Hosted Engine deep dive

Everything you always wanted to know about oVirt Hosted Engine (*but were afraid to ask)*

Simone Tiraboschi, Principal Software Engineer

January 2019

AGENDA

- **Brief getting started**
  - Packages
  - Services
- **Hosted engine deployment:**
  - Vintage flow
  - New node 0 flow
  - Cleanup
  - UI - cockpit
- **HE Maintenance types: local/global**
- **Technical details and internals**
- **Logs and troubleshooting:**
  - Backup and restore
  - Major issues
- **Q&A ???**

Getting started

- Hosted engine is the simplest way to ensure **HA capabilities** for oVirt engine/RHV manager
- The engine is installed on a VM (it saves hosts for virt purposes)
- The engine VM runs on hosts managed by the engine
- HE involved hosts can also run regular VMs
- Dedicated services on HE hosts will take care to keep the engine up

Related packages

- **ovirt-engine-appliance/rhvm-appliance**: it provides an up to date appliance with engine rpms
- **ovirt-hosted-engine-setup**: provides the setup and CLI utility
- **ovirt-hosted-engine-ha**: provides ha services:
    - **ovirt-ha-agent**:
        - Monitors local host state, engine VM status
        - Takes action if needed to ensure high availability
    - **ovirt-ha-broker:**
        - Liason between ovirt-ha-agent and:
            - Shared storage (metadata)
            - Local host status (monitoring)
        - Serializes requests
        - Separate, testable entity distinct from ovirt-ha-agent

ovirt-ha-broker

- Used by ovirt-ha-agent to read to/write from storage
- Has a set of monitors for host status:
  - **Ping** (gateway)
  - **CPU load**
  - **Memory** use
  - **Management** network bridge status
  - **Engine VM status** (at virt level)
  - **Engine status** (via http request to an health page)
- Listening socket: /var/run/ovirt-hosted-engine-ha/broker.socket

ovirt-ha-agent

- Ensures ovirt-engine VM high availability
- Uses configuration file written by setup (/etc/ovirt-hosted-engine/hosted-engine.conf):
  - Host id, storage config, gateway address to monitor, ...
- **If Engine VM is not running, it's started**
- **If Engine is non-responsive, VM is restarted**
- VM status read from vdsm getVmStats verb
- Engine status via engine liveness page: http://<engine-fqdn>/OvirtEngineWeb/HealthStatus

ovirt-ha-agent - host score

- **Single number** (scalar) representing a host's suitability for running the engine VM
- Range is 0 (unsuitable) to 3400 (all is well)
- May change
- Calculated based on host status: **each monitor** (ping, cpu load, gateway status, ...) has a **weight** and **contributes** to the score
- If VM is starting, starts on host with highest score If startup fails, host score is temporarily reduced, allows other hosts to try starting the VM
- If VM is running, and a different host has much higher score, VM is migrated to the better host
- Current migration/restart threshold is 800 points E.g. gateway failure will trigger migration, cpu load won't...

ovirt-ha-agent - host score penalties

https://github.com/oVirt/ovirt-hosted-engine-ha/blob/master/ovirt_hosted_engine_ha/agent/agent.conf

```
[score]
# NOTE: These values must be the same for all hosts in the HA cluster!
base-score=3400
gateway-score-penalty=1600
not-uptodate-config-penalty=1000
mgmt-bridge-score-penalty=600
free-memory-score-penalty=400
cpu-load-score-penalty=1000
engine-retry-score-penalty=50
cpu-load-penalty-min=0.4
cpu-load-penalty-max=0.9
```

ovirt-ha-agent - host score penalties

https://github.com/oVirt/ovirt-hosted-engine-ha/blob/master/ovirt_hosted_engine_ha/agent/states.py

Penalties are proportionally applied:

```
# Penalty is normalized to [0, max penalty] and is linear based on
# (magnitude of value within penalty range) / (size of range)
penalty = int(
    (load_average - score_cfg['cpu-load-penalty-min']) /
    (
        score_cfg['cpu-load-penalty-max'] -
        score_cfg['cpu-load-penalty-min']
    ) *
    score_cfg['cpu-load-score-penalty']
)
penalty = max(0, min(score_cfg['cpu-load-score-penalty'],
                     penalty))
```

ovirt-ha-agent - migrate/restart

https://github.com/oVirt/ovirt-hosted-engine-ha/blob/master/ovirt_hosted_engine_ha/agent/states.py

If a different host has a score which is significatively best, the engine VM got shut down on the first host and restarted on the second one (**cold migration!**)

```
if (new_data.best_score_host and
        new_data.best_score_host["host-id"] != new_data.host_id and
        new_data.best_score_host["score"] >= self.score(logger) +
        self.MIGRATION_THRESHOLD_SCORE):
    logger.error("Host %s (id %d) score is significantly better"
                 " than local score, shutting down VM on this host",
                 new_data.best_score_host['hostname'],
                 new_data.best_score_host["host-id"])
    return EngineStop(new_data)
```

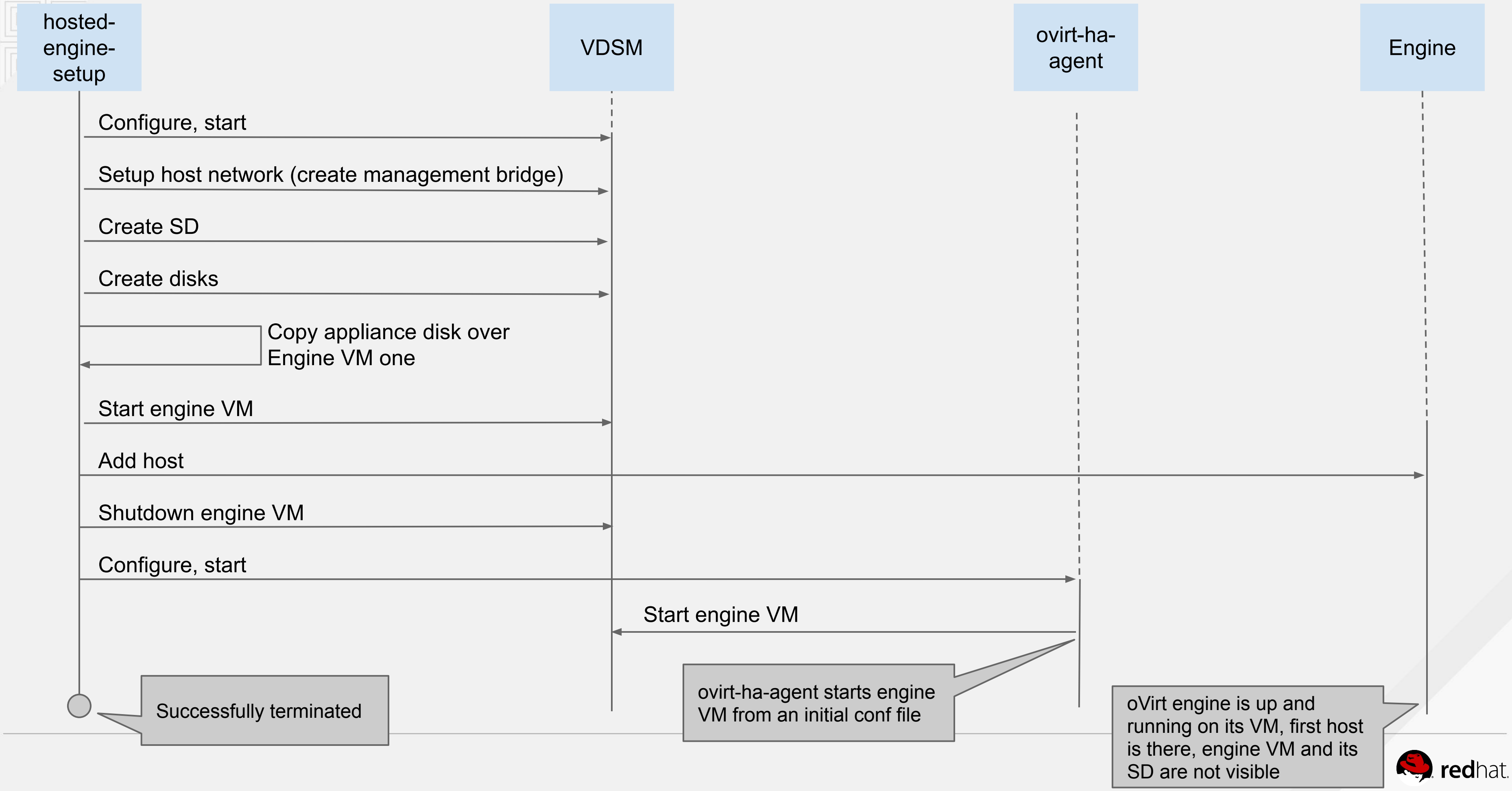
Local maintenance triggers a **live migration** instead

ovirt-ha-agent

- Only live hosts are considered for **startup/migration**... (let’s see maintenance mode...)
- If a **host hasn't updated** its metadata in a short while, it is **considered dead**
- **Startup algorithm** is eager/optimistic: if two hosts have same score, both will try to start VM
- **Sanlock** (volume lease) will allow **only one** to succeed
- **Race can** also **happen** due to hosts not seeing metadata updates at the same time

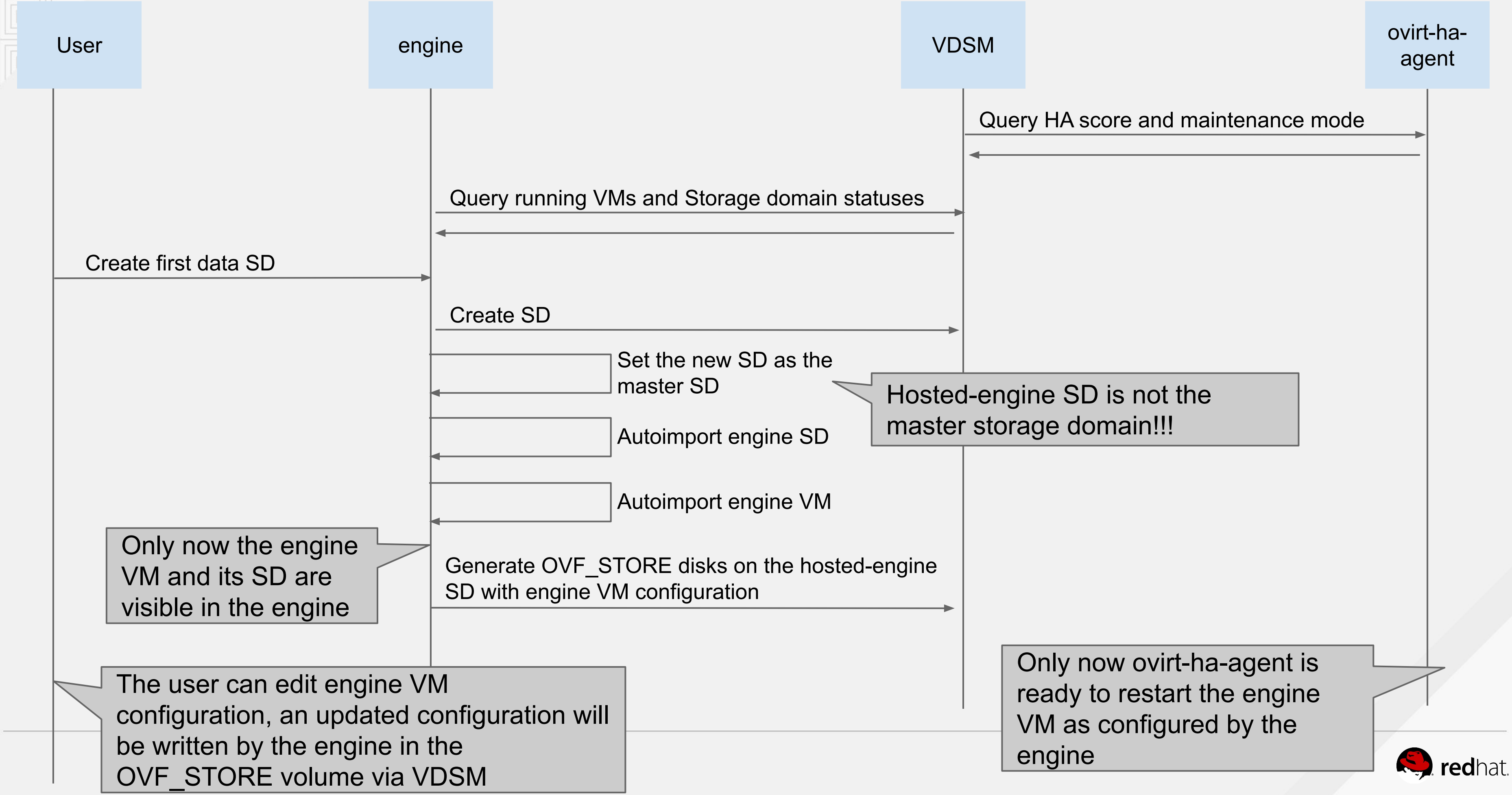
Deployment

Vintage flow: <=4.1, deprecated in 4.2, removed in 4.3



hosted-engine-setup
VDSM
ovirt-ha-agent
Engine
Configure, start
Setup host network (create management bridge)
Create SD
Create disks
Copy appliance disk over Engine VM one
Start engine VM
Add host
Shutdown engine VM
Configure, start
Start engine VM
Successfully terminated
ovirt-ha-agent starts engine VM from an initial conf file
oVirt engine is up and running on its VM, first host is there, engine VM and its SD are not visible

Vintage flow, part 2 - uncontrolled and up to the user



User

engine

VDSM

ovirt-ha-agent

Query HA score and maintenance mode

Query running VMs and Storage domain statuses

Create first data SD

Create SD

Set the new SD as the master SD

Hosted-engine SD is not the master storage domain!!!

Autoimport engine SD

Autoimport engine VM

Only now the engine VM and its SD are visible in the engine

Generate OVF_STORE disks on the hosted-engine SD with engine VM configuration

The user can edit engine VM configuration, an updated configuration will be written by the engine in the OVF_STORE volume via VDSM

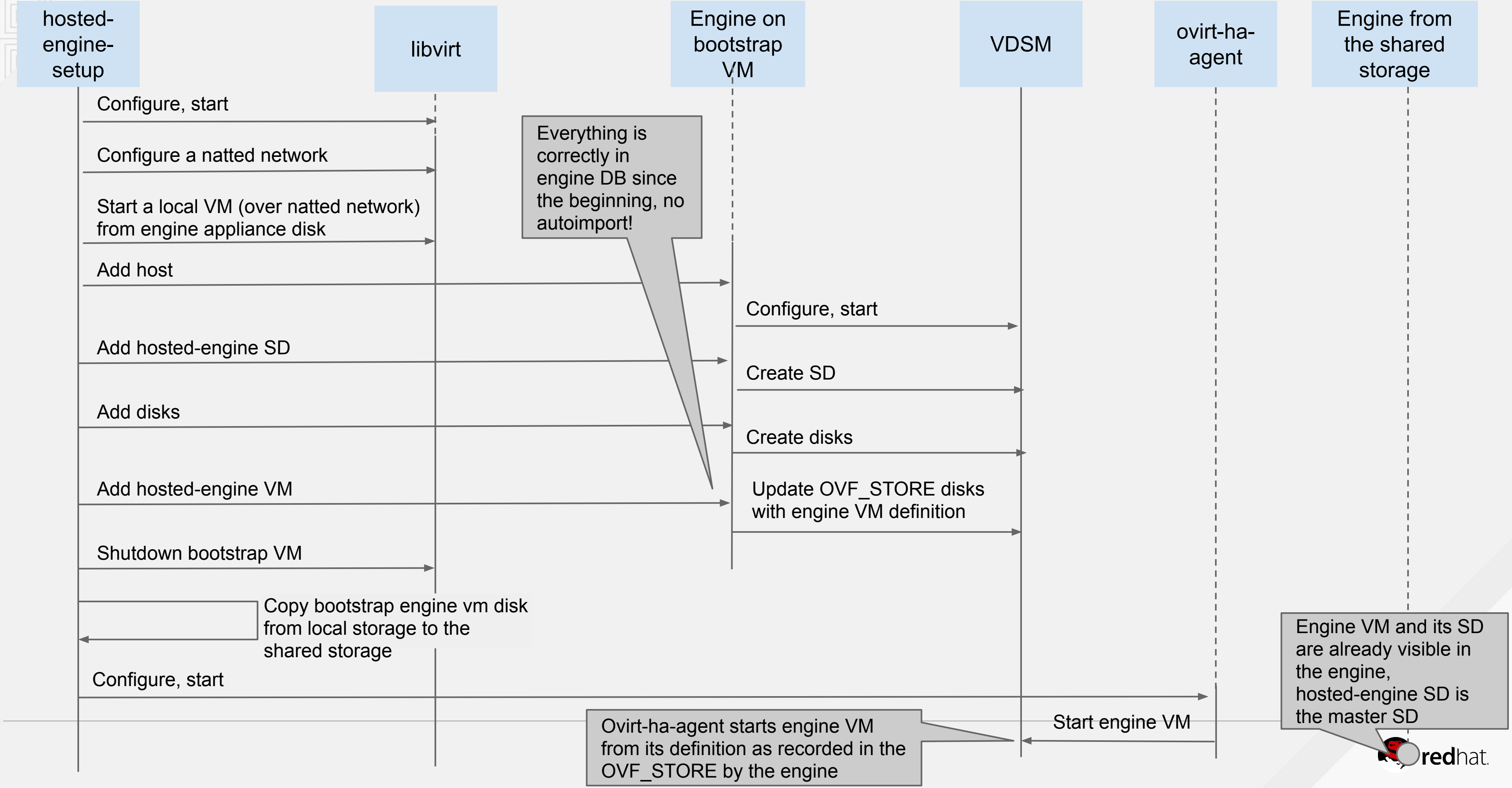
Only now ovirt-ha-agent is ready to restart the engine VM as configured by the engine

Current ansible based flow (node - 0): goals

- **Use standard engine flows, avoid re-implementing logic**
- Replace complicated hosted engine import code in engine

- **Keep all existing features**
- **Better input validation** - get rid of execute, fail, retry cycle
- Phase out OTOPI in favour of Ansible
- Keep **backward compatibility** with our CLI and Cockpit flows

Current ansible based flow (node - 0)



hosted-engine-setup

libvirt

Engine on bootstrap VM

VDSM

ovirt-ha-agent

Engine from the shared storage

Configure, start

Configure a natted network

Start a local VM (over natted network) from engine appliance disk

Everything is correctly in engine DB since the beginning, no autoimport!

Add host

Configure, start

Add hosted-engine SD

Create SD

Add disks

Create disks

Add hosted-engine VM

Update OVF_STORE disks with engine VM definition

Shutdown bootstrap VM

Copy bootstrap engine vm disk from local storage to the shared storage

Configure, start

Engine VM and its SD are already visible in the engine, hosted-engine SD is the master SD

Start engine VM

Ovirt-ha-agent starts engine VM from its definition as recorded in the OVF_STORE by the engine

Current ansible based flow (node - 0): benefits 1

- **Reverse** the vintage setup **flow** - **start ovirt-engine first** (instead of vdsm):
  - ovirt-engine already knows how to perform all the tasks we need, less bugs!
  - The bootstrap VM will use libvirt’s NATted network and local disk, deployment is simpler on networking side
  - /etc/hosts is configured to resolve all the necessary names properly

- Only collect information needed for the bootstrap ovirt-engine to start:
  1. Engine FQDN and credentials
  2. The first host’s (self) hostname

Current ansible based flow (node - 0): benefits 2

- Only once we have a locally running engine:
  1. Ask the user for storage connection data
  2. Immediate validation possible through running ovirt-engine

All flows use the standard engine and vdsm logic!

- No SPM ID issues
- No vdsm issues with “undocumented” APIs
- All engine DB records reflect the latest and greatest storage features

All VM features match standard VMs (memory hotplug, migration profiles…)

No autoimport code, no value guessing - all defined using standard flows

Summary

Vintage (deprecated) flow

1. CLI and cockpit based
2. Fully custom setup (OTOPI)
3. Time to finish: ~30 min
4. No upfront storage validation
5. Custom code and flows
6. HE specific VM configuration

Node 0 flow

1. CLI and cockpit based
2. Setup based on Ansible*
3. Time to finish: ~30 min
4. Storage connection validated
5. Standard code and flows
6. Standard HE VM configuration

* OTOPI wrapper left for backwards compatibility with answer files and so on

Cleanup

To cleanup a single HE host, from failed or successful deployments, the user can run:

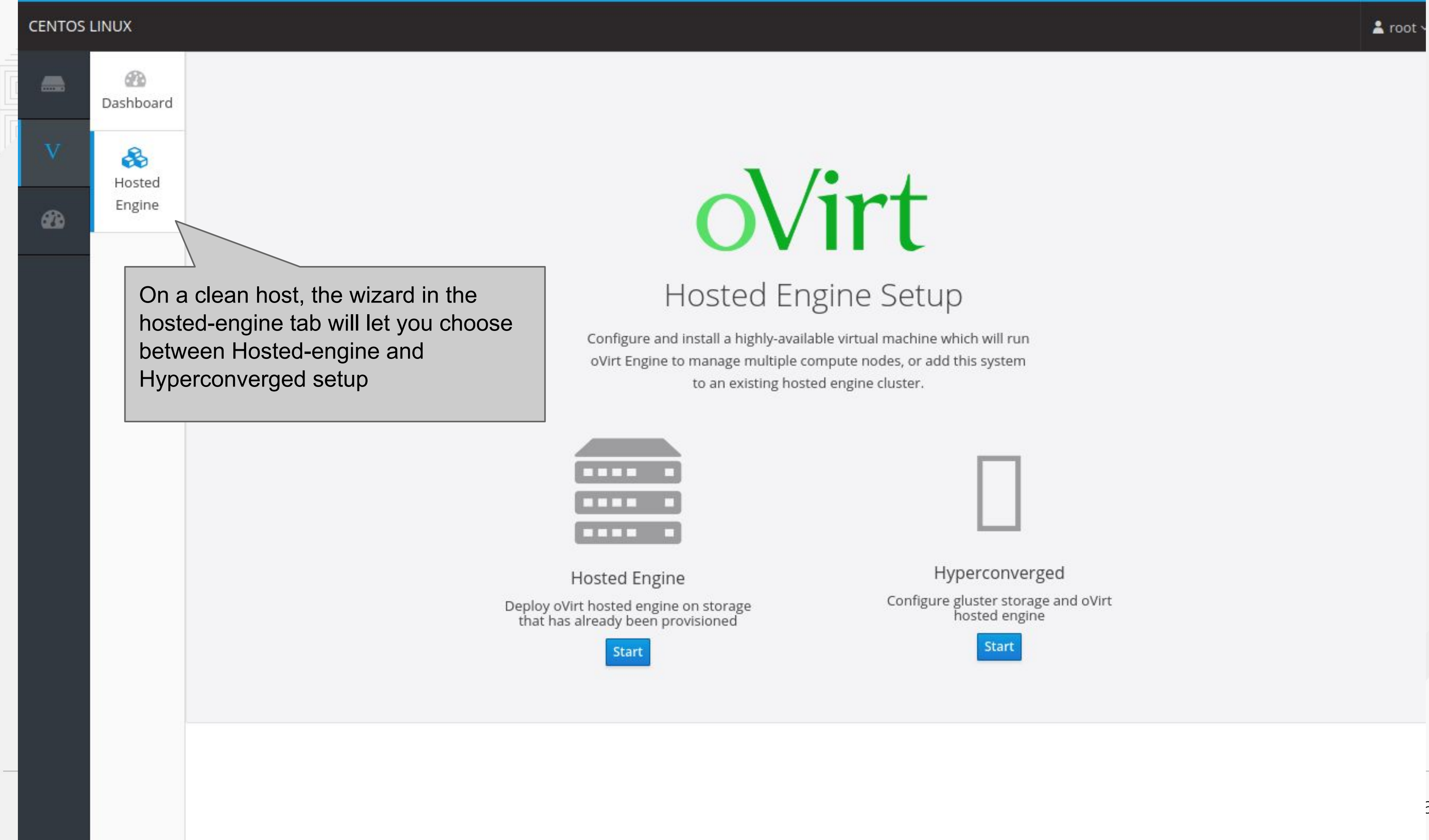
```
[root@c76he20190115h1 ~]#  ovirt-hosted-engine-cleanup
 This will de-configure the host to run ovirt-hosted-engine-setup from
scratch.
Caution, this operation should be used with care.

Are you sure you want to proceed? [y/n]
```

It will not touch at all the shared storage, it’s up to the user to clean that if needed

Deploying from cockpit

How it looks like



Hosted Engine Setup

Configure and install a highly-available virtual machine which will run oVirt Engine to manage multiple compute nodes, or add this system to an existing hosted engine cluster.

Hosted Engine

Deploy oVirt hosted engine on storage that has already been provisioned

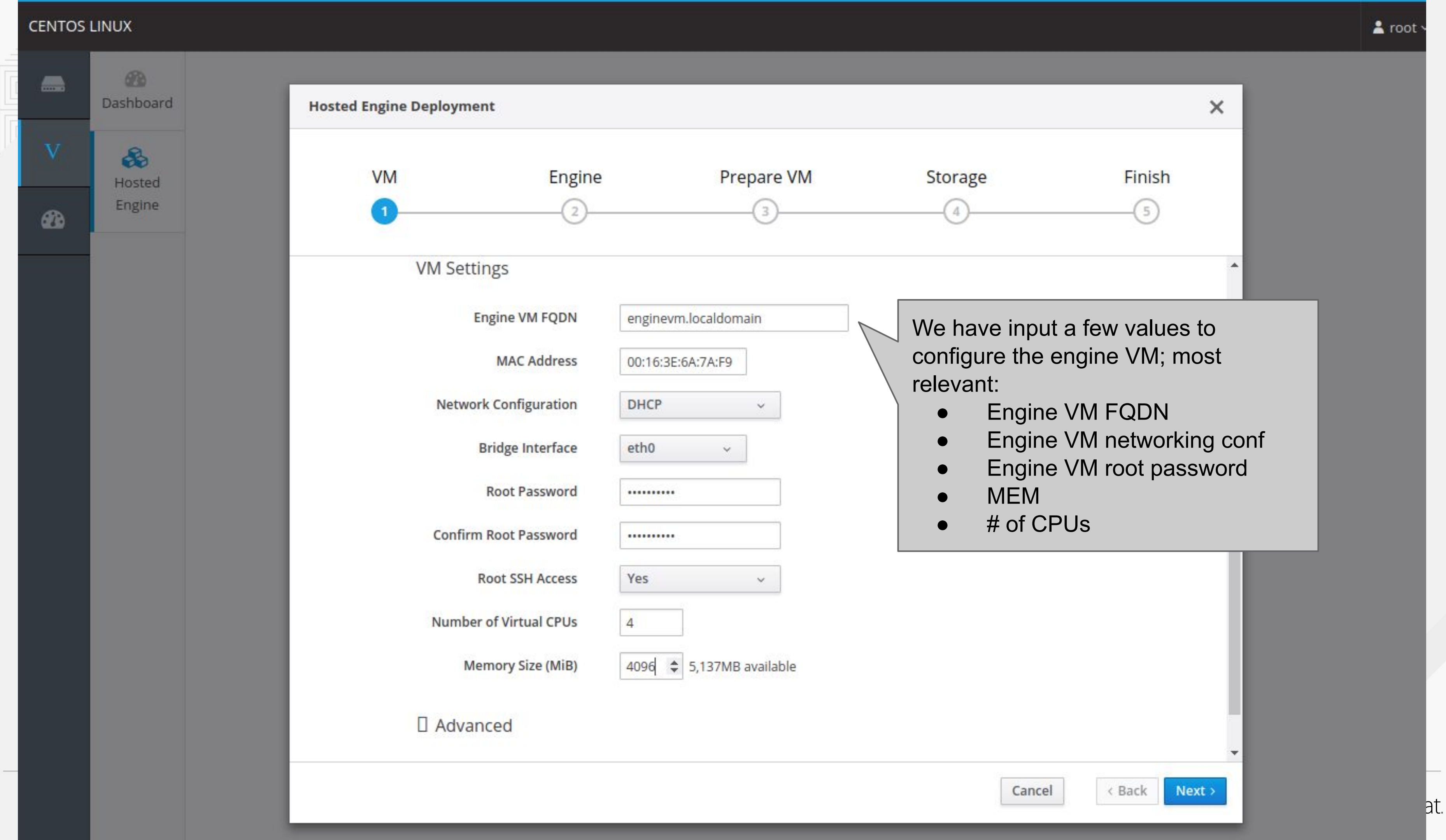
Start

Hyperconverged

Configure gluster storage and oVirt hosted engine

Start

On a clean host, the wizard in the hosted-engine tab will let you choose between Hosted-engine and Hyperconverged setup



Hosted Engine Deployment

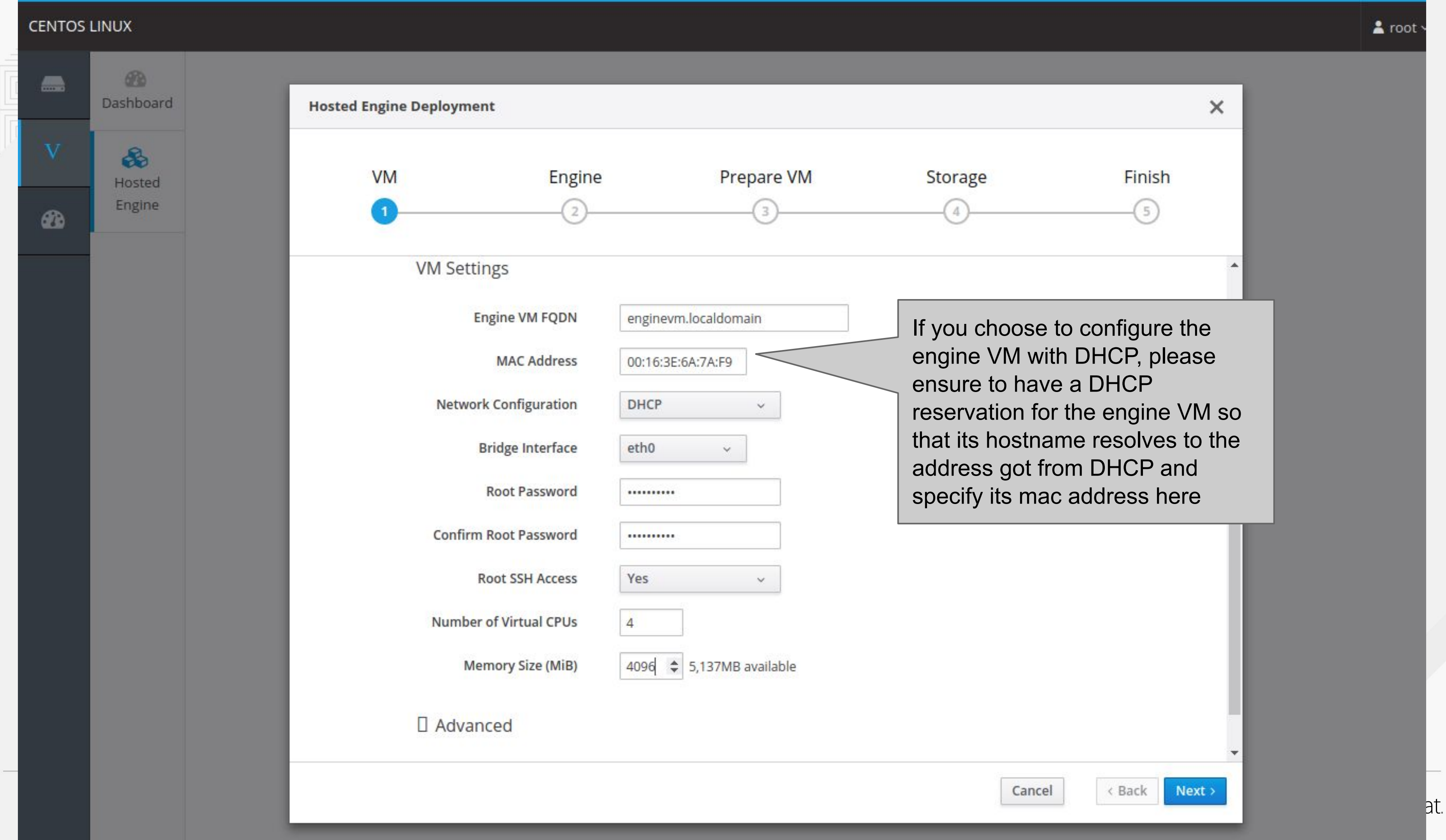
VM Settings

Engine VM FQDN	enginevm.localdomain
MAC Address	00:16:3E:6A:7A:F9
Network Configuration	DHCP
Bridge Interface	eth0
Root Password	••••••••••
Confirm Root Password	••••••••••
Root SSH Access	Yes
Number of Virtual CPUs	4
Memory Size (MiB)	4096 5,137MB available

□ Advanced

We have input a few values to configure the engine VM; most relevant:

- Engine VM FQDN
- Engine VM networking conf
- Engine VM root password
- MEM
- # of CPUs



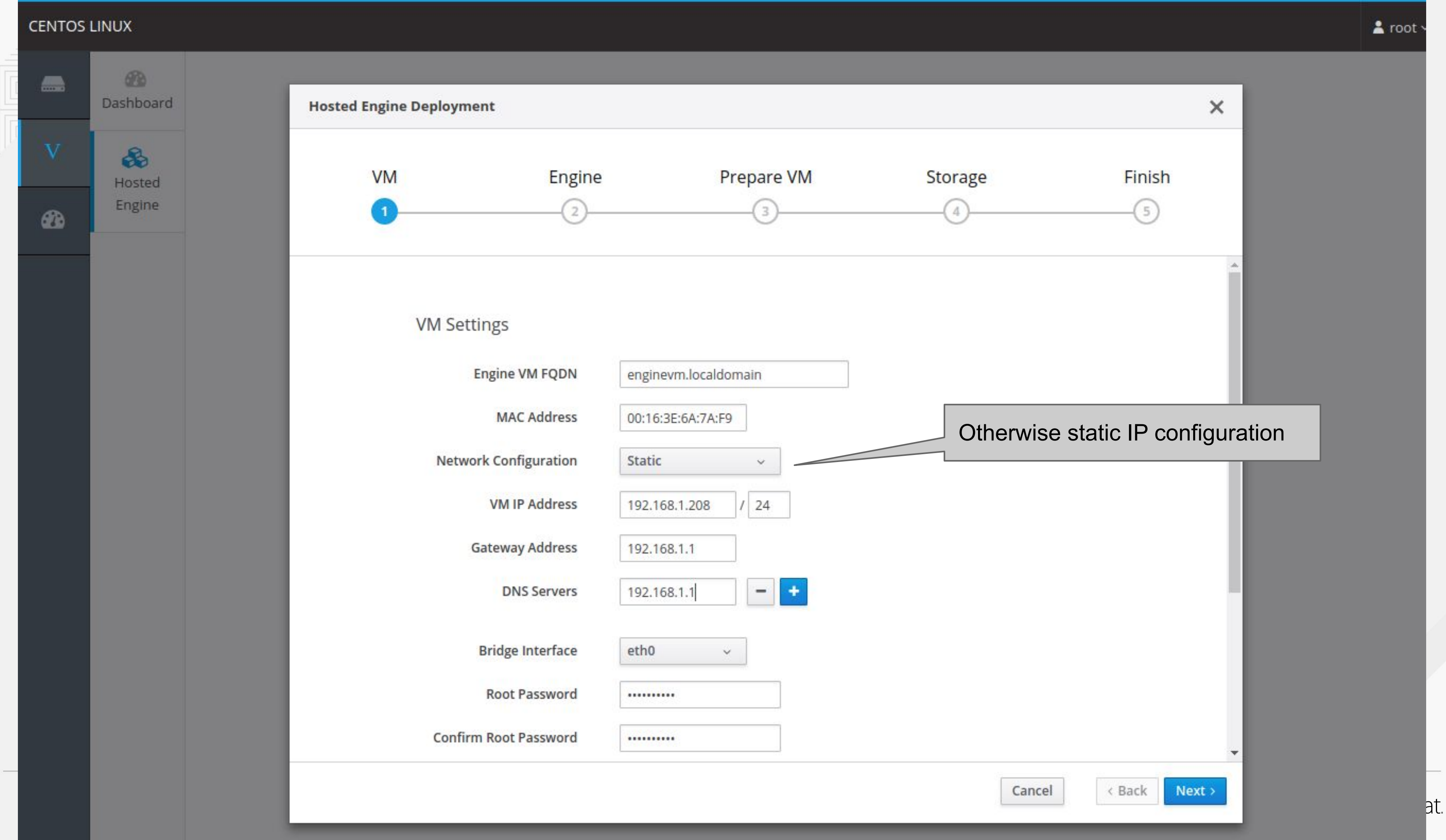
Hosted Engine Deployment

VM Settings

Engine VM FQDN	enginevm.localdomain
MAC Address	00:16:3E:6A:7A:F9
Network Configuration	DHCP
Bridge Interface	eth0
Root Password	••••••••••
Confirm Root Password	••••••••••
Root SSH Access	Yes
Number of Virtual CPUs	4
Memory Size (MiB)	4096 5,137MB available

□ Advanced

If you choose to configure the engine VM with DHCP, please ensure to have a DHCP reservation for the engine VM so that its hostname resolves to the address got from DHCP and specify its mac address here

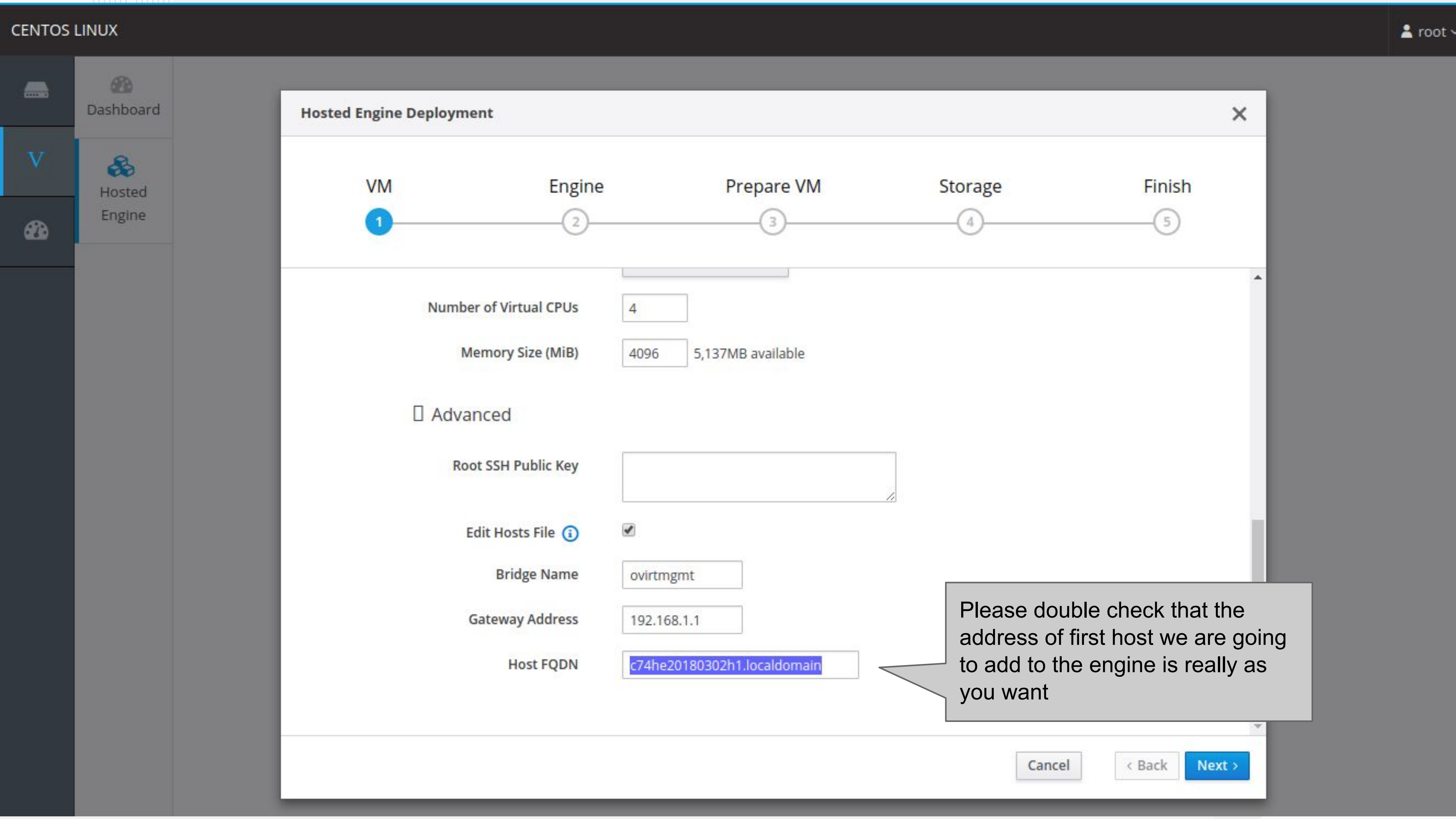


Hosted Engine Deployment

VM Settings

Engine VM FQDN	enginevm.localdomain
MAC Address	00:16:3E:6A:7A:F9
Network Configuration	Static
VM IP Address	192.168.1.208 / 24
Gateway Address	192.168.1.1
DNS Servers	192.168.1.1
Bridge Interface	eth0
Root Password	••••••••••
Confirm Root Password	••••••••••

Otherwise static IP configuration

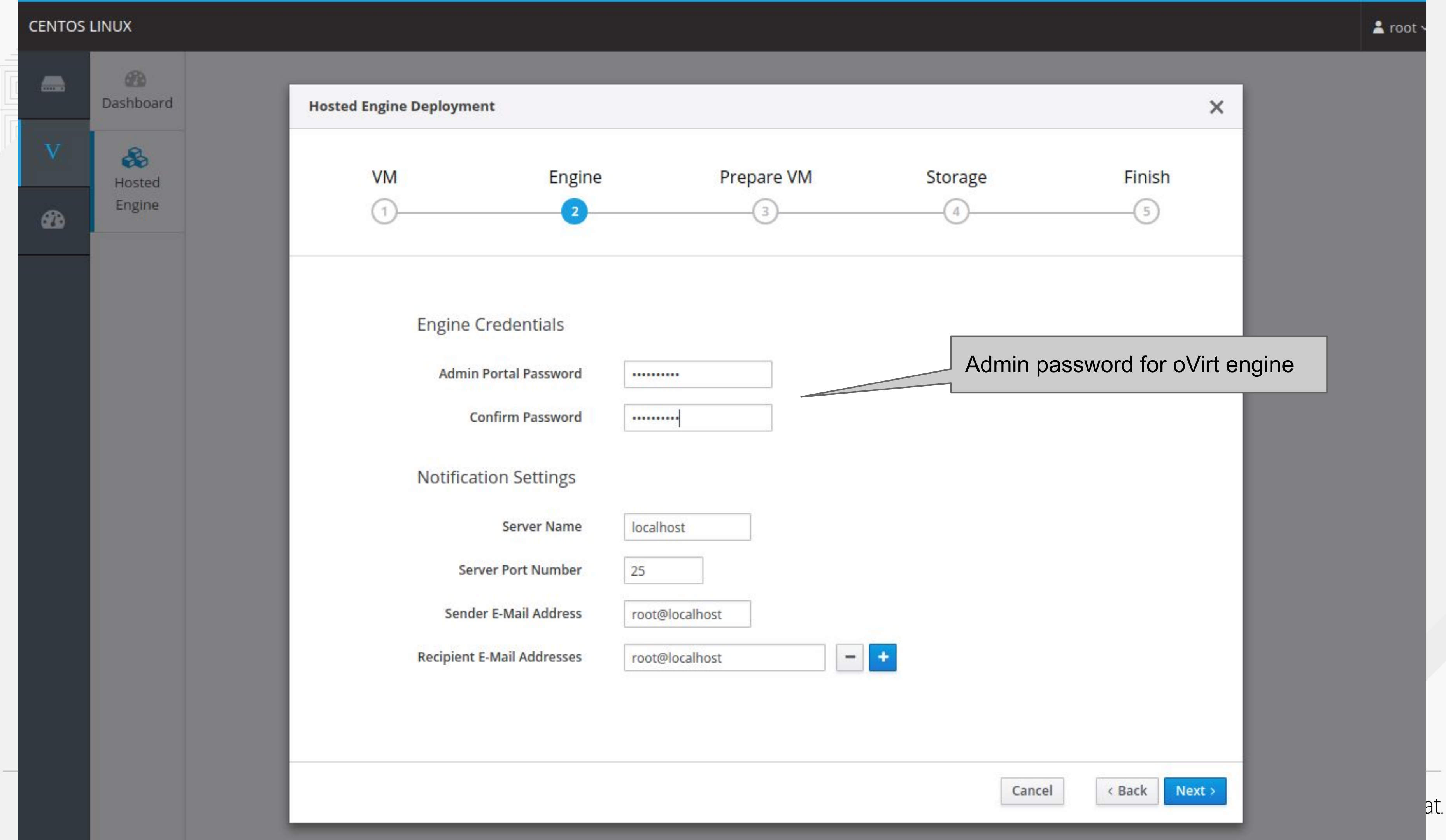


Number of Virtual CPUs	4
Memory Size (MiB)	4096 5,137MB available

□ Advanced

Root SSH Public Key	
Edit Hosts File ⓘ	☑
Bridge Name	ovirtmgmt
Gateway Address	192.168.1.1
Host FQDN	c74he20180302h1.localdomain

Please double check that the address of first host we are going to add to the engine is really as you want



Hosted Engine Deployment

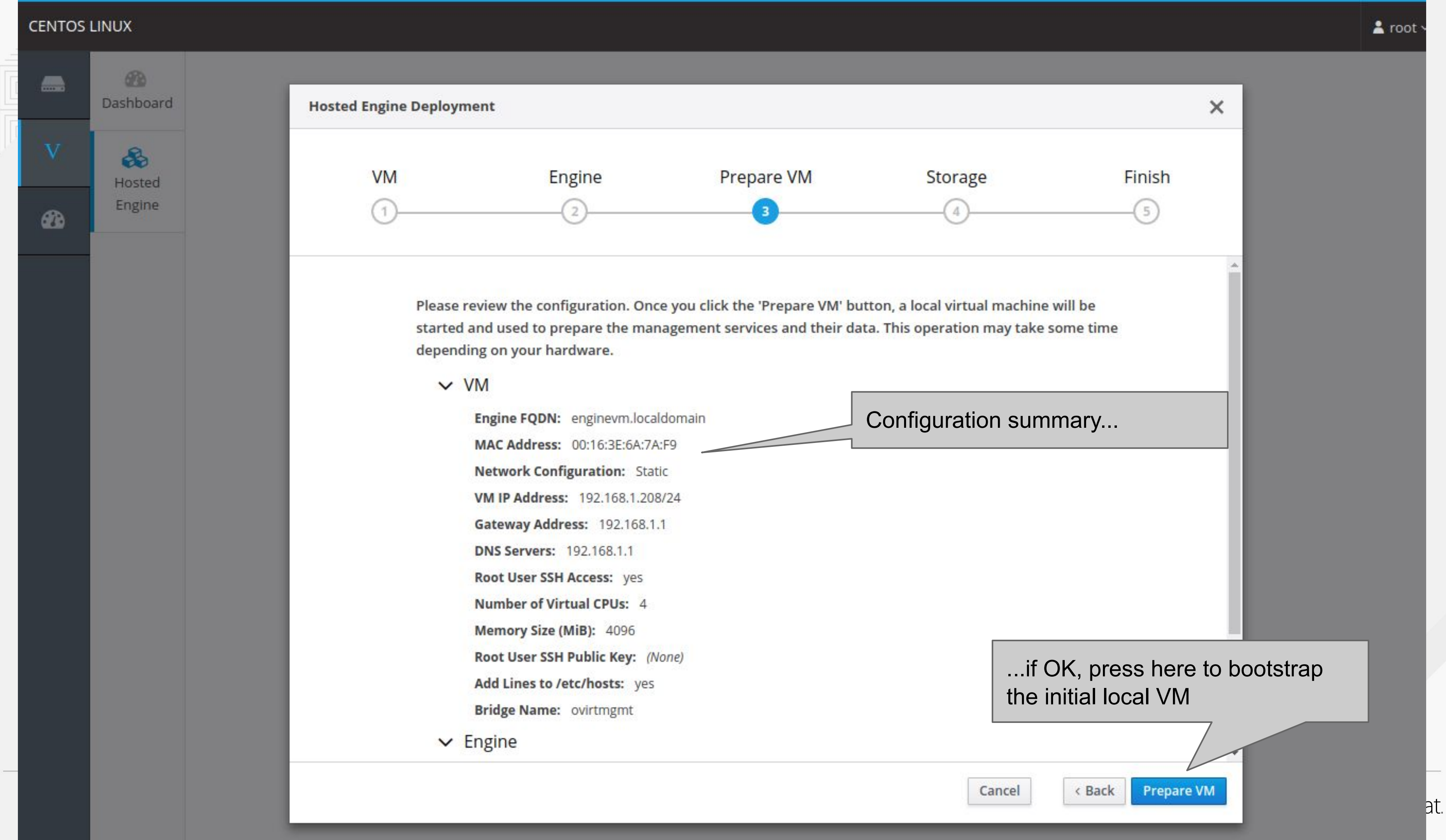
Engine Credentials

Admin Portal Password	••••••••••
Confirm Password	••••••••••

Notification Settings

Server Name	localhost
Server Port Number	25
Sender E-Mail Address	root@localhost
Recipient E-Mail Addresses	root@localhost

Admin password for oVirt engine



Hosted Engine Deployment

Please review the configuration. Once you click the 'Prepare VM' button, a local virtual machine will be started and used to prepare the management services and their data. This operation may take some time depending on your hardware.

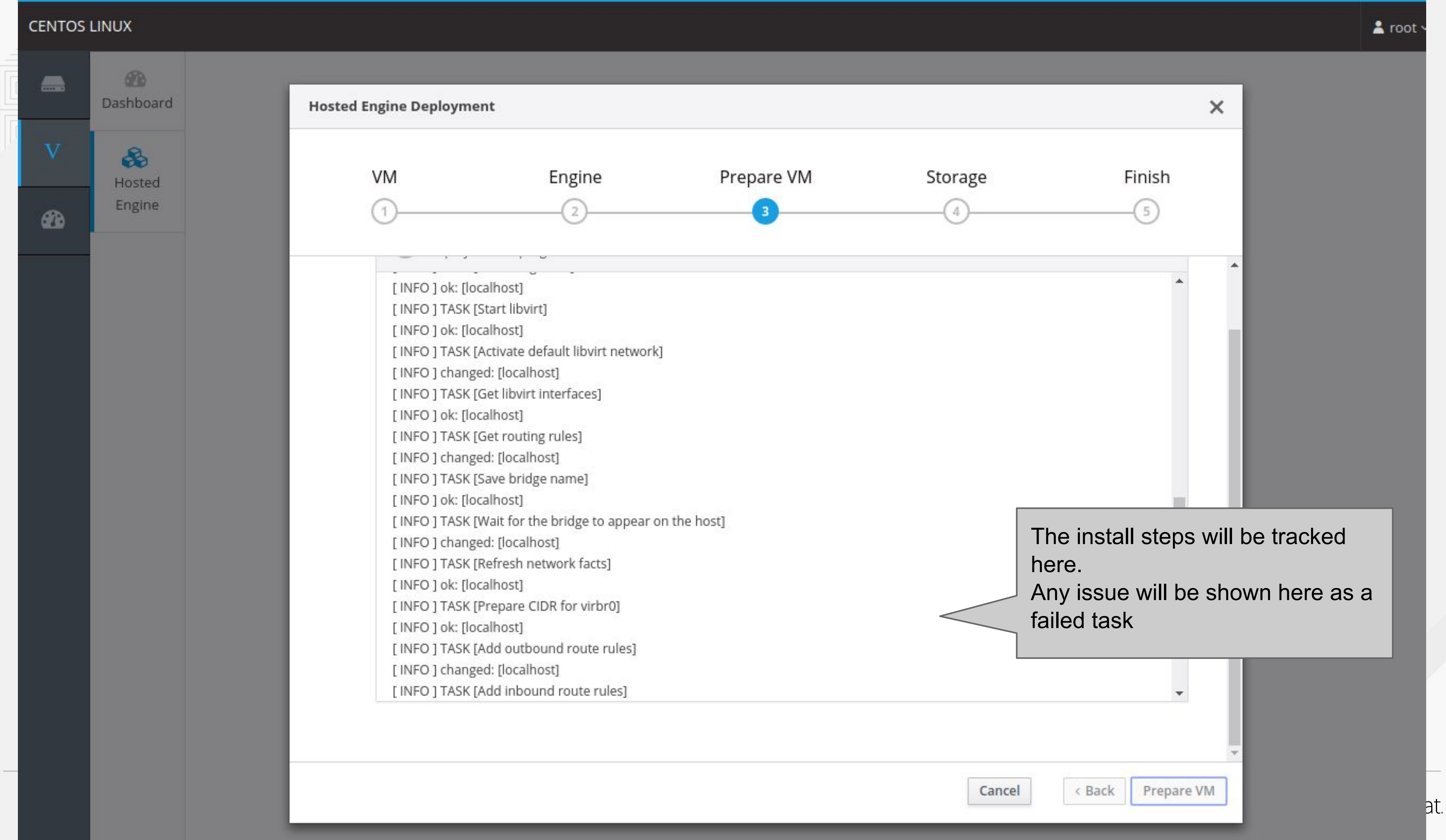
VM

Engine FQDN: enginevm.localdomain
MAC Address: 00:16:3E:6A:7A:F9
Network Configuration: Static
VM IP Address: 192.168.1.208/24
Gateway Address: 192.168.1.1
DNS Servers: 192.168.1.1
Root User SSH Access: yes
Number of Virtual CPUs: 4
Memory Size (MiB): 4096
Root User SSH Public Key: *(None)*
Add Lines to /etc/hosts: yes
Bridge Name: ovirtmgmt

Engine

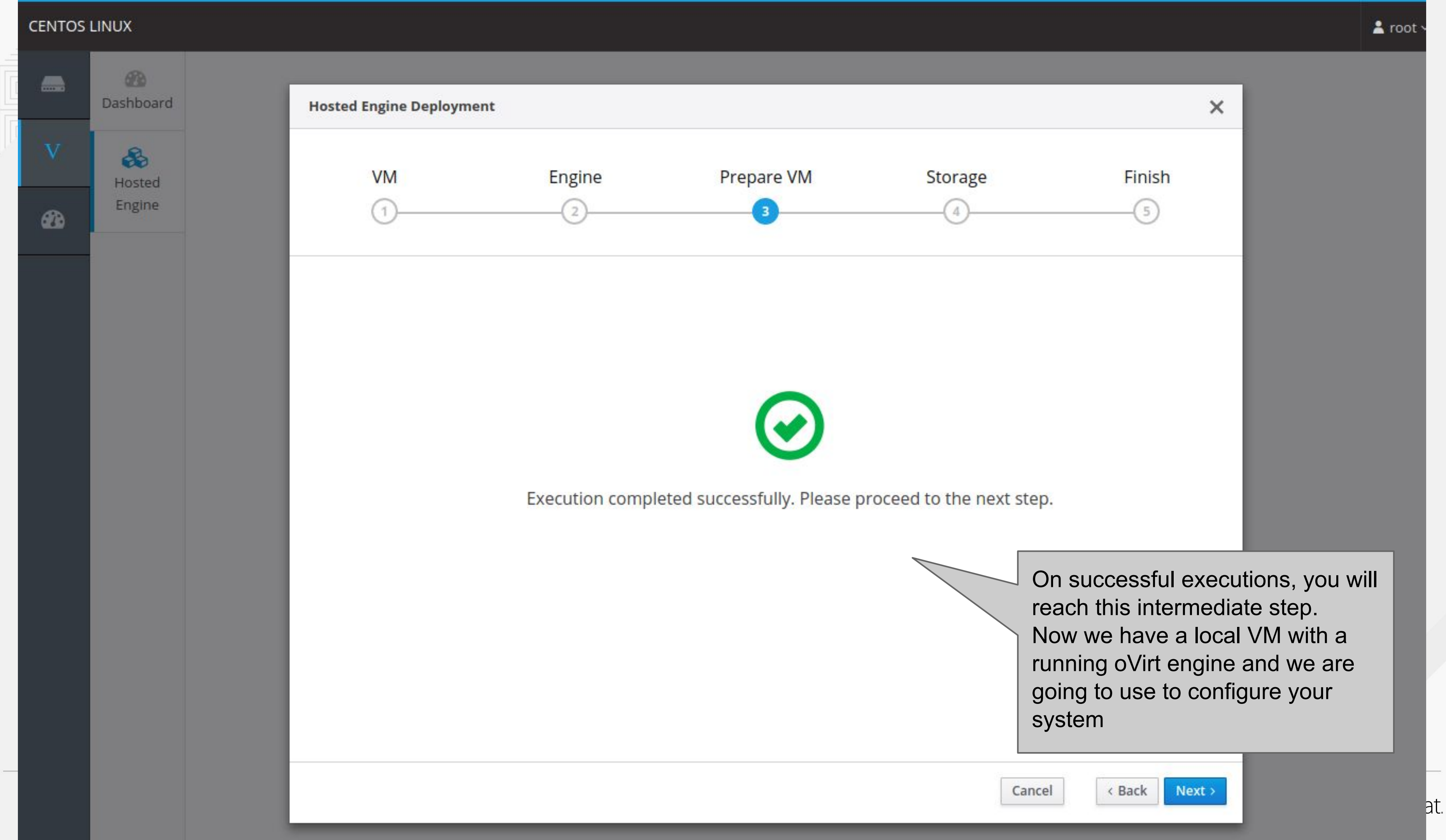
Configuration summary...

...if OK, press here to bootstrap the initial local VM



```
[ INFO ] ok: [localhost]
[ INFO ] TASK [Start libvirt]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Activate default libvirt network]
[ INFO ] changed: [localhost]
[ INFO ] TASK [Get libvirt interfaces]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Get routing rules]
[ INFO ] changed: [localhost]
[ INFO ] TASK [Save bridge name]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Wait for the bridge to appear on the host]
[ INFO ] changed: [localhost]
[ INFO ] TASK [Refresh network facts]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Prepare CIDR for virbr0]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Add outbound route rules]
[ INFO ] changed: [localhost]
[ INFO ] TASK [Add inbound route rules]
```

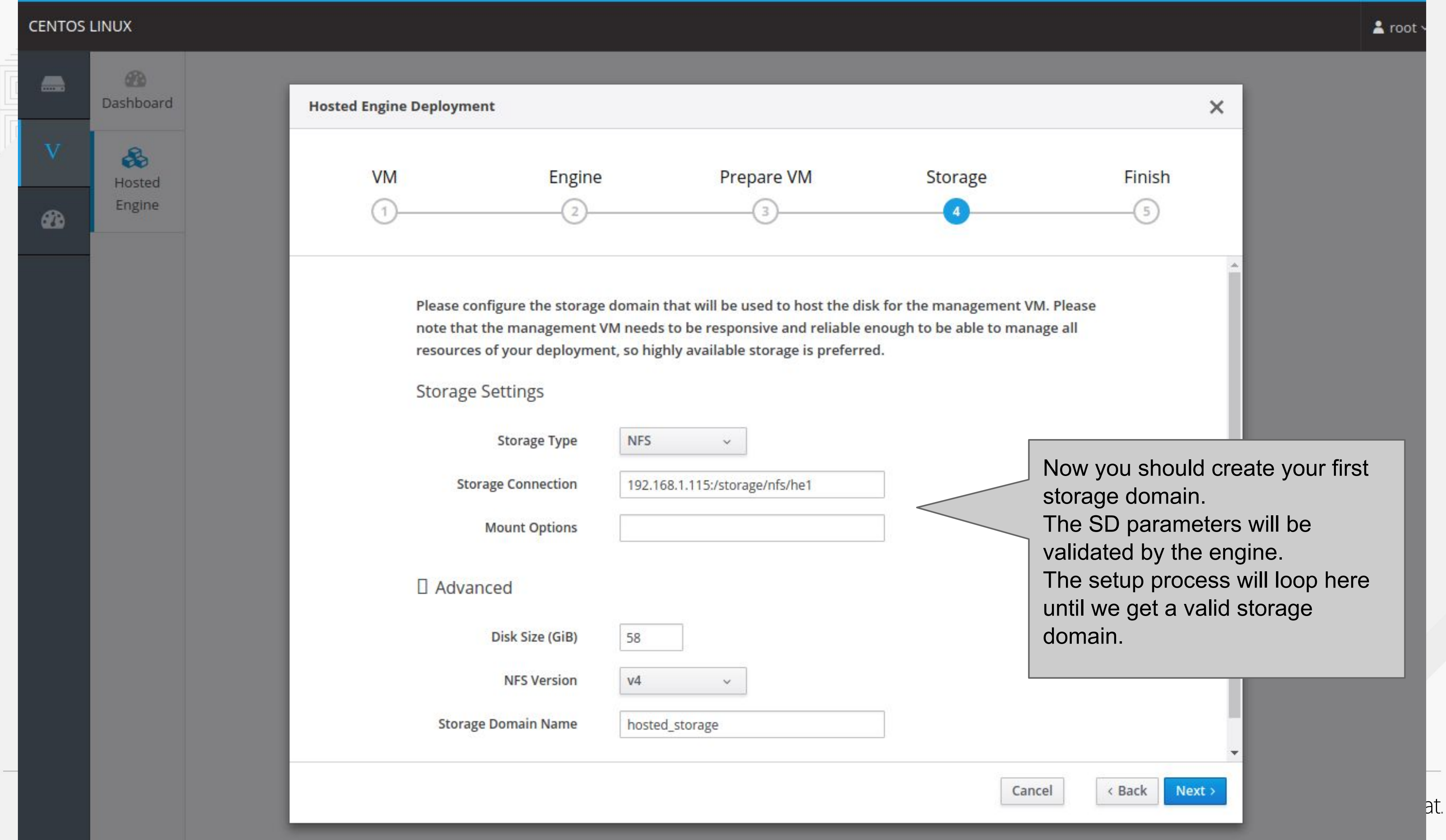
The install steps will be tracked here.
Any issue will be shown here as a failed task



Hosted Engine Deployment

Execution completed successfully. Please proceed to the next step.

On successful executions, you will reach this intermediate step. Now we have a local VM with a running oVirt engine and we are going to use to configure your system



Hosted Engine Deployment ✕

Please configure the storage domain that will be used to host the disk for the management VM. Please note that the management VM needs to be responsive and reliable enough to be able to manage all resources of your deployment, so highly available storage is preferred.

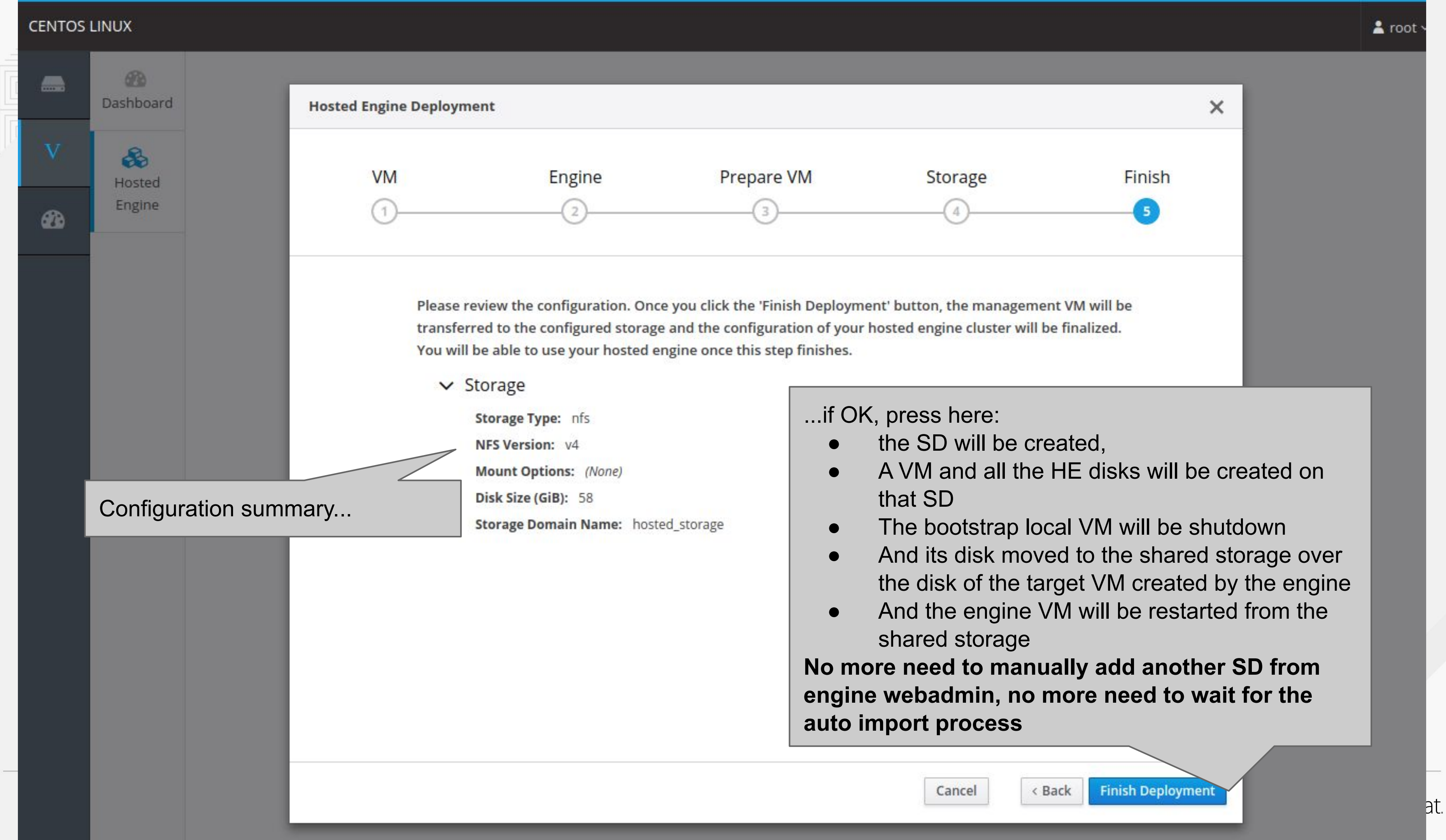
Storage Settings

Storage Type	NFS
Storage Connection	192.168.1.115:/storage/nfs/he1
Mount Options	

□ Advanced

Disk Size (GiB)	58
NFS Version	v4
Storage Domain Name	hosted_storage

Now you should create your first storage domain.
The SD parameters will be validated by the engine.
The setup process will loop here until we get a valid storage domain.



Hosted Engine Deployment

Please review the configuration. Once you click the 'Finish Deployment' button, the management VM will be transferred to the configured storage and the configuration of your hosted engine cluster will be finalized. You will be able to use your hosted engine once this step finishes.

Storage

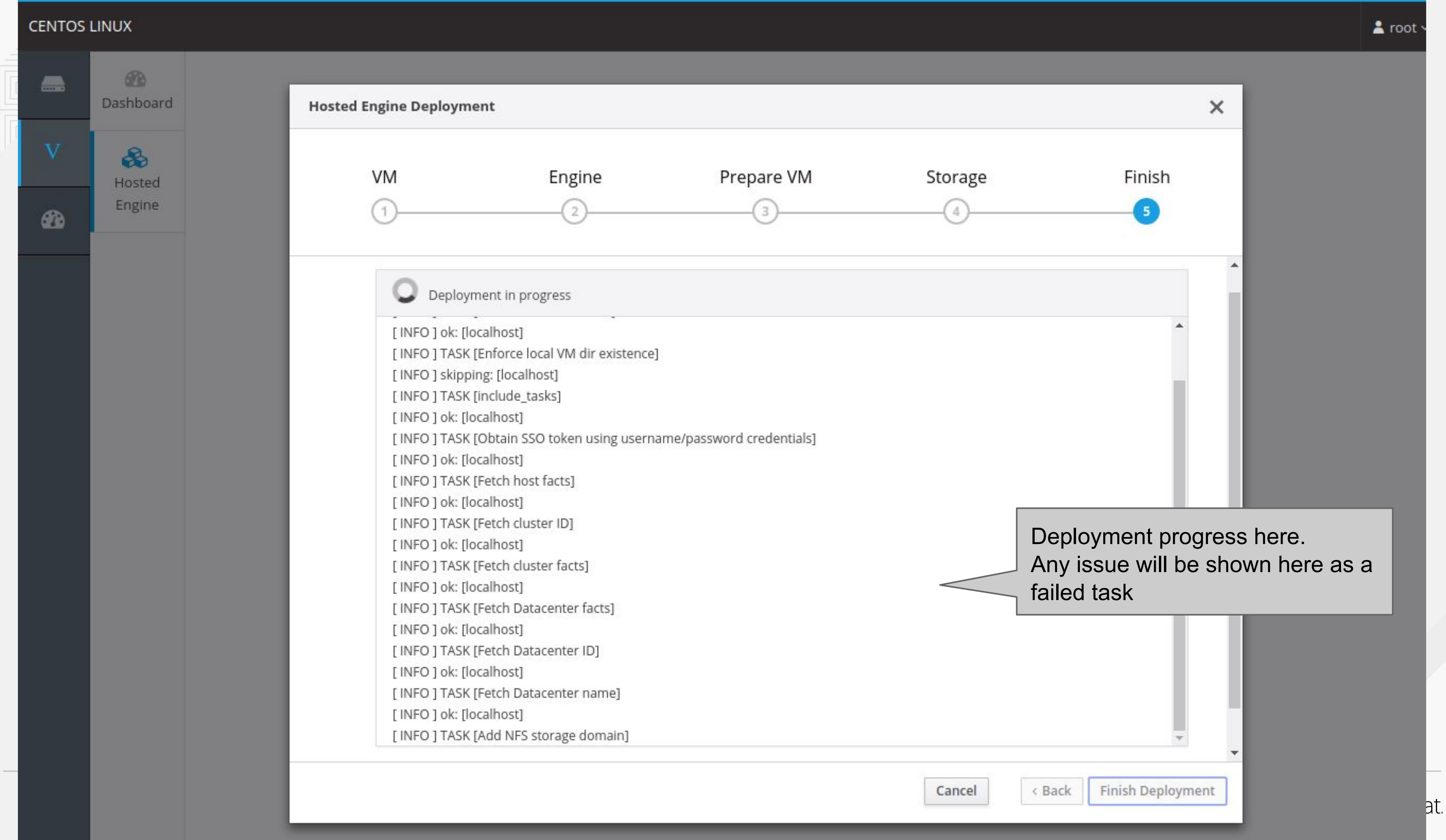
Storage Type:	nfs
NFS Version:	v4
Mount Options:	*(None)*
Disk Size (GiB):	58
Storage Domain Name:	hosted_storage

Configuration summary...

...if OK, press here:

- the SD will be created,
- A VM and all the HE disks will be created on that SD
- The bootstrap local VM will be shutdown
- And its disk moved to the shared storage over the disk of the target VM created by the engine
- And the engine VM will be restarted from the shared storage

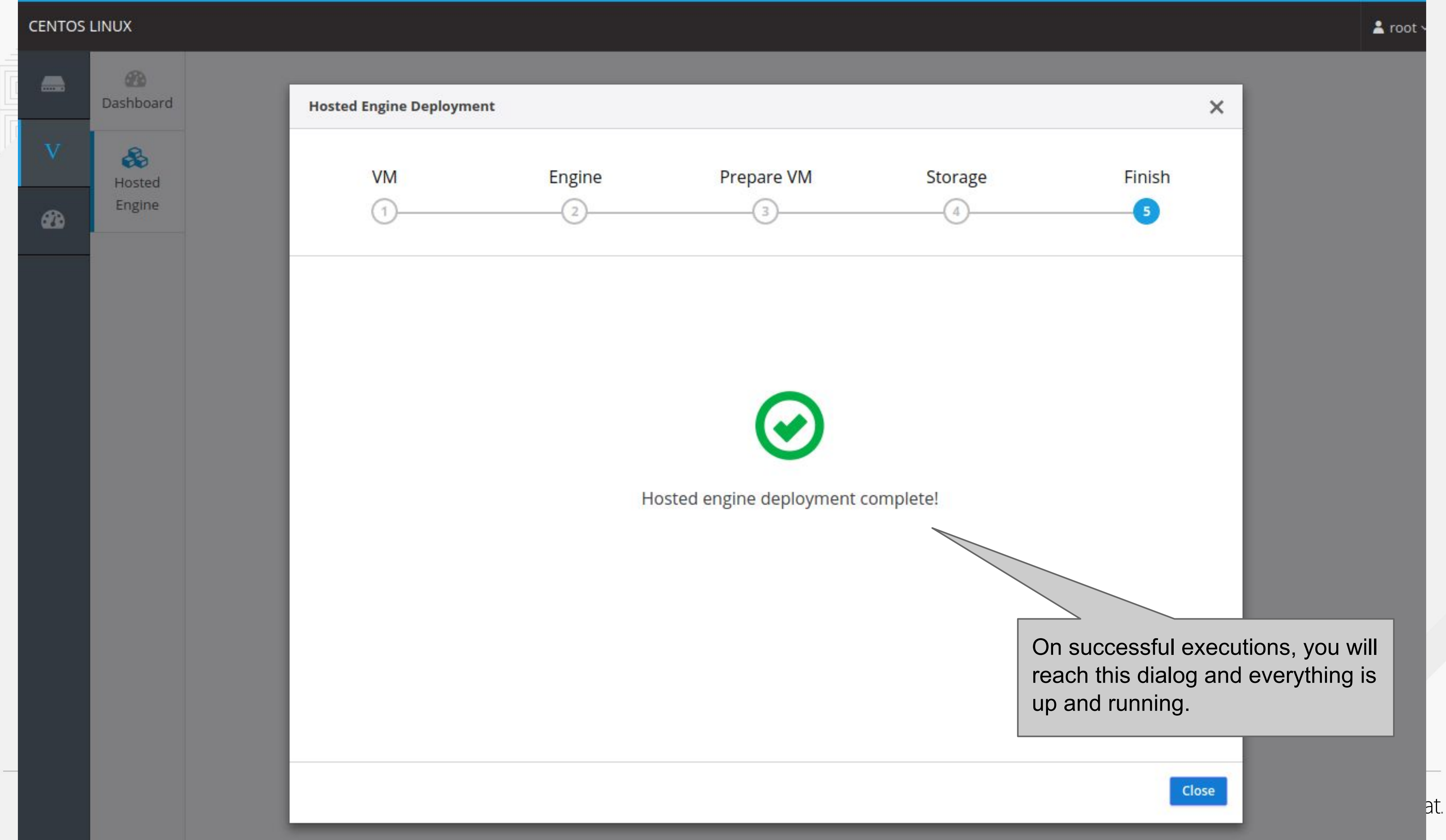
No more need to manually add another SD from engine webadmin, no more need to wait for the auto import process



Deployment in progress

```
[ INFO ] ok: [localhost]
[ INFO ] TASK [Enforce local VM dir existence]
[ INFO ] skipping: [localhost]
[ INFO ] TASK [include_tasks]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Obtain SSO token using username/password credentials]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch host facts]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch cluster ID]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch cluster facts]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch Datacenter facts]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch Datacenter ID]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Fetch Datacenter name]
[ INFO ] ok: [localhost]
[ INFO ] TASK [Add NFS storage domain]
```

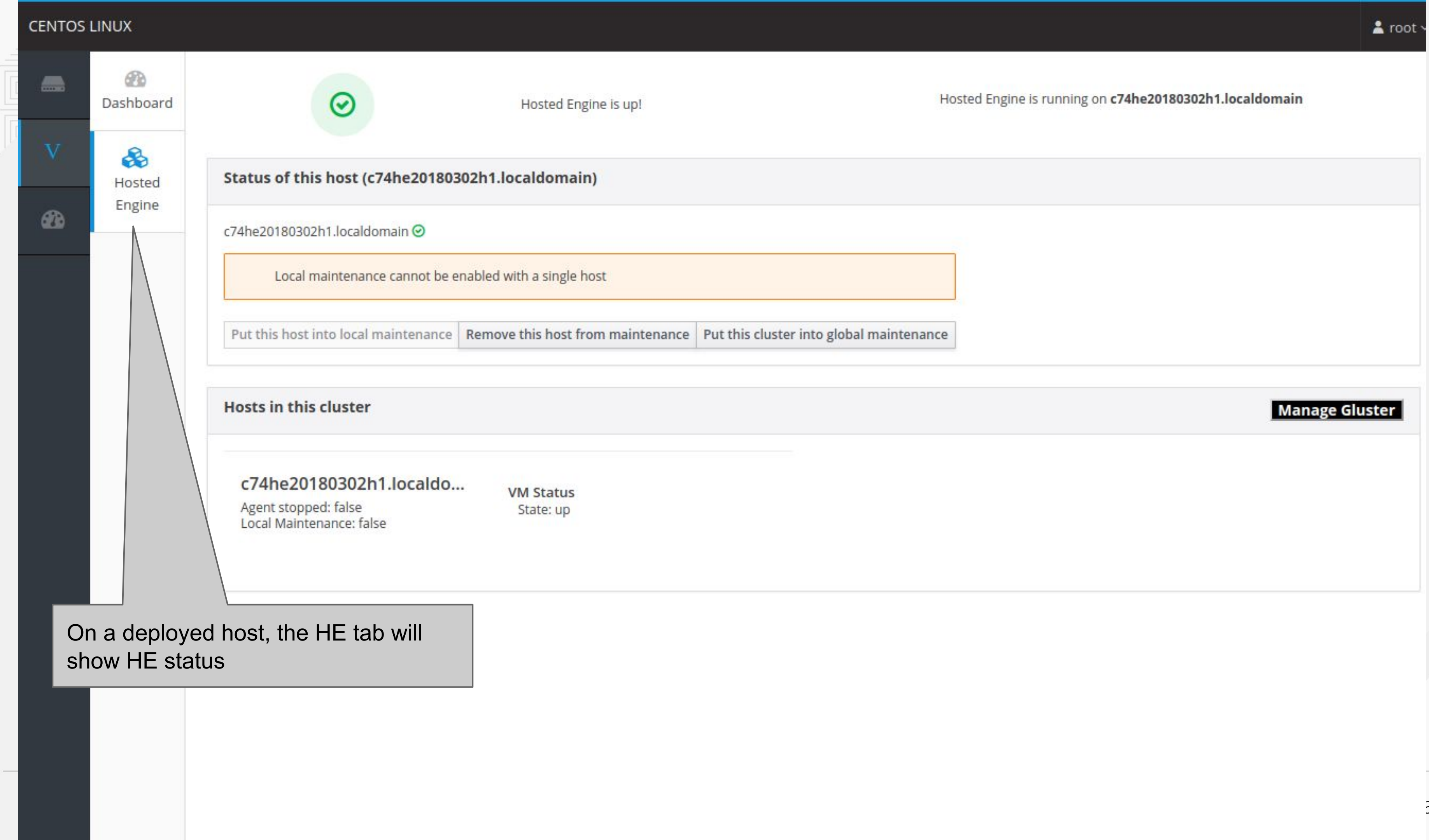
Deployment progress here.
Any issue will be shown here as a failed task



Hosted Engine Deployment

Hosted engine deployment complete!

On successful executions, you will reach this dialog and everything is up and running.


CENTOS LINUX
root
Dashboard
Hosted Engine
Put this host into local maintenance
Remove this host from maintenance
Put this cluster into global maintenance
Manage Gluster

Hosted Engine is up!

Hosted Engine is running on **c74he20180302h1.localdomain**

Status of this host (c74he20180302h1.localdomain)

c74he20180302h1.localdomain

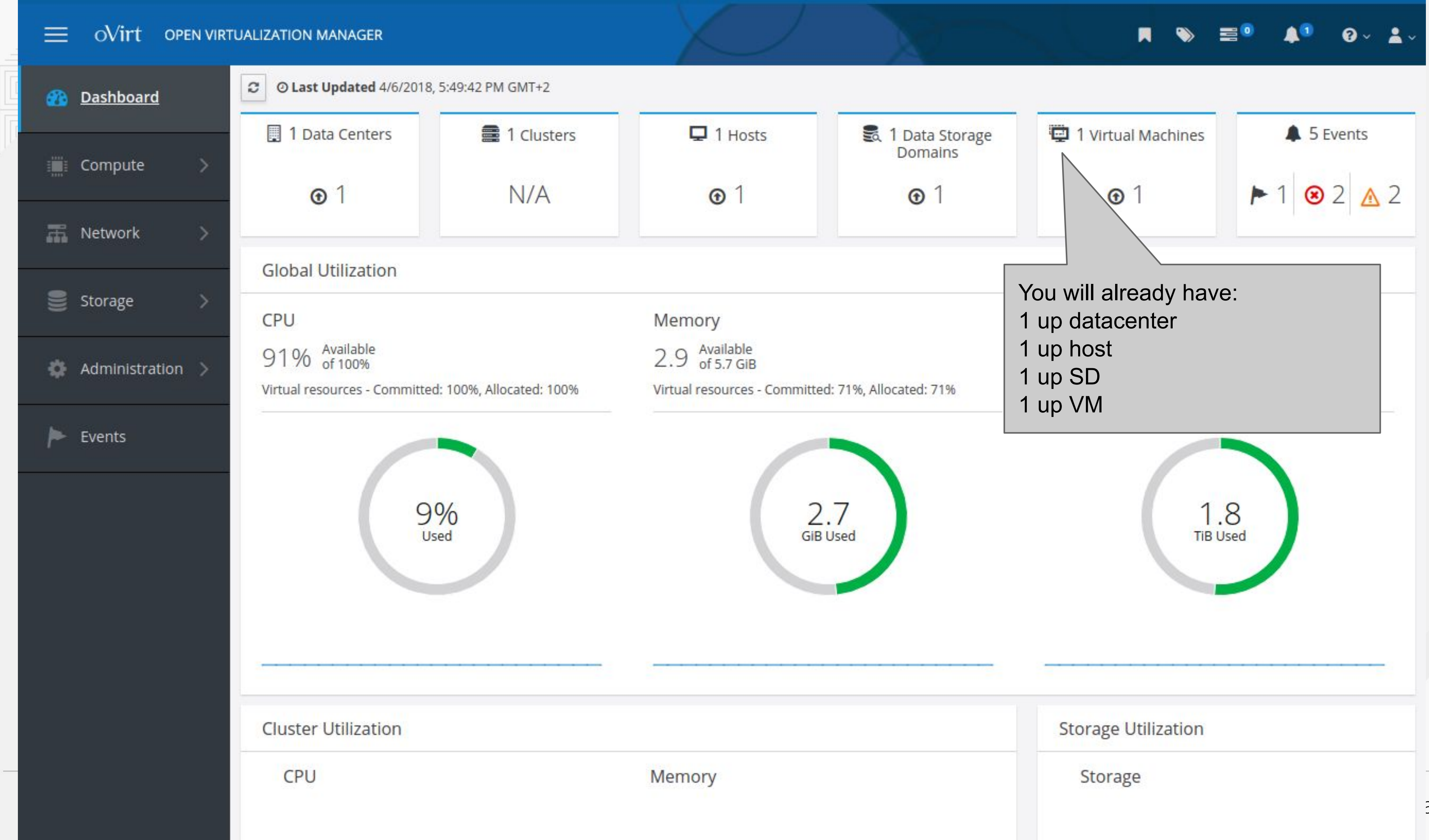
Local maintenance cannot be enabled with a single host

Hosts in this cluster

c74he20180302h1.localdo...
Agent stopped: false
Local Maintenance: false

VM Status
State: up

On a deployed host, the HE tab will show HE status



oVirt OPEN VIRTUALIZATION MANAGER

Dashboard

Compute

Network

Storage

Administration

Events

Last Updated 4/6/2018, 5:49:42 PM GMT+2

1 Data Centers	1 Clusters	1 Hosts	1 Data Storage Domains	1 Virtual Machines	5 Events
1	N/A	1	1	1	1 2 2

Global Utilization

CPU

91% Available of 100%

Virtual resources - Committed: 100%, Allocated: 100%

9% Used

Memory

2.9 Available of 5.7 GiB

Virtual resources - Committed: 71%, Allocated: 71%

2.7 GiB Used

1.8 TiB Used

Cluster Utilization

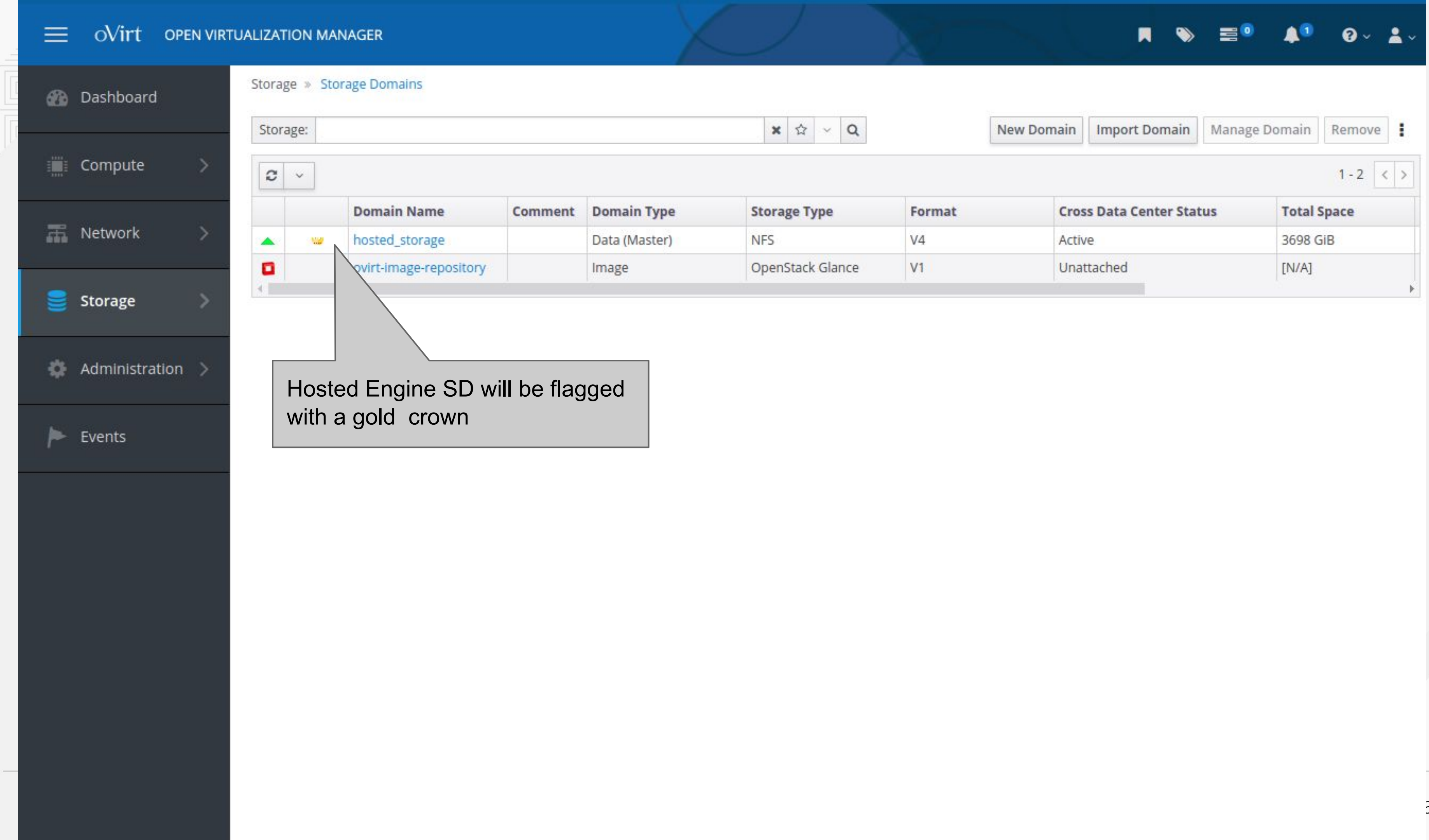
CPU

Memory

Storage Utilization

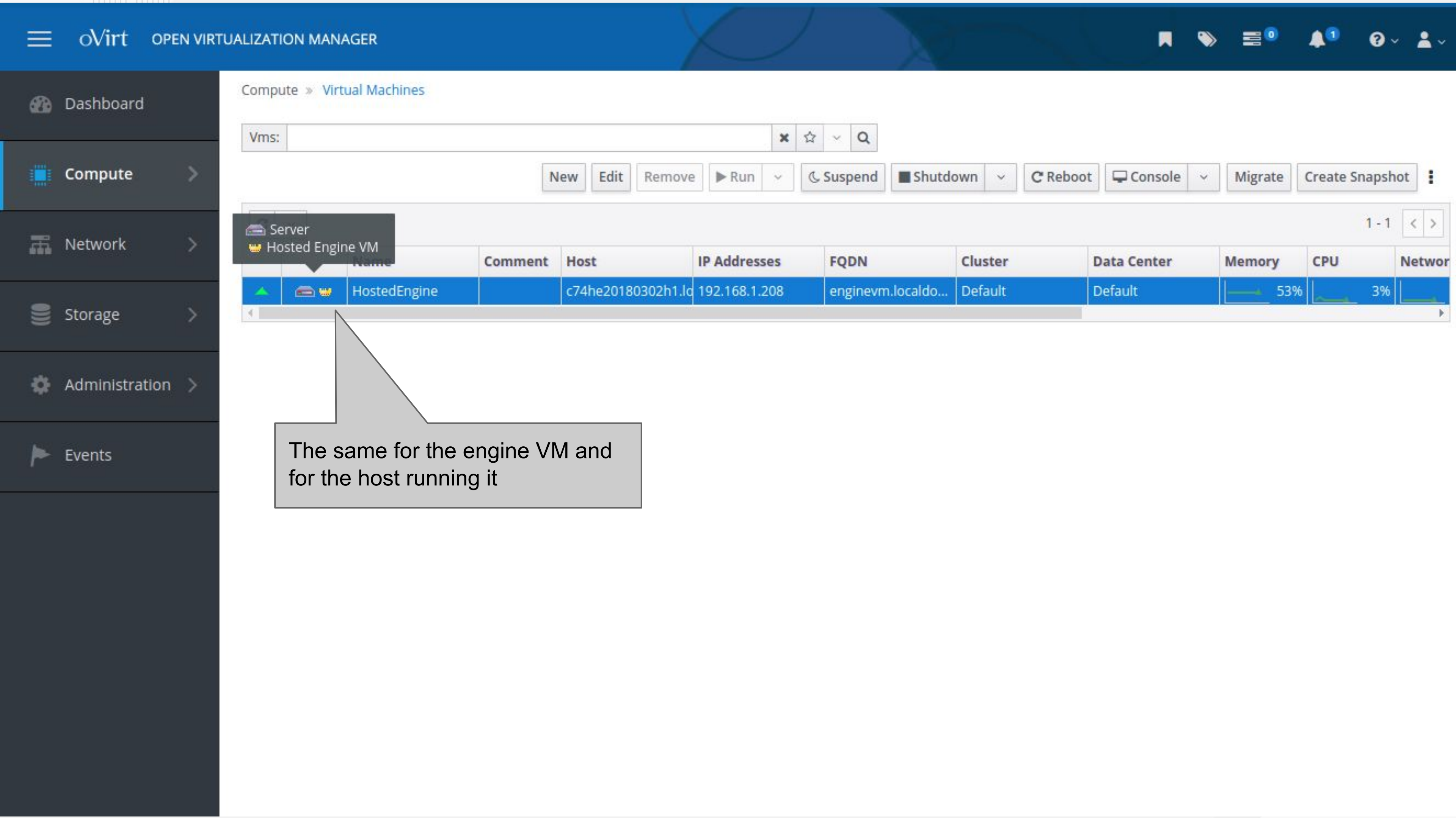
Storage

You will already have:
1 up datacenter
1 up host
1 up SD
1 up VM



		Domain Name	Comment	Domain Type	Storage Type	Format	Cross Data Center Status	Total Space
▲	♛	hosted_storage		Data (Master)	NFS	V4	Active	3698 GiB
■		ovirt-image-repository		Image	OpenStack Glance	V1	Unattached	[N/A]

Hosted Engine SD will be flagged with a gold crown



Compute » Virtual Machines

Vms:

		Name	Comment	Host	IP Addresses	FQDN	Cluster	Data Center	Memory	CPU	Networ
▲		HostedEngine		c74he20180302h1.lo	192.168.1.208	enginevm.localdo...	Default	Default	53%	3%	

The same for the engine VM and for the host running it

Maintenance

Maintenance modes

- For HE we have 2 maintenance modes:
  - **Global** maintenance mode
    - It’s for the engine VM: the VM will be not migrated/restarted by HA agent
    - It applies to all the HE hosts at the same time (it’s a flag on the shared storage)
    - It can be set from CLI or from the engine
  - **Local** maintenance mode
    - It’s for host related activities
    - It applies to a single host
    - It’s locally saved on host FS (/var/lib/ovirt-hosted-engine-ha/ha.conf)
    - Setting host maintenance mode from the engine will also imply hosted-engine local maintenance mode for the same host, the opposite is not true

Technical details and internals

Local configuration

/etc/ovirt-hosted-engine/hosted-engine.conf

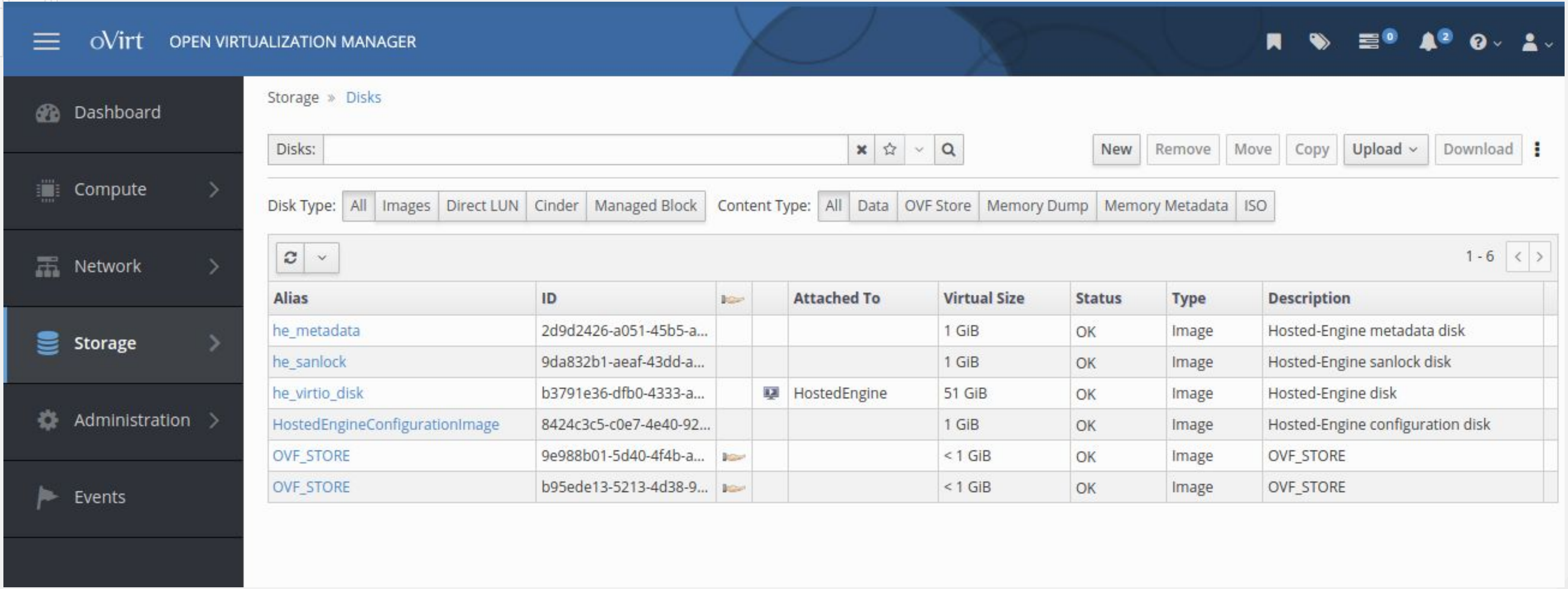
```
fqdn=enginevm.localdomain
vm_disk_id=b3791e36-dfb0-4333-a666-cc7a73d61abe
vm_disk_vol_id=62501681-897d-46ab-8225-c54181022c2b
vmid=ed1535c9-903b-415c-bd2f-f3346ca4f256
storage=192.168.1.115:/storage/nfs/he1
nfs_version=v4
mnt_options=
conf=/var/run/ovirt-hosted-engine-ha/vm.conf
host_id=1
console=vnc
domainType=nfs
spUUID=00000000-0000-0000-0000-00000000000
sdUUID=503ba198-5375-40a5-837d-f349eb908398
connectionUUID=e29cf818-5ee5-46e1-85c1-8aeefa33e95d
ca_cert=/etc/pki/vdsm/libvirt-spice/ca-cert.pem
ca_subject="C=EN, L=Test, O=Test, CN=Test"
vdsm_use_ssl=true
gateway=192.168.1.1
bridge=ovirtmgmt
metadata_volume_UUID=0772f336-f9b9-40a2-b384-15d80ab6b071
metadata_image_UUID=2d9d2426-a051-45b5-a482-390def1b4d2f
lockspace_volume_UUID=49524530-8889-47a4-a1f4-8089c03252b1
lockspace_image_UUID=9da832b1-aeaf-43dd-a995-896fc3fa58e8
conf_volume_UUID=78638fb3-96ca-41ff-ab13-2864c92315fc
conf_image_UUID=8424c3c5-c0e7-4e40-92c2-f4661dffe965

# The following are used only for iSCSI storage
iqn=
portal=
user=
password=
port=
```

It contains all the info needed to connect and access the hosted-engine storage domain also if the engine is down

There is only one value that changes between one HE host and the next one: **host_id**
Mixing up host_id with different hosts can cause huge headaches!

On the shared storage



Alias	ID			Attached To	Virtual Size	Status	Type	Description
he_metadata	2d9d2426-a051-45b5-a...				1 GiB	OK	Image	Hosted-Engine metadata disk
he_sanlock	9da832b1-aeaf-43dd-a...				1 GiB	OK	Image	Hosted-Engine sanlock disk
he_virtio_disk	b3791e36-dfb0-4333-a...			HostedEngine	51 GiB	OK	Image	Hosted-Engine disk
HostedEngineConfigurationImage	8424c3c5-c0e7-4e40-92...				1 GiB	OK	Image	Hosted-Engine configuration disk
OVF_STORE	9e988b01-5d40-4f4b-a...				< 1 GiB	OK	Image	OVF_STORE
OVF_STORE	b95ede13-5213-4d38-9...				< 1 GiB	OK	Image	OVF_STORE

On the shared storage

- **he_metadata**
  - It’s a kind of whiteboard use by hosted-engine host to communicate.
  - Each host writes its metadata (status, local maintenance, score, update timestamp…) in its specific sector (based on host_id, so it the got messed up…)
  - Each host can only write in its own sector but it will also read metadata from other hosts
- **he_sanlock**
  - Used with sanlock to protect engine VM disk and he_metadata global sector, based on host_id
- **he_virtio_disk**
  - It contains engine VM disk
- **HostedEngineConfigurationImage**
  - It contains a master copy of hosted-engine configuration used as a template adding a new HE host from the engine
- **OVF_STORE** (two copies)
  - It contains VM definition (ovf+xml for libvirt of all the VMs on that SD)
  - Periodically (1h or on updates) rewritten by the engine

Temporary files

`/var/run/ovirt-hosted-engine-ha/vm.conf`

On tmpfs, not persisted!

```
# Editing the hosted engine VM is only possible via the manager UI\API
# This file was generated at Sun Jan 20 01:37:27 2019

cpuType=Haswell-noTSX
emulatedMachine=pc-i440fx-rhel7.6.0
vmId=ed1535c9-903b-415c-bd2f-f3346ca4f256
smp=4
memSize=4096
maxVCpus=64
spiceSecureChannels=smain,sdisplay,sinputs,scursor,splayback,srecord,s
smartcard,susbredir
xmlBase64=PD94bWwgdmVyc2lvbj0nMS4wJyBlbmNvZGluZz0nVVRGLTgnPz4KPGRvbWFp
biB4bWxuczpvdmlydC10dW5lPSJodHRwOi8vb3ZpcnQub3JnL3ZtL3R1bmUvMS4wIiB4bW
xuczpvdmlydC12bT0iaHR0cDovL292aXJ0Lm9yZy92bS8xLjAiIHR5cGU9Imt2bSI+PG5h
bWU+SG9zdGVkRW5naW5lPC9uYW1lPjx1dWlkPmVkMTUzNWM5LTkwM2ItNDE1Yy1iZDJmLW
YzMzQ2Y2E0ZjI1NjwvdXVpZD48bWVtb3J5PjQxOTQzMDQ8L21lbW9yeT48Y3VycmVudE1l
bW9yeT40MTk0MzA0PC9jdXJyZW50TWVtb3J5Pjxpb3RocmVhZHM+MTwvaW90aHJlYWRzPj
xtYXhNZW1vcnkgc2xvdHM9IjE2Ij4xNjc3NzIxNjwvbWF4TWVtb3J5Pjx2Y3B1IGN1cnJl
bnQ9IjQiPjY0PC92Y3B1PjxzeXNpbmZvIHR5cGU9InNtYmlvcyI+PHN5c3RlbT48ZW50cn
kgbmFtZT0ibWFudWZhY3R1cmVyIj5vVmlydDwvZW50cnk+PGVudHJ5IG5hbWU9InByb2R1
Y3QiPk9TLU5BTUU6PC9lbnRyeT48ZW50cnkgbmFtZT0idmVyc2lvbiI+T1MtVkVSU0lPTj
o8L2VudHJ5PjxlbnRyeS
…
```

Periodically regenerated parsing OVF_STORE disks

Used by ovirt-ha-agent to start engine VM via vdsm

If the xml for libvirt is there (in base64) - engine >= 4.2 - vdsm will direct send that to libvirt ignoring everything else

Inside engine’s DB

```
[root@hosted-engine-01 tmp]# sudo -u postgres scl enable rh-postgresql10 -- psql -d engine -c "select vds_name, vds_spm_id, ha_score,
ha_configured, ha_active, ha_global_maintenance, ha_local_maintenance from vds"
   vds_name    | vds_spm_id | ha_score | ha_configured | ha_active | ha_global_maintenance | ha_local_maintenance |
is_hosted_engine_host
-------------+------------+----------+---------------+-----------+-----------------------+----------------------+-------------------
---
 host_mixed_1 |          1 |     3400 | t             | t         | f                     | f                    | f
 host_mixed_2 |          2 |     3400 | t             | t         | f                     | f                    | t
 host_mixed_3 |          3 |     3400 | t             | t         | f                     | f                    | f
```

It should **ALWAYS** match host_id used in /etc/ovirt-hosted-engine/hosted-engine.conf on each host!

ha_score as reported by the host

If hosted-engine is configured on this specific host

If hosted-engine is active on this specific host

If this host reported to be in HE global maintenance

vds is a view based on vds_static, vds_dynamic and vds_statistics

Computed by the engine according to VM info as reported by hosts

All of them refers to the last time the engine successfully queries VDSM!
The engine has no direct access to hosted-engine metadata!!!
This info could be outdated!

Troubleshooting

Setup Logs

All the logs are available under /var/log/ovirt-hosted-engine-setup/ on the host:

```
ovirt-hosted-engine-setup-ansible-get_network_interfaces-20183617355-er2ny7.log
ovirt-hosted-engine-setup-ansible-initial_clean-2018361784-i75e3e.log
ovirt-hosted-engine-setup-ansible-bootstrap_local_vm-20183617821-lhrq77.log
ovirt-hosted-engine-setup-ansible-create_storage_domain-201836172019-wh5qok.log
ovirt-hosted-engine-setup-ansible-create_target_vm-201836172117-o0rrxl.log
ovirt-hosted-engine-setup-ansible-final_clean-201836173040-ugmzwx.log
```

We have at least 6 playbooks until the end so at least 6 log files (1 more for FC device discovery, 2 more for iSCSI discovery and login)

The setup is also trying to extract engine logs from engine VM disks, not always possible

Other relevant logs on the host

Other relevant logs on the host are:

```
/var/log/messages
/var/log/vdsm/vdsm.log
/var/log/vdsm/supervdsm.log
/var/log/libvirt/qemu/HostedEngineLocal.log
/var/log/libvirt/qemu/HostedEngine.log
```

Other relevant logs on the engine VM

Other relevant logs on the engine VM are:

```
/var/log/messages
```

```
/var/log/ovirt-engine/engine.log
```

```
/var/log/ovirt-engine/server.log
```

```
/var/log/ovirt-engine/host-deploy/ovirt-host-deploy-20180406171311-c74he20180302h1.localdomain-301aca6d.log
```

```
/var/log/ovirt-engine/host-deploy/ovirt-host-deploy-ansible-20180406171312-c74he20180302h1.localdomain-301aca6d.
log
```

The target VM started from the shared storage should be accessible on the FQDN you set from any host in the network.

The bootstrap local VM runs over a natted network so it's accessible only from the host where it's running; /etc/hosts should already contain a record for that pointing on the right address on the natted subnet.

If not accessible over the network other means to reach it are:

```
hosted-engine --console
```

```
remote-viewer vnc://<host_address>:5900 (hosted-engine --add-console-password to set a
temporary VNC password)
```

HA services logs

Relevant logs:

```
/var/log/ovirt-hosted-engine-ha/agent.log
/var/log/ovirt-hosted-engine-ha/broker.log
```

The logs are usually at INFO level, the logs can be set at debug level editing /etc/ovirt-hosted-engine-ha/agent-log.conf and broker-log.conf

```
[loggers]
keys=root

[handlers]
keys=syslog,logfile

[formatters]
keys=long,sysform

[logger_root]
level=INFO
handlers=syslog,logfile
propagate=0

[handler_syslog]
```

Set DEBUG here and restart the service

Backup and restore

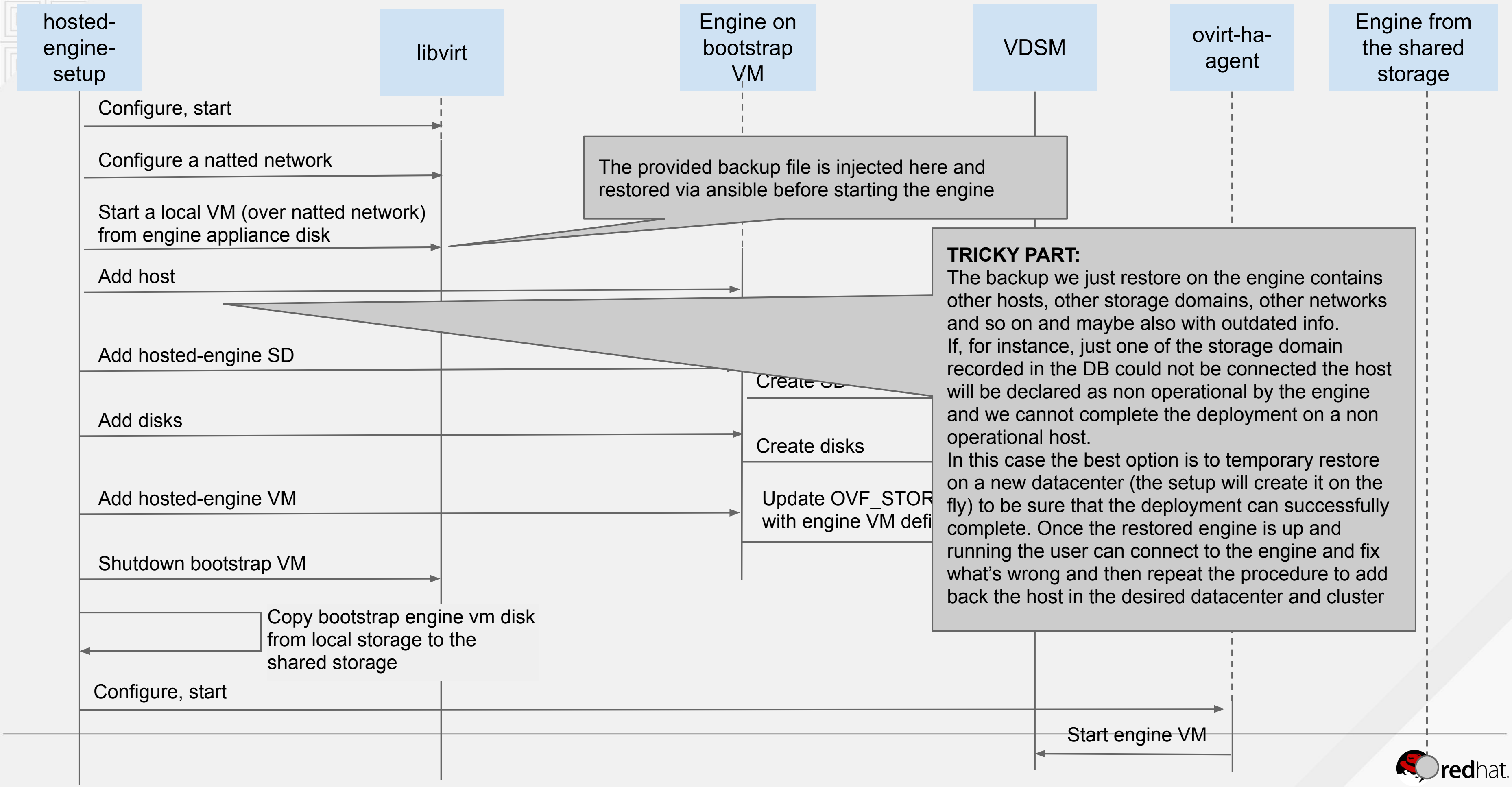
- Since hosted-engine relies on a **volume lease** (VM lease weren't available in oVirt when we started hosted-engine), we cannot rely on **disk snapshots** for the engine VM
- For the same reason we cannot live storage migrate the engine VM from its storage domain to a different one
- The best approach to backup hosted-engine is periodically running **engine-backup** inside the VM
- With 4.2.7 we introduced the capability to restore a backup on the fly deploying a new hosted-engine VM:

```
hosted-engine --deploy    --restore-from-file=file

     Restore an engine backup file during the deployment
```

- The same tool could be used to **migrate** the engine VM from **one storage domain** to a **new one** or from **bare metal** to **hosted-engine** via backup and restore
- **CAREFULLY follow the documentation since ending with two active engine VM acting at the same time over the same VMs could be really destructive**

Restore: how it works



hosted-engine-setup

libvirt

Engine on bootstrap VM

VDSM

ovirt-ha-agent

Engine from the shared storage

Configure, start

Configure a natted network

Start a local VM (over natted network) from engine appliance disk

The provided backup file is injected here and restored via ansible before starting the engine

Add host

Add hosted-engine SD

Create SD

Add disks

Create disks

Add hosted-engine VM

Update OVF_STOR with engine VM defi

Shutdown bootstrap VM

Copy bootstrap engine vm disk from local storage to the shared storage

Configure, start

Start engine VM

TRICKY PART:
The backup we just restore on the engine contains other hosts, other storage domains, other networks and so on and maybe also with outdated info.
If, for instance, just one of the storage domain recorded in the DB could not be connected the host will be declared as non operational by the engine and we cannot complete the deployment on a non operational host.
In this case the best option is to temporary restore on a new datacenter (the setup will create it on the fly) to be sure that the deployment can successfully complete. Once the restored engine is up and running the user can connect to the engine and fix what’s wrong and then repeat the procedure to add back the host in the desired datacenter and cluster

Major issues: 1. spm host id mashup

host_id is **locally stored on each host** in /etc/ovirt-hosted-engine/hosted-engine.conf

It's also stored for each host in the **engine DB** as vds_spm_id

If, due to human errors, reactivation of old decommissioned hosts, backup… two hosts tries to use the same spm host id the stranger behaviour could be reported:

- Incongruent HA scores reported by the hosts
- Sanlock issues
- …

SOLUTION: manually align all the host_id locally saved on host side to what's recorded in the engine DB (**master copy!**)

Major issues: 2. corrupted metadata volume

Due to storage issues or power outages and so on the **metadata** volume can be **corrupted**, if so it can be cleaned running (just of one of hosted-engine hosts):

```
[root@c76he20190115h1 ~]# hosted-engine --clean-metadata --help
Usage: /sbin/hosted-engine --clean_metadata [--force-cleanup]
[--host-id=<id>]
    Remove host's metadata from the global status database.
    Available only in properly deployed cluster with properly st
    agent.

    --force-cleanup  This option overrides the safety checks. Use at your
own
                     risk DANGEROUS.

    --host-id=<id>  Specify an explicit host id to clean
```

If specified, it will ignore any safety check about other active hosts

If specified, it will clean only the sector of a specific host

Major issues: 3. corrupted lockspace volume

Due to storage issues or power outages and so on the **lockspace** volume can be **corrupted**, you can notice it from sanlock related errors in broker.log and sanlock.log. If so it can be cleaned with

```
# on each HE host

systemctl stop ovirt-ha-agent ovirt-ha-broker

sanlock client shutdown -f 1 # carefully, it could trigger the watchdog and reboot

# on a single host

hosted-engine --reinitialize-lockspace

# on each HE host

systemctl start ovirt-ha-agent ovirt-ha-broker
```

Major issues: 4. Engine-setup refuses to execute

- Engine-setup is going to shut down the engine VM for setup tasks while, if not in global maintenance mode, ovirt-ha-agent is going to motor engine health and potentially restart the engine VM to gain a working engine
- Shutting down the engine VM in the middle of a yum, engine or DB upgrade could lead to bad results so engine-setup is trying to detect global maintenance status as recorded in the engine DB
- The point is that the DB could contain outdated info that the engine is not able to refresh for different reasons and so we can have a false positive where engine-setup refuses to upgrade requiring global maintenance mode while the env is already there

SOLUTION: if, and only if, the user is absolutely sure that the engine is really in global maintenance mode, he can execute engine-setup with `--otopi-environment=OVESETUP_CONFIG/continueSetupOnHEVM=bool:True` to completely skip the check about global maintenance mode as recorded in the DB (potentially dangerous!)

Major issues: 5. Engine VM refuses to start

- The user can edit the engine VM from the engine itself.
- The new configuration will be written in the OVF_STORE and consumed on the next run (the OVF_STORE update will be sync, expect a few seconds delay to complete).
- The engine should validate the configuration preventing upfront major mistakes
- If for any reason the VM could not start with the new configuration the user can still:

```
# copy /var/run/ovirt-hosted-engine-ha/vm.conf somewhere else

# edit it as needed (rememend that the libvirt XML will win over anything else if there, so fix it or remove it)

# start the engine VM with customized configuration with:

hosted-engine --vm-start --vm-conf=/root/myvm.conf

# fix what’s wrong from the engine, and try a new boot from the OVF_STORE definition
```

Q&A ???

Q&A: questions from session 1

- **Ansible deployment failed - where to look for errors?**

/var/log/ovirt-hosted-engine-setup/ovirt-hosted-engine-setup-ansible*
It's always in verbose mode

- **Hosted Engine VM is unmanaged - how to recover?**

It could happen in 4.1 if for any instance the autoimport code was failing. It couldn't happen anymore by design with the new flow.

- **HE VM is corrupted, what is the best way to recover?**

The best way to recover is using the restore flow starting from the last backup. If no backup is available, try a fresh deployment manually importing all the existing storage domains at the end and manually adding back all the hosts.

- **How can we modify HE VM from vm.conf and persist it?**

vm.conf is periodically regenerated starting from OVF_STORE content, if the engine is available the best option is to edit the VM configuration from the engine. If the engine is not available, we can make a copy of vm.conf, fix what is needed and than try to start the engine VM with a custom vm.conf with `hosted-engine --vm-start --vm-conf=/root/myvm.conf`
No change is going to be persisted and so the user has to manually fix the engine VM definition again from the engine one up and running.

Q&A: questions from session 2

- **Cannot add host to HE cluster (deploy as HE host)?**

Check engine.log and host-deploy logs on the engine VM

- **Host vm-status is different from the rest of the cluster?**

Probably one of the host is failing to access the storage to write its score or read other ones from the metadata volume.
Please check update timestamps and broker.log for storage related errors.

- **HE VM would not start - what to do?**

Check agent.log, broker.log, libvirt and vdsm logs for errors. Fix as required (for instance regenerate lockspace or metadata volume or provide a custom vm.conf).

- **When recovering from backup - which appliance to take? For instance we are on 4.2 now, but original deployment was on 4.1**

Normally the best option is latest .z version of the same major used at backup time since engine-backup is not supposed to be an upgrade tool.
For instance if the backup was took with 4.1 it will be safer to apply it on 4.1 based appliance.
On migrations it will be safer to upgrade before the migration if needed.