Migrazione cluster OpenShift Cattolica

Report attività installazione cluster OpenShift

Sommario

Sommario **2**

Disegno infrastrutturale **5**

Update rispetto al disegno infrastrutturale 5

Ambiente di produzione: 5

Ambienti di non produzione: 6

Schema di rete 7

Schema di storage 7

Preparazione **8**

Preparazione del Bastion 8

Copia dei certificati vCenter 8

Test di raggiungibilità 9

Creazione del keypair SSH 9

Pacchetti di installazione e client 10

Cluster OCP-SVIL (NoProd) **11**

Operazioni pre installazione 11

Configurazioni di rete 11

Configurazioni vSphere 11

Test DNS e allocazione IP 12

Installazione del cluster 12

Preparazione dell'install-config.yml 13

Installazione del cluster 15

Verifica dell'installazione 17

Applicazione di test 19

Validazione dell'installazione 20

Operazioni post installazione 22

Configurazione dello storage per l'Image Registry 22

Autenticazione 24

Collegamento ad Active Directory 24

Sincronizzazione dei gruppi Active Directory 26

Sincronizzazione automatica 27

Autorizzazione 31

Aggiunta del ruolo cluster-admin 31

Rimozione del self-provisioning 32

Cluster OCP-TEST (NoProd) **33**

Operazioni pre installazione 33

Configurazioni di rete 33

Configurazioni vSphere 33

Cluster OCP-BF (NoProd) **34**

Operazioni pre installazione 34

Configurazioni di rete 34
Configurazioni vSphere 34

Cluster OCP-COLL (NoProd) 35
Operazioni pre installazione 35
Configurazioni di rete 35
Configurazioni vSphere 35

Cluster OCP-PROD (Prod) 36
Operazioni pre installazione 36
Configurazioni di rete 36
Configurazioni vSphere 36
Test DNS e allocazione IP 37
Installazione del cluster 37
Preparazione dell'install-config.yml 37
Installazione del cluster 39
Creazione dei MachineSet per i worker 41
Verifica dell'installazione 43
Applicazione di test 45
Validazione dell'installazione 46
Operazioni post installazione 48
Configurazione dello storage per l'Image Registry 48
Autenticazione 50
Collegamento ad Active Directory 50
Sincronizzazione dei gruppi Active Directory 53
Sincronizzazione automatica 55
Autorizzazione 60
Aggiunta del ruolo cluster-admin 60
Rimozione del self-provisioning 60

Task post-installazione 61
Integrazione con pipelines per Jenkins 61
Service Account Jenkins 61
Custom Roles 61
Custom Role Binding 62
Service Account per Mia Console 63
Service Account per Mia Console 63
Cluster Role per Mia Console 63
Cluster Role Binding 64
Installazione Filebeat 64

Comandi utili 65
Problematiche di Header 65
Scaling MachineSet 66
Caricamento certificati custom 66

Ingress controller certificate 66
API Certificate 67
Visualizzazione di consumo di risorse 67
Formattazione di output 67
Documentazione risorse da CLI 67

Appendice 1: jenkins-role Custom Roles 68

Appendice 2: mia-console-role Custom Role 73

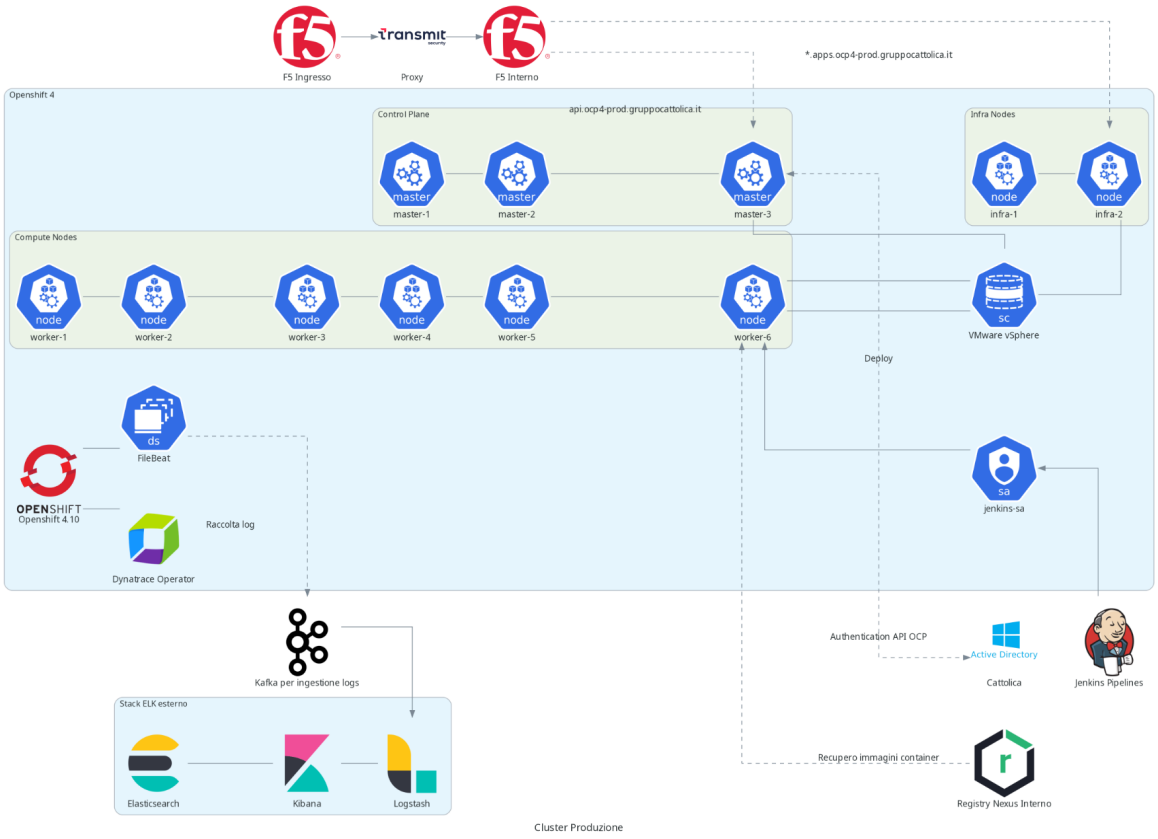
Appendice 3: Risorse per Filebeat 75

Disegno infrastrutturale

Update rispetto al disegno infrastrutturale

Rispetto a quanto esplicato nel precedente documento, sono state apportate alcune modifiche all'infrastruttura proposta:

Ambiente di produzione:



In seguito anche ad un confronto con Red Hat, è stato deciso di optare per l'aggiunta di due nodi di tipologia **Infra**.

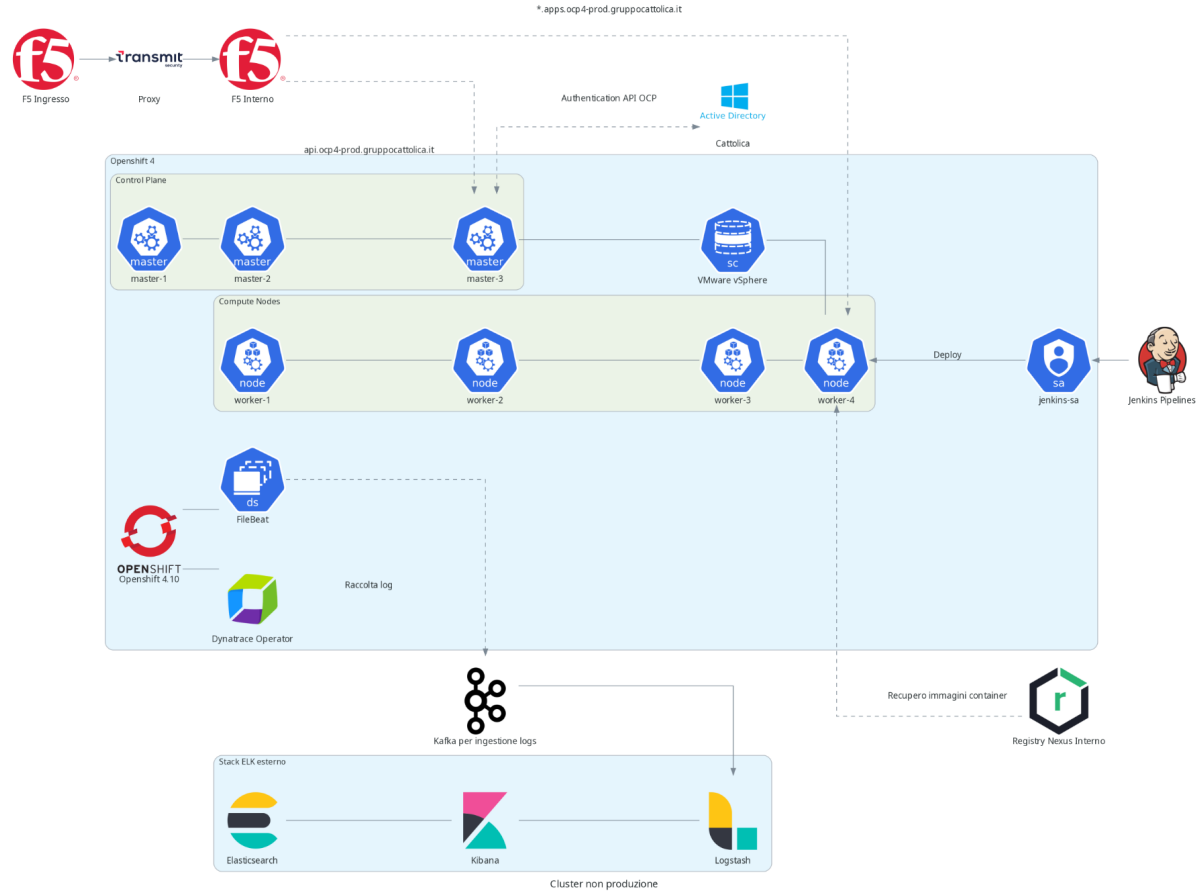
I nodi con questo particolare ruolo:

- sono a tutti gli effetti dei worker computazionali
- possono ospitare solo alcune categorie di applicazioni come il monitoraggio e gli Ingress Controller, che non sono control plane ma nemmeno applicativi del cliente
- come i nodi master, non necessitano di subscription e quindi vanno a preservare i restanti worker dai workload di infrastruttura

Abbiamo ritenuto che l'ambiente **ocp-prod** necessitasse di due nodi infra per queste ragioni:

- possibilità di bilanciare il traffico in ingresso dall'F5 BIG-IP verso due nodi che ospiteranno certamente le due repliche degli Ingress Controller[1]
- I workload infrastrutturali come il Prometheus vengono tolti dai worker, risparmiando cpu e memoria

Ambienti di non produzione:



In seguito alla migrazione delle applicazioni, è stato evidenziato che la somma delle *resource request* dei vari deploy superava le risorse disponibili del cluster. Queste resource non comportano un effettivo consumo di risorse sul nodo, ma solo una loro preallocazione numerica.

Per risolvere questa problematica, è stato scelto di allocare un ulteriore nodo per ogni ambiente di non produzione, in modo da distribuire in maniera uniforme il vario carico di lavoro e non venire impattati dalla problematica sopra descritta.

N.B. Questa problematica non era stata riscontrata nel vecchio cluster di preproduzione in quanto vi era un dimensionamento diverso con un numero di nodi maggiore. Questo comportava un maggior numero di risorse pre allocabili numericamente.

[1] In un'installazione IPI, infatti, la destinazione dei Pod dei router non è predicibile

Schema di rete

Riportiamo solo lo schema definitivo di rete approvato dal cliente:

Label	OCP4_PROD	OCP4_COLL	OCP4_BF	OCP4_TEST	OCP4_SVIL
VLAN	359	365	361	362	363
Subnet	172.22.10.0	172.22.10.128	172.22.10.160	172.22.10.192	172.22.10.224
Mask	/26	/27	/27	/27	/27
n. Host	62	30	30	30	30
Net Address	.0	.128	.160	.192	.224
GW	.1	.129	.161	.193	.225
API	.32	.132	.164	.196	.228
IGRES	.33	.133	.165	.197	.229
DHCP Scope	.34 - .62	134 - .158	.166 - .190	.198 - .222	.230 - .254

Schema di storage

Riportiamo la lista dei Datastore da utilizzare

Label	**Datastore**
ocp-prod	[VM_FLASH_PROD_209_DR]
ocp-coll	[VM_FLASH_COLL_LINUX_039_NODR]
ocp-bf	[VM_FLASH_COLL_LINUX_039_NODR]
ocp-test	[VM_FLASH_COLL_LINUX_039_NODR]
ocp-svil	[VM_FLASH_COLL_LINUX_039_NODR]

⚠️ **NOTA BENE**

Per il cluster **ocp-prod** useremo il Datastore [VM_FLASH_PROD_209_DR] per i soli nodi Master. I primi due worker servono solo per effettuare l'installazione e verranno cancellati successivamente in favore dei nuovi.

Vedi la sezione Creazione dei MachineSet per i worker.

Preparazione

Preparazione del Bastion

È stato preparato un nodo bastion per lanciare l'installazione dei cluster.

Bastion FQDN	`mn901ansi.gca.net`
Username	*[utente LDAP ricevuto via e-mail]*
Password	*[password LDAP ricevuta via e-mail]*
Shared folder	`/CattApp/OCP4`

Copia dei certificati vCenter

Collegarsi al nodo bastion e spostarsi nella cartella condivisa:

```
$ cd /CattApp/OCP4
```

Scaricare i certificati dal vCenter (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere →* *Adding vCenter root CA certificates to your system trust*).:

```
$ curl https://catvcsa101.cattolica.gca.net/certs/download.zip
```

Una volta aperto l'archivio, bisogna copiare i certificati nel sistema:

```
$ sudo cp certs/lin/* /etc/pki/ca-trust/source/anchors
```

quindi forziamo la rilettura dei certificati:

```
$ sudo update-ca-trust extract
```

Test di raggiungibilità

Dal nodo bastion, verifichiamo che sia il vCenter, sia i nodi ESXi siano raggiungibili sulla porta 443 (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere →* Prerequisites).

Test verso il vCenter:

```
$ ncat -c '' -w 1 -v catvcsa101.cattolica.gca.net 443
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Connected to 172.25.244.11:443.
```

Test verso un ESXi a campione:

```
$ ncat -c '' -w 1 -v catesx143.cattolica.gca.net 443
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Connected to 172.25.244.63:443.
```

> ⚠️ **NOTA BENE**
>
> Oltre che il nodo bastion, dove gira l' `openshift-install`, anche il futuro cluster OpenShift dovrà essere in grado di raggiungere le API del vCenter.
>
> In un'installazione IPI, infatti, è il control plane finale che si occupa di comandare al vCenter la creazione dei nodi worker.

Creazione del keypair SSH

Creiamo un keypair SSH che andremo ad usare per l'accesso al cluster (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere →* Generating a key pair for cluster node SSH access):

```
$ mkdir /CattApp/OCP4/ssh_keypair

$ cd /CattApp/OCP4/ssh_keypair

$ ssh-keygen -t ed25519 -N '' -C '' -f \
  /CattApp/OCP4/ssh_keypair/id_ed25519
```

Pacchetti di installazione e client

Download del pacchetti di installazione:

```
$ cd /CattApp/OCP4

$ curl -O \
https://mirror.openshift.com/pub/openshift-v4/x86_64/clients/ocp/stable-4.10/openshift-install-linux.tar.gz

$ tar xfvz openshift-install-linux.tar.gz

$ sudo mv openshift-install /usr/local/bin/

$ openshift-install completion bash > openshift_install_bash_completion

$ sudo mv openshift_install_bash_completion /etc/bash_completion.d/
```

Download del client e configurazione (vedi *OpenShift Container Platform → 4.10 → CLI tools → Chapter 2. OpenShift CLI (oc)* → *Configuring the OpenShift CLI*):

```
$ curl -O \
https://mirror.openshift.com/pub/openshift-v4/x86_64/clients/ocp/stable-4.10/openshift-client-linux.tar.gz

$ tar xfvz openshift-client-linux.tar.gz

$ sudo mv kubectl oc /usr/local/bin/

$ kubectl completion bash > kubectl_bash_completion

$ oc completion bash > oc_bash_completion

$ sudo mv oc_bash_completion kubectl_bash_completion /etc/bash_completion.d/
```

Se necessario, installare il pacchetto bash-completion:

```
$ sudo yum install bash-completion
```

Cluster OCP-SVIL (NoProd)

In questa sezione verranno riportate tutte le operazioni pre, durante e post installazione del cluster in oggetto.

Operazioni pre installazione

Configurazioni di rete

Di seguito sono riportati i dati della configurazione di rete per il cluster.

Label	OCP4_SVIL
Subnet	172.22.10.224
Mask	27
GW	.225
API	.228
IGRES	.229
DHCP Scope	.230 - .254

Configurazioni vSphere

Di seguito sono riportati i dati per l'installazione del cluster su VMware vSphere.

vCenter FQDN	catvcsa101.cattolica.gca.net
Username	cattolica\srv_ocp4_user
Password	*[non riportata in documentazione]*
Datacenter	CAT-ROZZANO
Datastore	VM_FLASH_COLL_LINUX_039_NODR
Network	OCP4_SVIL
Cluster	CAT_LINUX
OCP API VIP	api.ocp-svil.gruppocattolica.it
OCP Ingress VIP	*.apps.ocp-svil.gruppocattolica.it

Test DNS e allocazione IP

Effettuiamo le prove per verificare la presenza dei record DNS dell'API server e dell'Ingress Controller.

```
$ dig +short A api.ocp-svil.gruppocattolica.it
172.22.10.228

$ dig +short A testwc.apps.ocp-svil.gruppocattolica.it
172.22.10.229

$ dig +short -x 172.22.10.228
api.ocp-svil.gruppocattolica.it
```

Installazione del cluster

Per l'installazione del cluster OCP su VMware vSphere ci atteniamo al documento Installing a cluster on vSphere with network customizations (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere*).

Preparazione dell'install-config.yml

Creiamo una cartella dedicata al cluster:

```
$ mkdir -p /CattApp/OCP4/ocp-svil/

$ cd /CattApp/OCP4/ocp-svil/
```

All'interno della cartella, prepariamo il file `install-config.yml` per installare il cluster in modalità unattended:

```
apiVersion: v1
baseDomain: gruppocattolica.it
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    vsphere:
      cpus: 8
      coresPerSocket: 2
      memoryMB: 32768
      osDisk:
        diskSizeGB: 150
  replicas: 3
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    vsphere:
      cpus: 4
      coresPerSocket: 2
      memoryMB: 16384
      osDisk:
        diskSizeGB: 150
  replicas: 3
metadata:
  name: ocp-svil
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
```

```
  machineNetwork:
  - cidr: 172.22.10.224/27
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    apiVIP: 172.22.10.228
    cluster: CAT-LINUX
    datacenter: CAT-ROZZANO
    defaultDatastore: VM_FLASH_COLL_LINUX_039_NODR
    diskType: thick
    folder: ocp-svil
    ingressVIP: 172.22.10.229
    network: OCP4_SVIL
    password: [non riportata in documentazione]
    username: cattolica\srv_ocp4_user
    vCenter: catvcsa101.cattolica.gca.net
publish: External
pullSecret: '{"auths":{"cloud.openshift.com":{"auth":...}}}'
sshKey: 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIBuFTRyRxcH7UDzl9jXzGQYwNW+ICpENzY5vm4tEWmKR'
```

Installazione del cluster

Quindi procediamo con la creazione degli asset di configurazione:

```
$ openshift-install create install-config --dir ocp-svil
```

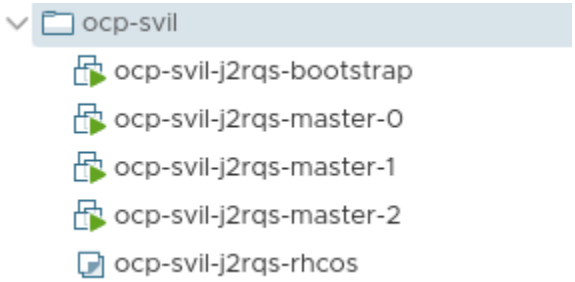
Se non ci sono errori, installiamo il cluster con:

```
$ openshift-install create cluster --dir ocp-svil --log-level=info
```

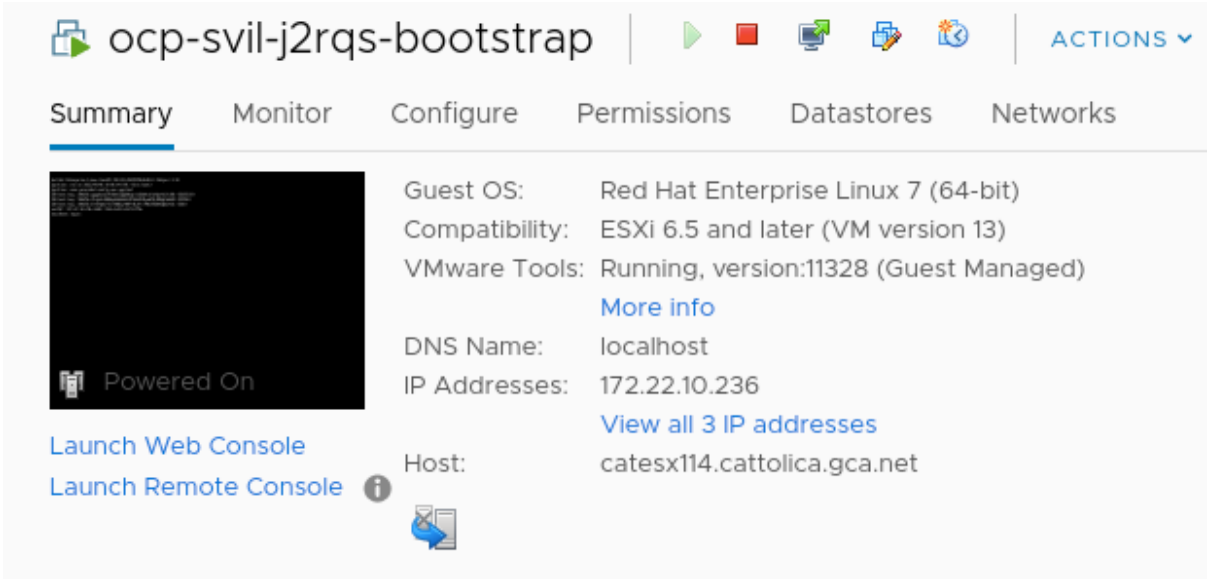
Su un'altro terminale, seguiamo la creazione del cluster dai log:

```
$ tail -f ocp-svil/.openshift_install.log
```

Contemporaneamente, seguiamo la creazione del cluster dal vSphere Client:



Le vm partono e ricevono correttamente un indirizzo IP dal DHCP:



Attendiamo quindi che il completamento dell'installazione:

```
$ openshift-install create cluster --dir ocp-svil --log-level=info
INFO Consuming Install Config from target directory
INFO Obtaining RHCOS image file from
'https://rhcos.mirror.openshift.com/art/storage/releases/rhcos-4.10/410.84.202207061638
-0/x86_64/rhcos-410.84.202207061638-0-vmware.x86_64.ova?sha256='
INFO The file was found in cache:
~/.cache/openshift-installer/image_cache/rhcos-410.84.202207061638-0-vmware.x86_64.ova.
Reusing...
INFO Creating infrastructure resources...
INFO Waiting up to 20m0s (until 12:26PM) for the Kubernetes API at
https://api.ocp-svil.gruppocattolica.it:6443...
INFO API v1.23.5+012e945 up
INFO Waiting up to 30m0s (until 12:38PM) for bootstrapping to complete...
INFO Destroying the bootstrap resources...
INFO Waiting up to 40m0s (until 12:59PM) for the cluster at
https://api.ocp-svil.gruppocattolica.it:6443 to initialize...
INFO Waiting up to 10m0s (until 12:43PM) for the openshift-console route to be
created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/CattApp/OCP4/ocp-svil/auth/kubeconfig'
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp-svil.gruppocattolica.it
INFO Login to the console with user: "kubeadmin", and password: "[non riportata in
```

```
documentazione]"
INFO Time elapsed: 29m11s
```

Verifica dell'installazione

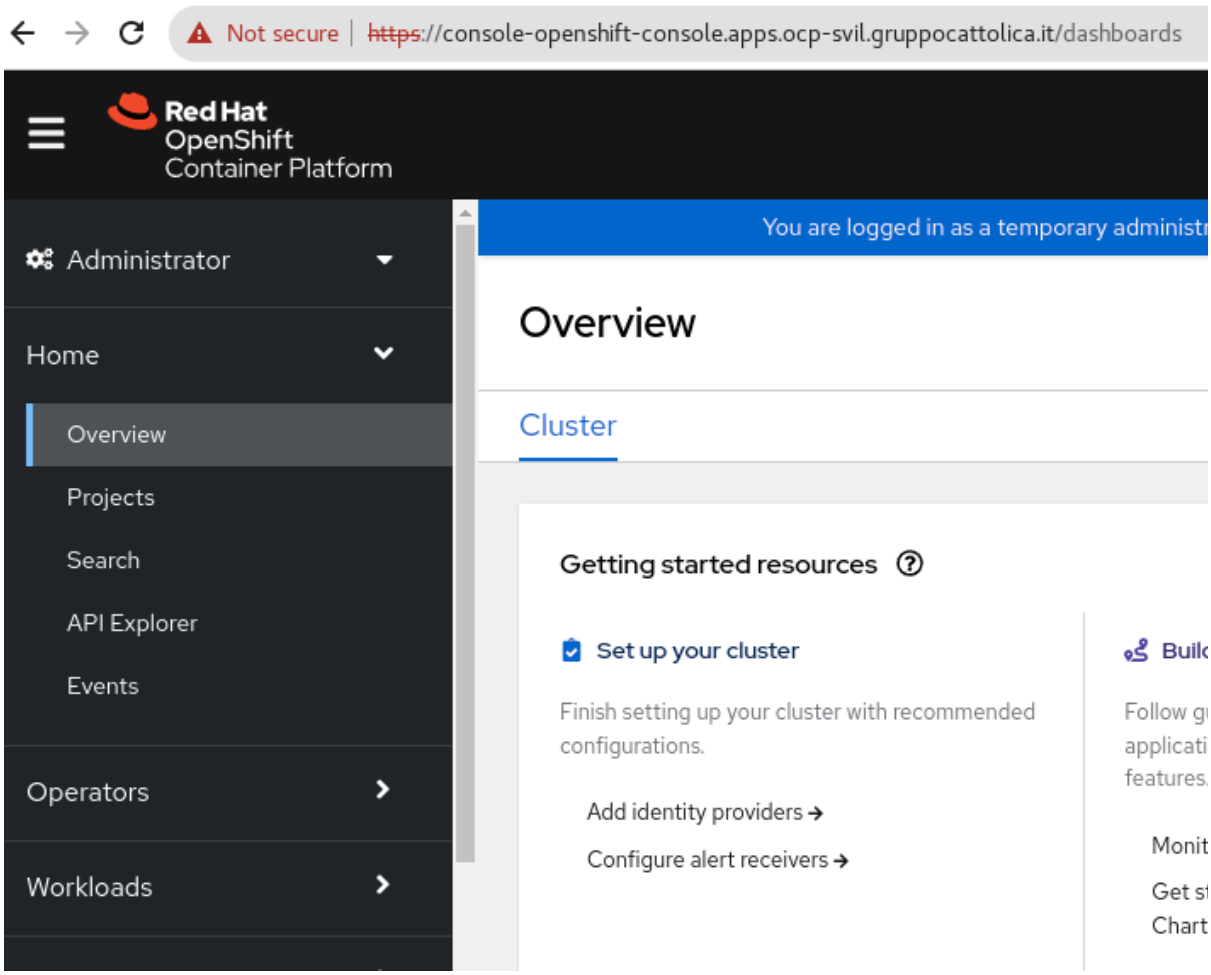
Una volta completata, utilizziamo le credenziali forniteci dall'installer e proviamo a connetterci (vedi *OpenShift Container Platform → 4.10 → CLI tools → Chapter 2. OpenShift CLI (oc) →* [*Logging in to the cluster by using the CLI*]):

```
$ oc login -u kubeadmin https://api.ocp-svil.gruppocattolica.it:6443
Login successful.

You have access to 65 projects, the list has been suppressed. You can list all projects
with 'oc projects'

Using project "default".
```

E verifichiamo la lista dei nodi:

```
$ oc get nodes
NAME                          STATUS   ROLES    AGE   VERSION
ocp-svil-j2rqs-master-0       Ready    master   23m   v1.23.5+012e945
ocp-svil-j2rqs-master-1       Ready    master   24m   v1.23.5+012e945
ocp-svil-j2rqs-master-2       Ready    master   23m   v1.23.5+012e945
ocp-svil-j2rqs-worker-lm2px   Ready    worker   12m   v1.23.5+012e945
ocp-svil-j2rqs-worker-wb4zq   Ready    worker   12m   v1.23.5+012e945
ocp-svil-j2rqs-worker-wp8m6   Ready    worker   12m   v1.23.5+012e945
```

Proviamo quindi ad accedere dalla web console:



L'installazione è andata a buon fine.

Applicazione di test

Andreamo a creare un'applicazione di test con strategia S2I in modo da verificare:

- la corretta creazione di un Pod
- la funzionalità dell'OpenShift Image Registry
- la raggiungibilità tramite IngressController

Creiamo un project di test:

```
$ oc new-project test-app
```

Creiamo un'applicazione con il S2I:

```
$ oc new-app https://github.com/sclorg/cakephp-ex
```

Attendiamo che la build sia completata:

```
$ oc get builds
NAME           TYPE     FROM          STATUS     STARTED       DURATION
cakephp-ex-1   Source   Git@69743d9   Complete   7 min ago     1m37s
```

Creiamo una rotta:

```
$ oc create route edge cakephp-ex --service=cakephp-ex
```

E proviamo a raggiungere l'applicazione da un host esterno (e.g. il bastion):

```
$ curl -k -s https://$(oc get route/cakephp-ex -o jsonpath='{.spec.host}') | head

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to OpenShift</title>
```

Il test è andato a buon fine.

Validazione dell'installazione

È possibile effettuare una validazione dell'installazione, effettuando alcuni test suggeriti nella documentazione ufficiale (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter* *24. Validating an installation*).

Tutti i test proposti nel documento sono stati effettuati e conclusi con successo. Ne riportiamo solo alcuni per archivio:

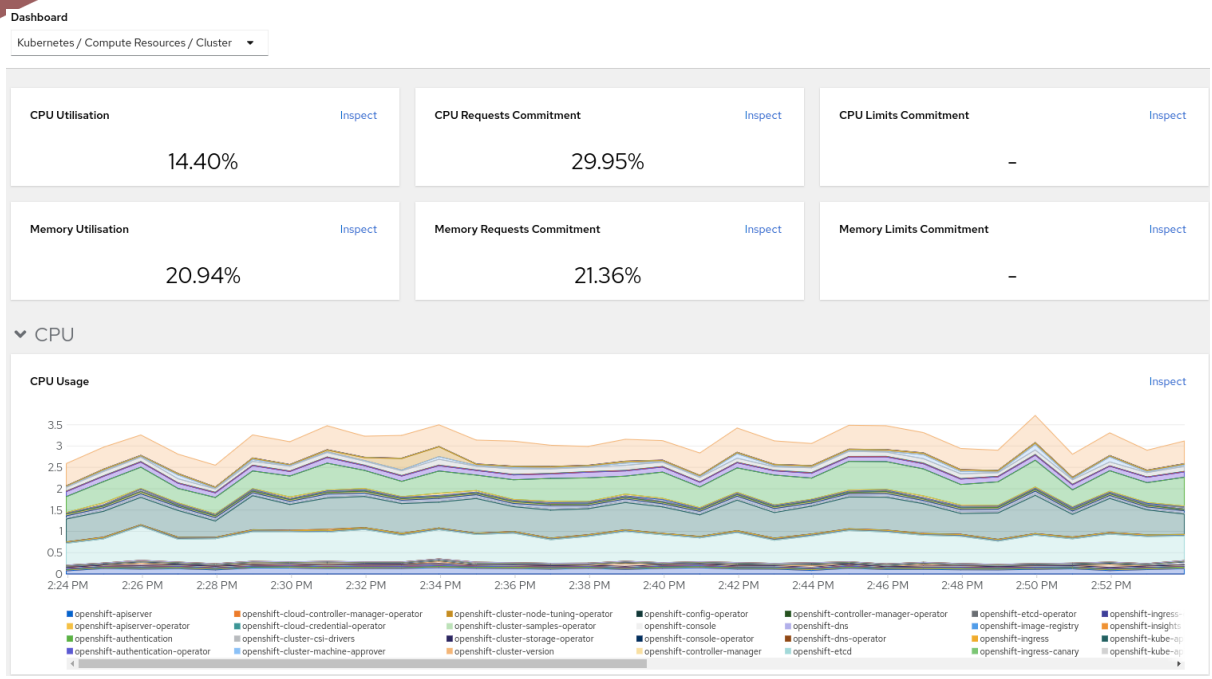
```
oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.10.28   True        False         148m    Cluster vers...
```

```
$ oc get clusterversion -o jsonpath='{.items[0].spec}{"\n"}'
{"channel":"stable-4.10","clusterID":"8fb77f5a-461e-470a-b1ec-5fe14845f61d"}
```

```
$ oc adm top nodes
NAME                          CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
ocp-svil-j2rqs-master-0       1528m        43%    5689Mi          38%
ocp-svil-j2rqs-master-1       1831m        52%    8445Mi          56%
ocp-svil-j2rqs-master-2       1319m        37%    5558Mi          37%
ocp-svil-j2rqs-worker-lm2px   558m         7%     4676Mi          15%
ocp-svil-j2rqs-worker-wb4zq   482m         6%     4298Mi          13%
ocp-svil-j2rqs-worker-wp8m6   250m         3%     2014Mi          6%
```

Di seguito riportiamo il pannello sinottico della Overview del cluster alla pagina https://console-openshift-console.apps.ocp-svil.gruppocattolica.it/dashboards

Di seguito riportiamo i grafici della dashboard generale del cluster alla pagina https://console-openshift-console.apps.ocp-svil.gruppocattolica.it/monitoring/dashboards/grafana-dashboard-k8s-resources-cluster



⚠️ NOTA BENE

Dal momento che l'installazione è stata effettuata su un cluster VMware vSphere 6.5.0, non sarà possibile passare alla versione minor successiva di OpenShift (la **4.11**) senza prima aggiornare il vSphere stesso.

Infatti, il cluster OpenShift riporta:

```
$ oc adm upgrade
Cluster version is 4.10.28
Upgradeable=False
  Reason:
VSphereCSIDriverOperatorCR_VMwareVSphereController_check_deprecated_hw_version
  Message: Cluster operator storage should not be upgraded between minor versions:
VSphereCSIDriverOperatorCRUpgradeable: VMwareVSphereControllerUpgradeable: node
ocp-svil-j2rqs-worker-lm2px has hardware version vmx-13, which is below the minimum
required version 15
```

Si raccomanda quindi di effettuare l'aggiornamento alla versione 7.0.2 o successiva sia del vCenter, sia dei nodi ESXi. Successivamente, passare alla versione hardware 15 (vedi *OpenShift Container Platform → **4.11** → Installing → Chapter 20. Installing on vSphere →* [*VMware vSphere infrastructure requirements*]).

Operazioni post installazione

Configurazione dello storage per l'Image Registry

Non avendo vSphere una sorgente di storage di default a oggetti, al termine dell'installazione l'image-registry rimane volutamente non installato.

Procederemo quindi alla preparazione di un `PersistentVolume` a blocchi, di tipo rwo, quindi senza poter scalare lo stesso Image Registry su più repliche. Ci atteniamo a quanto riportato nella documentazione ufficiale (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere →* *Configuring block registry storage for VMware vSphere*).

Editiamo la risorsa config.imageregistry.operator.openshift.io e aggiungiamo il provisioning dello storage:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster \
--type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

Quindi prepariamo il `PersistentVolume`:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage
  namespace: openshift-image-registry
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

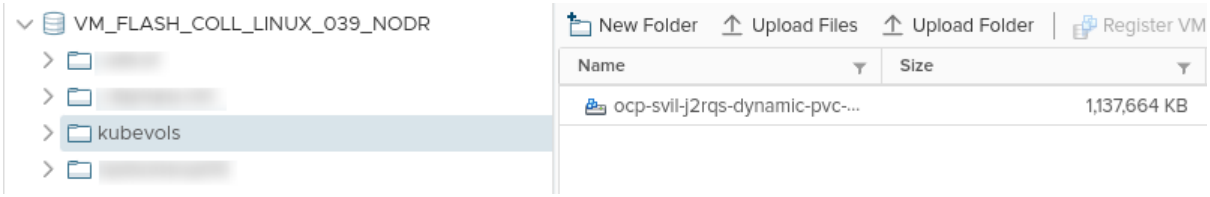
E lo creiamo:

```
$ oc apply -f image-registry-pvc.yml -n openshift-image-registry
```

Verifichiamo che il PVC sia in stato di Bound:

```
$ oc get pvc/image-registry-storage -n openshift-image-registry

image-registry-storage Bound pvc-7bee5d3b  100Gi  RWO  thin  2m
```

Un disco vmdk è stato creato nella folder `kubevols` del Datastore:



Quindi editiamo nuovamente la risorsa config.imageregistry.operator.openshift.io

```
$ oc edit config.imageregistry.operator.openshift.io/cluster
```

con le seguenti modifiche:

```
spec:
  managementState: Managed
  storage:
    pvc:
      claim: image-registry-storage
```

Attendiamo quindi che l'image-registry-operator provvede al re-deploy dei Pod dellImage Registry:

```
$ oc get co/image-registry
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry   4.10.28   True        False         False      25s
```

Autenticazione

Collegamento ad Active Directory

Creiamo un identity provider di tipo LDAP con tutte le personalizzazioni necessarie per Active Directory (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 7. Configuring identity providers →* *Configuring an LDAP identity provider*).

Andiamo a definire nel file `ldap-secret.yml` un secret con la password dell'utente **srv_ocp4_user** che avrà il ruolo di bindDN:

```
apiVersion: v1
stringData:
  bindPassword: [non riportata in documentazione]
kind: Secret
metadata:
  name: ldap-secret
type: Opaque
```

e lo creiamo:

```
$ oc apply -f ldap-secret.yml -n openshift-config
```

Modifichiamo la risorsa `oauth/cluster` e aggiungiamo il provider LDAP:

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  annotations:
    include.release.openshift.io/ibm-cloud-managed: "true"
    include.release.openshift.io/self-managed-high-availability: "true"
    include.release.openshift.io/single-node-developer: "true"
    release.openshift.io/create-only: "true"
  name: cluster
spec:
  identityProviders:
  - ldap:
      attributes:
        email:
        - mail
        id:
        - sAMAccountName
        name:
```

```
        - displayName
        preferredUsername:
        - sAMAccountName
      bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
      bindPassword:
        name: ldap-secret

      # Nel caso si volesse passare l'autenticazione ad ActiveDirectory a trust,
abilitare la ca
      # e configurare la chiave "insecure: false"
      # ca:
      #   name: ca-config-map
      insecure: true

      # Nella seguente chiave url sono indicati i soli gruppi che possono autenticarsi
al cluster.
      url:
"ldap://cattolica.gca.net:389/dc=cattolica,dc=gca,dc=net?sAMAccountName?sub?(&(objectCl
ass=user)(memberOf:1.2.840.113556.1.4.1941:=CN=OCP4_NoProd_Login,OU=Openshift,OU=Utenze
di Servizio,OU=All Users,DC=cattolica,DC=gca,DC=net))"
    mappingMethod: claim
    name: Cattolica Active Directory
    type: LDAP
```

In questo modo, possono effettuare la login solo gli utenti filtrati dalla seguente regola:

```
(&(objectClass=user)(memberOf:1.2.840.113556.1.4.1941:=CN=OCP4_NoProd_Login,OU=Openshif
t,OU=Utenze di Servizio,OU=All Users,DC=cattolica,DC=gca,DC=net))
```

Il gruppo **OCP4_NoProd_Login** è un gruppo fittizio che contiene tutti gli altri gruppi del ramo **OU=Openshift**. Questo perché non è possibile insistere su una OU in un filtro utente. Mentre la parte **memberOf:1.2.840.113556.1.4.1941:=** permette di ottenere ricorsivamente tutti gli utenti che fanno parte di un certo gruppo (vedi Active Directory Group Related Searches).

E sostituiamo la configurazione con un replace:

```
$ oc replace -f oauth-cluster.yml
```

Attendiamo quindi che l'authentication-operator provvede al re-deploy dei Pod del servizio oAut:

```
$ oc get co/authentication
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
```

```
authentication   4.10.28   True        False         False      16s
```

Quindi proviamo una login con utenza aziendale:

```
$ oc login -u benedettia https://api.ocp-svil.gruppocattolica.it:6443
```

Sincronizzazione dei gruppi Active Directory

A differenza degli utenti, la creazione dei gruppi non passa dall'oAuth provider. Dobbiamo quindi procedere con una sincronizzazione diretta verso LDAP (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 17. Syncing LDAP groups →* [*About the Active Directory configuration file*]).

Creiamo un file chiamato `sync.yaml`:

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://cattolica.gca.net:389
bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
bindPassword: [non riportata in documentazione]
insecure: true
augmentedActiveDirectory:
  groupsQuery:
    baseDN: "OU=Openshift,OU=Utenze di Servizio,OU=All
Users,dc=cattolica,dc=gca,dc=net"
    scope: sub
    derefAliases: never
    pageSize: 100
  groupUIDAttribute: dn
  groupNameAttributes: [ cn ]
  usersQuery:
    baseDN: "dc=cattolica,dc=gca,dc=net"
    scope: sub
    derefAliases: never
    filter: (objectClass=user)
    pageSize: 100
  userNameAttributes: [ sAMAccountName ]
  groupMembershipAttributes: [ memberOf ]
```

In questo modo, verranno sincronizzati automaticamente tutti i gruppi facente parte del ramo **OU=Openshift**. La ricerca degli utenti deve avere invece il baseDN generale perché i membri dei gruppi potrebbero trovarsi in un ramo qualsiasi dell'albero LDAP.

La risorsa `LDAPSyncConfig` in realtà non esiste in OpenShift. Per usarla dobbia passarla al seguente comando:

```
$ oc adm groups sync --sync-config=config.yaml --confirm
```

Si può ripetere questo comando ogni volta che si vorranno sincronizzare manualmente i gruppi di AD con quelli di OpenShift.

Sincronizzazione automatica

Oltre alla possibilità di sincronizzare i gruppi manualmente, abbiamo deciso di programmare una sincronizzazione automatica su base giornaliera (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 17. Syncing LDAP groups →* *Automatically syncing LDAP groups*).

Innanzitutto creiamo un project dove andremo a definire una risorsa `CronJob` che si occupi di effettuare la sincronizzazione ad intervalli regolari:

```
$ oc new-project ldap-sync
```

Quindi andiamo a definire nel file `ldap-secret.yml` lo stesso secret con la password di Active Directory che avevamo creato per l'identity provider:

```
apiVersion: v1
stringData:
  bindPassword: [non riportata in documentazione]
kind: Secret
metadata:
  name: ldap-secret
type: Opaque
```

e lo creiamo:

```
$ oc apply -f ldap-sync-secret.yml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-service-account.yaml` il `ServiceAccount` che verrà utilizzato per la sync:

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: ldap-group-syncer
```

```
  namespace: ldap-sync
```

e lo creiamo:

```
$ oc apply -f ldap-sync-service-account.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cluster-role.yaml` il `ClusterRole` che verrà utilizzato per la sync:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ldap-group-syncer
rules:
  - apiGroups:
      - ''
      - user.openshift.io
    resources:
      - groups
    verbs:
      - get
      - list
      - create
      - update
```

e lo creiamo:

```
$ oc apply -f ldap-sync-cluster-role.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cluster-role-binding.yaml` la `ClusterRoleBinding` che verrà utilizzata per la sync:

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ldap-group-syncer
subjects:
  - kind: ServiceAccount
    name: ldap-group-syncer
    namespace: ldap-sync
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
```

```
  name: ldap-group-syncer
```

e la creiamo:

```
$ oc apply -f ldap-sync-cluster-role-binding.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-config-map.yaml` la `ConfigMap` che verrà utilizzata per la sync:

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
data:
  # Il file sync.yaml fornisce le configurazioni usate nella CronJob ldap-group-syncer
  # dal comando 'oc adm groups sync'.
  # La password viene recuperata nel CronJob direttamente dal Secret.
  sync.yaml: |
    kind: LDAPSyncConfig
    apiVersion: v1
    url: ldap://cattolica.gca.net:389
    bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
    bindPassword:
      file: "/etc/secrets/bindPassword"
    insecure: true
    augmentedActiveDirectory:
      groupsQuery:
        baseDN: "OU=Openshift,OU=Utenze di Servizio,OU=All
Users,dc=cattolica,dc=gca,dc=net"
        scope: sub
        derefAliases: never
        pageSize: 100
      groupUIDAttribute: dn
      groupNameAttributes: [ cn ]
      usersQuery:
        baseDN: "dc=cattolica,dc=gca,dc=net"
        scope: sub
        derefAliases: never
        filter: (objectClass=user)
        pageSize: 100
      userNameAttributes: [ sAMAccountName ]
```

```
      groupMembershipAttributes: [ memberOf ]
```

e la creiamo:

```
$ oc apply -f ldap-sync-config-map.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cron-job.yaml` il `CronJob` che verrà utilizzato per la sync:

```
kind: CronJob
apiVersion: batch/v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
spec:
  schedule: "13 0 * * *"
  concurrencyPolicy: Forbid
  jobTemplate:
    spec:
      backoffLimit: 0
      ttlSecondsAfterFinished: 1800
      template:
        spec:
          containers:
            - name: ldap-group-sync
              image: "registry.redhat.io/openshift4/ose-cli:latest"
              command:
                - "/bin/bash"
                - "-c"
                - "oc adm groups sync --sync-config=/etc/config/sync.yaml --confirm"
              volumeMounts:
                - mountPath: "/etc/config"
                  name: "ldap-sync-volume"
                - mountPath: "/etc/secrets"
                  name: "ldap-bind-password"
                # Per utilizzare una connessione trust su TLS
                # definire una ca e decommentare le seguenti righe:
                #- mountPath: "/etc/ldap-ca"
                #  name: "ldap-ca"
          volumes:
            - name: "ldap-sync-volume"
              configMap:
                name: "ldap-group-syncer"
            - name: "ldap-bind-password"
```

```
            secret:
              secretName: "ldap-secret"
          # Per utilizzare una connessione trust su TLS
          # definire una ca e decommentare le seguenti righe:
          #- name: "ldap-ca"
          #  configMap:
          #    name: "ca-config-map"
        restartPolicy: "Never"
        terminationGracePeriodSeconds: 30
        activeDeadlineSeconds: 500
        dnsPolicy: "ClusterFirst"
        serviceAccountName: "ldap-group-syncer"
```

> ⚠️ **NOTA BENE**
>
> È necessario indicare il comando `oc` che il Job dovrà eseguire, esattamente come provato precedentemente nella sincronizzazione manuale.

e lo creiamo:

```
$ oc apply -f ldap-sync-cron-job.yaml -n ldap-sync
```

A questo punto non ci resta che monitorare i Job e i gruppi che ci aspettiamo vengano creati.

Autorizzazione

Una volta disponibili i gruppi Active Directory, procediamo con l'assegnazione dei permessi tramite paradigma RBAC.

Aggiunta del ruolo cluster-admin

Aggiungiamo a tutti gli utenti del gruppo **OCP4_NoProd_ClusterAdmins** i permessi forniti dal ruolo **cluster-admin** (gli stessi privilegi dell'utente **kubeadmin**):

```
$ oc adm policy add-cluster-role-to-group cluster-admin \
OCP4_NoProd_ClusterAdmins --rolebinding-name=cluster-admins
```

Rimozione del self-provisioning

Per evitare che qualsiasi utente LDAP loggato al cluster possa crearsi un nuovo Project/Namespace, rimuoviamo il role **self-provisioner** a tutti i futuri utenti loggati:

```
$ oc adm policy remove-cluster-role-from-group \
self-provisioner system:authenticated:oauth
```

Cluster OCP-TEST (NoProd)

Dal momento che si tratta di un altro cluster NoProd con le medesime configurazioni dei precedenti, riportiamo in questa sezione le sole modifiche da apportare rispetto al cluster OCP-SVIL.

Per i dettagli, attenersi alla documentazione di OCP-SVIL.

Operazioni pre installazione

Configurazioni di rete

Di seguito sono riportati i dati della configurazione di rete per il cluster.

Label	`OCP4_TEST`
Subnet	`172.22.10.192`
Mask	`27`
GW	`.193`
API	`.196`
IGRES	`.197`
DHCP Scope	`.198 - .222`

Configurazioni vSphere

Di seguito sono riportati i dati per l'installazione del cluster su VMware vSphere.

vCenter FQDN	`catvcsa101.cattolica.gca.net`
Username	`cattolica\srv_ocp4_user`
Password	*[non riportata in documentazione]*
Datacenter	`CAT-ROZZANO`
Datastore	`VM_FLASH_COLL_LINUX_039_NODR`
Network	`OCP4_TEST`
Cluster	`CAT_LINUX`
OCP API VIP	`api.ocp-test.gruppocattolica.it`
OCP Ingress VIP	`*.apps.ocp-test.gruppocattolica.it`

Cluster OCP-BF (NoProd)

Dal momento che si tratta di un altro cluster NoProd con le medesime configurazioni dei precedenti, riportiamo in questa sezione le sole modifiche da apportare rispetto al cluster OCP-SVIL.

Per i dettagli, attenersi alla documentazione di OCP-SVIL.

Operazioni pre installazione

Configurazioni di rete

Di seguito sono riportati i dati della configurazione di rete per il cluster.

Label	`OCP4_BF`
Subnet	`172.22.10.160`
Mask	`27`
GW	`.161`
API	`.164`
IGRES	`.165`
DHCP Scope	`.166 - .190`

Configurazioni vSphere

Di seguito sono riportati i dati per l'installazione del cluster su VMware vSphere.

vCenter FQDN	`catvcsa101.cattolica.gca.net`
Username	`cattolica\srv_ocp4_user`
Password	*[non riportata in documentazione]*
Datacenter	`CAT-ROZZANO`
Datastore	`VM_FLASH_COLL_LINUX_039_NODR`
Network	`OCP4_BF`
Cluster	`CAT_LINUX`
OCP API VIP	`api.ocp-bf.gruppocattolica.it`
OCP Ingress VIP	`*.apps.ocp-bf.gruppocattolica.it`

Cluster OCP-COLL (NoProd)

Dal momento che si tratta di un altro cluster NoProd con le medesime configurazioni dei precedenti, riportiamo in questa sezione le sole modifiche da apportare rispetto al cluster OCP-SVIL.

Per i dettagli, attenersi alla documentazione di OCP-SVIL.

Operazioni pre installazione

Configurazioni di rete

Di seguito sono riportati i dati della configurazione di rete per il cluster.

Label	`OCP4_COLL`
Subnet	`172.22.10.128`
Mask	`27`
GW	`.129`
API	`.132`
IGRES	`.133`
DHCP Scope	`.134 - .158`

Configurazioni vSphere

Di seguito sono riportati i dati per l'installazione del cluster su VMware vSphere.

vCenter FQDN	`catvcsa101.cattolica.gca.net`
Username	`cattolica\srv_ocp4_user`
Password	*[non riportata in documentazione]*
Datacenter	`CAT-ROZZANO`
Datastore	`VM_FLASH_COLL_LINUX_039_NODR`
Network	`OCP4_COLL`
Cluster	`CAT_LINUX`
OCP API VIP	`api.ocp-coll.gruppocattolica.it`
OCP Ingress VIP	`*.apps.ocp-coll.gruppocattolica.it`

Cluster OCP-PROD (Prod)

In questa sezione verranno riportate tutte le operazioni pre, durante e post installazione del cluster in oggetto.

Operazioni pre installazione

Configurazioni di rete

Di seguito sono riportati i dati della configurazione di rete per il cluster.

Label	OCP4_PROD
Subnet	172.22.10.0
Mask	26
GW	.1
API	.32
IGRES	.33
DHCP Scope	.34 - .62

Configurazioni vSphere

Di seguito sono riportati i dati per l'installazione del cluster su VMware vSphere.

vCenter FQDN	catvcsa101.cattolica.gca.net
Username	cattolica\srv_ocp4_user
Password	*[non riportata in documentazione]*
Datacenter	CAT-ROZZANO
Datastore	VM_FLASH_PROD_213_DR
Network	OCP4_PROD
Cluster	CAT_LINUX
OCP API VIP	api.ocp-prod.gruppocattolica.it
OCP Ingress VIP	*.apps.ocp-prod.gruppocattolica.it

Test DNS e allocazione IP

Effettuiamo le prove per verificare la presenza dei record DNS dell'API server e dell'Ingress Controller.

```
$ dig +short A api.ocp-prod.gruppocattolica.it
172.22.10.32

$ dig +short A testwc.apps.ocp-prod.gruppocattolica.it
172.22.10.33

$ dig +short -x 172.22.10.32
api.ocp-prod.gruppocattolica.it
```

Installazione del cluster

Per l'installazione del cluster OCP su VMware vSphere ci atteniamo al documento Installing a cluster on vSphere with network customizations (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere*).

Preparazione dell'install-config.yml

Creiamo una cartella dedicata al cluster:

```
$ mkdir -p /CattApp/OCP4/ocp-svil/

$ cd /CattApp/OCP4/ocp-svil/
```

All'interno della cartella, prepariamo il file `install-config.yml` per installare il cluster in modalità unattended:

```
apiVersion: v1
baseDomain: gruppocattolica.it
compute:
- architecture: amd64
  hyperthreading: Enabled
  name: worker
  platform:
    vsphere:
      cpus: 8
      coresPerSocket: 2
```

```
      memoryMB: 32768
      osDisk:
        diskSizeGB: 150
  replicas: 2
controlPlane:
  architecture: amd64
  hyperthreading: Enabled
  name: master
  platform:
    vsphere:
      cpus: 8
      coresPerSocket: 2
      memoryMB: 32768
      osDisk:
        diskSizeGB: 150
  replicas: 3
metadata:
  name: ocp-prod
networking:
  clusterNetwork:
  - cidr: 10.128.0.0/14
    hostPrefix: 23
  machineNetwork:
  - cidr: 172.22.10.0/26
  networkType: OpenShiftSDN
  serviceNetwork:
  - 172.30.0.0/16
platform:
  vsphere:
    apiVIP: 172.22.10.32
    cluster: CAT-LINUX
    datacenter: CAT-ROZZANO
    defaultDatastore: VM_FLASH_PROD_213_DR
    diskType: thick
    folder: ocp-prod
    ingressVIP: 172.22.10.33
    network: OCP4_PROD
    password: [non riportata in documentazione]
    username: cattolica\srv_ocp4_user
    vCenter: catvcsa101.cattolica.gca.net
publish: External
pullSecret: '{"auths":{"cloud.openshift.com":{"auth":...}}}'
sshKey: 'ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIBuFTRyRxcH7UDzl9jXzGQYwNW+ICpENzY5vm4tEWmKR'
```

Installazione del cluster

Quindi procediamo con la creazione degli asset di configurazione:

```
$ openshift-install create install-config --dir ocp-prod
```

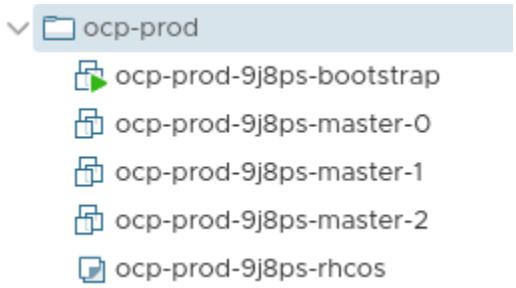
Se non ci sono errori, installiamo il cluster con:

```
$ openshift-install create cluster --dir ocp-svil --log-level=info
```

Su un'altro terminale, seguiamo la creazione del cluster dai log:

```
$ tail -f ocp-svil/.openshift_install.log
```

Contemporaneamente, seguiamo la creazione del cluster dal vSphere Client:



Le vm partono e ricevono correttamente un indirizzo IP dal DHCP:

Attendiamo quindi che il completamento dell'installazione:

```
$ openshift-install create cluster --dir ocp-prod --log-level=info
INFO Consuming Install Config from target directory
INFO Obtaining RHCOS image file from
'https://rhcos.mirror.openshift.com/art/storage/releases/rhcos-4.10/410.84.202207061638
-0/x86_64/rhcos-410.84.202207061638-0-vmware.x86_64.ova?sha256='
INFO The file was found in cache:
~/.cache/openshift-installer/image_cache/rhcos-410.84.202207061638-0-vmware.x86_64.ova.
Reusing...
INFO Creating infrastructure resources...
INFO Waiting up to 20m0s (until 12:26PM) for the Kubernetes API at
https://api.ocp-prod.gruppocattolica.it:6443...
INFO API v1.23.5+012e945 up
INFO Waiting up to 30m0s (until 12:38PM) for bootstrapping to complete...
INFO Destroying the bootstrap resources...
INFO Waiting up to 40m0s (until 12:59PM) for the cluster at
https://api.ocp-prod.gruppocattolica.it:6443 to initialize...
INFO Waiting up to 10m0s (until 12:43PM) for the openshift-console route to be
created...
INFO Install complete!
INFO To access the cluster as the system:admin user when using 'oc', run 'export
KUBECONFIG=/CattApp/OCP4/ocp-prod/auth/kubeconfig'
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp-prod.gruppocattolica.it
INFO Login to the console with user: "kubeadmin", and password: "[non riportata in
documentazione]"
INFO Time elapsed: 29m11s
```

Creazione dei MachineSet per i worker

A differenza dei cluster di NoProd, il cluster di produzione conta 6 nodi worker. Per questioni di disponibilità di storage, è stato scelto di distribuire i nodi su tre Datastore diversi:

MachineSet	**Datastore**
ocp-prod-6vvc5-worker-209	VM_FLASH_PROD_209_DR
ocp-prod-6vvc5-worker-215	VM_FLASH_PROD_215_DR
ocp-prod-6vvc5-worker-258	VM_FLASH_PROD_258_DR

Definiamo quindi il primo MachineSet come nell'esempio:

```
apiVersion: machine.openshift.io/v1beta1
kind: MachineSet
metadata:
  labels:
    machine.openshift.io/cluster-api-cluster: ocp-prod-6vvc5
  name: ocp-prod-6vvc5-worker-209
  namespace: openshift-machine-api
spec:
  replicas: 2
  selector:
    matchLabels:
      machine.openshift.io/cluster-api-cluster: ocp-prod-6vvc5
      machine.openshift.io/cluster-api-machineset: ocp-prod-6vvc5-worker-209
  template:
    metadata:
      labels:
        machine.openshift.io/cluster-api-cluster: ocp-prod-6vvc5
        machine.openshift.io/cluster-api-machine-role: worker
        machine.openshift.io/cluster-api-machine-type: worker
        machine.openshift.io/cluster-api-machineset: ocp-prod-6vvc5-worker-209
    spec:
      lifecycleHooks: {}
      metadata: {}
      providerSpec:
        value:
          apiVersion: machine.openshift.io/v1beta1
          credentialsSecret:
            name: vsphere-cloud-credentials
          diskGiB: 150
          kind: VSphereMachineProviderSpec
          memoryMiB: 32768
```

```
        metadata:
          creationTimestamp: null
        network:
          devices:
          - networkName: OCP4_PROD
        numCPUs: 8
        numCoresPerSocket: 2
        snapshot: ""
        template: ocp-prod-6vvc5-rhcos
        userDataSecret:
          name: worker-user-data
        workspace:
          datacenter: CAT-ROZZANO
          datastore: VM_FLASH_PROD_209_DR
          folder: /CAT-ROZZANO/vm/ocp-prod
          resourcePool: /CAT-ROZZANO/host/CAT-LINUX/Resources
          server: catvcsa101.cattolica.gca.net
```

e lo creiamo:

```
$ oc apply -f ocp-prod-6vvc5-worker-209.yml
```

e attendiamo la creazione delle relative `Machine` e successivi `Node`:

```
$ oc get machinesets -n openshift-machine-api
NAME                         DESIRED   CURRENT   READY   AVAILABLE   AGE
ocp-prod-6vvc5-worker        2         2         2       2           47h
ocp-prod-6vvc5-worker-209    2         2         2       2           47h

$ oc get machines -n openshift-machine-api
NAME                               PHASE     TYPE   REGION   ZONE   AGE
ocp-prod-6vvc5-master-0            Running                          2d
ocp-prod-6vvc5-master-1            Running                          2d
ocp-prod-6vvc5-master-2            Running                          2d
ocp-prod-6vvc5-worker-jd7km        Running                          2d
ocp-prod-6vvc5-worker-8iuyw        Running                          2d
ocp-prod-6vvc5-worker-209-b7fhh    Running                          47h
ocp-prod-6vvc5-worker-209-bbsdh    Running                          47h

$ oc get nodes
NAME                               STATUS   ROLES          AGE
ocp-prod-6vvc5-master-0            Ready    master         2d
ocp-prod-6vvc5-master-1            Ready    master         2d
```

```
ocp-prod-6vvc5-master-2           Ready    master         2d
ocp-prod-6vvc5-worker-jd7km       Ready    worker         2d
ocp-prod-6vvc5-worker-8iuyw       Ready    worker         2d
ocp-prod-6vvc5-worker-209-b7fhh   Ready    worker         47h
ocp-prod-6vvc5-worker-209-bbsdh   Ready    worker         47h
```

Ripetiamo quindi l'operazione per i successivi due `MachineSet`.

Al termine della procedura, scaliamo a 0 il `MachineSet` originario:

```
$ oc scale --replicas=0  machineset/ocp-prod-6vvc5-worker \
-n openshift-machine-api
```

Quindi attendiamo che i suoi `Node` e le sue `Machine` scompaiano e lo cancelliamo:

```
$ oc delete machineset/ocp-prod-6vvc5-worker -n openshift-machine-api
```

Verifica dell'installazione

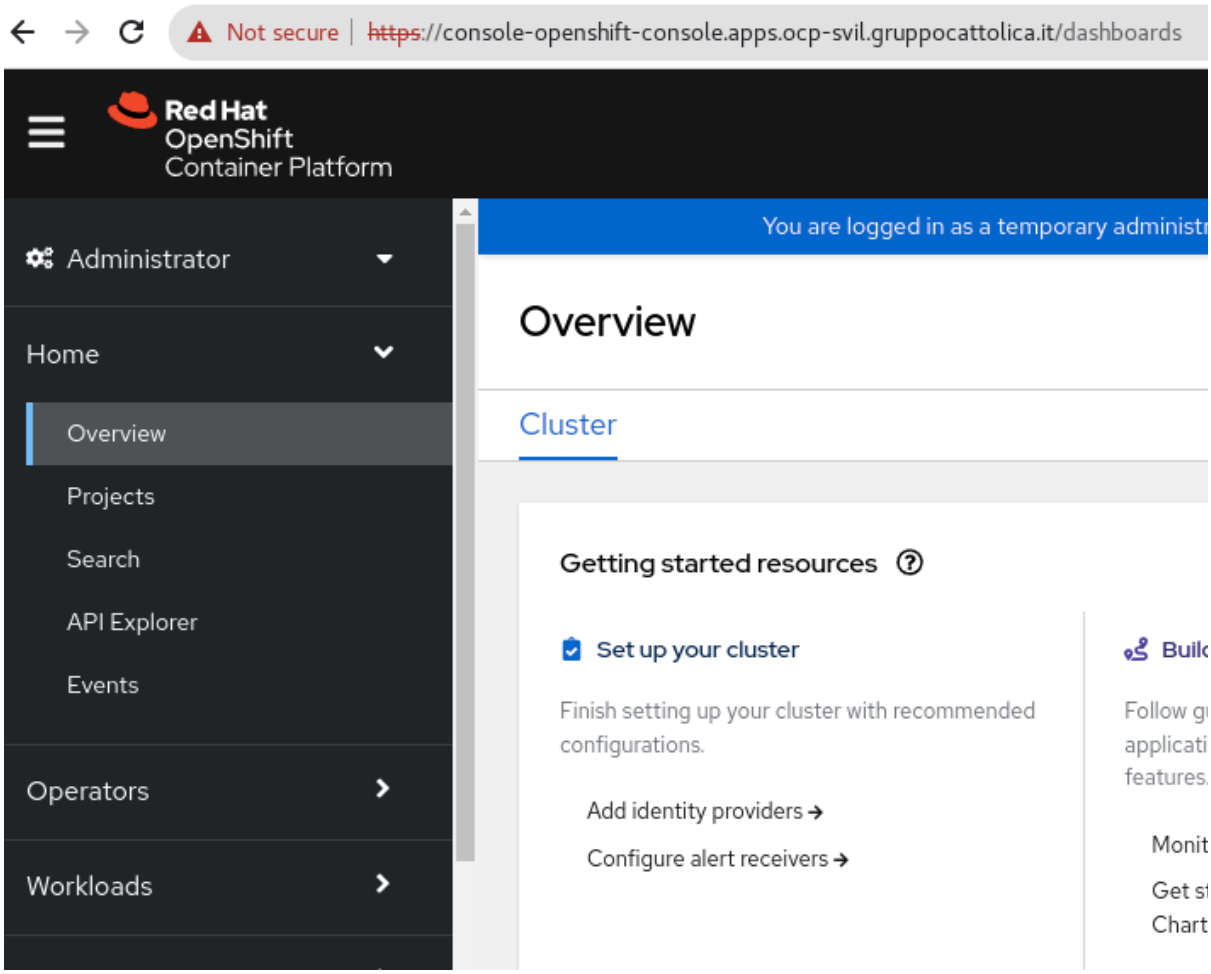
Una volta completata, utilizziamo le credenziali forniteci dall'installer e proviamo a connetterci (vedi *OpenShift Container Platform → 4.10 → CLI tools → Chapter 2. OpenShift CLI (oc) →* *Logging in to the cluster by using the CLI*):

```
$ oc login -u kubeadmin https://api.ocp-prod.gruppocattolica.it:6443
Login successful.

You have access to 65 projects, the list has been suppressed. You can list all projects
with 'oc projects'

Using project "default".
```

E verifichiamo la lista dei nodi:

```
$ oc get nodes
NAME                          STATUS   ROLES    AGE   VERSION
ocp-prod-6vvc5-master-0       Ready    master   27m   v1.23.5+012e945
ocp-prod-6vvc5-master-1       Ready    master   27m   v1.23.5+012e945
ocp-prod-6vvc5-master-2       Ready    master   27m   v1.23.5+012e945
ocp-prod-6vvc5-worker-vcngq   Ready    worker   12m   v1.23.5+012e945
ocp-prod-6vvc5-worker-zp74k   Ready    worker   12m   v1.23.5+012e945
```

Proviamo quindi ad accedere dalla web console:



L’installazione è andata a buon fine.

Applicazione di test

Andreamo a creare un'applicazione di test con strategia S2I in modo da verificare:

- la corretta creazione di un Pod
- la funzionalità dell'OpenShift Image Registry
- la raggiungibilità tramite IngressController

Creiamo un project di test:

```
$ oc new-project test-app
```

Creiamo un'applicazione con il S2I:

```
$ oc new-app https://github.com/sclorg/cakephp-ex
```

Attendiamo che la build sia completata:

```
$ oc get builds
NAME           TYPE     FROM          STATUS     STARTED       DURATION
cakephp-ex-1   Source   Git@69743d9   Complete   7 min ago     1m37s
```

Creiamo una rotta:

```
$ oc create route edge cakephp-ex --service=cakephp-ex
```

E proviamo a raggiungere l'applicazione da un host esterno (e.g. il bastion):

```
$ curl -k -s https://$(oc get route/cakephp-ex -o jsonpath='{.spec.host}') | head

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
  <title>Welcome to OpenShift</title>
```

Il test è andato a buon fine.

Validazione dell'installazione

È possibile effettuare una validazione dell'installazione, effettuando alcuni test suggeriti nella documentazione ufficiale (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter* *24. Validating an installation*).

Tutti i test proposti nel documento sono stati effettuati e conclusi con successo. Ne riportiamo solo alcuni per archivio:

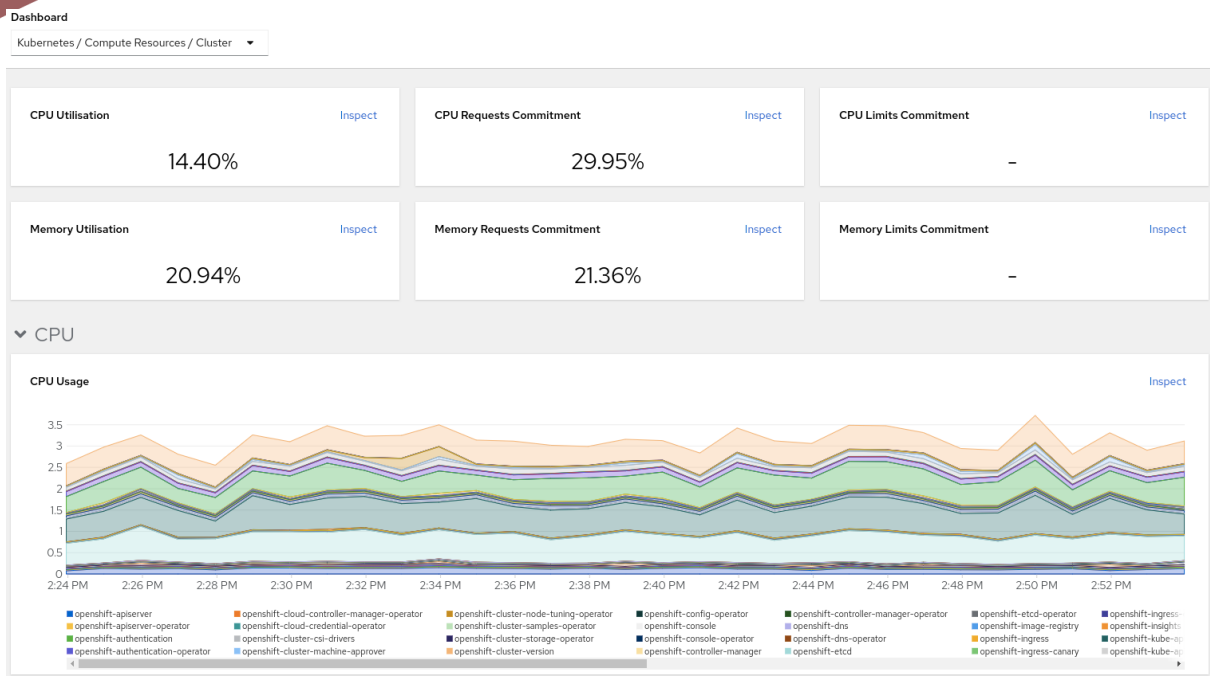
```
oc get clusterversion
NAME      VERSION   AVAILABLE   PROGRESSING   SINCE   STATUS
version   4.10.28   True        False         148m    Cluster vers...
```

```
$ oc get clusterversion -o jsonpath='{.items[0].spec}{"\n"}'
{"channel":"stable-4.10","clusterID":"3fbeac69-3e85-4463-a94b-cb73992bbdb1"}
```

```
$ oc adm top nodes
NAME                          CPU(cores)   CPU%   MEMORY(bytes)   MEM%
ocp-prod-6vvc5-master-0       1110m        14%    5075Mi          16%
ocp-prod-6vvc5-master-1       1453m        19%    5710Mi          18%
ocp-prod-6vvc5-master-2       1173m        15%    6818Mi          21%
ocp-prod-6vvc5-worker-vcngq   811m         10%    3432Mi          11%
ocp-prod-6vvc5-worker-zp74k   699m         9%     3547Mi          11%
```

Di seguito riportiamo il pannello sinottico della Overview del cluster alla pagina https://console-openshift-console.apps.ocp-svil.gruppocattolica.it/dashboards

Di seguito riportiamo i grafici della dashboard generale del cluster alla pagina https://console-openshift-console.apps.ocp-svil.gruppocattolica.it/monitoring/dashboards/grafana-dashboard-k8s-resources-cluster



⚠️ NOTA BENE

Dal momento che l'installazione è stata effettuata su un cluster VMware vSphere 6.5.0, non sarà possibile passare alla versione minor successiva di OpenShift (la **4.11**) senza prima aggiornare il vSphere stesso.

Infatti, il cluster OpenShift riporta:

```
$ oc adm upgrade
Cluster version is 4.10.28
Upgradeable=False
  Reason:
VSphereCSIDriverOperatorCR_VMwareVSphereController_check_deprecated_hw_version
  Message: Cluster operator storage should not be upgraded between minor versions:
VSphereCSIDriverOperatorCRUpgradeable: VMwareVSphereControllerUpgradeable: node
ocp-svil-j2rqs-worker-lm2px has hardware version vmx-13, which is below the minimum
required version 15
```

Si raccomanda quindi di effettuare l'aggiornamento alla versione 7.0.2 o successiva sia del vCenter, sia dei nodi ESXi. Successivamente, passare alla versione hardware 15 (vedi *OpenShift Container Platform → **4.11** → Installing → Chapter 20. Installing on vSphere →* [*VMware vSphere infrastructure requirements*]).

Operazioni post installazione

Configurazione dello storage per l'Image Registry

Non avendo vSphere una sorgente di storage di default a oggetti, al termine dell'installazione l'image-registry rimane volutamente non installato.

Procederemo quindi alla preparazione di un `PersistentVolume` a blocchi, di tipo rwo, quindi senza poter scalare lo stesso Image Registry su più repliche. Ci atteniamo a quanto riportato nella documentazione ufficiale (vedi *OpenShift Container Platform → 4.10 → Installing → Chapter 20. Installing on vSphere →* [*Configuring block registry storage for VMware vSphere*]).

Editiamo la risorsa config.imageregistry.operator.openshift.io e aggiungiamo il provisioning dello storage:

```
$ oc patch config.imageregistry.operator.openshift.io/cluster \
--type=merge -p '{"spec":{"rolloutStrategy":"Recreate","replicas":1}}'
```

Quindi prepariamo il `PersistentVolume`:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: image-registry-storage
  namespace: openshift-image-registry
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
```

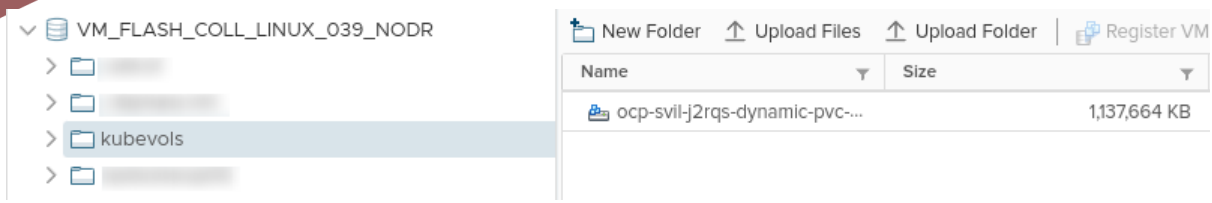
E lo creiamo:

```
$ oc apply -f image-registry-pvc.yml -n openshift-image-registry
```

Verifichiamo che il PVC sia in stato di Bound:

```
$ oc get pvc/image-registry-storage -n openshift-image-registry

image-registry-storage Bound pvc-7bee5d3b  100Gi  RWO  thin  2m
```

Un disco vmdk è stato creato nella folder `kubevols` del Datastore:



Quindi editiamo nuovamente la risorsa config.imageregistry.operator.openshift.io

```
$ oc edit config.imageregistry.operator.openshift.io/cluster
```

con le seguenti modifiche:

```
spec:
  managementState: Managed
  storage:
    pvc:
      claim: image-registry-storage
```

Attendiamo quindi che l'image-registry-operator provvede al re-deploy dei Pod dellImage Registry:

```
$ oc get co/image-registry
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
image-registry   4.10.28   True        False         False      25s
```

Autenticazione

Collegamento ad Active Directory

Creiamo un identity provider di tipo LDAP con tutte le personalizzazioni necessarie per Active Directory (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 7. Configuring identity providers →* [*Configuring an LDAP identity provider*]).

Andiamo a definire nel file `ldap-secret.yml` un secret con la password dell'utente **srv_ocp4_user** che avrà il ruolo di bindDN:

```
apiVersion: v1
stringData:
  bindPassword: [non riportata in documentazione]
kind: Secret
metadata:
  name: ldap-secret
type: Opaque
```

e lo creiamo:

```
$ oc apply -f ldap-secret.yml -n openshift-config
```

Modifichiamo la risorsa `oauth/cluster` e aggiungiamo il provider LDAP:

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  annotations:
    include.release.openshift.io/ibm-cloud-managed: "true"
    include.release.openshift.io/self-managed-high-availability: "true"
    include.release.openshift.io/single-node-developer: "true"
    release.openshift.io/create-only: "true"
  name: cluster
spec:
  identityProviders:
  - ldap:
      attributes:
        email:
        - mail
        id:
        - sAMAccountName
        name:
```

```
        - displayName
        preferredUsername:
        - sAMAccountName
      bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
      bindPassword:
        name: ldap-secret

      # Nel caso si volesse passare l'autenticazione ad ActiveDirectory a trust,
abilitare la ca
      # e configurare la chiave "insecure: false"
      # ca:
      #   name: ca-config-map
      insecure: true

      # Nella seguente chiave url sono indicati i soli gruppi che possono autenticarsi
al cluster.
      url:
"ldap://cattolica.gca.net:389/dc=cattolica,dc=gca,dc=net?sAMAccountName?sub?(&(objectCl
ass=user)(memberOf:1.2.840.113556.1.4.1941:=CN=OCP4_NoProd_Login,OU=Openshift,OU=Utenze
di Servizio,OU=All Users,DC=cattolica,DC=gca,DC=net))"
    mappingMethod: claim
    name: Cattolica Active Directory
    type: LDAP
```

In questo modo, possono effettuare la login solo gli utenti filtrati dalla seguente regola:

```
(&(objectClass=user)(memberOf:1.2.840.113556.1.4.1941:=CN=OCP4_NoProd_Login,OU=Openshif
t,OU=Utenze di Servizio,OU=All Users,DC=cattolica,DC=gca,DC=net))
```

Il gruppo **OCP4_NoProd_Login** è un gruppo fittizio che contiene tutti gli altri gruppi del ramo **OU=Openshift**. Questo perché non è possibile insistere su una OU in un filtro utente. Mentre la parte **memberOf:1.2.840.113556.1.4.1941:=** permette di ottenere ricorsivamente tutti gli utenti che fanno parte di un certo gruppo (vedi Active Directory Group Related Searches).

E sostituiamo la configurazione con un replace:

```
$ oc replace -f oauth-cluster.yml
```

Attendiamo quindi che l'authentication-operator provvede al re-deploy dei Pod del servizio oAut:

```
$ oc get co/authentication
NAME             VERSION   AVAILABLE   PROGRESSING   DEGRADED   SINCE   MESSAGE
```

```
authentication   4.10.28   True        False         False      16s
```

Quindi proviamo una login con utenza aziendale:

```
$ oc login -u benedettia https://api.ocp-svil.gruppocattolica.it:6443
```

Sincronizzazione dei gruppi Active Directory

A differenza degli utenti, la creazione dei gruppi non passa dall'oAuth provider. Dobbiamo quindi procedere con una sincronizzazione diretta verso LDAP (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 17. Syncing LDAP groups →* [*About the Active Directory configuration file*]).

Creiamo un file chiamato `sync.yaml`:

```
kind: LDAPSyncConfig
apiVersion: v1
url: ldap://cattolica.gca.net:389
bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
bindPassword: [non riportata in documentazione]
insecure: true
augmentedActiveDirectory:
  groupsQuery:
    baseDN: "OU=Openshift,OU=Utenze di Servizio,OU=All
Users,dc=cattolica,dc=gca,dc=net"
    scope: sub
    derefAliases: never
    pageSize: 100
  groupUIDAttribute: dn
  groupNameAttributes: [ cn ]
  usersQuery:
    baseDN: "dc=cattolica,dc=gca,dc=net"
    scope: sub
    derefAliases: never
    filter: (objectClass=user)
    pageSize: 100
  userNameAttributes: [ sAMAccountName ]
  groupMembershipAttributes: [ memberOf ]
```

In questo modo, verranno sincronizzati automaticamente tutti i gruppi facente parte del ramo **OU=Openshift**. La ricerca degli utenti deve avere invece il baseDN generale perché i membri dei gruppi potrebbero trovarsi in un ramo qualsiasi dell'albero LDAP.

La risorsa `LDAPSyncConfig` in realtà non esiste in OpenShift. Per usarla dobbia passarla al seguente comando:

```
$ oc adm groups sync --sync-config=config.yaml --confirm
```

Si può ripetere questo comando ogni volta che si vorranno sincronizzare manualmente i gruppi di AD con quelli di OpenShift.

Sincronizzazione automatica

Oltre alla possibilità di sincronizzare i gruppi manualmente, abbiamo deciso di programmare una sincronizzazione automatica su base giornaliera (vedi *OpenShift Container Platform → 4.10 → Authentication and authorization → Chapter 17. Syncing LDAP groups →* [*Automatically syncing LDAP groups*]).

Innanzitutto creiamo un project dove andremo a definire una risorsa `CronJob` che si occupi di effettuare la sincronizzazione a intervalli regolari:

```
$ oc new-project ldap-sync
```

Quindi andiamo a definire nel file `ldap-secret.yml` lo stesso secret con la password di Active Directory che avevamo creato per l'identity provider:

```
apiVersion: v1
stringData:
  bindPassword: [non riportata in documentazione]
kind: Secret
metadata:
  name: ldap-secret
type: Opaque
```

e lo creiamo:

```
$ oc apply -f ldap-sync-secret.yml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-service-account.yaml` il `ServiceAccount` che verrà utilizzato per la sync:

```
kind: ServiceAccount
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
```

e lo creiamo:

```
$ oc apply -f ldap-sync-service-account.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cluster-role.yaml` il `ClusterRole` che verrà utilizzato per la sync:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: ldap-group-syncer
rules:
  - apiGroups:
      - ''
      - user.openshift.io
    resources:
      - groups
    verbs:
      - get
      - list
      - create
      - update
```

e lo creiamo:

```
$ oc apply -f ldap-sync-cluster-role.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cluster-role-binding.yaml` la `ClusterRoleBinding` che verrà utilizzata per la sync:

```
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: ldap-group-syncer
subjects:
  - kind: ServiceAccount
    name: ldap-group-syncer
    namespace: ldap-sync
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: ldap-group-syncer
```

e la creiamo:

```
$ oc apply -f ldap-sync-cluster-role-binding.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-config-map.yaml` la `ConfigMap` che verrà utilizzata per la sync:

```yaml
kind: ConfigMap
apiVersion: v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
data:
  # Il file sync.yaml fornisce le configurazioni usate nella CronJob ldap-group-syncer
  # dal comando 'oc adm groups sync'.
  # La password viene recuperata nel CronJob direttamente dal Secret.
  sync.yaml: |
    kind: LDAPSyncConfig
    apiVersion: v1
    url: ldap://cattolica.gca.net:389
    bindDN: CN=srv_ocp4_user,OU=Openshift,OU=Utenze di Servizio,OU=All
Users,DC=cattolica,DC=gca,DC=net
    bindPassword:
      file: "/etc/secrets/bindPassword"
    insecure: true
    augmentedActiveDirectory:
      groupsQuery:
        baseDN: "OU=Openshift,OU=Utenze di Servizio,OU=All
Users,dc=cattolica,dc=gca,dc=net"
        scope: sub
        derefAliases: never
        pageSize: 100
      groupUIDAttribute: dn
      groupNameAttributes: [ cn ]
      usersQuery:
        baseDN: "dc=cattolica,dc=gca,dc=net"
        scope: sub
        derefAliases: never
        filter: (objectClass=user)
        pageSize: 100
      userNameAttributes: [ sAMAccountName ]
      groupMembershipAttributes: [ memberOf ]
```

e la creiamo:

```
$ oc apply -f ldap-sync-config-map.yaml -n ldap-sync
```

Andiamo a definire nel file `ldap-sync-cron-job.yaml` il `CronJob` che verrà utilizzato per la sync:

```yaml
kind: CronJob
apiVersion: batch/v1
metadata:
  name: ldap-group-syncer
  namespace: ldap-sync
spec:
  schedule: "13 0 * * *"
  concurrencyPolicy: Forbid
  jobTemplate:
    spec:
      backoffLimit: 0
      ttlSecondsAfterFinished: 1800
      template:
        spec:
          containers:
            - name: ldap-group-sync
              image: "registry.redhat.io/openshift4/ose-cli:latest"
              command:
                - "/bin/bash"
                - "-c"
                - "oc adm groups sync --sync-config=/etc/config/sync.yaml --confirm"
              volumeMounts:
                - mountPath: "/etc/config"
                  name: "ldap-sync-volume"
                - mountPath: "/etc/secrets"
                  name: "ldap-bind-password"
                # Per utilizzare una connessione trust su TLS
                # definire una ca e decommentare le seguenti righe:
                #- mountPath: "/etc/ldap-ca"
                #  name: "ldap-ca"
          volumes:
            - name: "ldap-sync-volume"
              configMap:
                name: "ldap-group-syncer"
            - name: "ldap-bind-password"
              secret:
                secretName: "ldap-secret"
            # Per utilizzare una connessione trust su TLS
            # definire una ca e decommentare le seguenti righe:
            #- name: "ldap-ca"
            #  configMap:
            #    name: "ca-config-map"
          restartPolicy: "Never"
          terminationGracePeriodSeconds: 30
          activeDeadlineSeconds: 500
```

```
          dnsPolicy: "ClusterFirst"
          serviceAccountName: "ldap-group-syncer"
```

> ⚠️ **NOTA BENE**
>
> È necessario indicare il comando `oc` che il Job dovrà eseguire, esattamente come provato precedentemente nella sincronizzazione manuale.

e lo creiamo:

```
$ oc apply -f ldap-sync-cron-job.yaml -n ldap-sync
```

A questo punto non ci resta che monitorare i Job e i gruppi che ci aspettiamo vengano creati.

Autorizzazione

Una volta disponibili i gruppi Active Directory, procediamo con l'assegnazione dei permessi tramite paradigma RBAC.

Aggiunta del ruolo cluster-admin

Aggiungiamo a tutti gli utenti del gruppo **OCP4_NoProd_ClusterAdmins** i permessi forniti dal ruolo **cluster-admin** (gli stessi privilegi dell'utente **kubeadmin**):

```
$ oc adm policy add-cluster-role-to-group cluster-admin \
OCP4_NoProd_ClusterAdmins --rolebinding-name=cluster-admins
```

Rimozione del self-provisioning

Per evitare che qualsiasi utente LDAP loggato al cluster possa crearsi un nuovo Project/Namespace, rimuoviamo il role **self-provisioner** a tutti i futuri utenti loggati:

```
$ oc adm policy remove-cluster-role-from-group \
self-provisioner system:authenticated:oauth
```

Task post-installazione

In seguito all'installazione dei vari cluster, è stato necessario andare ad aggiungere alcune componenti su tutti i cluster Openshift creati. Nello specifico:

- Integrazione con pipelines per Jenkins
- Service Account per Mia Console
- DaemonSet per Filebeat

Integrazione con pipelines per Jenkins

Per tutti i cluster è stato necessario creare un Service Account per poter eseguire le varie operazione di deploy applicativi, andando ad usare la soluzione di pipeline Jenkins.

Per fare questo sono state create tre tipologie di risorse:

- Il Service Account specifico di Jenkins (*jenkins-ci*)
- I custom roles che vengono utilizzati dal suddetto service account
- I relativi custom role binding

Service Account Jenkins

Il service account *jenkins-ci* è stato creato sotto il namespace *devops*, presente in tutti i cluster installati, con la seguente risorsa:

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: jenkins-ci
 namespace: devops
```

Sono stati estratti poi i token necessari per permettere l'autenticazione tra Jenkins e i cluster Openshift.

Custom Roles

Per svolgere le funzioni necessarie per l'esecuzione delle pipelines di Jenkins è stato necessario creare un Custom Cluster Roles per il service account[2].

[2] La specifica del Custom Roles è in Appendice 1.

Custom Role Binding

Il custom role ed il ruolo di self-provisioner sono stati assegnati al service account *jenkins-ci* mediante due cluster role binding:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: jenkins-role
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: jenkins-role
subjects:
- kind: ServiceAccount
 name: jenkins-ci
 namespace: devops
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: self-provisioner-jenkins
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: self-provisioner
subjects:
- kind: ServiceAccount
 name: jenkins-ci
 namespace: devops
```

Service Account per Mia Console

Per integrare i cluster Openshift con la console applicativa di Mia Platform è stato necessario andare a creare delle risorse dedicate. Nello specifico:

- Service Account *mia-console-sa*
- Cluster Role specifici
- Cluster Role Binding

Service Account per Mia Console

Per permettere alla console applicativa di Mia-Platform di eseguire le varie operazioni, è stato necessario andare a creare dei service account *mia-console-sa* con la seguente definizione:

```
apiVersion: v1
kind: ServiceAccount
metadata:
 name: mia-console-sa
 namespace: devops
```

Sono stati estratti poi i token necessari per permettere l'autenticazione con i cluster Openshift.

Cluster Role per Mia Console

Per fornire a questo service account i permessi adeguati, è stato necessario creare un Custom Cluster Role chiamato *mia-console-role*.[3]

[3] La specifica del Custom Roles è in Appendice 2.

Cluster Role Binding

Per assegnare questo Cluster Role al service account creato, è stato creata un cluster role binding fatto nel seguente modo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: mia-console-rolebinding
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: mia-console-role
subjects:
- kind: ServiceAccount
 name: mia-console-sa
 namespace: devops
```

Installazione Filebeat

Filebeat è usato per esportare i log dei cluster verso il cluster Kafka, il quale invia poi i log verso un cluster ELK esterno. Lo strumento è stato scelto in quanto è già in uso all'interno del perimetro di Cattolica.

Per poter fare ciò sono state create alcune risorse[4] nel namespace *filebeat-logging* creato su ogni cluster.

Oltre alla creazione delle risorse necessarie, è stato anche assegnato il **Security Context Constraint** *privileged* al service account *filebeat* con il seguente comando:

```
$ oc adm policy add-scc-to-user privileged \
   system:serviceaccount:filebeat-logging:filebeat
```

[4] Per lo yaml delle configurazioni si rimanda all'Appendice 3

Comandi utili

Si listano qui una serie di comandi *non banali* che possono tornare utili.

Problematiche di Header

DA HAProxy 2.2 (ossia da OCP 4.4) viene implementata la normalizzazione della capitalizzazione degli header HTTP ricevuti andando a imporli come lower case.

Questo, in caso di applicazioni che sono *case sensitive* per quanto riguarda gli header HTTP, può comportare delle problematiche[5].

Per risolvere la problematica si va a modificare la configurazione con il comando:

```
$ oc -n openshift-ingress-operator edit ingresscontrollers/default
```

E si va ad aggiungere la seguente parte:

```
[...]
spec:
   httpHeaders:
    headerNameCaseAdjustments:
    - [Header1]
    - ...
    - [HeaderN]
```

Con *[Header1] .. [HeaderN]* gli header nella forma (uppercase, lowercase, ecc) che vogliamo che vengano passati all'applicazione. Non è possibile disabilitare la normalizzazione degli header.

Questo non è attivo di default, ma deve essere poi applicato per singola *route*, andando ad annotarla con il comando:

```
$ oc annotate routes/route-target haproxy.router.openshift.io/h1-adjust-case=true
```

[5] **N.B.** Questo comportamento **non rispetta la specifica del protocollo HTTP**: https://www.rfc-editor.org/rfc/rfc2616

Scaling MachineSet

Dato che l'installazione è stata fatta con metodologia IPI, abbiamo come vantaggio la possibilità di gestire la creazione e la configurazione di nuovi nodi direttamente da parte di OCP.

Per fare questo andiamo a identificare quali *MachineSets* sono disponibili:

```
$ oc get machinesets.machine.openshift.io -n openshift-machine-api
```

Da questo scegliamo il *MachineSet* che vogliamo scalare e lo imponiamo con il comando:

```
$ oc scale machinesets.machine.openshift.io \
  --replicas=4 -n openshift-machine-api machineset-name
```

Dopo poco inizia automaticamente la creazione del nuovo nodo su VMWare.

Caricamento certificati custom

Per modificare i certificati degli endpoint *api.*.gruppocattolica.it* o quelli esposti dagli ingress controller bisogna andare ad applicare le seguenti operazioni.

Ingress controller certificate

Carico l'eventuale nuova CA del nuovo certificato

```
$ oc create configmap custom-ca --from-file=ca-bundle.crt=</path/to/example-ca.crt> -n
openshift-config
```

Carico il certificato **wildcard** firmato dalla CA:

```
$ oc create secret tls <secret> --cert=</path/to/cert.crt> --key=</path/to/cert.key> -n
openshift-ingress
```

Inserisco che la CA custom che ho creato è una CA Trusted

```
$ oc patch proxy/cluster --type=merge
--patch='{"spec":{"trustedCA":{"name":"custom-ca"}}}'
```

Imposto che gli ingress controller vadano ad usare i certificati custom caricati

```
$ oc patch ingresscontroller.operator default --type=merge -p
'{"spec":{"defaultCertificate": {"name": "<secret>"}}}' -n openshift-ingress-operator
```

Per modificare i certificati esposti dalle API di OpenShift, basta andare a creare un secret con il certificato custom specifico per le *api*:

```
$ oc create secret tls <secret> --cert=</path/to/cert.crt> --key=</path/to/cert.key> -n openshift-config
```

In seguito basta imporre nella configurazione dell'API operator di utilizzare il certificato caricato (l'**FQDN** in questione è l'endpoint api che viene esposto e per cui i certificati verranno utilizzati) :

```
$ oc patch apiserver cluster --type=merge -p '{"spec":{"servingCerts": {"namedCertificates": [{"names": ["<FQDN>"], "servingCertificate": {"name": "<secret>"}}]}}}'
```

Visualizzazione di consumo di risorse

Per visualizzare quante risorse sono consumate effettivamente su un nodo, possiamo lanciare il comando:

```
$ oc adm top nodes [node-name]
```

Per visualizzare quante risorse vengono consumate effettivamente da un pod possiamo lanciare il comando:

```
$ oc adm top pods -A [--sort-by=cpu|memory]
```

Con l'opzione *–container*, si va ad avere la distinta per container e non per pod.

Formattazione di output

Quando viene fatto il comando di *oc get [risorsa]* è possibile utilizzare l'opzione *-o jsonpath={query}* oppure *-o custom-coulms=[colonne da estrarre]*.

Ad esempio:

```
$ oc get pods -A -o \
jsonpath='{range.items[*]}{.metadata.name}{"\n"}{.spec.containers[*].resources.requests}{"\n"}{end}'
$ oc get pods -A -o \
custom-columns=NAME:.metadata.name,NS:.metadata.namespace,ReqCPU:.spec.containers[*].resources.requests.cpu,ReqMem:.spec.containers[*].resources.requests.memory
```

Documentazione risorse da CLI

Durante la manipolazione di risorse è molto comodo andare ad esplorare la documentazione dei vari campi delle varie risorse. Per fare questo può essere usato il comando *oc explain*:

```
$ oc explain pod
$ ox explain pod.spec.affinity
```

Appendice 1: jenkins-role Custom Roles

```
apiVersion: authorization.openshift.io/v1
kind: ClusterRole
metadata:
 name: jenkins-role
rules:
- apiGroups:
 - autoscaling
 attributeRestrictions: null
 resources:
 - horizontalpodautoscalers
 verbs:
 - get
 - list
 - watch
- apiGroups:
 - autoscaling
 attributeRestrictions: null
 resources:
 - horizontalpodautoscalers/status
 verbs:
 - update
- apiGroups:
 - batch
 attributeRestrictions: null
 resources:
 - cronjobs
 - jobs
 verbs:
 - create
 - delete
 - deletecollection
 - get
 - list
 - patch
 - update
 - watch
- apiGroups:
 - ""
 attributeRestrictions: null
 resources:
 - pods
 verbs:
 - create
 - delete
```

```
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  attributeRestrictions: null
  resources:
  - configmaps
  - replicationcontrollers
  - replicationcontrollers/scale
  - services
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  attributeRestrictions: null
  resources:
  - secrets
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  attributeRestrictions: null
  resources:
  - pods/status
  - replicationcontrollers/status
  - resourcequotas/status
  verbs:
  - get
  - list
```

```
  - watch
- apiGroups:
  - apps
  attributeRestrictions: null
  resources:
  - deployments
  - deployments/rollback
  - deployments/scale
  - replicasets
  - replicasets/scale
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  - apps.openshift.io
  attributeRestrictions: null
  resources:
  - deploymentconfigs
  - deploymentconfigs/scale
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  - apps.openshift.io
  attributeRestrictions: null
  resources:
  - deploymentconfigrollbacks
  - deploymentconfigs/instantiate
  - deploymentconfigs/rollback
  verbs:
  - create
- apiGroups:
  - ""
```

```
  - apps.openshift.io
  attributeRestrictions: null
  resources:
  - deploymentconfigs/log
  - deploymentconfigs/status
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  - image.openshift.io
  attributeRestrictions: null
  resources:
  - imagestreamimages
  - imagestreammappings
  - imagestreams
  - imagestreams/secrets
  - imagestreamtags
  verbs:
  - create
  - delete
  - deletecollection
  - get
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  - image.openshift.io
  attributeRestrictions: null
  resources:
  - imagestreams/status
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  - image.openshift.io
  attributeRestrictions: null
  resources:
  - imagestreams/layers
  verbs:
  - get
  - update
```

```
- apiGroups:
  - ""
  - image.openshift.io
  attributeRestrictions: null
  resources:
  - imagestreamimports
  verbs:
  - create
```

Appendice 2: mia-console-role Custom Role

```
apiVersion: authorization.openshift.io/v1
kind: ClusterRole
metadata:
 name: mia-console-role
rules:
- apiGroups:
 - ""
 attributeRestrictions: null
 resources:
 - pods
 verbs:
 - get
 - list
- apiGroups:
 - ""
 attributeRestrictions: null
 resources:
 - pods/log
 verbs:
 - get
 - list
- apiGroups:
 - apps
 attributeRestrictions: null
 resources:
 - deployments
 verbs:
 - get
 - list
- apiGroups:
 - ""
 - apps.openshift.io
 attributeRestrictions: null
 resources:
 - deploymentconfigs
 verbs:
 - get
 - list
- apiGroups:
 - metrics.k8s.io
 attributeRestrictions: null
```

```
resources:
- pods
verbs:
- list
```

Appendice 3: Risorse per Filebeat

Oltre alle risorse qui descritte è stato creato anche i secret contenenti i certificati per autenticarsi con il cluster Kafka target.

Inoltre, le risorse sono state customizzate a seconda (in concomitanza con i commenti nello YAML sottostante) dell'ambiente in modo opportuno.

```yaml
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: metricbeat-daemonset-config
 namespace: filebeat-logging
 labels:
   k8s-app: metricbeat
data:
 metricbeat.yml: |-
   metricbeat.config.modules:
     path: ${path.config}/modules.d/*.yml
     reload.enabled: false

   processors:
     - add_cloud_metadata:
     - add_fields:
         target: environment
         fields:
           # Cambiato a seconda del cluster in cui è stato deployato
           type: prod

   output.kafka:
     hosts: ['${KAFKA_HOST1}', '${KAFKA_HOST2}', '${KAFKA_HOST3}']
     topic: '${KAFKA_TOPIC}'
     compression: gzip
     max_message_bytes: 1000000
     ssl:
       certificate_authorities: ["/etc/certificates/ca.crt"]
       certificate: "/etc/certificates/abaco.crt"
       key: "/etc/certificates/abaco.key"
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: metricbeat-daemonset-modules
 namespace: filebeat-logging
 labels:
   k8s-app: metricbeat
data:
```

```
 beat-xpack.yml: |-
   - module: beat
     xpack.enabled: true
     period: 10s
     hosts: ["http://localhost:5067"]
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: filebeat-config
 namespace: filebeat-logging
 labels:
   k8s-app: filebeat
data:
 filebeat.yml: |-
   filebeat.inputs:
   - type: container
     paths:
       - /var/log/containers/*.log
     processors:
       - add_kubernetes_metadata:
           host: ${NODE_NAME}
           matchers:
           - logs_path:
               logs_path: "/var/log/containers/"
     multiline:
       type: pattern
       pattern: '(^\{|^\d+\.|^[A-Z])'
       negate: true
       match: after
       flush_pattern: '^\}$'
       timeout: 200ms

   processors:
     - add_cloud_metadata:
     - add_host_metadata:
     - add_fields:
         target: environment
         fields:
           # Cambiato a seconda del cluster in cui è stato deployato
           type: prod

   output.kafka:
     hosts: ['${KAFKA_HOST1}', '${KAFKA_HOST2}', '${KAFKA_HOST3}']
     topic: '${KAFKA_TOPIC}'
     compression: gzip
     max_message_bytes: 1000000
```

```
    ssl:
      certificate_authorities: ["/etc/certificates/ca.crt"]
      certificate: "/etc/certificates/abaco.crt"
      key: "/etc/certificates/abaco.key"

  monitoring.enabled: false
  monitoring.cluster_uuid: "M-qRFQe4QyWYGE_u80Eo_g"
  http.enabled: true
  http.port: 5067
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
 name: filebeat
 namespace: filebeat-logging
 labels:
   k8s-app: filebeat
spec:
 selector:
   matchLabels:
     k8s-app: filebeat
 template:
   metadata:
     labels:
       k8s-app: filebeat
   spec:
     serviceAccountName: filebeat
     terminationGracePeriodSeconds: 30
     hostNetwork: true
     dnsPolicy: ClusterFirstWithHostNet
     containers:
     - name: metricbeat
       image: docker.elastic.co/beats/metricbeat:7.11.1
       args: [
         "-c", "/etc/metricbeat.yml",
         "-e",
         "-system.hostfs=/hostfs",
       ]
       env:
       - name: KAFKA_TOPIC
         value: metricbeat
       - name: KAFKA_HOST1
         value: 172.21.244.60:9093
       - name: KAFKA_HOST2
         value: 172.21.244.61:9093
       - name: KAFKA_HOST3
         value: 172.21.244.62:9093
```

```
      - name: NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      securityContext:
        runAsUser: 0
        privileged: true
      resources:
        limits:
          memory: 200Mi
        requests:
          cpu: 100m
          memory: 100Mi
      volumeMounts:
      - name: metricbeat-config
        mountPath: /etc/metricbeat.yml
        readOnly: true
        subPath: metricbeat.yml
      - name: metricbeat-data
        mountPath: /usr/share/metricbeat/data
      - name: modules
        mountPath: /usr/share/metricbeat/modules.d
        readOnly: true
    - name: filebeat
      image: docker.elastic.co/beats/filebeat:7.11.1
      args: [
        "-c", "/etc/filebeat.yml",
        "-e",
      ]
      env:
      # Cambiato a seconda del cluster in cui è stato deployato
      - name: KAFKA_TOPIC
        value: abaco-ocp4-prod
      - name: KAFKA_HOST1
        value: 172.21.244.60:9093
      - name: KAFKA_HOST2
        value: 172.21.244.61:9093
      - name: KAFKA_HOST3
        value: 172.21.244.62:9093
      - name: NODE_NAME
        valueFrom:
          fieldRef:
            fieldPath: spec.nodeName
      securityContext:
        runAsUser: 0
        privileged: true
      resources:
```

```
        limits:
          memory: 200Mi
        requests:
          cpu: 100m
          memory: 100Mi
      volumeMounts:
      - name: filebeat-config
        mountPath: /etc/filebeat.yml
        readOnly: true
        subPath: filebeat.yml
      - name: filebeat-data
        mountPath: /usr/share/filebeat/data
      - name: varlibdockercontainers
        mountPath: /var/lib/docker/containers
        readOnly: true
      - name: varlog
        mountPath: /var/log
        readOnly: true
      - name: filebeat-credentials
        mountPath: /etc/certificates
        readOnly: true
    volumes:
    - name: metricbeat-config
      configMap:
        defaultMode: 0640
        name: metricbeat-daemonset-config
    - name: modules
      configMap:
        defaultMode: 0640
        name: metricbeat-daemonset-modules
    - name: metricbeat-data
      hostPath:
        path: /var/lib/metricbeat-data
        type: DirectoryOrCreate
    - name: filebeat-config
      configMap:
        defaultMode: 0640
        name: filebeat-config
    - name: filebeat-credentials
      secret:
        secretName: filebeat-credentials
    - name: varlibdockercontainers
      hostPath:
        path: /var/lib/docker/containers
    - name: varlog
      hostPath:
        path: /var/log
```

```
      - name: filebeat-data
        hostPath:
          path: /var/lib/filebeat-data
          type: DirectoryOrCreate
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 name: filebeat
subjects:
- kind: ServiceAccount
 name: filebeat
 namespace: filebeat-logging
roleRef:
 kind: ClusterRole
 name: filebeat
 apiGroup: rbac.authorization.k8s.io
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 name: filebeat
 labels:
    k8s-app: filebeat
rules:
- apiGroups: [""]
 resources:
 - namespaces
 - pods
 - nodes
 verbs:
 - get
 - watch
 - list
- apiGroups: ["apps"]
 resources:
    - replicasets
 verbs: ["get", "list", "watch"]
---
apiVersion: v1
kind: ServiceAccount
metadata:
 name: filebeat
 namespace: filebeat-logging
 labels:
    k8s-app: filebeat
---
```