sudo mkdir -p </path/to/repos>
   ```

   If you can re-connect this server to the disconnected LAN and use it as the repository server, store the files locally. If you cannot, use USB-connected storage so you can transport the software to a repository server in your disconnected LAN

9. Sync the packages and create the repository for each of them

   ○ For on-premise installations on x86_64 servers, run the following command

     ```
     $ sudo for repo in \
     rhel-7-server-rpms \
     rhel-7-server-extras-rpms \
     rhel-7-server-ansible-2.6-rpms \
     rhel-7-server-ose-3.11-rpms
     ```

```
do
  reposync --gpgcheck -lm --repoid=${repo}
--download_path=</path/to/repos>
  createrepo -v </path/to/repos/>${repo} -o </path/to/repos/>${repo}
Done
```

Obtaining images.

Pull the required container images:

1. Start the Docker daemon:
   ```
   $ sudo systemctl start docker
   ```

2. Pull all of the required OpenShift Container Platform infrastructure component images. Replace <tag> with the version to install. For example, specify v3.11.98 for the latest version. You can specify a different minor version.
   Pull all of the required OpenShift Container Platform component images for the optional components. Replace <tag>with the version to install. For example, specify v3.11.98 for the latest version. You can specify a different minor version.
   Pull the Red Hat-certified Source-to-Image (S2I) builder images that you intend to use in your OpenShift Container Platform environment.

- We created script that performs pull of all relevant docker images -

  pull_infrastructure_component_images.sh

```
#! /bin/bash
for image in \
apb-base:v3.11.98 \
apb-tools:v3.11.98 \
automation-broker-apb:v3.11.98 \
csi-attacher:v3.11.98 \
csi-driver-registrar:v3.11.98 \
csi-livenessprobe:v3.11.98 \
csi-provisioner:v3.11.98 \
grafana:v3.11.98 \
image-inspector:v3.11.98 \
local-storage-provisioner:v3.11.98 \
manila-provisioner:v3.11.98 \
mariadb-apb:v3.11.98 \
mediawiki:v3.11.98 \
mediawiki-apb:v3.11.98 \
```

```
mysql-apb:v3.11.98 \
ose-ansible:v3.11.98 \
ose-ansible-service-broker:v3.11.98 \
ose-cli:v3.11.98 \
ose-cluster-autoscaler:v3.11.98 \
ose-cluster-capacity:v3.11.98 \
ose-cluster-monitoring-operator:v3.11.98 \
ose-console:v3.11.98 \
ose-configmap-reloader:v3.11.98 \
ose-control-plane:v3.11.98 \
ose-deployer:v3.11.98 \
ose-descheduler:v3.11.98 \
ose-docker-builder:v3.11.98 \
ose-docker-registry:v3.11.98 \
ose-efs-provisioner:v3.11.98 \
ose-egress-dns-proxy:v3.11.98 \
ose-egress-http-proxy:v3.11.98 \
ose-egress-router:v3.11.98 \
ose-haproxy-router:v3.11.98 \
ose-hyperkube:v3.11.98 \
ose-hypershift:v3.11.98 \
ose-keepalived-ipfailover:v3.11.98 \
ose-kube-rbac-proxy:v3.11.98 \
ose-kube-state-metrics:v3.11.98 \
ose-metrics-server:v3.11.98 \
ose-node:v3.11.98 \
ose-node-problem-detector:v3.11.98 \
ose-operator-lifecycle-manager:v3.11.98 \
ose-ovn-kubernetes:v3.11.98 \
ose-pod:v3.11.98 \
ose-prometheus-config-reloader:v3.11.98 \
ose-prometheus-operator:v3.11.98 \
ose-recycler:v3.11.98 \
ose-service-catalog:v3.11.98 \
ose-template-service-broker:v3.11.98 \
ose-tests:v3.11.98 \
ose-web-console:v3.11.98 \
postgresql-apb:v3.11.98 \
registry-console:v3.11.98 \
snapshot-controller:v3.11.98 \
snapshot-provisioner:v3.11.98; \
do \
  sudo docker pull registry.access.redhat.com/openshift3/$image
done
```

```
sudo docker pull registry.access.redhat.com/rhel7/etcd:3.2.22

for image in \
metrics-cassandra:v3.11.98 \
metrics-hawkular-metrics:v3.11.98 \
metrics-hawkular-openshift-agent:v3.11.98 \
metrics-heapster:v3.11.98 \
metrics-schema-installer:v3.11.98 \
oauth-proxy:v3.11.98 \
ose-logging-curator5:v3.11.98 \
ose-logging-elasticsearch5:v3.11.98 \
ose-logging-eventrouter:v3.11.98 \
ose-logging-fluentd:v3.11.98 \
ose-logging-kibana5:v3.11.98 \
prometheus:v3.11.98 \
prometheus-alert-buffer:v3.11.98 \
prometheus-alertmanager:v3.11.98 \
prometheus-node-exporter:v3.11.98; \
do \
  sudo docker pull registry.access.redhat.com/openshift3/$image
done

for image in \
cfme-openshift-postgresql \
cfme-openshift-memcached \
cfme-openshift-app-ui \
cfme-openshift-app \
cfme-openshift-embedded-ansible \
cfme-openshift-httpd \
cfme-httpd-configmap-generator; \
do \
  sudo docker pull registry.access.redhat.com/cloudforms46/$image
done

for image in \
rhgs-server-rhel7 \
rhgs-volmanager-rhel7 \
rhgs-gluster-block-prov-rhel7 \
rhgs-s3-server-rhel7; \
do \
  sudo docker pull registry.access.redhat.com/rhgs3/$image
done

for image in \
jboss-amq-6/amq63-openshift \
jboss-datagrid-7/datagrid71-openshift \
jboss-datagrid-7/datagrid71-client-openshift \
jboss-datavirt-6/datavirt63-openshift \
jboss-datavirt-6/datavirt63-driver-openshift \
jboss-decisionserver-6/decisionserver64-openshift \
jboss-processserver-6/processserver64-openshift \
jboss-eap-6/eap64-openshift \
jboss-eap-7/eap70-openshift \
jboss-webserver-3/webserver31-tomcat7-openshift \
jboss-webserver-3/webserver31-tomcat8-openshift \
openshift3/jenkins-2-rhel7 \
openshift3/jenkins-agent-maven-35-rhel7 \
openshift3/jenkins-agent-nodejs-8-rhel7 \
openshift3/jenkins-slave-base-rhel7 \
openshift3/jenkins-slave-maven-rhel7 \
openshift3/jenkins-slave-nodejs-rhel7 \
rhscl/mongodb-32-rhel7 \
rhscl/mysql-57-rhel7 \
rhscl/perl-524-rhel7 \
rhscl/php-56-rhel7 \
rhscl/postgresql-95-rhel7 \
rhscl/python-35-rhel7 \
redhat-sso-7/sso70-openshift \
rhscl/ruby-24-rhel7 \
redhat-openjdk-18/openjdk18-openshift \
redhat-sso-7/sso71-openshift \
rhscl/nodejs-6-rhel7 \
rhscl/mariadb-101-rhel7; \
do \
  sudo docker pull registry.access.redhat.com/$image
done
```

Exporting images

Since your environment does not have access to your internal network and requires physical media to transfer content, export the images to compressed files.

1. Create a directory to store your compressed images in and change to it:

```
$ mkdir </path/to/images>
$ cd </path/to/images>
```

2. Export the OpenShift Container Platform infrastructure component images:
    a. For on-premise installations on x86_64 servers, run the following command:

```
$ sudo docker save -o ose3-images.tar \
    registry.redhat.io/openshift3/apb-base \
    registry.redhat.io/openshift3/apb-tools \
    registry.redhat.io/openshift3/automation-broker-apb \
    registry.redhat.io/openshift3/csi-attacher \
    registry.redhat.io/openshift3/csi-driver-registrar \
    registry.redhat.io/openshift3/csi-livenessprobe \
    registry.redhat.io/openshift3/csi-provisioner \
    registry.redhat.io/openshift3/grafana \
    registry.redhat.io/openshift3/image-inspector \
    registry.redhat.io/openshift3/local-storage-provisioner \
    registry.redhat.io/openshift3/manila-provisioner \
    registry.redhat.io/openshift3/mariadb-apb \
    registry.redhat.io/openshift3/mediawiki \
    registry.redhat.io/openshift3/mediawiki-apb \
    registry.redhat.io/openshift3/mysql-apb \
    registry.redhat.io/openshift3/ose-ansible \
    registry.redhat.io/openshift3/ose-ansible-service-broker \
    registry.redhat.io/openshift3/ose-cli \
    registry.redhat.io/openshift3/ose-cluster-autoscaler \
    registry.redhat.io/openshift3/ose-cluster-capacity \
    registry.redhat.io/openshift3/ose-cluster-monitoring-operator \
    registry.redhat.io/openshift3/ose-console \
    registry.redhat.io/openshift3/ose-configmap-reloader \
    registry.redhat.io/openshift3/ose-control-plane \
    registry.redhat.io/openshift3/ose-deployer \
    registry.redhat.io/openshift3/ose-descheduler \
    registry.redhat.io/openshift3/ose-docker-builder \
    registry.redhat.io/openshift3/ose-docker-registry \
    registry.redhat.io/openshift3/ose-efs-provisioner \
    registry.redhat.io/openshift3/ose-egress-dns-proxy \
    registry.redhat.io/openshift3/ose-egress-http-proxy \
    registry.redhat.io/openshift3/ose-egress-router \
    registry.redhat.io/openshift3/ose-haproxy-router \
    registry.redhat.io/openshift3/ose-hyperkube \
    registry.redhat.io/openshift3/ose-hypershift \
    registry.redhat.io/openshift3/ose-keepalived-ipfailover \
    registry.redhat.io/openshift3/ose-kube-rbac-proxy \
    registry.redhat.io/openshift3/ose-kube-state-metrics \
    registry.redhat.io/openshift3/ose-metrics-server \
    registry.redhat.io/openshift3/ose-node \
    registry.redhat.io/openshift3/ose-node-problem-detector \
    registry.redhat.io/openshift3/ose-operator-lifecycle-manager \
    registry.redhat.io/openshift3/ose-ovn-kubernetes \
    registry.redhat.io/openshift3/ose-pod \
    registry.redhat.io/openshift3/ose-prometheus-config-reloader \
    registry.redhat.io/openshift3/ose-prometheus-operator \
    registry.redhat.io/openshift3/ose-recycler \
    registry.redhat.io/openshift3/ose-service-catalog \
    registry.redhat.io/openshift3/ose-template-service-broker \
    registry.redhat.io/openshift3/ose-tests \
    registry.redhat.io/openshift3/ose-web-console \
    registry.redhat.io/openshift3/postgresql-apb \
    registry.redhat.io/openshift3/registry-console \
    registry.redhat.io/openshift3/snapshot-controller \
    registry.redhat.io/openshift3/snapshot-provisioner
    registry.redhat.io/rhel7/etcd
```

b. For optional components, to export them, run the following command:

```
$ sudo docker save -o ose3-optional-imags.tar \
    registry.redhat.io/openshift3/metrics-cassandra \
    registry.redhat.io/openshift3/metrics-hawkular-metrics \
    registry.redhat.io/openshift3/metrics-hawkular-openshift-agent \
    registry.redhat.io/openshift3/metrics-heapster \
    registry.redhat.io/openshift3/metrics-schema-installer \
    registry.redhat.io/openshift3/oauth-proxy \
    registry.redhat.io/openshift3/ose-logging-curator5 \
    registry.redhat.io/openshift3/ose-logging-elasticsearch5 \
    registry.redhat.io/openshift3/ose-logging-eventrouter \
    registry.redhat.io/openshift3/ose-logging-fluentd \
    registry.redhat.io/openshift3/ose-logging-kibana5 \
    registry.redhat.io/openshift3/prometheus \
    registry.redhat.io/openshift3/prometheus-alert-buffer \
    registry.redhat.io/openshift3/prometheus-alertmanager \
    registry.redhat.io/openshift3/prometheus-node-exporter \
    registry.redhat.io/cloudforms46/cfme-openshift-postgresql \
    registry.redhat.io/cloudforms46/cfme-openshift-memcached \
    registry.redhat.io/cloudforms46/cfme-openshift-app-ui \
    registry.redhat.io/cloudforms46/cfme-openshift-app \
    registry.redhat.io/cloudforms46/cfme-openshift-embedded-ansible
\
    registry.redhat.io/cloudforms46/cfme-openshift-httpd \
    registry.redhat.io/cloudforms46/cfme-httpd-configmap-generator \
    registry.redhat.io/rhgs3/rhgs-server-rhel7 \
    registry.redhat.io/rhgs3/rhgs-volmanager-rhel7 \
    registry.redhat.io/rhgs3/rhgs-gluster-block-prov-rhel7 \
    registry.redhat.io/rhgs3/rhgs-s3-server-rhel7
```

c. Export the S2I builder images that you pulled.

```
sudo docker save -o ose3-builder-images.tar \

registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat7-openshift \
      registry.access.redhat.com/jboss-amq-6/amq63-openshift \
      registry.access.redhat.com/jboss-datagrid-7/datagrid71-openshift \
      registry.access.redhat.com/jboss-datagrid-7/datagrid71-client-openshift \
      registry.access.redhat.com/jboss-datavirt-6/datavirt63-openshift \
      registry.access.redhat.com/jboss-datavirt-6/datavirt63-driver-openshift \

registry.access.redhat.com/jboss-decisionserver-6/decisionserver64-openshift \

registry.access.redhat.com/jboss-processserver-6/processserver64-openshift \
      registry.access.redhat.com/jboss-eap-6/eap64-openshift \
      registry.access.redhat.com/jboss-eap-7/eap70-openshift \

registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat7-openshift \

registry.access.redhat.com/jboss-webserver-3/webserver31-tomcat8-openshift \
      registry.access.redhat.com/openshift3/jenkins-2-rhel7 \
      registry.access.redhat.com/openshift3/jenkins-agent-maven-35-rhel7 \
      registry.access.redhat.com/openshift3/jenkins-agent-nodejs-8-rhel7 \
      registry.access.redhat.com/openshift3/jenkins-slave-base-rhel7 \
      registry.access.redhat.com/openshift3/jenkins-slave-maven-rhel7 \
      registry.access.redhat.com/openshift3/jenkins-slave-nodejs-rhel7 \
      registry.access.redhat.com/rhscl/mongodb-32-rhel7 \
      registry.access.redhat.com/rhscl/mysql-57-rhel7 \
      registry.access.redhat.com/rhscl/perl-524-rhel7 \
      registry.access.redhat.com/rhscl/php-56-rhel7 \
      registry.access.redhat.com/rhscl/postgresql-95-rhel7 \
      registry.access.redhat.com/rhscl/python-35-rhel7 \
      registry.access.redhat.com/redhat-sso-7/sso70-openshift \
      registry.access.redhat.com/rhscl/ruby-24-rhel7 \
      registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift \
      registry.access.redhat.com/redhat-sso-7/sso71-openshift \
      registry.access.redhat.com/rhscl/nodejs-6-rhel7 \
      registry.access.redhat.com/rhscl/mariadb-101-rhel7
```

d. Copy the compressed files from your Internet-connected host to your internal host.

```
-rw-------. 1 root root  12G Dec 20 13:00 ose3-images.tar
-rw-------. 1 root root 7.3G Dec 20 14:29 ose3-optional-images.tar
-rw-------. 1 root root  11G Dec 20 14:53 ose3-builder-images.tar
```

Exporting repos

Since your environment does not have access to your internal network and requires physical media to transfer content, export the repos to compressed files.

1. Create a directory to store your compressed images in and change to it:

   ```
   $ mkdir </path/to/repos>
   $ cd </path/to/repos>
   ```

2. Export the OpenShift Container Platform infrastructure component repos:

   ```
   $ sudo tar -czvf ocp3.11.98.repo.tgz ocprepo/
   ```

3. Copy the compressed files from your Internet-connected host to your internal host.

   ```
   -rw-r--r--.  1 root root 36G Dec 20 13:57 ocp3.11.98.repo.tgz
   ```

Setup on internal (not connected to the internet) VM: ocprepo

1. Deploy instance using RHEL 7.6 image
   Note: httpd should already be installed on the image
2. Copy prepackaged tar files to /var/www/html/
3. Unarchive the repos in the /var/www/html/repos
4. Check mode on repos directory
   chmod -R 7777 repos
5. Open firewalld port 80/tcp

   ```
    systemctl status firewalld.service
    systemctl start firewalld.service
    systemctl enable firewalld.service
   firewall-cmd --permanent --add-port=80/tcp
    firewall-cmd --reload
   ```

6. Enable and Start httpd service

4.3 Offline Container registry deployment

This section covers the setup of an offline container registry. The procedure is based on the installation guide [Creating an Offline Container Registry for openshift 3.11]

Setup when using offline server

1. Deploy instance using RHEL 7.6 image (alternatively use the same instance as in section 4.2)
2. Install local registry

   ```
   yum docker-distribution -y
   systemctl enable docker-distribution
   systemctl start docker-distribution
   ```

3. Edit /etc/docker/daemon.json so it has the local registry as insecure:

   ```
   {
           "insecure-registries" : ["<registryIp>:5000"]
   ```

}

4. Restart docker process
5. Re-Tag images
   Run the following command for each image:
   note: <tag> value must match the value defined in local_registry_images.yaml on directory. In this deployment tags are 'v3.11.98' , 'v3.11', 'latest':
   ```
   docker tag <registryIp>:5000/<image>:<tag>
   ```
6. Push to local registry
   ```
   docker push <image>:<tag>
   ```
9. Open firewalld port 5000/tcp
   ```
   systemctl status firewalld.service
   systemctl start firewalld.service
   systemctl enable firewalld.service
   firewall-cmd --permanent --add-port=5000/tcp
   firewall-cmd --reload
   ```

4.4 Ansible Tower

At the start we downloaded the tar file from this website: ansible tower tar file**.**

20 GB of dedicated hard disk space for Tower service nodes and for nodes containing a database 150 GB+ recommended. Mount this disks to /var in the node.
After this we edited the inventory file to our needs.
Then we run the ./setup.sh script and it installed ansible tower.
Edit the /etc/ansible/ansible.cfg forks   = <number that you want> (default is 5, we increased it to 100)
Initially we need to install govc on the ansible tower server: guide to install GOVC.
For backup run the ./setup.sh -b script.
After installation License required to be able to complete initial configuration. We used https://store.ansible.com/redhat/tower_license/ - Ansible Enterprise license generator FOR RED HATTERS (need valid Red Hat email address - for internal use only)

4.5 central grafana

Follow these steps to install central grafana: central grafana

4.6 Template creation in vSphere for OCP VMs

First we created in vSphere rhel 7.6 template that contains a minimal installation with some addons to it:

- We enabled the disk uuid option.
- Disabled the swap.
- Enabled Network Manager service.
- Enabled SElinux in enforced mode.
- Enabled the IP forwarding.

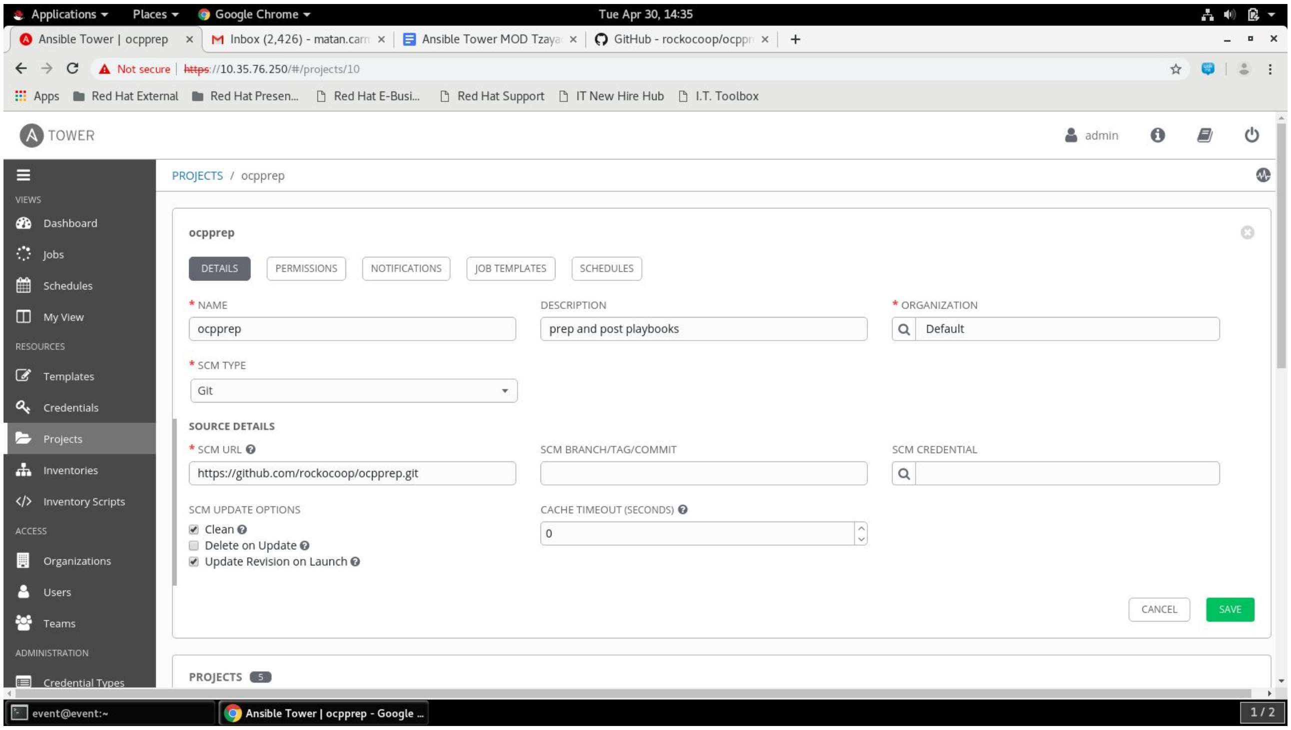
- We entered the public key of the ansible tower server in the authorized_keys file.

Created a private key of the root user of the ansible tower, because this is necessary for the ansible tower passwordless ssh access to managed hosts. When we start a job we need to choose which private key we want to use.

4.7 Ansible Tower configurations and definitions

We created three projects:

- The first project was ocpprep which contains the content of a git repository ocpprep repository:



ocpprep

* NAME: ocpprep

DESCRIPTION: prep and post playbooks

* ORGANIZATION: Default

* SCM TYPE: Git

SOURCE DETAILS

* SCM URL: https://github.com/rockocoop/ocpprep.git

SCM BRANCH/TAG/COMMIT:

SCM CREDENTIAL:

SCM UPDATE OPTIONS

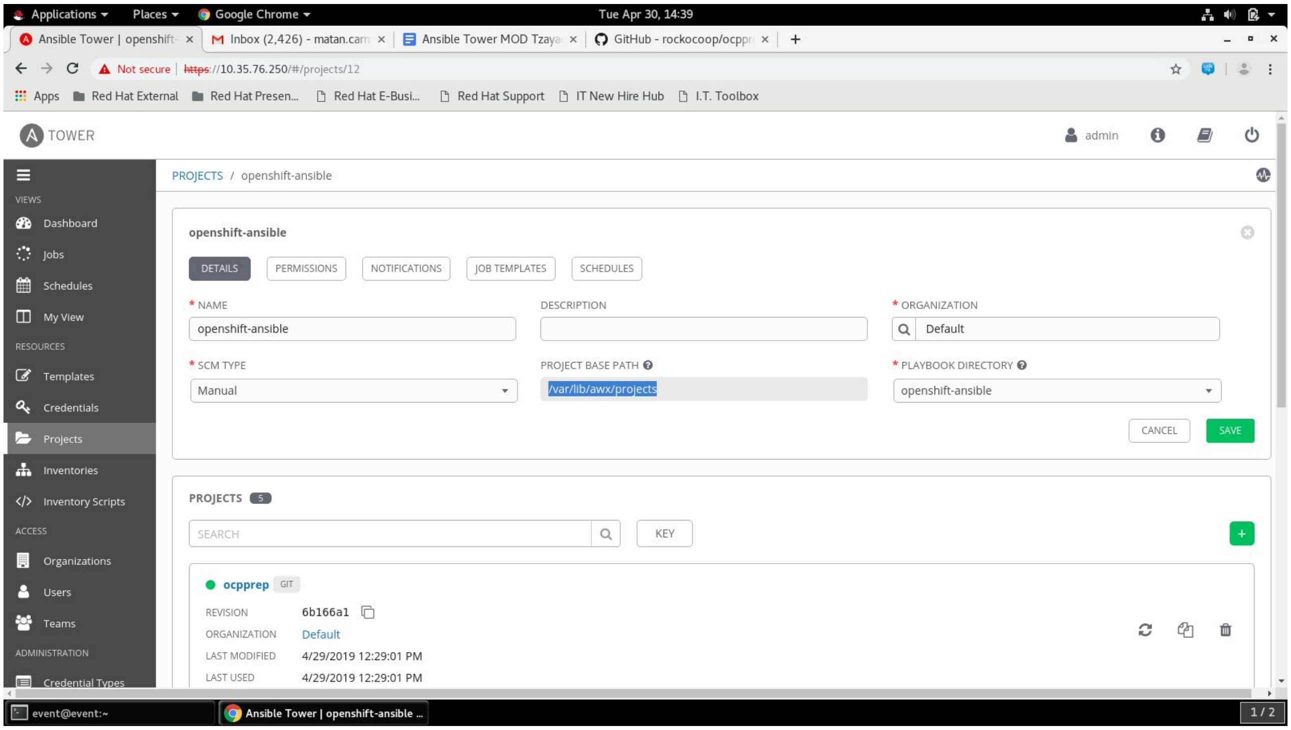
☑ Clean

☐ Delete on Update

☑ Update Revision on Launch

CACHE TIMEOUT (SECONDS): 0

- The second project was openshift-ansible which contains all the installation playbooks and it's located locally on the tower server under /var/lib/awx/projects folder:



openshift-ansible

* NAME: openshift-ansible

DESCRIPTION:

* ORGANIZATION: Default

* SCM TYPE: Manual

PROJECT BASE PATH: /var/lib/awx/projects

* PLAYBOOK DIRECTORY: openshift-ansible

PROJECTS 5

ocpprep GIT

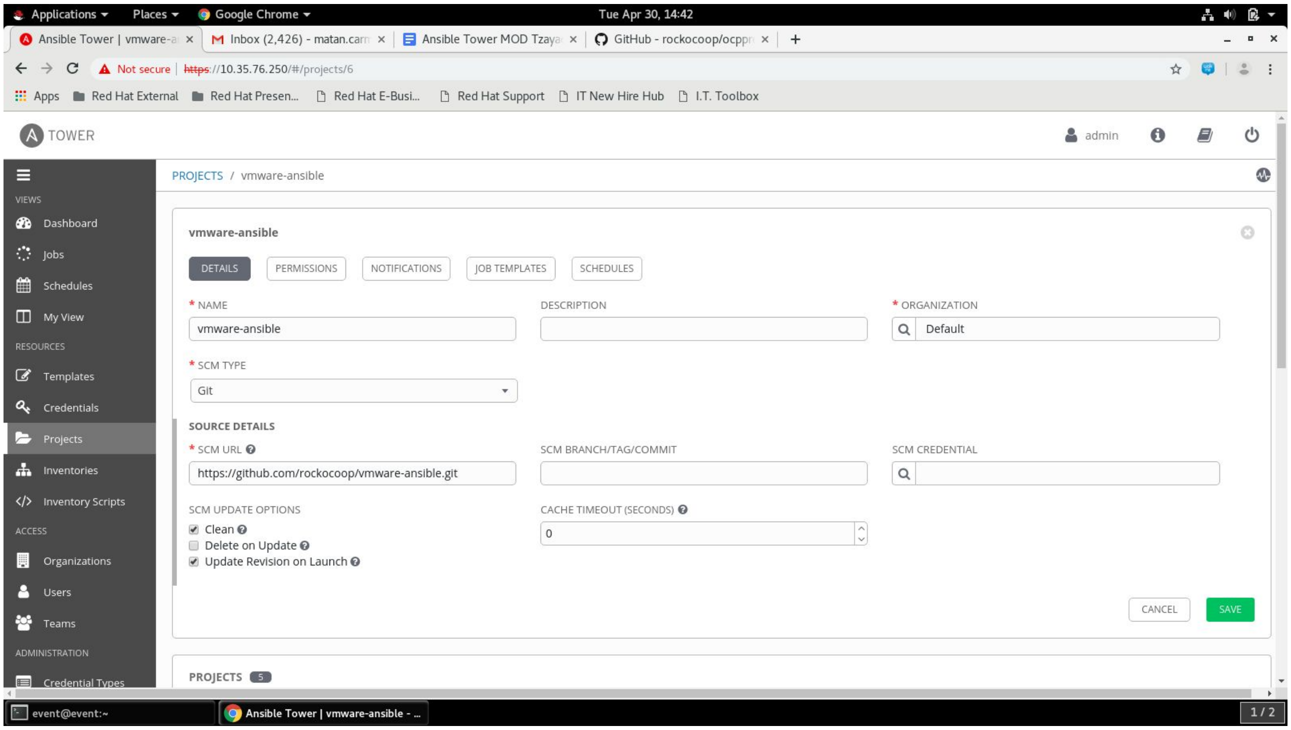
REVISION 6b166a1

ORGANIZATION Default

LAST MODIFIED 4/29/2019 12:29:01 PM

LAST USED 4/29/2019 12:29:01 PM

- The third project was vmware-ansible which contained the playbooks that created the VMs of the OCP cluster or destroy them vmware-ansible:



vmware-ansible

* NAME: vmware-ansible

DESCRIPTION:

* ORGANIZATION: Default

* SCM TYPE: Git

SOURCE DETAILS

* SCM URL: https://github.com/rockocoop/vmware-ansible.git

SCM BRANCH/TAG/COMMIT:

SCM CREDENTIAL:

SCM UPDATE OPTIONS

☑ Clean

☐ Delete on Update

☑ Update Revision on Launch

CACHE TIMEOUT (SECONDS): 0

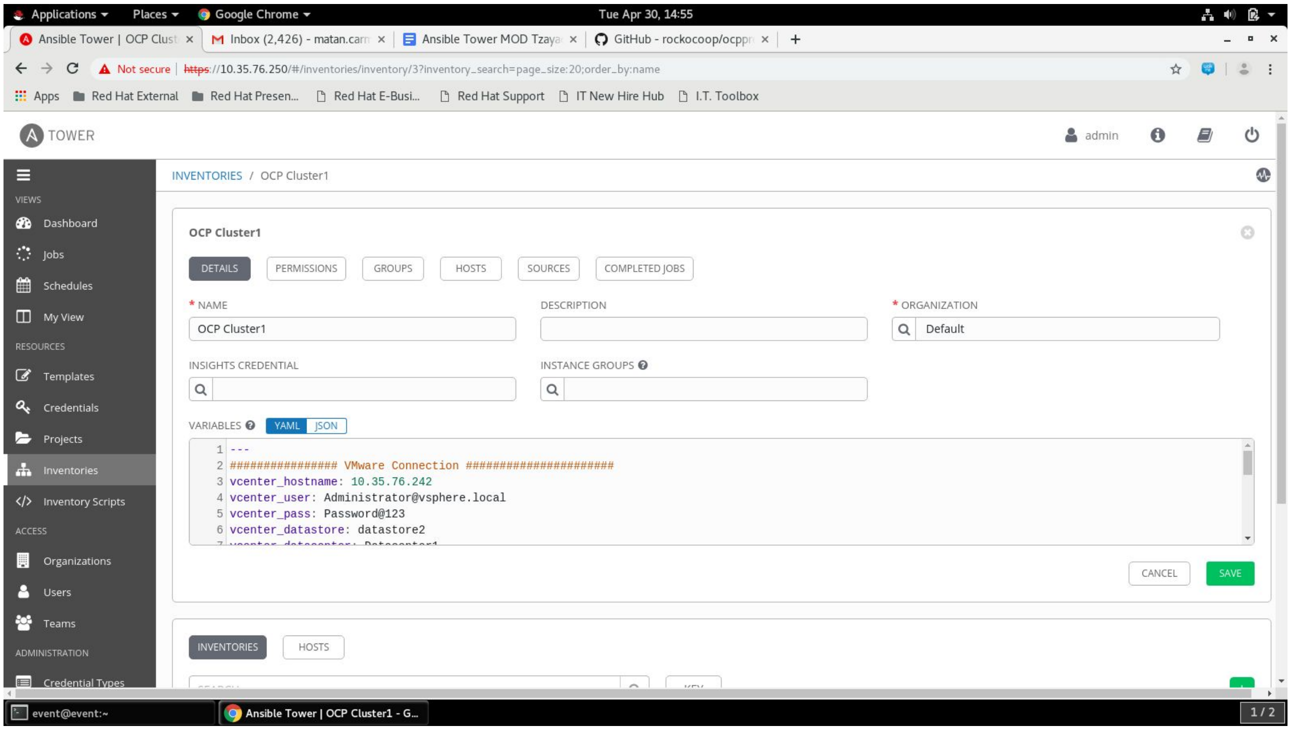
PROJECTS 5

We also created two more project that looked like the first and the third projects but the difference was that the playbooks were located locally on the tower server instead of being on git and we tested it and it worked.

After that we created inventory for the jobs. The first section of the inventory is the details, it contains variables on all of the hosts:

```
---
################ VMware Connection ######################
vcenter_hostname: 10.35.76.242
vcenter_user: Administrator@vsphere.local
vcenter_pass: Password@123
vcenter_datastore: datastore2
vcenter_datacenter: Datacenter1
vcenter_cluster: Cluster1
vcenter_vmtemplate: rhel76
vcenter_folder_full: /Cluster1/OCP1/

#################General Parameters#######################
ansible_ssh_common_args: '-o StrictHostKeyChecking=no -o ControlMaster=auto -o ControlPersist=60s -o PreferredAuthentications=publickey'
ansible_become: false
cluster_timezone: UTC
#ntp=["10.56.190.1","10.56.190.2"]
searchdomain: ocp1.zayadtest.com
dns: ["10.35.76.249","8.8.8.8"]
openshift_docker_insecure_registries: 10.35.76.249:5000
openshift_docker_blocked_registries: registry.access.redhat.com,docker.io
yumrepo_url: "10.35.76.249/ocprepo"
keepalived_vip: 10.35.76.209
keepalived_interface: ens192
keepalived_vrrpid: 1
routervialb: true
Roottemppas: Password@123

#connect to external grafana
grafanaURL: "http://10.35.76.246:3000"
grafanaPass: "Password@123"
grafanaClusterName: "OCP1"
grafmas: "ocpmas1.ocp1.zayadtest.com"
prometheusURL: "https://prometheus-k8s-openshift-monitoring.apps.openshift.ocp1.zayadtest.com"
```

And it looked like this:



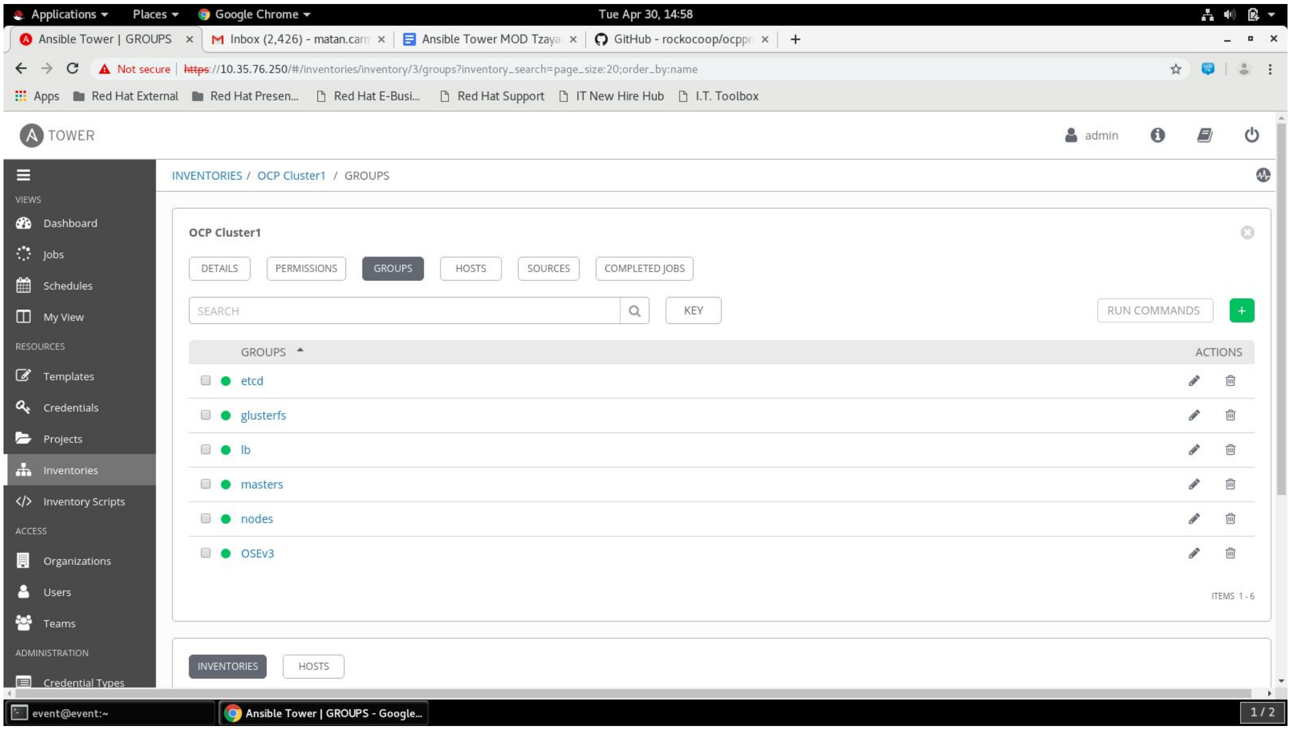
OCP Cluster1

* NAME	DESCRIPTION	* ORGANIZATION
OCP Cluster1		Default
INSIGHTS CREDENTIAL	INSTANCE GROUPS	

VARIABLES

```
---
################ VMware Connection ######################
vcenter_hostname: 10.35.76.242
vcenter_user: Administrator@vsphere.local
vcenter_pass: Password@123
vcenter_datastore: datastore2
vcenter_datacenter: Datacenter1
```

After this we created the groups of the hosts:



GROUPS	ACTIONS
etcd	
glusterfs	
lb	
masters	
nodes	
OSEv3	

ITEMS 1 - 6

- Etcd: this group contains the hosts that was meant to serve as etcd, the parameter that relates to the etcd hosts:
  ```
  ---
  pv_etcd: sdc
  ```
- Glusterfs: this group contains the hosts that was meant to serve as glusterfs, it doesn't have any special variables that relates to the glusterfs hosts.
- Lb: this group contains the hosts that was meant to serve as load balancer, the parameter that relates to the lb hosts:
  ```
  ---
  vmCPUs: 2
  vmMemory: 4096
  vmDisk: 40
  ```
- Masters: this group contains the hosts that was meant to serve as masters, it doesn't have any special variables that relates to the masters hosts.
- Nodes: this group contains all the hosts in the cluster except of the load balancers, the parameter that relates to the nodes hosts:
  ```
  ---
  vmCPUs: 4
  vmMemory: 16384
  vmDisk: 40
  pv_device: sdb
  ```
- OSEv3: this group contains the all the other groups (if you configures that the etcd role will be on the masters so you don't need to put the etcd group in the OSEv3 group), the parameter that relates to the OSEv3 hosts:
  ```
  ---
  #################General Parameters#######################
  ansible_ssh_user: root
  ansible_become: true
  debug_level: 2
  openshift_deployment_type: openshift-enterprise
  openshift_image_tag: v3.11.98
  openshift_release: 3.11.98
  openshift_pkg_version: -3.11.98
  openshift_disable_check: memory_availability,disk_availability,docker_image_availability
  openshift_master_default_subdomain: apps.openshift.ocp1.zayadtest.com
  openshift_master_cluster_hostname: openshift.ocp1.zayadtest.com
  openshift_master_cluster_public_hostname: openshift.ocp1.zayadtest.com

  #################The identity providers using HTpasswd#######################
  openshift_master_identity_providers: [{'name': 'htpasswd_auth', 'login': 'true', 'challenge': 'true', 'kind': 'HTPasswdPasswordIdentityProvider'}]
  openshift_master_htpasswd_users: {'admin':'$apr1$udcx1.9E$XZmOZkWhIRa5lzxIcBOxz1'}

  #################Disconnected environment#######################
  oreg_url: 10.35.76.249:5000/openshift3/ose-${component}:${version}
  ```

```
openshift_examples_modify_imagestreams: True
openshift_enable_service_catalog: False
template_service_broker_install: True
#openshift_template_service_broker_namespaces: ['openshift']
#ansible_service_broker_install: true
#ansible_service_broker_local_registry_whitelist: ['.*-apb$']
os_firewall_use_firewalld: True
openshift_hosted_manage_registry: True
openshift_cluster_monitoring_operator_install: True
openshift_metrics_install_metrics: True
openshift_logging_install_logging: True
openshift_logging_es_nodeselector: {"node-role.kubernetes.io/infra": "true"}
openshift_logging_es_memory_limit: 2Gi
openshift_node_groups: [{‘name’:’node-config-master’, ‘labels’: [‘node-role.kubernetes.io/master=true’,’runtime=docker’]}, {‘name’:’node-config-infra’, ‘labels’: [‘node-role.kubernetes.io/infra=true’,’runtime=docker’,’logging-infra=elasticsearch]},{‘name’:’node-config-compute’, ‘labels’: [‘node-role.kubernetes.io/compute=true’,’runtime=docker’], ‘edits’: [{ ‘key’:’kubeletArguments.pods-per-core’, ‘value’: [‘20’]}]}]
# Network sdn plugin changed to openshift-ovs-multitenant
os_sdn_network_plugin_name: redhat/openshift-ovs-multitenant

################ Vsphere backend pv ######################
openshift_cloudprovider_kind: vsphere
openshift_cloudprovider_vsphere_username: administrator@vsphere.local
openshift_cloudprovider_vsphere_password: Password@123
openshift_cloudprovider_vsphere_host: 10.35.76.242
openshift_cloudprovider_vsphere_datacenter: Datacenter1
openshift_cloudprovider_vsphere_cluster: Cluster1
openshift_cloudprovider_vsphere_datastore: datastore3
openshift_cloudprovider_vsphere_folder: /Cluster1/OCP1/

# Set the metrics dynamic storage
openshift_master_dynamic_provisioning_enabled: True
openshift_metrics_storage_kind: dynamic
openshift_metrics_cassandra_storage_type: dynamic
openshift_metrics_cassandra_pvc_size: 15Gi
# Set the logging dynamic storage
openshift_logging_storage_kind: dynamic
openshift_logging_es_pvc_dynamic: True
openshift_logging_es_pvc_size: 20Gi
openshift_logging_storage_access_modes: ['ReadWriteOnce']
# Set the registry dynemic storage
openshift_hosted_registry_storage_kind: dynamic
openshift_hosted_registry_storage_access_modes: ['ReadWriteOnce']
```

```
openshift_hosted_registry_storage_volume_size: 10Gi
# Set the prometheus dynamic storage
openshift_prometheus_storage_kind: dynamic
openshift_cluster_monitoring_operator_prometheus_storage_enabled: true
openshift_cluster_monitoring_operator_prometheus_storage_capacity: 50Gi
# Set the prometheus-alertmanager dynemic storage
openshift_prometheus_alertmanager_storage_kind: dynamic
openshift_cluster_monitoring_operator_alertmanager_storage_enabled: true
openshift_cluster_monitoring_operator_alertmanager_storage_capacity: 2Gi
# Set the prometheus-alertbuffer dynamic storage
openshift_prometheus_alertbuffer_storage_kind: dynamic
openshift_prometheus_alertbuffer_storage_type: pvc
openshift_prometheus_alertbuffer_pvc_size: 10Gi
openshift_prometheus_pvc_access_modes: [ReadWriteOnce]

# if you want to use also with glusterfs as a pv you need to uncomment it.
#openshift_storage_glusterfs_image: 10.35.76.249:5000/rhgs3/rhgs-server-rhel7:v3.11
#openshift_storage_glusterfs_block_image:10.35.76.249:5000/rhgs3/rhgs-gluster-block-prov-rhel7:v3.11
#openshift_storage_glusterfs_heketi_image: 10.35.76.249:5000/rhgs3/rhgs-volmanager-rhel7:v3.11
```

Initially we create the templates that defines which playbook will run. We have several templates:

- Deploy VMware: this template runs the playbooks/provisioningVMware.yml from vmware-ansible project, this playbook run 3 roles:
    1. Vmware_create_vm: this role creates the cluster vms, the main.yml looks like this:

```
---

- name: Abort Play when IP is in use
  command: ping -c1 {{ ansible_ssh_host }}
  delegate_to: localhost
  register: ping_result
  failed_when: >
    ("" not in ping_result.stderr)
  changed_when: False

- meta: end_play
  when: ping_result.rc==0


- name:
  debug:
    msg: " IP Address {{ ansible_ssh_host }} is free"
    verbosity: 3
  when: ping_result.rc==1
```

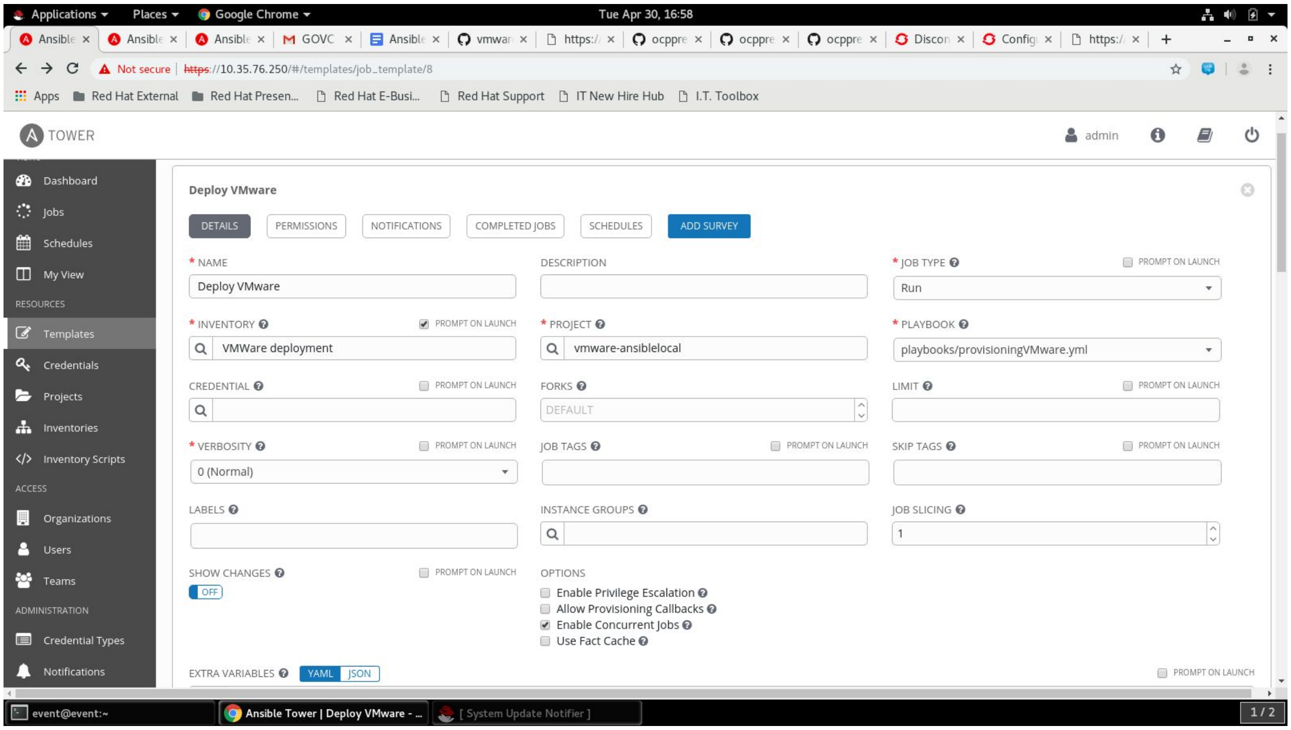
```
- name: Creat a new VM
  block:
    - set_fact:
        disk_size: "{{ vmDisk }}"
        memory: "{{ vmMemory }}"
        cpu: "{{ vmCPUs }}"

    - vmware_guest:
        hostname: "{{ vcenter_hostname }}"
        username: "{{ vcenter_user }}"
        password: "{{ vcenter_pass }}"
        datacenter: "{{ vcenter_datacenter }}"
        name: "{{ hostname }}"
        template: "{{ vcenter_vmtemplate }}"
        resource_pool: "{{ vcenter_resource_pool | default(omit) }}"
        cluster: "{{ vcenter_cluster }}"
        state: poweredon
        folder: "{{ vcenter_folder_full }}"
        validate_certs: False
        is_template: no
        disk:
          - size_gb: "{{ disk_size }}"
            type: thin
            datastore: "{{ vcenter_datastore }}"
        hardware:
          memory_mb: "{{ memory }}"
          num_cpus: "{{ cpu }}"
        networks:
         - name: "{{ vlan }}"
           ip: "{{ ansible_ssh_host }}"
           netmask: "{{ netmask }}"
           gateway: "{{ gateway }}"
        customization:
          dns_servers:
          - "{{ dns[0] }}"
          - "{{ dns[1] }}"
        wait_for_ip_address: yes
      register: new_vm
      delegate_to: localhost
      failed_when: >
        (new_vm.failed==true)
  rescue:
    - debug:
        msg: " VM {{ hostname }} already Exists"
        verbosity: 3
    - meta: end_play



  - name: New VM Status Message
  debug:
    msg: " New VM {{ hostname }}, {{ ansible_ssh_host }}, is Up and Running"
```

2. Vmware_add_disk: this role adds a disk to all the cluster nodes except the lb hosts, this is the main.yml:

```
---

- name: Load govc Vars
  include_vars: "{{ role_path }}/vars/govc_vars.yml"

- name: Get VM Info/State
  command: "govc vm.info -vm.ip={{ ansible_ssh_host }} -k=true"
  environment:
    GOVC_USERNAME: "{{ govc_username }}"
    GOVC_PASSWORD: "{{ govc_password }}"
    GOVC_INSECURE: "{{ govc_insecure }}"
    GOVC_DATASTORE: "{{ govc_datastore}}"
    GOVC_DATACENTER: "{{ govc_datacenter}}"
    GOVC_URL: "{{ govc_url }}"
  register: govc_vm_info
  changed_when: False
  delegate_to: localhost
  failed_when: >
    ("no such VM" in govc_vm_info.stdout)

- name: Add SCSI to existing VM
  command: "govc device.scsi.add -vm={{ hostname }} -type=pvscsi
-k=true"
  environment:
    GOVC_USERNAME: "{{ govc_username }}"
    GOVC_PASSWORD: "{{ govc_password }}"
    GOVC_INSECURE: "{{ govc_insecure }}"
    GOVC_DATASTORE: "{{ govc_datastore}}"
    GOVC_DATACENTER: "{{ govc_datacenter}}"
    GOVC_URL: "{{ govc_url }}"
  register: govc_scsi_add
  changed_when: False
  delegate_to: localhost
  failed_when: >
    ("pvscsi-100" not in govc_scsi_add.stdout)
  with_items:
    - "{{disks}}"
  loop_control:
    loop_var: scsi

- name: wait for reconfiguration
  pause: seconds=10

- name: Add Disk to VM
  block:
  #  Add /dev/sdb Disk to existing VM
    - command: >
        govc vm.disk.create
        -vm={{ hostname }}
        -name='{{ hostname }}/{{ hostname }}_{{index}}'
        -ds='/{{ vcenter_datacenter }}/datastore/{{ vcenter_datastore
}}'
        -mode=independent_persistent
        -controller pvscsi-100{{index + 1 }}
        -eager=false
        -thick=false
        -size {{ item }}G
        -k=true
      environment:
        GOVC_USERNAME: "{{ govc_username }}"
        GOVC_PASSWORD: "{{ govc_password }}"
        GOVC_INSECURE: "{{ govc_insecure }}"
        GOVC_DATASTORE: "{{ govc_datastore}}"
        GOVC_DATACENTER: "{{ govc_datacenter}}"
        GOVC_URL: "{{ govc_url }}"
      register: govc_sdb_add
      changed_when: False
      delegate_to: localhost
      failed_when: >
        ("Creating disk" not in govc_sdb_add.stdout)
      with_items:
        - "{{disks}}"
      loop_control:
        index_var: index
```

3. Vm_power: this role restart all the vms, the main.yml:

```
---

- name: Power Control
  vmware_guest:
    hostname: "{{ vcenter_hostname }}"
    username: "{{ vcenter_user }}"
    password: "{{ vcenter_pass }}"
    validate_certs: no
    cluster: "{{ vcenter_cluster }}"
    name: "{{ hostname }}"
    state: "{{ vm_state }}"
  delegate_to: localhost

- pause: seconds=30

#- name: Wait 300 seconds for remote system to be available, but only
start checking after 30 seconds
#   wait_for_connection:
#     delay: 30
   # timeout: 300
```



Deploy VMware

DETAILS | PERMISSIONS | NOTIFICATIONS | COMPLETED JOBS | SCHEDULES | ADD SURVEY

* NAME: Deploy VMware	DESCRIPTION:	* JOB TYPE (□ PROMPT ON LAUNCH): Run
* INVENTORY (☑ PROMPT ON LAUNCH): VMWare deployment	* PROJECT: vmware-ansiblelocal	* PLAYBOOK: playbooks/provisioningVMware.yml
CREDENTIAL (□ PROMPT ON LAUNCH):	FORKS: DEFAULT	LIMIT (□ PROMPT ON LAUNCH):
* VERBOSITY (□ PROMPT ON LAUNCH): 0 (Normal)	JOB TAGS (□ PROMPT ON LAUNCH):	SKIP TAGS (□ PROMPT ON LAUNCH):
LABELS:	INSTANCE GROUPS:	JOB SLICING: 1
SHOW CHANGES (□ PROMPT ON LAUNCH): OFF	OPTIONS: □ Enable Privilege Escalation □ Allow Provisioning Callbacks ☑ Enable Concurrent Jobs □ Use Fact Cache	

EXTRA VARIABLES: YAML | JSON (□ PROMPT ON LAUNCH)

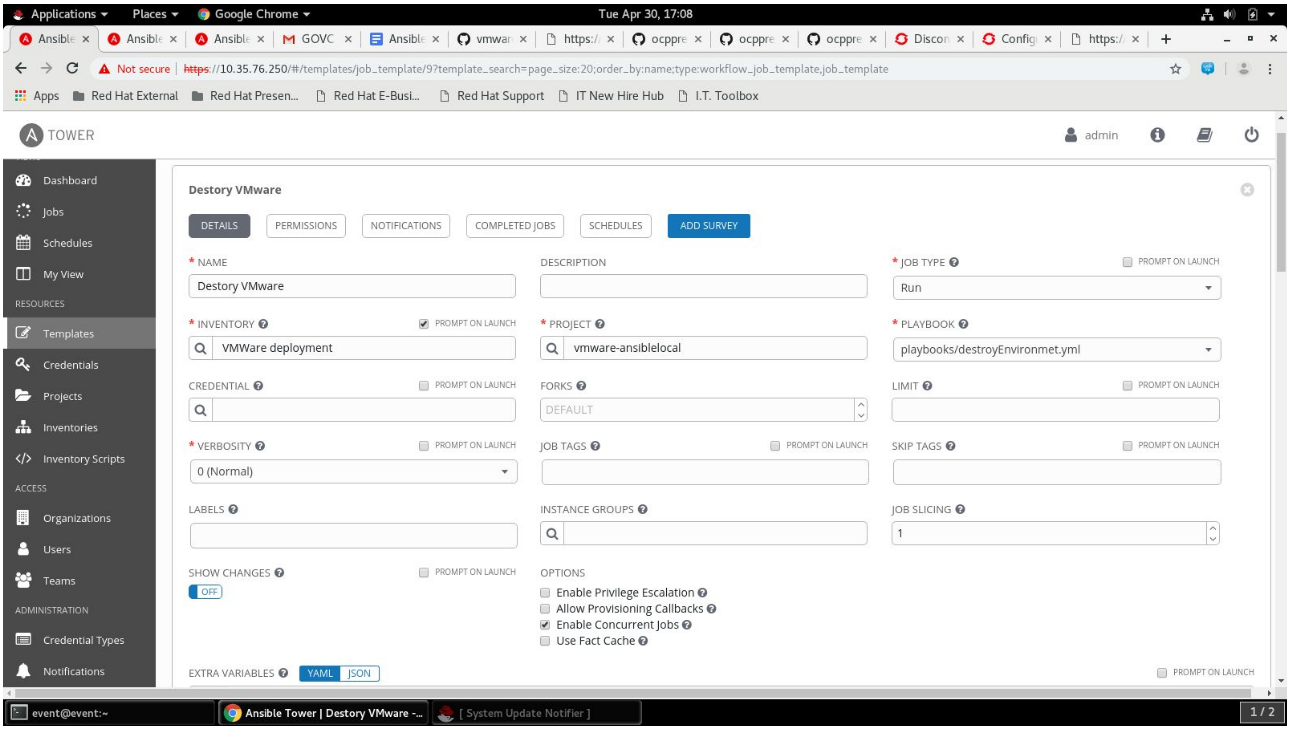
- The second template is Destroy VMware, this template runs the playbook playbooks/destroyEnvironmet.yml from vmware-ansible project, this playbook run 3 roles:
    1. Vm_power: this role poweroff all the cluster vms, this is the main.yml:

```
---

- name: Power Control
  vmware_guest:
    hostname: "{{ vcenter_hostname }}"
    username: "{{ vcenter_user }}"
    password: "{{ vcenter_pass }}"
    validate_certs: no
    cluster: "{{ vcenter_cluster }}"
    name: "{{ hostname }}"
    state: "{{ vm_state }}"
  delegate_to: localhost
```

```
- pause: seconds=30

#- name: Wait 300 seconds for remote system to be available, but only start checking after 30 seconds
#  wait_for_connection:
#    delay: 30
  # timeout: 300
```

2. Vmware_remove_vm: this role erase all the cluster vms, this is the main.yml:

```
- name: Load govc Vars
  include_vars: "{{ role_path }}/vars/govc_vars.yml"

- name: Remove "{{ hostname }}"
  vmware_guest:
    hostname: "{{ vcenter_hostname }}"
    username: "{{ vcenter_user }}"
    password: "{{ vcenter_pass }}"
    validate_certs: no
    cluster: "{{ vcenter_cluster }}"
    name: "{{ hostname }}"
    state: absent
  delegate_to: localhost
  register: facts
```



Destory VMware

* NAME: Destory VMware
DESCRIPTION:
* JOB TYPE: Run — □ PROMPT ON LAUNCH
* INVENTORY: VMWare deployment — ☑ PROMPT ON LAUNCH
* PROJECT: vmware-ansiblelocal
* PLAYBOOK: playbooks/destroyEnvironmet.yml
CREDENTIAL: — □ PROMPT ON LAUNCH
FORKS: DEFAULT
LIMIT: — □ PROMPT ON LAUNCH
* VERBOSITY: 0 (Normal) — □ PROMPT ON LAUNCH
JOB TAGS: — □ PROMPT ON LAUNCH
SKIP TAGS: — □ PROMPT ON LAUNCH
LABELS:
INSTANCE GROUPS:
JOB SLICING: 1
SHOW CHANGES: OFF — □ PROMPT ON LAUNCH
OPTIONS
□ Enable Privilege Escalation
□ Allow Provisioning Callbacks
☑ Enable Concurrent Jobs
□ Use Fact Cache
EXTRA VARIABLES: YAML JSON — □ PROMPT ON LAUNCH

- The third template is Pre-Install, this template run the playbook playbooks/pre-install.yml from ocpprep project, this playbook performs pre install tasks on the cluster nodes, this playbooks runs this playbooks:

1. Setup_internal_repos.yml: this playbook configure all the repository sources for openshift in the cluster servers, the main.yml:

```
---
- name: mkdir oldrepos
  file:
   path: /etc/yum.repos.d/.oldrepos
   state: directory
  tags: mkdir_oldrepos

- name: Check to see if any repo files found in /etc/yum.repos.d
  find:
    paths: "/etc/yum.repos.d"
    patterns: "*.repo"
  register: repo_files

- name: Move all yum.repo files to /etc/yum.repos.d/.oldrepos
  shell: "mv /etc/yum.repos.d/*.repo /etc/yum.repos.d/.oldrepos/"
  tags: mv_repo
  when: repo_files.matched|int > 0

- name: RHEL Openshift repos setup
  template:
    src: templates/setup_internal_repos.yml
    dest: /etc/yum.repos.d/ose.repo
```

2. Then it installs the supported packages on all the servers:

```
- name: install support packages
     package:
       name: "{{item}}"
       state: present
     with_items:
       - wget
       - git
       - net-tools
       - bind-utils
       - yum-utils
       - iptables-services
       - bridge-utils
       - bash-completion
       - kexec-tools
       - sos
       - psacct
       - atomic-openshift-docker-excluder
       - atomic-openshift-excluder
       - ntp
```

3. Then it’s configuring the ntp, update resolv file for dns and builds hosts file:

```
- name:     Configure NTP | Add server lines to ntp.conf
    template:
      src:    templates/ntpconf.j2
      dest:   /etc/ntp.conf
    when:     ntp is defined
    notify:   ntp_restart

  - name:     Configure NTP | start ntp daemon
    service:
      name: ntpd
      enabled: yes
      state: started

  - name: update resolv file for dns
    template:
      src: templates/resolv.j2
      dest: /etc/resolv.conf
    when: dns is defined

  - name: set hostname
    hostname:
      name: "{{ hostname }}"

  - name: Build hosts file
    lineinfile:
      path: /etc/hosts
      line: "{{ hostvars[item].ansible_default_ipv4.address }} {{ hostvars[item].hostname }}"
      state: present
    when: 'hostvars[item].ansible_default_ipv4.address is defined'
    with_items:
      - "{{ groups['OSEv3'] }}"

  handlers:
  - name: ntp_restart
    service:
      name: ntpd
      state: restarted

 - name: update /etc/hosts on all nodes with ivpcoe-vip
    lineinfile:
      dest: /etc/hosts
      line: '{{ keepalived_vip }}  {{ openshift_master_cluster_public_hostname }}'
    when: keepalived_vip is defined
```

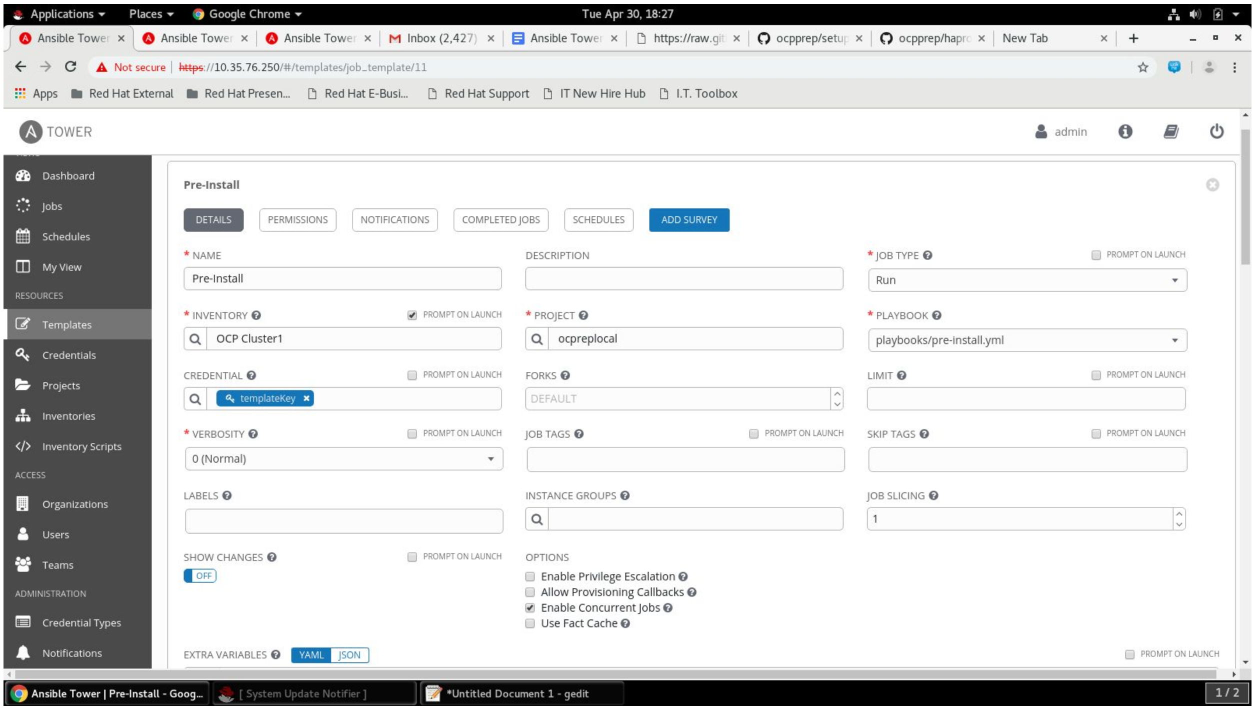
4. Configure the keepalive on the load balancers:

```
- hosts: lb
  become: true
  tasks:


  - name: install keepalived on lb nodes
    package:
      name: "{{item}}"
    with_items:
     - keepalived
    when: keepalived_vip is defined

  - name: create keepalived.conf
    template:
      src: ../templates/keepalived.conf
      dest: /etc/keepalived/keepalived.conf
    when: keepalived_vip is defined

  - name: restart keepalived
    service:
      name: keepalived
      state: restarted
      enabled: yes
    when: keepalived_vip is defined
```

5. Initially it installs docker on all servers except the lbs:

```
- name: Install Docker
  hosts: nodes
  become: true
  gather_facts: true
  serial: "100%"

  tasks:


  - name: Install Docker | install packages
    package:
      name: "{{ item }}"
      state: present
    with_items:
      - docker
      - ethtool
    notify: docker_restart
```

```
- name: Install Docker | setup docker storage
  template:
    src: templates/docker-storage-setup.j2
    dest: /etc/sysconfig/docker-storage-setup
  notify: docker_restart
- name: Verify presence of /dev/docker-vg/docker-pool
  stat:
    path: /dev/docker-vg/docker-pool
  register: docker_vg_status

- name: Run docker-storage-setup
  command: /usr/bin/docker-storage-setup
  when: docker_vg_status.stat.islnk is not defined

- name: Install Docker | enable docker
  service:
    name: docker
    state: started
    enabled: yes

- name: setup etcd storage
  include_tasks: include/etcdlvol.yml
  when: inventory_hostname in groups['etcd'] and pv_etcd is defined

handlers:
- name: docker_restart
  service:
    name: docker
    state: restarted
```



Pre-Install

* NAME: Pre-Install

DESCRIPTION:

* JOB TYPE: Run — □ PROMPT ON LAUNCH

* INVENTORY: OCP Cluster1 — ☑ PROMPT ON LAUNCH

* PROJECT: ocpreplocal

* PLAYBOOK: playbooks/pre-install.yml

CREDENTIAL: templateKey — □ PROMPT ON LAUNCH

FORKS: DEFAULT

LIMIT: — □ PROMPT ON LAUNCH

* VERBOSITY: 0 (Normal) — □ PROMPT ON LAUNCH

JOB TAGS: — □ PROMPT ON LAUNCH

SKIP TAGS: — □ PROMPT ON LAUNCH

LABELS:

INSTANCE GROUPS:

JOB SLICING: 1

SHOW CHANGES: OFF — □ PROMPT ON LAUNCH

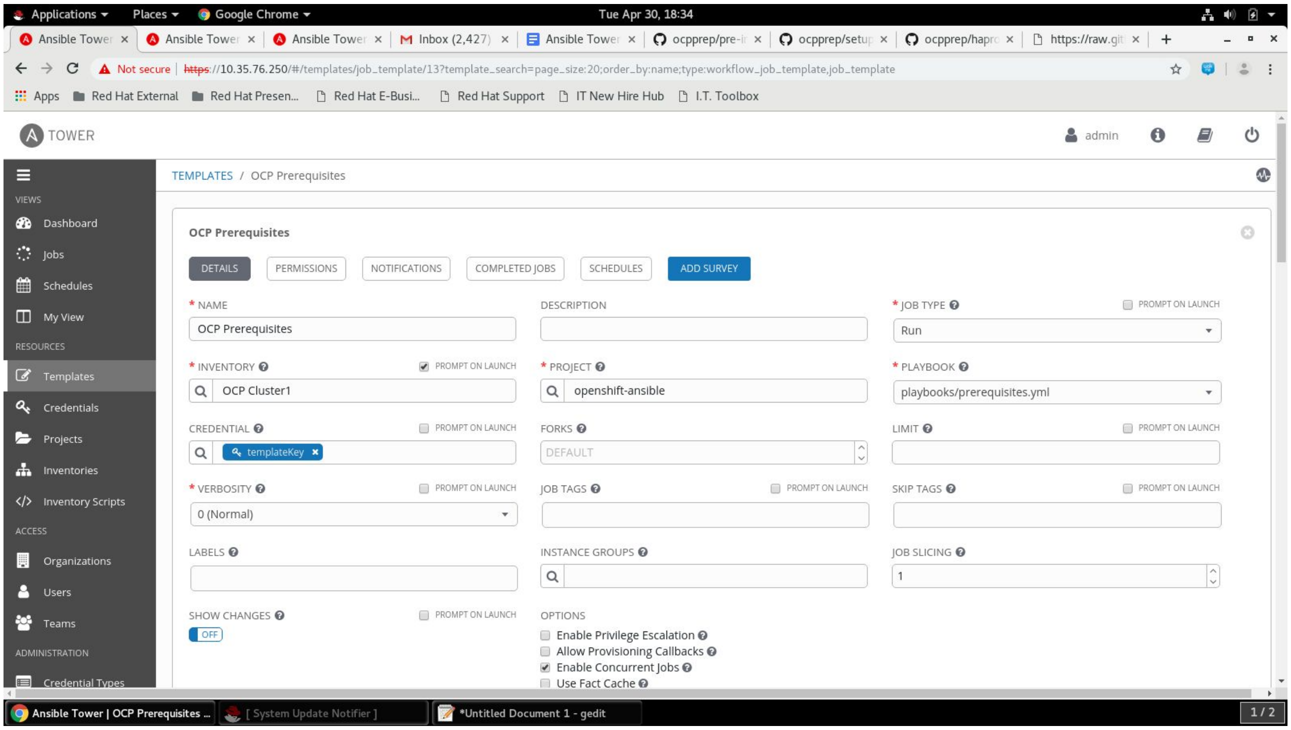
OPTIONS
□ Enable Privilege Escalation
□ Allow Provisioning Callbacks
☑ Enable Concurrent Jobs
□ Use Fact Cache

EXTRA VARIABLES: YAML JSON — □ PROMPT ON LAUNCH

- The fourth template is OCP Prerequisites, it runs the playbook playbooks/Prerequisites.yml from the project openshift-ansible, this playbook runs the Prerequisites tasks that needs to be done before the deployment. This is the playbook:

```
---
# l_scale_up_hosts may be passed in via various scaleup plays.
- name: Fail openshift_kubelet_name_override for new hosts
  hosts: "{{ l_scale_up_hosts | default('nodes') }}"
  tasks:
  - name: Fail when openshift_kubelet_name_override is defined
    fail:
      msg: "openshift_kubelet_name_override Cannot be defined for new hosts"
    when:
    - openshift_kubelet_name_override is defined
    - openshift_cloudprovider_kind | default('', true) != 'azure'

- import_playbook: init/main.yml
  vars:
    l_install_base_packages: True
    l_repo_hosts: "{{ l_scale_up_hosts | default('oo_all_hosts') }}"

- import_playbook: init/validate_hostnames.yml
  when: not (skip_validate_hostnames | default(False))

# This is required for container runtime for crio, only needs to run once.
- name: Configure os_firewall
```

```
  hosts: "{{ l_scale_up_hosts | default(l_default_firewall_hosts) }}"
  vars:
    l_default_firewall_hosts:
"oo_masters_to_config:oo_etcd_to_config:oo_lb_to_config:oo_nfs_to_config:oo_nodes_to_config"
  roles:
  - role: os_firewall

- import_playbook: container-runtime/private/setup_storage.yml

- import_playbook: container-runtime/private/config.yml
```

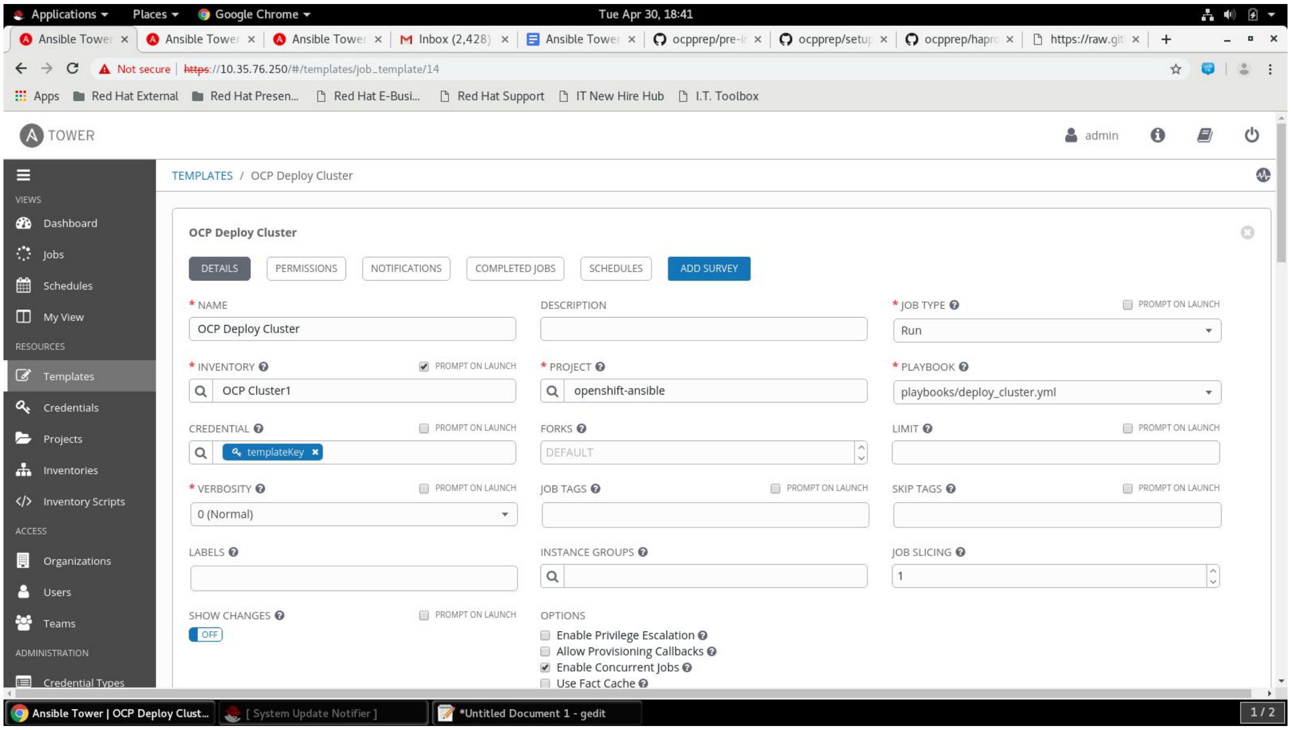


OCP Prerequisites

* NAME: OCP Prerequisites	DESCRIPTION:	* JOB TYPE: Run □ PROMPT ON LAUNCH
* INVENTORY: OCP Cluster1 ☑ PROMPT ON LAUNCH	* PROJECT: openshift-ansible	* PLAYBOOK: playbooks/prerequisites.yml
CREDENTIAL: templateKey □ PROMPT ON LAUNCH	FORKS: DEFAULT	LIMIT: □ PROMPT ON LAUNCH
* VERBOSITY: 0 (Normal) □ PROMPT ON LAUNCH	JOB TAGS: □ PROMPT ON LAUNCH	SKIP TAGS: □ PROMPT ON LAUNCH
LABELS:	INSTANCE GROUPS:	JOB SLICING: 1
SHOW CHANGES: OFF □ PROMPT ON LAUNCH	OPTIONS: □ Enable Privilege Escalation □ Allow Provisioning Callbacks ☑ Enable Concurrent Jobs □ Use Fact Cache	

- The fifth template is OCP Deploy Cluster, it runs the playbook playbooks/deploy_cluster.yml from the project openshift-ansible, this playbook runs the deployment tasks of the openshift cluster. This is the playbook:

```
---
- import_playbook: init/main.yml
  vars:
    l_networkman_check_hosts: "oo_nodes_to_config"

- import_playbook: openshift-checks/private/install.yml

- import_playbook: openshift-node/private/bootstrap.yml

- import_playbook: common/private/control_plane.yml

- import_playbook: openshift-node/private/join.yml
- import_playbook: common/private/components.yml
```



OCP Deploy Cluster

* NAME OCP Deploy Cluster	DESCRIPTION	* JOB TYPE □ PROMPT ON LAUNCH Run
* INVENTORY ☑ PROMPT ON LAUNCH OCP Cluster1	* PROJECT openshift-ansible	* PLAYBOOK playbooks/deploy_cluster.yml
CREDENTIAL □ PROMPT ON LAUNCH templateKey	FORKS DEFAULT	LIMIT □ PROMPT ON LAUNCH
* VERBOSITY □ PROMPT ON LAUNCH 0 (Normal)	JOB TAGS □ PROMPT ON LAUNCH	SKIP TAGS □ PROMPT ON LAUNCH
LABELS	INSTANCE GROUPS	JOB SLICING 1
SHOW CHANGES □ PROMPT ON LAUNCH OFF	OPTIONS □ Enable Privilege Escalation □ Allow Provisioning Callbacks ☑ Enable Concurrent Jobs □ Use Fact Cache	

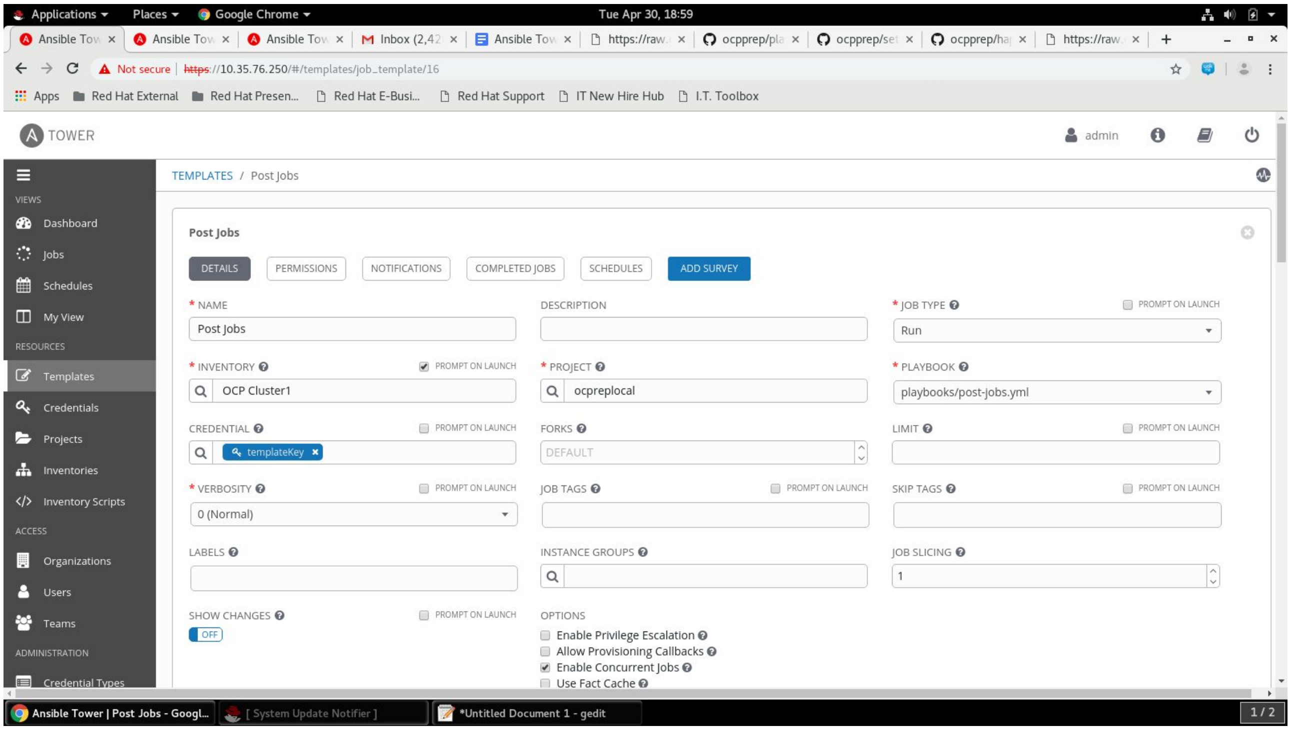
- The sixth template is Post Jobs, this template runs the playbook playbooks/post-jobs.yml from the ocpprep project, this playbook run some tasks:
    1. First it creates a datasource for the cluster in the central grafana:

```
- hosts:              localhost
  become:               true
  gather_facts:         false
  serial:             1

  tasks:

    - name: copy jq binary to master
      copy:
        src: "../supportBinary/jq-linux64"
        dest: "/opt/jq"
        mode: 777
      delegate_to: "{{grafmas}}"
      when: grafmas is defined

    - name: retrieve password
      shell:  oc get secret grafana-datasources -n openshift-monitoring -o json | /opt/jq
'.data["prometheus.yaml"]' | tr -d '"'| base64 -d|grep -i basicAuthPassword |awk '{print $2}'|tr
-d ','|tr -d '"'
      register: promPass
      delegate_to: "{{grafmas}}"
      when: grafmas is defined

    - name: Create datasource for cluster
      grafana_datasource:
        name: "{{grafanaClusterName}}"
        grafana_url: "{{grafanaURL}}"
        grafana_user: "admin"
        grafana_password: "{{grafanaPass}}"
        ds_type: "prometheus"
        url: "{{prometheusURL}}"
        basic_auth_user: "internal"
        basic_auth_password: "{{promPass.stdout}}"
        validate_certs: False
        tls_skip_verify: true
      when: grafmas is defined
```

2. Then it adds some configuration for the load balancers:

```
- hosts:                lb
  become:               true
  gather_facts:         false
  serial:               1

  tasks:

    - name: open port 80
      firewalld:
        port: 80/tcp
        permanent: true
        immediate: yes
        state: enabled

    - name: open port 443
      firewalld:
        port: 443/tcp
        permanent: true
        immediate: yes
        state: enabled

    - name: open port 2379
      firewalld:
        port: 2379/tcp
        permanent: true
        immediate: yes
        state: enabled
```

```
- name: restart firewalld
  service:
    name: firewalld
    state: restarted

- name: update haproxy config on lb
  template:
    src: templates/haproxy.cfg
    dest: haproxy.cfg
  when: routervialb is defined and routervialb == true
  delegate_to: localhost

- name: copy haproxy cfg to lb
  copy:
    src: haproxy.cfg
    dest: /etc/haproxy/

- name: selinux to open 2379
  seport:
    ports: 2379
    proto: tcp
    setype: http_port_t
    state: present
- name: restart haproxy
  service:
    name: haproxy
    state: restarted
```



Not secure | ~~https~~://10.35.76.250/#/templates/job_template/16

TEMPLATES / Post Jobs

Post Jobs

* NAME: Post Jobs

DESCRIPTION:

* JOB TYPE: Run — □ PROMPT ON LAUNCH

* INVENTORY: OCP Cluster1 — ☑ PROMPT ON LAUNCH

* PROJECT: ocpreplocal

* PLAYBOOK: playbooks/post-jobs.yml

CREDENTIAL: templateKey — □ PROMPT ON LAUNCH

FORKS: DEFAULT

LIMIT: — □ PROMPT ON LAUNCH

* VERBOSITY: 0 (Normal) — □ PROMPT ON LAUNCH

JOB TAGS: — □ PROMPT ON LAUNCH

SKIP TAGS: — □ PROMPT ON LAUNCH

LABELS:

INSTANCE GROUPS:

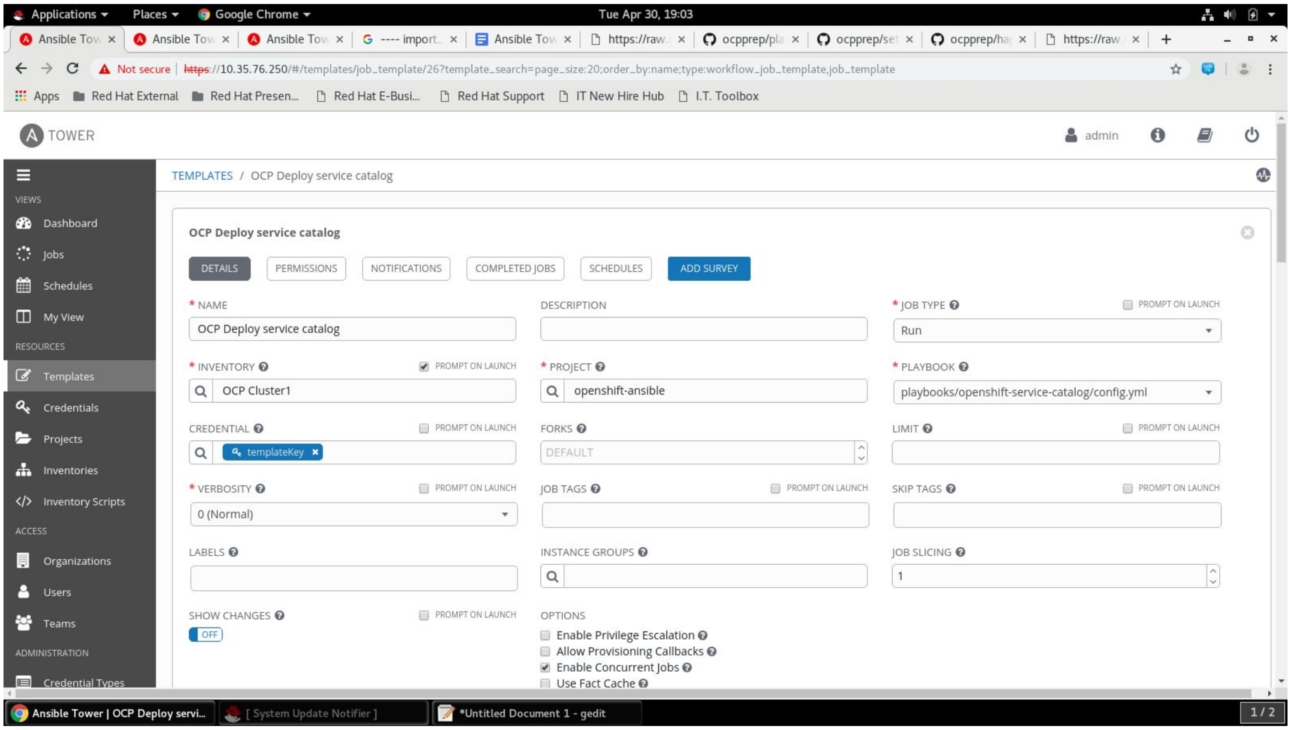
JOB SLICING: 1

SHOW CHANGES: OFF — □ PROMPT ON LAUNCH

OPTIONS
□ Enable Privilege Escalation
□ Allow Provisioning Callbacks
☑ Enable Concurrent Jobs
□ Use Fact Cache

- The seventh template is OCP Deploy service catalog, this template runs the playbook playbooks/openshift-service-catalog/config.yml from the openshift-ansible project. This playbook runs the deployment of the service catlog:

```
---
- import_playbook: ../init/main.yml
  vars:
    l_init_fact_hosts: "oo_masters_to_config:oo_etcd_to_config"
    l_openshift_version_set_hosts: "oo_masters_to_config:!oo_first_master"
    l_sanity_check_hosts: "{{ groups['oo_masters_to_config'] }}"

- import_playbook: private/config.yml
```

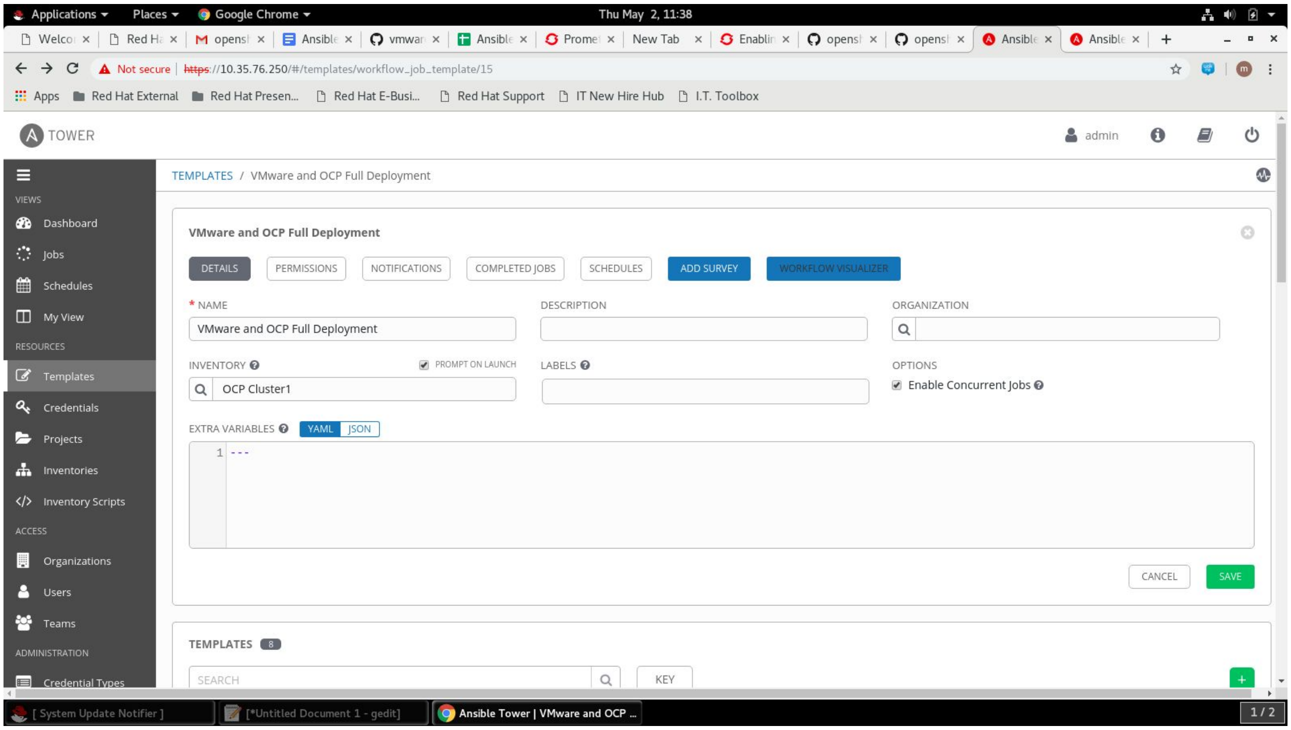

Applications Places Google Chrome
Tue Apr 30, 19:03
Not secure https://10.35.76.250/#/templates/job_template/26?template_search=page_size:20;order_by:name;type:workflow_job_template,job_template
Apps Red Hat External Red Hat Presen... Red Hat E-Busi... Red Hat Support IT New Hire Hub I.T. Toolbox
TOWER
admin
VIEWS
Dashboard
Jobs
Schedules
My View
RESOURCES
Templates
Credentials
Projects
Inventories
Inventory Scripts
ACCESS
Organizations
Users
Teams
ADMINISTRATION
Credential Types
DETAILS
PERMISSIONS
NOTIFICATIONS
COMPLETED JOBS
SCHEDULES
ADD SURVEY
Ansible Tower | OCP Deploy servi...
[System Update Notifier]
*Untitled Document 1 - gedit
1 / 2

TEMPLATES / OCP Deploy service catalog

OCP Deploy service catalog

- * NAME: OCP Deploy service catalog
- DESCRIPTION:
- * JOB TYPE: Run (□ PROMPT ON LAUNCH)
- * INVENTORY: OCP Cluster1 (☑ PROMPT ON LAUNCH)
- * PROJECT: openshift-ansible
- * PLAYBOOK: playbooks/openshift-service-catalog/config.yml
- CREDENTIAL: templateKey (□ PROMPT ON LAUNCH)
- FORKS: DEFAULT
- LIMIT: (□ PROMPT ON LAUNCH)
- * VERBOSITY: 0 (Normal) (□ PROMPT ON LAUNCH)
- JOB TAGS: (□ PROMPT ON LAUNCH)
- SKIP TAGS: (□ PROMPT ON LAUNCH)
- LABELS:
- INSTANCE GROUPS:
- JOB SLICING: 1
- SHOW CHANGES: OFF (□ PROMPT ON LAUNCH)
- OPTIONS:
  - □ Enable Privilege Escalation
  - □ Allow Provisioning Callbacks
  - ☑ Enable Concurrent Jobs
  - □ Use Fact Cache

- The last template is VMware and OCP Full Deployment, this is a template from kind workflow, that means that it runs other jobs in order that you choose:


Applications Places Google Chrome
Thu May 2, 11:38
Not secure https://10.35.76.250/#/templates/workflow_job_template/15
Apps Red Hat External Red Hat Presen... Red Hat E-Busi... Red Hat Support IT New Hire Hub I.T. Toolbox
TOWER
admin
VIEWS
Dashboard
Jobs
Schedules
My View
RESOURCES
Templates
Credentials
Projects
Inventories
Inventory Scripts
ACCESS
Organizations
Users
Teams
ADMINISTRATION
Credential Types
DETAILS
PERMISSIONS
NOTIFICATIONS
COMPLETED JOBS
SCHEDULES
ADD SURVEY
WORKFLOW VISUALIZER
CANCEL
SAVE
SEARCH
KEY
[System Update Notifier]
[*Untitled Document 1 - gedit]
Ansible Tower | VMware and OCP ...
1 / 2

TEMPLATES / VMware and OCP Full Deployment

VMware and OCP Full Deployment

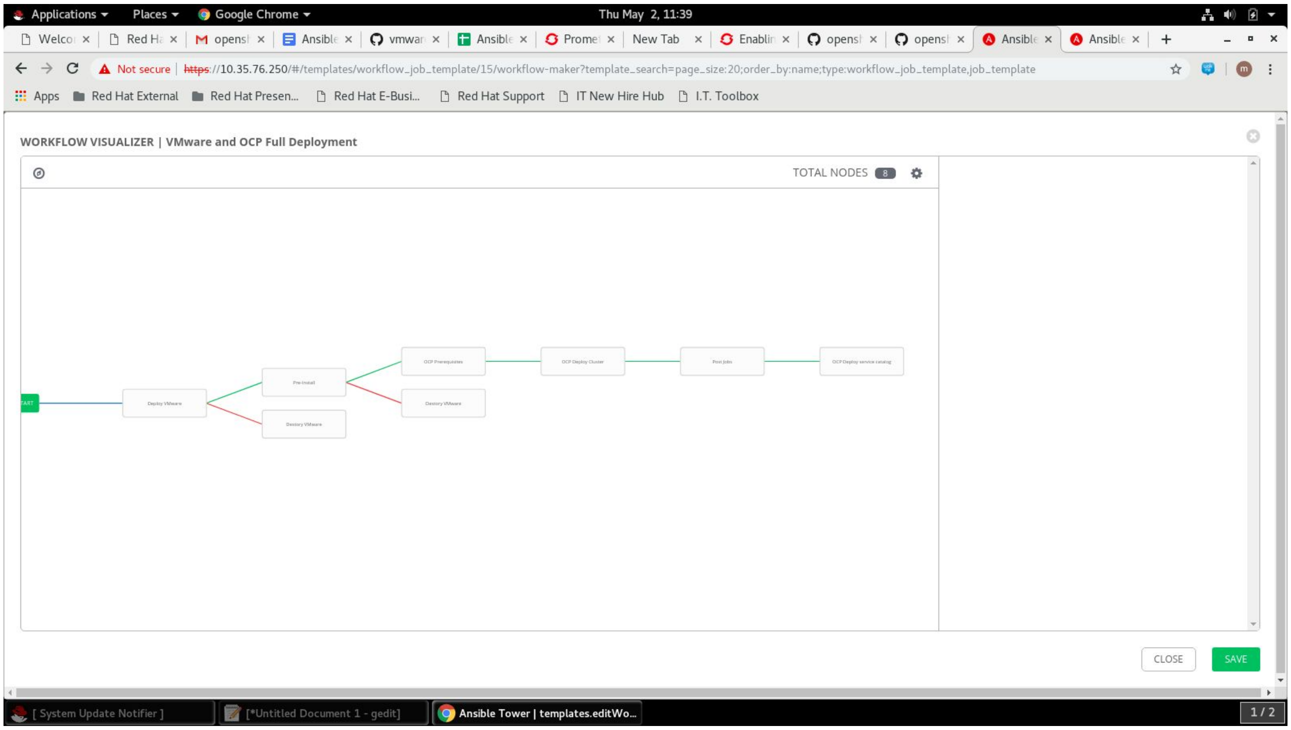
- * NAME: VMware and OCP Full Deployment
- DESCRIPTION:
- ORGANIZATION:
- INVENTORY: OCP Cluster1 (☑ PROMPT ON LAUNCH)
- LABELS:
- OPTIONS: ☑ Enable Concurrent Jobs
- EXTRA VARIABLES: YAML JSON

```
---
```

TEMPLATES 8

The order of the workflow is: first you run the Deploy VMware template, if it fails it runs the Destroy VMware template and exit. If the Deploy VMware works the workflow continues to template Pre-Install, if it fails it

runs the Destroy VMware template and exit.if the Pre-Install works the workflow continues to template OCP Prerequisites, if it fails the main template stop, if it works it continues to template OCP Deploy Cluster, if it fails the main template stop, if it works it continues to template Post Jobs, if it fails the main template stop, if it works it continues to template OCP Deploy service catalog, if it fails the main template stop, if it works that means that the workflow template worked.
And this is the workflow:


Applications
Places
Google Chrome
Thu May 2, 11:39
Not secure
https://10.35.76.250/#/templates/workflow_job_template/15/workflow-maker?template_search=page_size:20;order_by:name;type:workflow_job_template,job_template
Apps
Red Hat External
Red Hat Presen...
Red Hat E-Busi...
Red Hat Support
IT New Hire Hub
I.T. Toolbox
WORKFLOW VISUALIZER | VMware and OCP Full Deployment
TOTAL NODES 8
START
Deploy VMware
Pre-Install
Destory VMware
OCP Prerequisites
Destory VMware
OCP Deploy Cluster
Post Jobs
OCP Deploy service catalog
CLOSE
SAVE
[System Update Notifier]
[*Untitled Document 1 - gedit]
Ansible Tower | templates.editWo...
1 / 2

5. ENGAGEMENT CLOSURE AND RECOMMENDATIONS.

5.1 Knowledge Transfer

For this engagement the knowledge transfer was provided during the engagement. On this environment knowledge transfer was specifically provided to the following individual(s):

Client Staff Member	Notes

NAME_OF_MOD Mamram_STAFF_MEMBER Matan Carmeli	● Assisted with the data gathering process for the environment. ● Provided all the network details for the environment (including physical and virtual). ● Served as our full team member and learned as engagement was in progress ● Took active part in this journal preparation and demo

Table 5-1: Knowledge Transfer.

5.2 Testing

Testing was conducted in accordance with a predefined test plan as documented in Appendix 7.1.
Testing was conducted by the consultant and witnessed by a representative of MOD - Mamram as follows:

Description	Value
Overall Test Results	Passed
Testing completed on date:	01/05 - 02/05-2019
Testing Witnessed by:	01/05 - 02/05-2019

Table 4-2: Knowledge Transfer.

5.3 Engagement Observations

Overall, the engagement went smoothly and provided a good opportunity for the Customer’s team member to get hands on experience with Red Hat technologies and support practices. These skills can be improved by participation in Red Hat training programs.

5.4 Training and Certification

Specific course descriptions and availability can be found on Red Hat’s website under Training and Certification. The following courses are recommended:

Red Hat Certified System Administrator (RHCSA)

Verizon should invest in certifying two team members as RHCSA’s within their IT support staff in order to ensure the team can adequately manage and maintain the expanding environment. Various courses are available from Red Hat to train RHCSA candidates and prepare them for the certification exam, including ground up and fast tracks for existing Unix and Linux admins – The following course is a likely fit for Example.com’s experienced UNIX admin staff:

RHCSA Rapid Track course with exam (RH200)

“The RHCSA Rapid Track course with exam (RH200) is designed for experienced Linux and Unix system administrators who want to become accredited with the RHCSA certification. Students will learn to manage a Linux server, including installation and configuration of local components and services, as well as connections to existing network services. To successfully navigate this accelerated course, students must already have solid command line skills and know how to access man pages for help.”

Red Hat Automation with Ansible 1 (DO 407)

Red Hat Automation with Ansible 2: Ansible Tower (DO409)
Course Objectives

- *Deploy and use Red Hat Ansible Tower to manage existing Ansible projects, playbooks, and roles*
- *Use the visual dashboard to centrally launch, control, and monitor Ansible jobs*
- *Configure users and teams and use them to control access to systems, projects, and other resources through* role-based *access controls*
- *Automatically schedule Ansible jobs and update the host inventory*
- Perform basic maintenance and ad*ministration of the Red Hat Ansible Tower* installation
- Use the Red Hat Ansible Tower API to launch jobs from existing templates

6. APPENDIXES

6.1 Appendix A: RH Ansible Tower and OCP 3.11 Accelerator Test Matrix and Test result

ID	Test	Method	Expected Result	Notes	Pass/ Fail
S1	Ansible Tower Define Job Template	CLI or Dashboard	Template should be created successfully		Pass
S2	Ansible Tower - Define Project	CLI or Dashboard	Projects should be created successfully		Pass
S3	Ansible Tower - Inventory files editing	CLI or Dashboard	Different Inventory files should be edited successfully		Pass
S4	Ansible Tower - Define Workflow template	CLI or Dashboard	Template should be created successfully		Pass
S5	Ansible Tower - Running Jobs simultaneously	CLI or Dashboard	Two workflow jobs running simultaneously against two different inventory files	When working with Source GitHab - the same job run in sequential mode	Pass
S6	VMDK as persistent storage provisioning	CLI or Dashboard	Persistent Volume with storage class vsphere-storage		Pass

			should be created, PVC should be created		
S7	GlusterFS as persistent storage provisioning	CLI or Dashboard	Persistent Volume with storage class glusterfs-storage should be created, PVC should be created		Pass
S8	Two OCP cluster should be created including VM creation, preparation, OCP deployment, postinstall jobs	CLI or Dashboard	Workflow Templates should be completed successfully and as a result two ocp clusters should be up and running		Pass
S9	OCP - Admin role should be provided to admin User	CLI or Dashboard	Admin user should be able to manage cluster		Pass
S10	OCP - New projects creation	CLI or Dashboard	New Projects should be created successfully		Pass
S11	OCP - Dynamic PVC should be provisioned	CLI or Dashboard	Dynamic PVC should be created successfully		Pass
S12	OCP - new app creation	CLI or Dashboard	At least one App should be created successfully	Image streams should be edited before with definition of insecure registry	Pass
S13	Ansible Tower - both Clusters should be connected to the Central Grafana	CLI or Dashboard	In Central Grafana both OCP Clusters should be visualized successfully.		Pass

OTHER APPENDIX GOES HERE.

Copying-a-full-git-repo

https://docs.openshift.com/container-platform/3.11/install_config/configuring_vsphere.html

https://grafana.com/docs/installation/rpm/

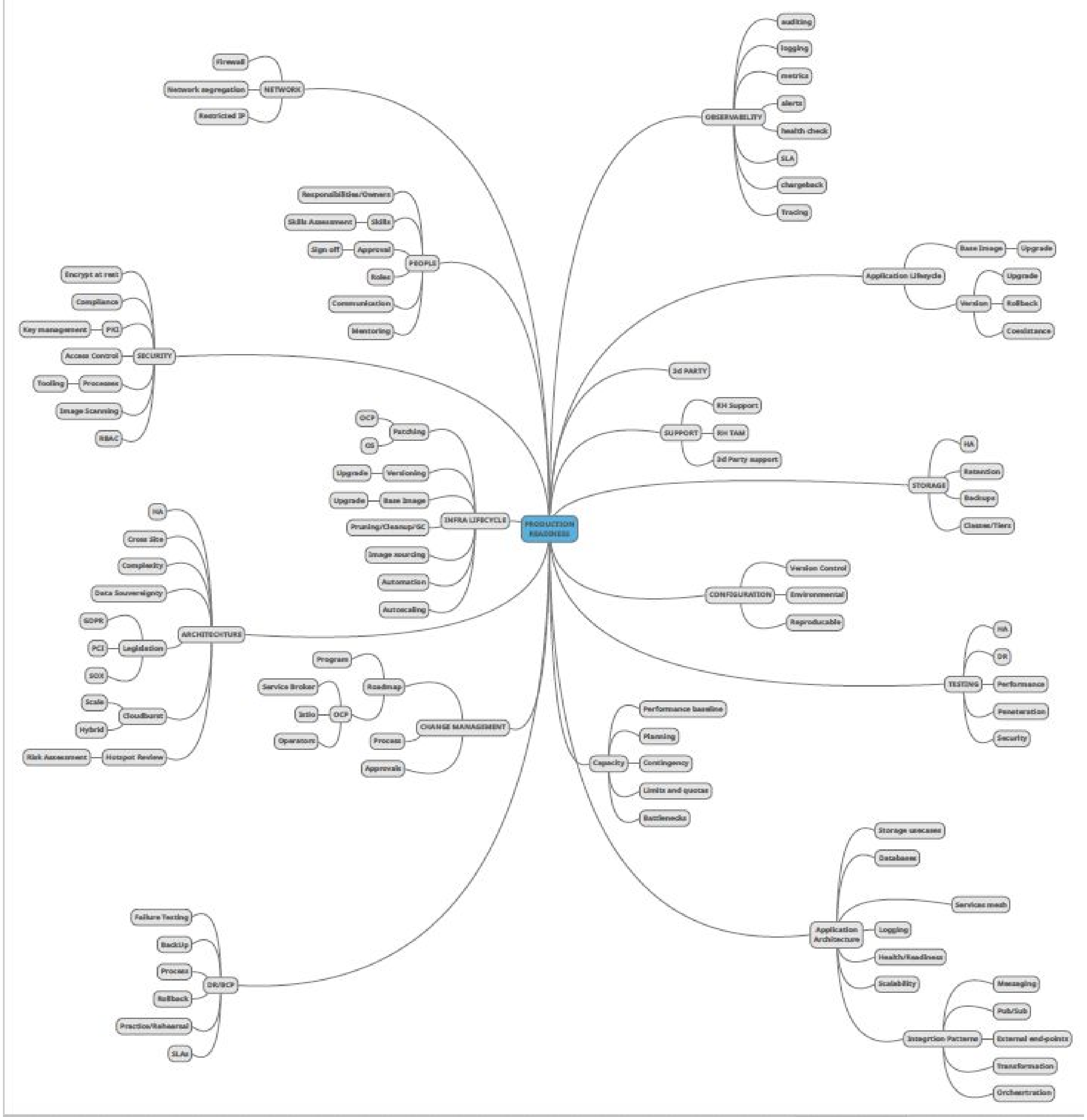
https://www.digitalocean.com/community/tutorials/how-to-use-prometheus-to-monitor-your-centos-7-server

https://docs.vmware.com/en/VMware-vSphere/6.7/com.vmware.vsphere.html.hostclient.doc/GUID-ED3ECA21-5763-4919-8947-A819A17980FB.html
https://access.redhat.com/articles/3344101

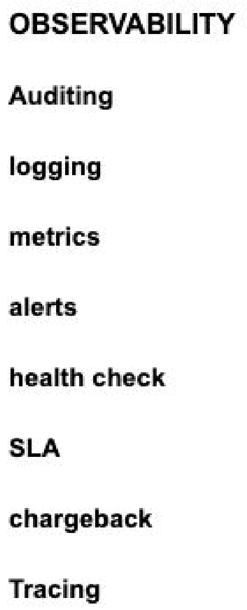
EOD Reports

Worksheet

Production Readiness Map



PRODUCTION READINESS



OBSERVABILITY

Auditing

logging

metrics

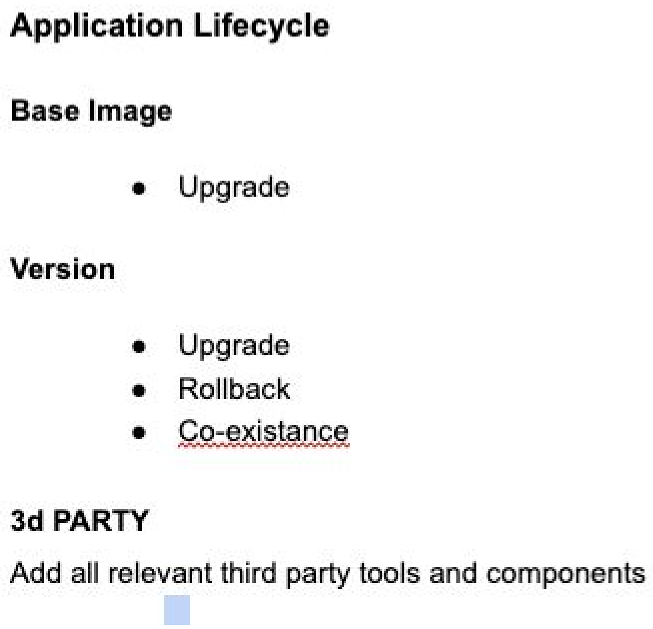
alerts

health check

SLA

chargeback

Tracing



Application Lifecycle

Base Image

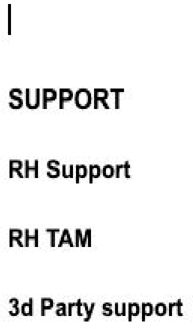
- Upgrade

Version

- Upgrade
- Rollback
- Co-existance

3d PARTY

Add all relevant third party tools and components

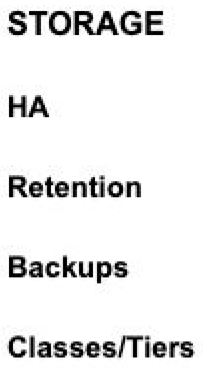


SUPPORT

RH Support

RH TAM

3d Party support



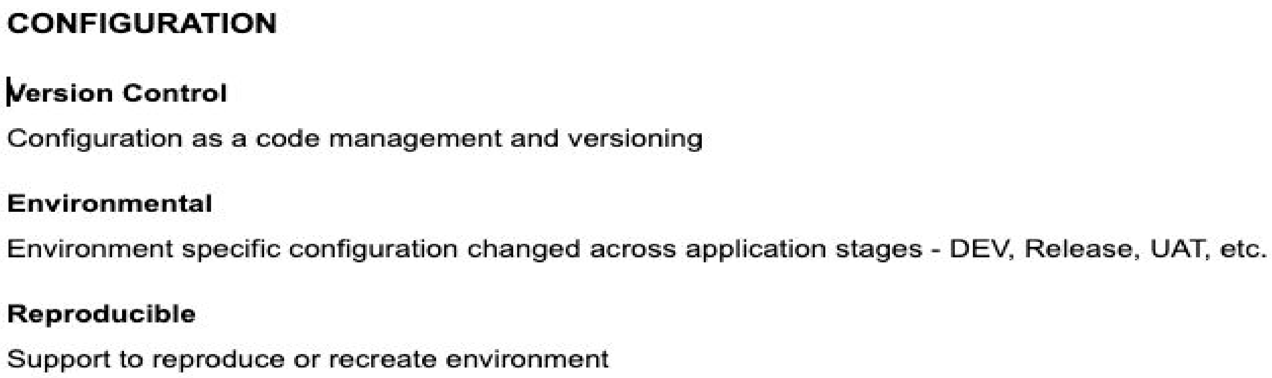
STORAGE

HA

Retention

Backups

Classes/Tiers



CONFIGURATION

Version Control

Configuration as a code management and versioning

Environmental

Environment specific configuration changed across application stages - DEV, Release, UAT, etc.

Reproducible

Support to reproduce or recreate environment

TESTING

HA

DR

Performance

Penetration

Security

Capacity

Performance baseline

Planning

Contingency

Limits and quotas

Bottlenecks

Application Architecture

Storage use cases

Databases

Services mesh

Logging

Health/Readiness

Scalability

Integration Patterns

- Messaging
- Pub/Sub
- External end-points
- Transformation
- Orchestration

SECURITY

Encrypt at rest

Compliance

PKI

- Key management

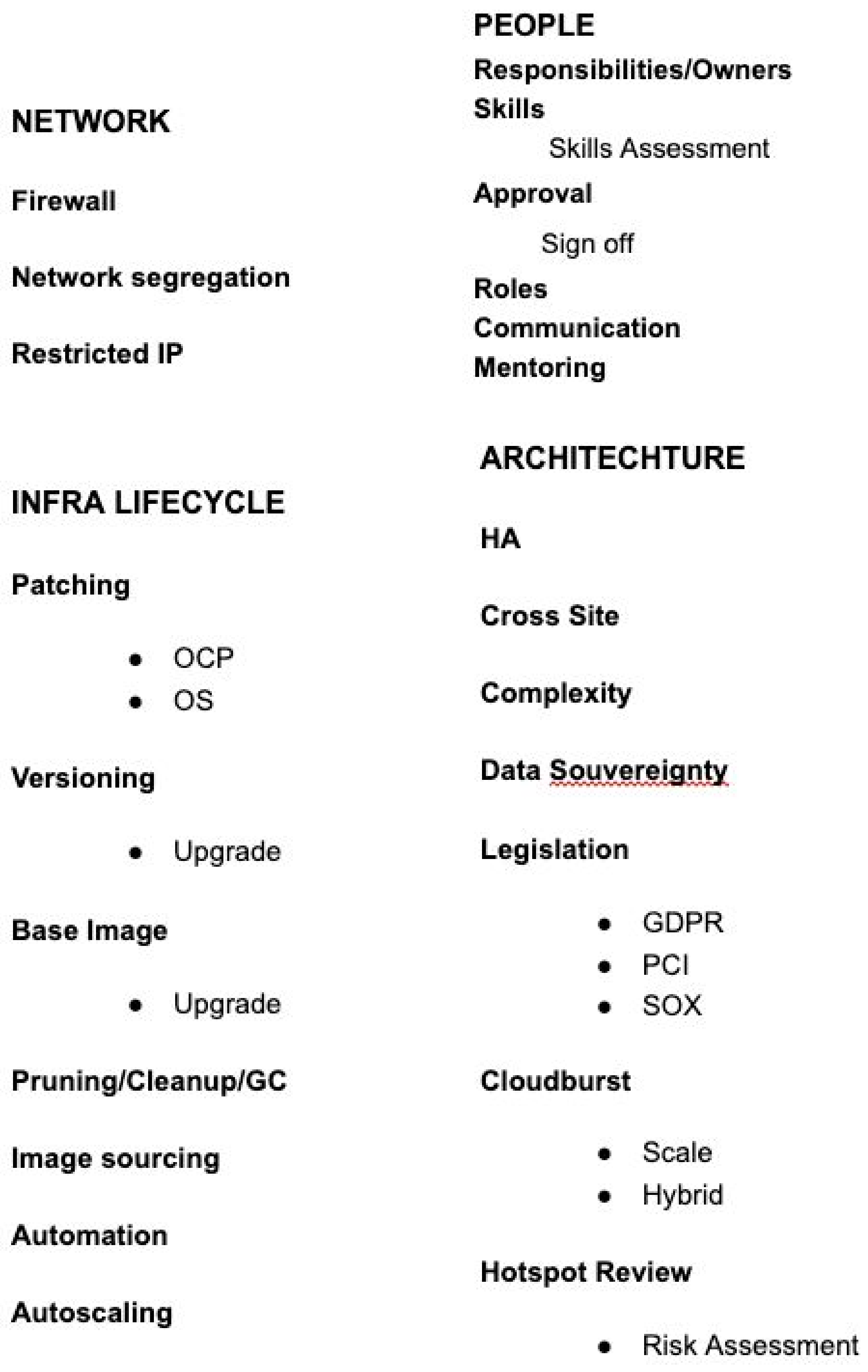
Access Control

Processes

- Tooling

Image Scanning

RBAC



NETWORK

Firewall

Network segregation

Restricted IP

INFRA LIFECYCLE

Patching

- OCP
- OS

Versioning

- Upgrade

Base Image

- Upgrade

Pruning/Cleanup/GC

Image sourcing

Automation

Autoscaling

PEOPLE

Responsibilities/Owners

Skills

Skills Assessment

Approval

Sign off

Roles

Communication

Mentoring

ARCHITECHTURE

HA

Cross Site

Complexity

Data Souvereignty

Legislation

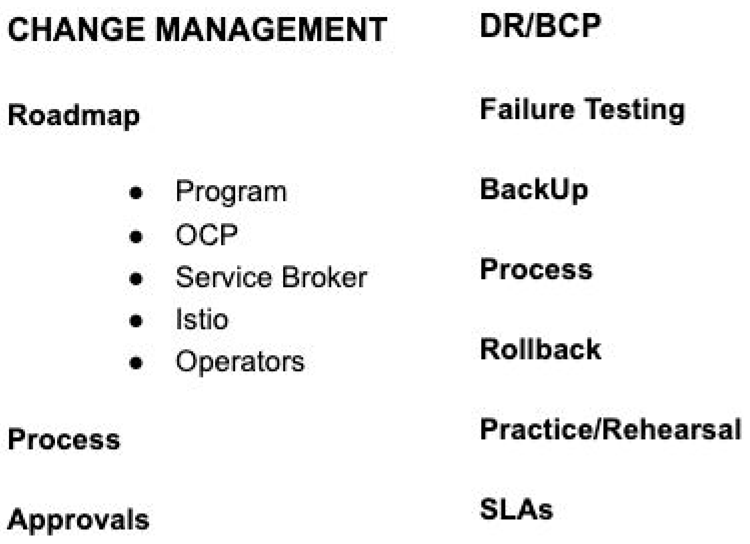
- GDPR
- PCI
- SOX

Cloudburst

- Scale
- Hybrid

Hotspot Review

- Risk Assessment



CHANGE MANAGEMENT

Roadmap

- Program
- OCP
- Service Broker
- Istio
- Operators

Process

Approvals

DR/BCP

Failure Testing

BackUp

Process

Rollback

Practice/Rehearsal

SLAs