Consiglio Nazionale delle Ricerche
Istituto di Calcolo e Reti ad Alte Prestazioni

Creazione, installazione e deploy di una piattaforma basata sullo stack Elastic Search tramite docker e Kubernetes per lo sviluppo di applicazioni di cybersecurity

Gianluigi Folino, Sabrina Celia

RT-ICAR-CS-21-10 **Novembre 2021**

Consiglio Nazionale delle Ricerche, Istituto di Calcolo e Reti ad Alte Prestazioni (ICAR)
– Sede di Cosenza, Via P. Bucci 8-9C, 87036 Rende, Italy, URL: www.icar.cnr.it
– Sezione di Napoli, Via P. Castellino 111, 80131 Napoli, URL: www.icar.cnr.it
– Sezione di Palermo, Via Ugo La Malfa, 153, 90146 Palermo, URL: www.icar.cnr.it

SOMMARIO

Introduzione

Docker, è un progetto open source che utilizza le funzionalità del kernel **Linux** per garantire l'isolamento fra container ed è delle tecnologie più importanti basate sui container. Prima della diffusione in produzione dei container lo sviluppo software si serviva prevalentemente della virtualizzazione, tecnologia che mediante l'utilizzo di hypervisor (di tipo 1 o di tipo 2) permette condividere il medesimo hardware di un server fisico tra più sistemi operativi eseguiti in macchine virtuali differenti, la virtualizzazione ha sicuramente molti vantaggi (isolamento, riduzione dei costi etc.), ma porta con sé degli svantaggi rispetto ai container. I container, rispetto alle macchine virtuali, permettono di risparmiare in termini di **risorse utilizzate** per il loro funzionamento.

Kubernetes, basato su tecnologia open source, permette l'orchestrazione dei container in modo da garantire anche una certa resilienza e scalabilità dell'infrastruttura.

Lo stack *Elastic Search* è formato da strumenti open source pensati per raccogliere, analizzare e visualizzare dati in tempo reale. E' formato da tre tool principali: Elasticsearch, Logstash e Kibana, che lo hanno reso una delle soluzioni più efficaci ed efficienti nell'ambito della raccolta e analisi dati, soprattutto nell'ambito della cybersecurity, con le sue componenti aggiuntive: Elastic Security e le componenti di Machine Learning e Anomaly Detection.

Perciò in questo rapporto si è scelto di usare Kubernetes, come strumento di orchestrazione e specificare tutto la parte tecnica che ha riguardato l'installazione di una Virtual Machine, la configurazione del Docker Registry, e infine la creazione, la pubblicazione e il deploy di un'immagine basata su Elastic Search.

Tale architettura che può girare su cloud o su cluster ad alte prestazioni potrà essere utilizzata per sviluppare soluzioni per la cybersecurity basata sullo stack Elastic Search.

Di seguito, un indice dei passi utilizzati per il deploy e l'installazione e, quindi, tutta la descrizione dell'intero processo.

Creazione della VM

Prima di creare una VM è necessario caricare l'iso della distribuzione da installare nel datastore

Nella sezione storage, si seleziona il datastore e tramite la funzione Datastore browser si esegue l'upload di un'immagine iso.

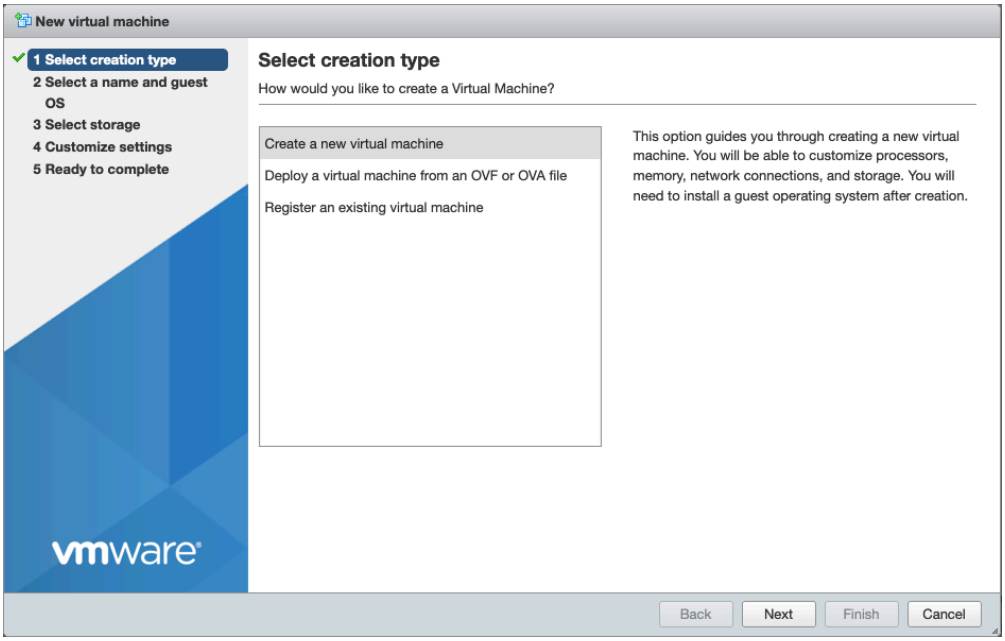
Ad esempio si può usare

https://cdimage.debian.org/cdimage/archive/10.10.0/amd64/iso-cd/debian-10.10.0-amd64-netinst.iso

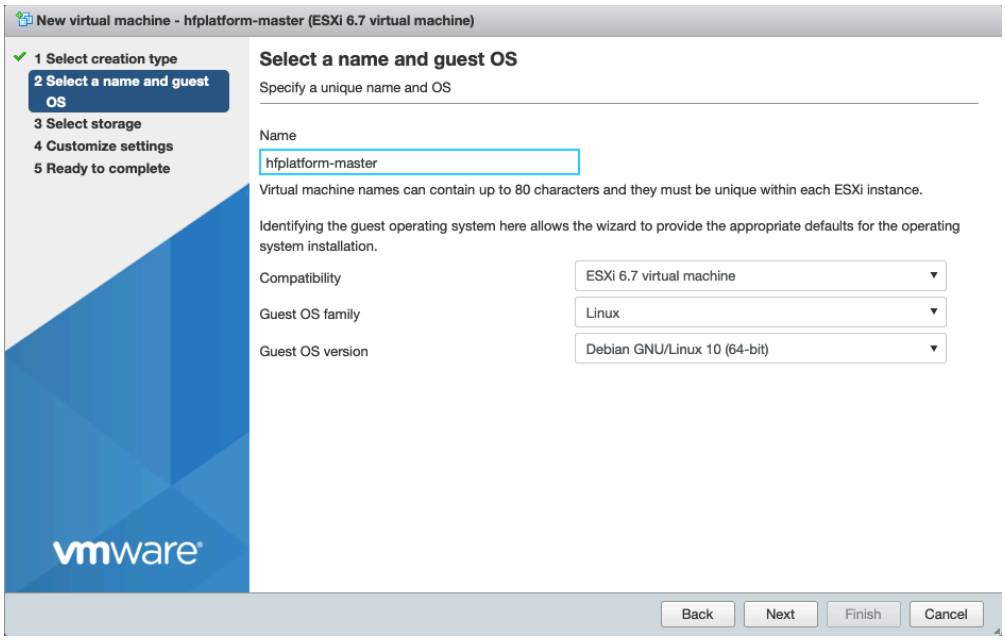
Creazione della VM

Nella sezione Virtual Machines creare una VM utilizzando le funzioni (pulsanti)

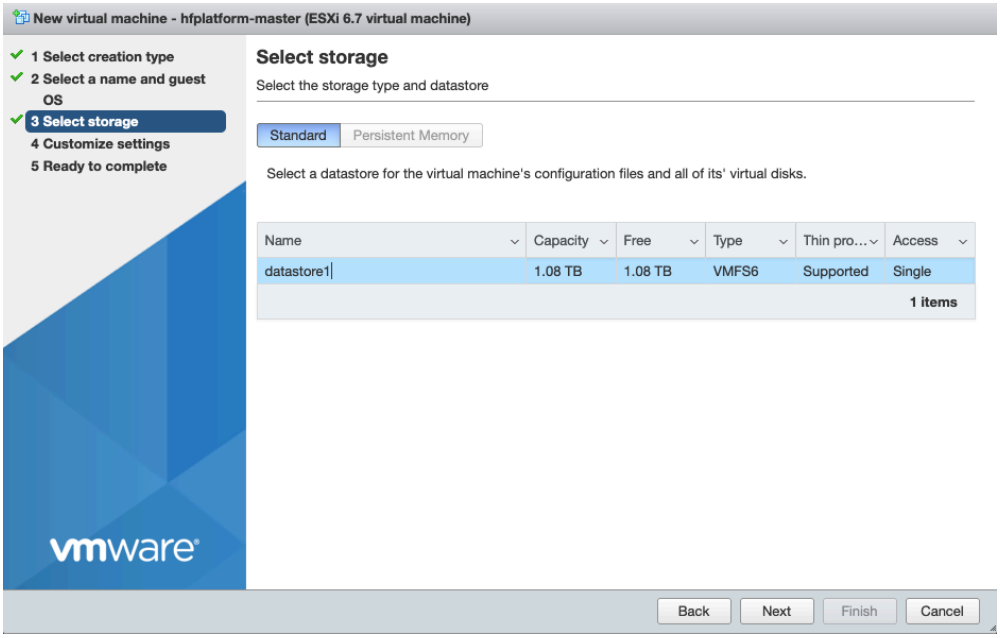
1. Create/Register VM
2. Create new virtual machine



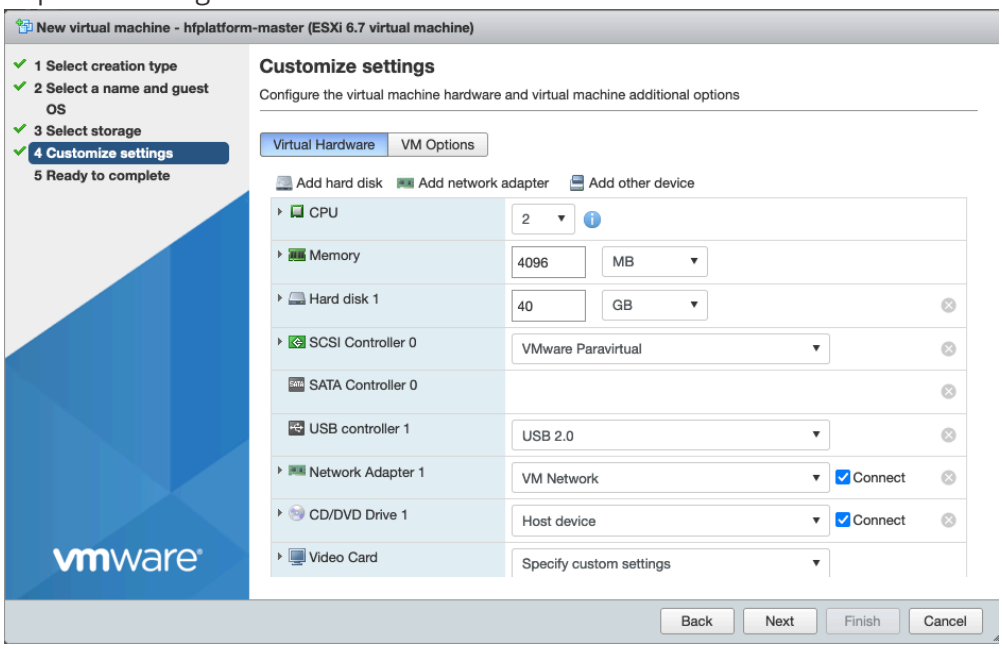
3. Inserire i parametri della vm
    1. Name: hfplatform-master
    2. Compatibility ESXi 6.7
    3. OS family: linux
    4. OS version: debian 10 - 64 bit



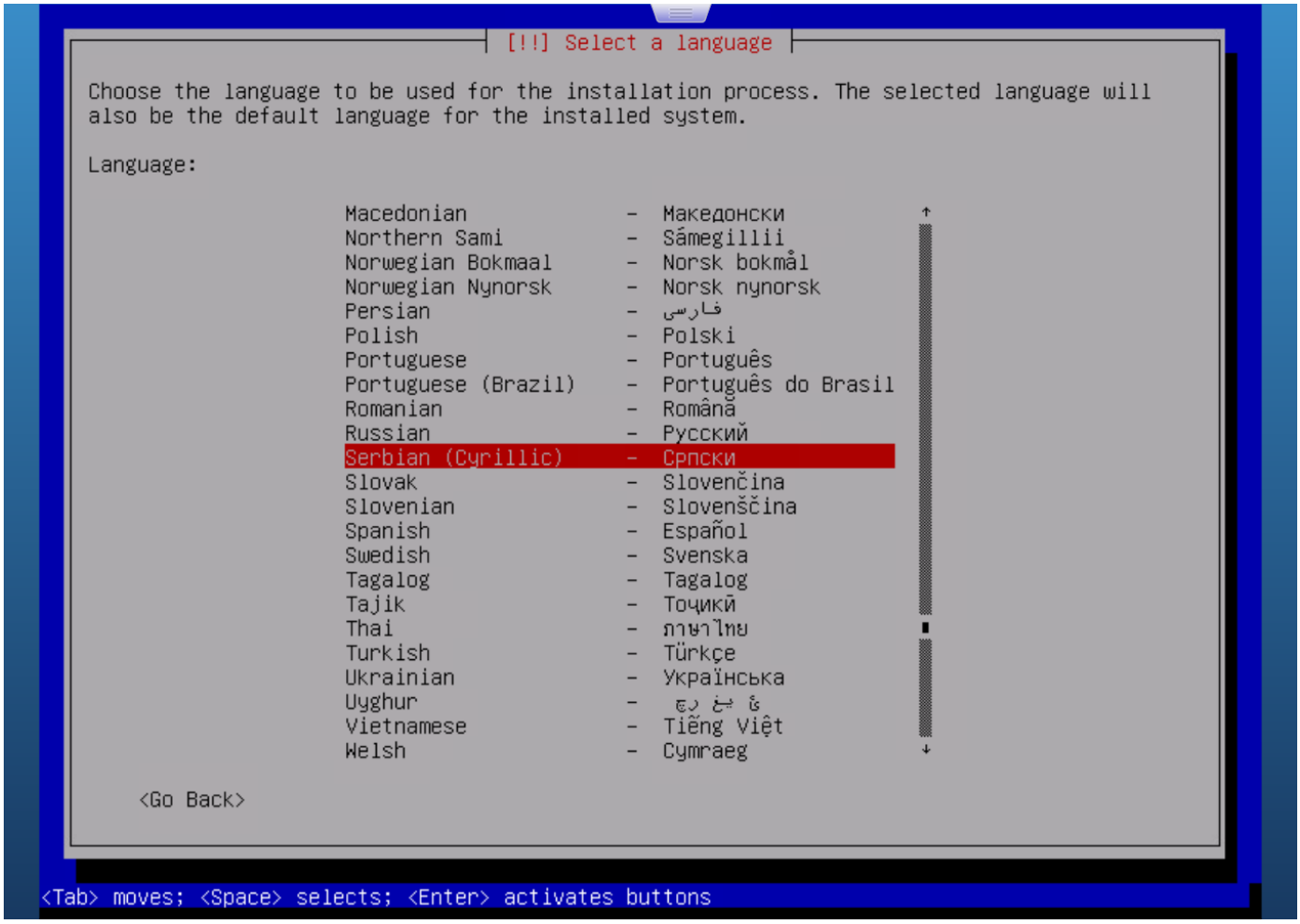
4. Select datastore

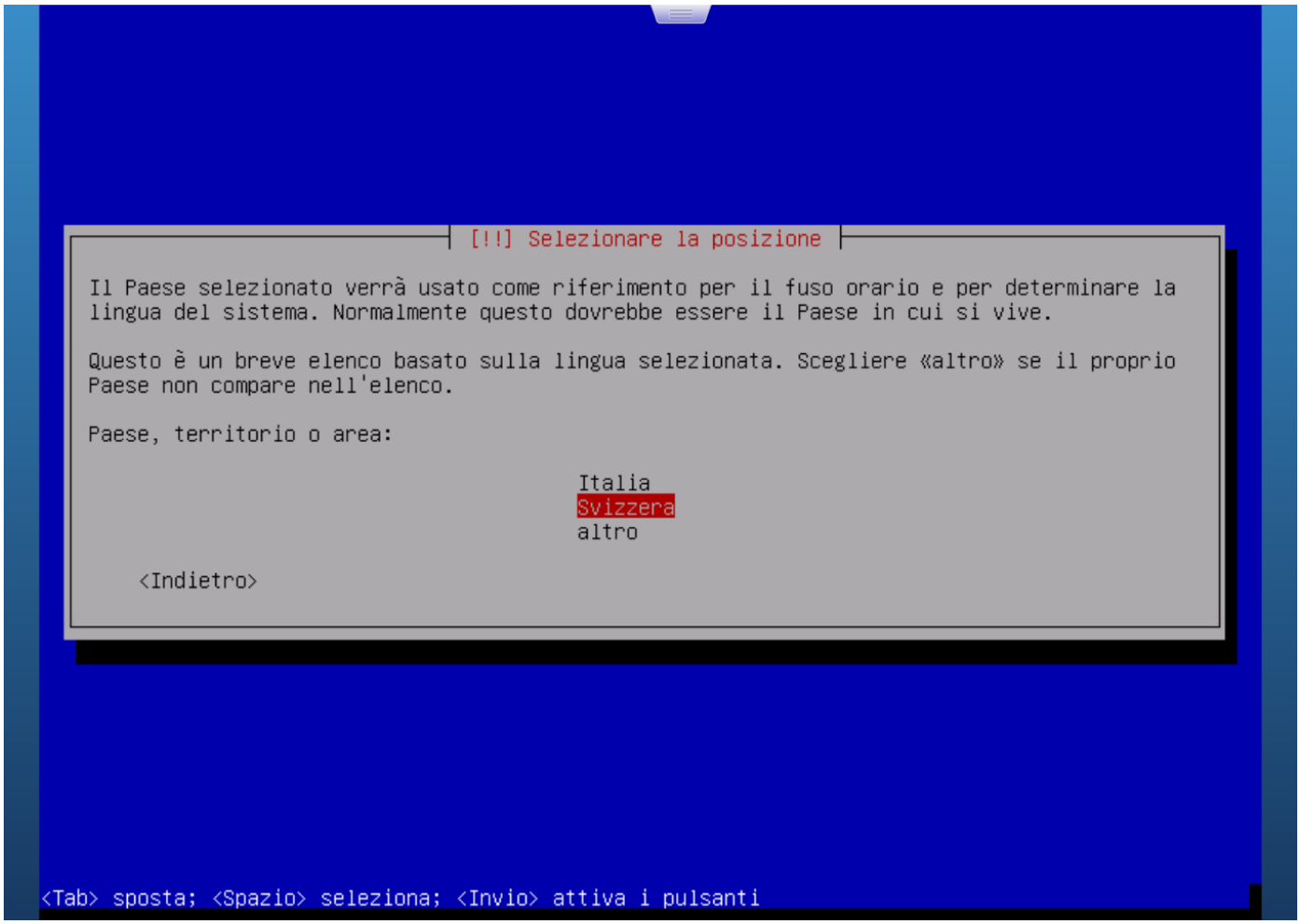


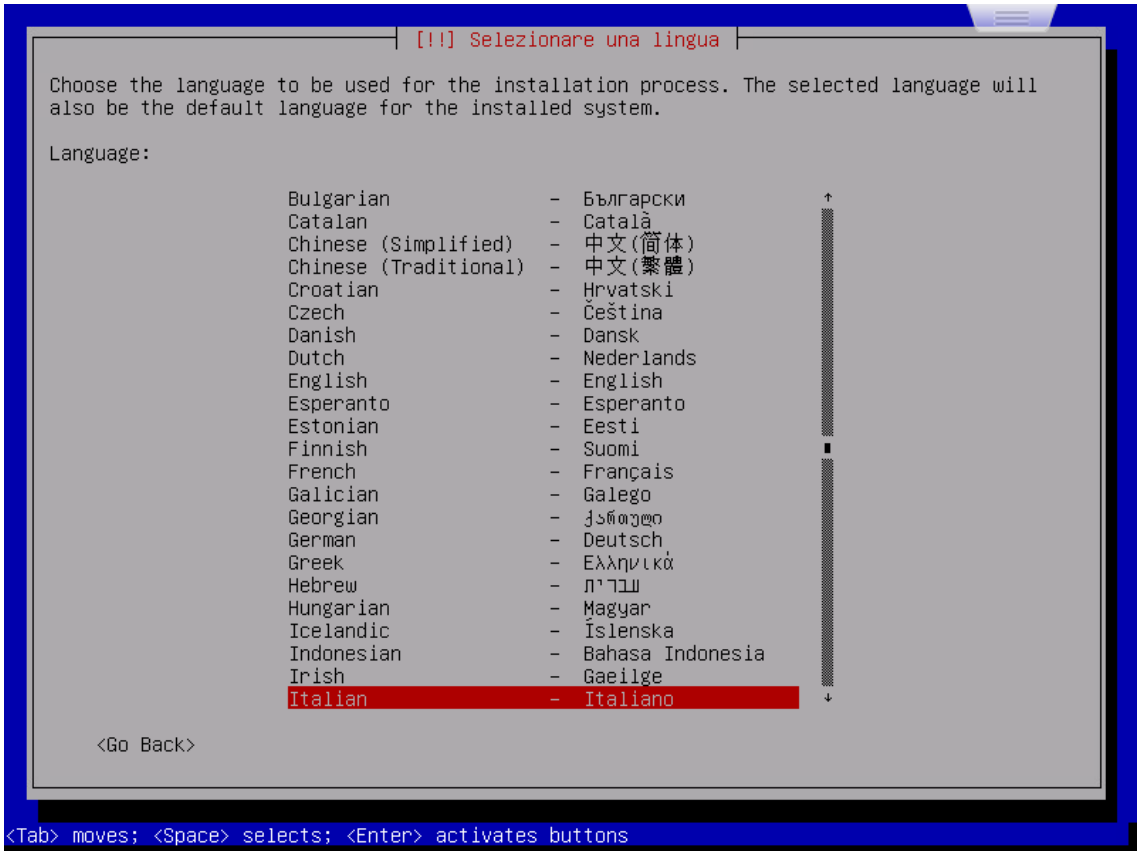
5. Configurazione dell'hardware
    1. Vcpu: 2
    2. Memory: 4096 MB
    3. hard disk: 40 GB
    4. Per il device CD/DVD Media scegliere Datastore ISO e selezionare l'iso precedentemente caricata e impostare il flag Connected

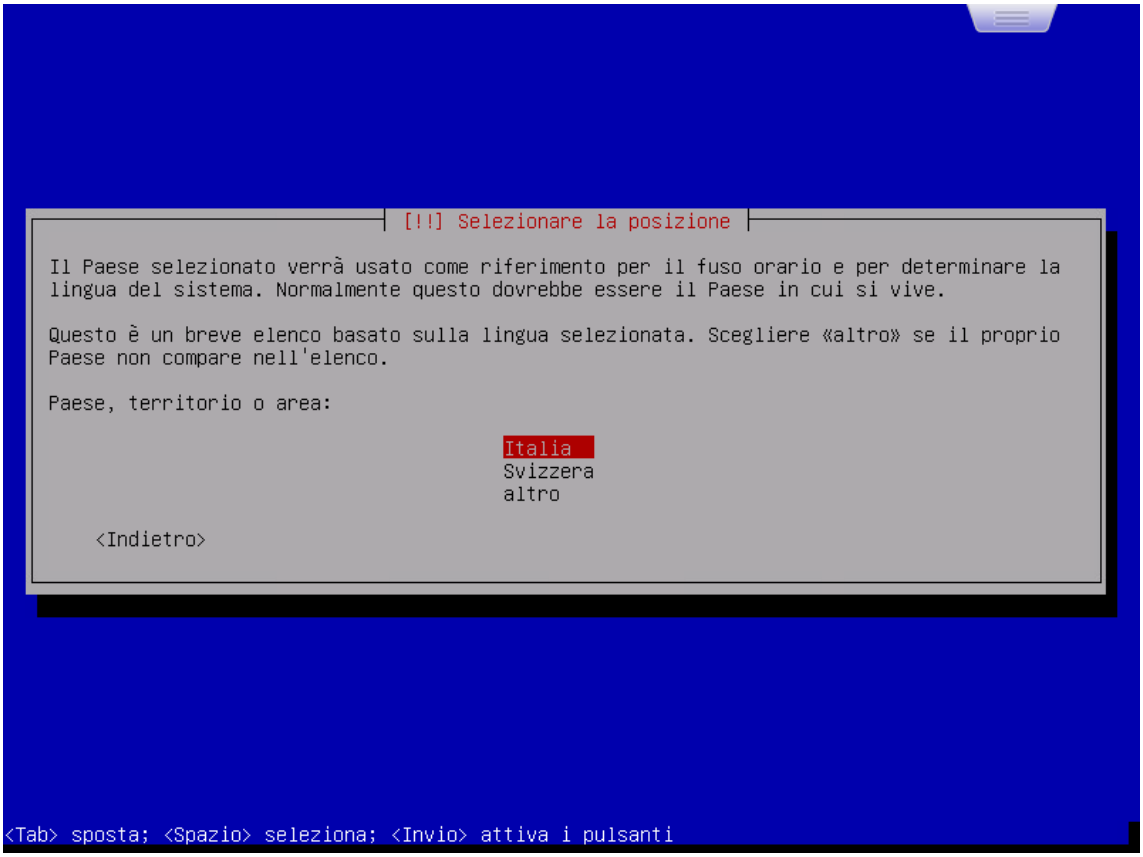


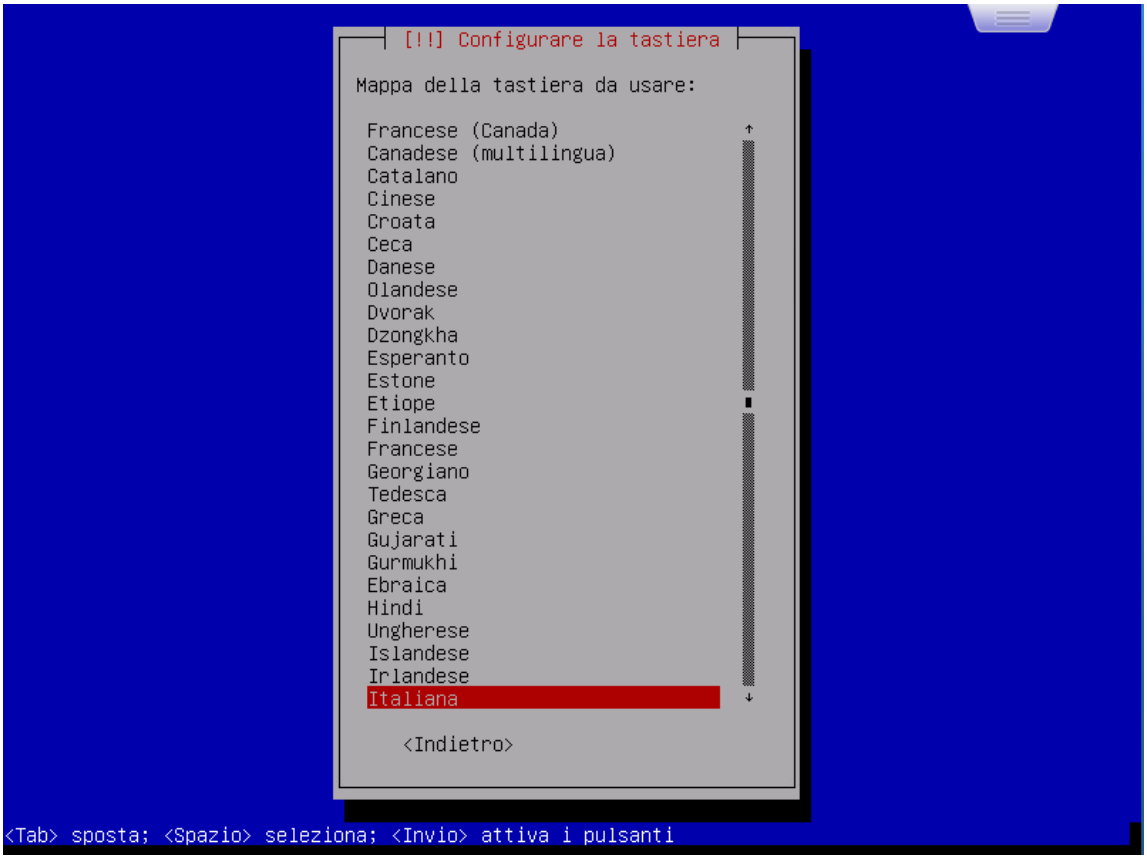
Avviare la VM e aprire una console. Procedere con l'installazione.


[!!] Select a language
Choose the language to be used for the installation process. The selected language will also be the default language for the installed system.
Language:
Macedonian - Македонски
Northern Sami - Sámegillii
Norwegian Bokmaal - Norsk bokmål
Norwegian Nynorsk - Norsk nynorsk
Persian - فارسی
Polish - Polski
Portuguese - Português
Portuguese (Brazil) - Português do Brasil
Romanian - Română
Russian - Русский
Serbian (Cyrillic) - Српски
Slovak - Slovenčina
Slovenian - Slovenščina
Spanish - Español
Swedish - Svenska
Tagalog - Tagalog
Tajik - Тоҷикӣ
Thai - ภาษาไทย
Turkish - Türkçe
Ukrainian - Українська
Uyghur - ئۇيغۇرچە
Vietnamese - Tiếng Việt
Welsh - Cymraeg
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons


[!!] Selezionare la posizione
Il Paese selezionato verrà usato come riferimento per il fuso orario e per determinare la lingua del sistema. Normalmente questo dovrebbe essere il Paese in cui si vive.
Questo è un breve elenco basato sulla lingua selezionata. Scegliere «altro» se il proprio Paese non compare nell'elenco.
Paese, territorio o area:
Italia
Svizzera
altro
<Indietro>
<Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti


[!!] Selezionare una lingua
Choose the language to be used for the installation process. The selected language will also be the default language for the installed system.
Language:
Bulgarian - Български
Catalan - Català
Chinese (Simplified) - 中文(简体)
Chinese (Traditional) - 中文(繁體)
Croatian - Hrvatski
Czech - Čeština
Danish - Dansk
Dutch - Nederlands
English - English
Esperanto - Esperanto
Estonian - Eesti
Finnish - Suomi
French - Français
Galician - Galego
Georgian - ქართული
German - Deutsch
Greek - Ελληνικά
Hebrew - עברית
Hungarian - Magyar
Icelandic - Íslenska
Indonesian - Bahasa Indonesia
Irish - Gaeilge
Italian - Italiano
<Go Back>
<Tab> moves; <Space> selects; <Enter> activates buttons


[!!] Selezionare la posizione
Il Paese selezionato verrà usato come riferimento per il fuso orario e per determinare la lingua del sistema. Normalmente questo dovrebbe essere il Paese in cui si vive.
Questo è un breve elenco basato sulla lingua selezionata. Scegliere «altro» se il proprio Paese non compare nell'elenco.
Paese, territorio o area:
Italia
Svizzera
altro
<Indietro>
<Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti



Quando richiesto impostare il nome host come hfplatform-master e come dominio hfmaster.local

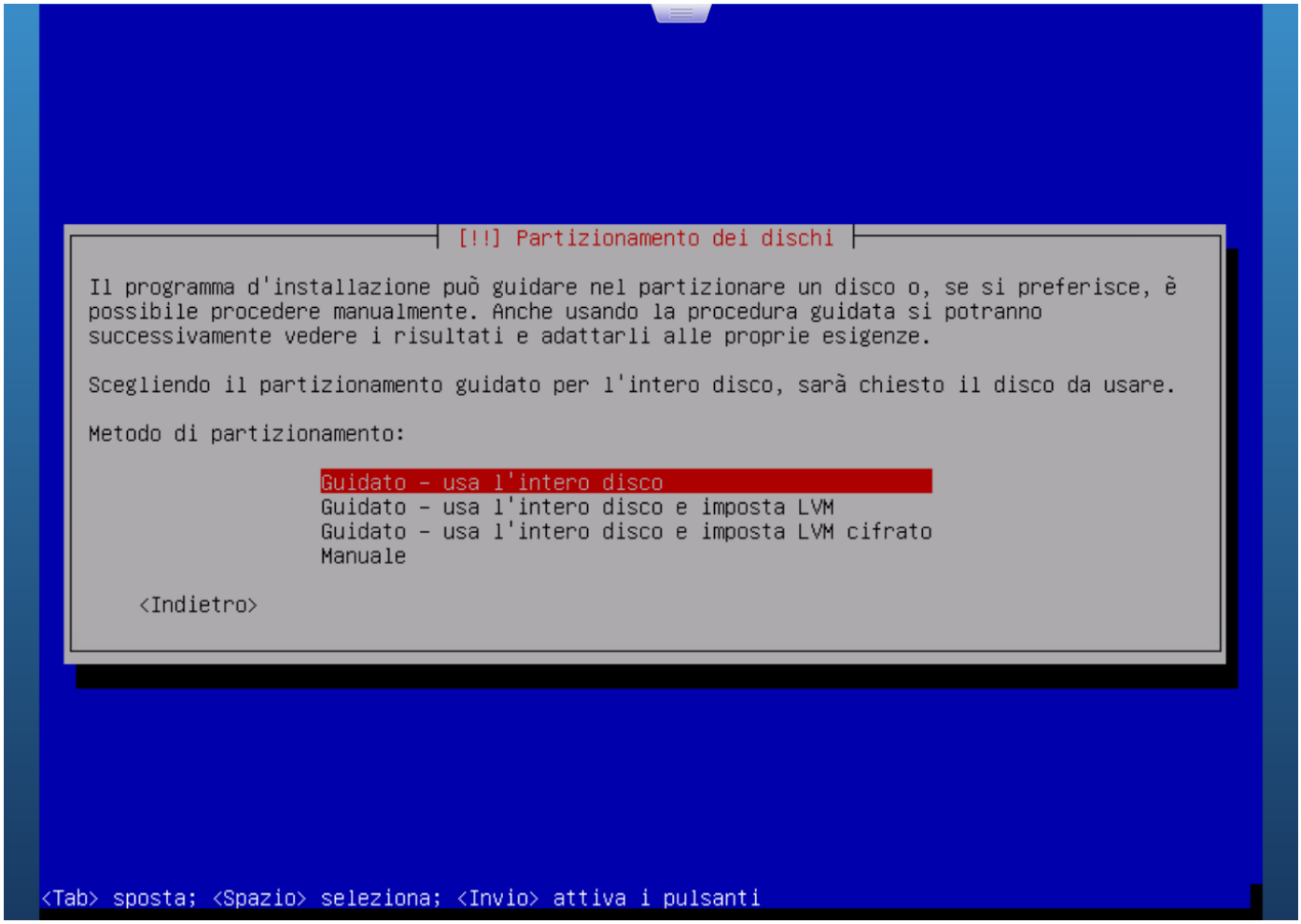
Account

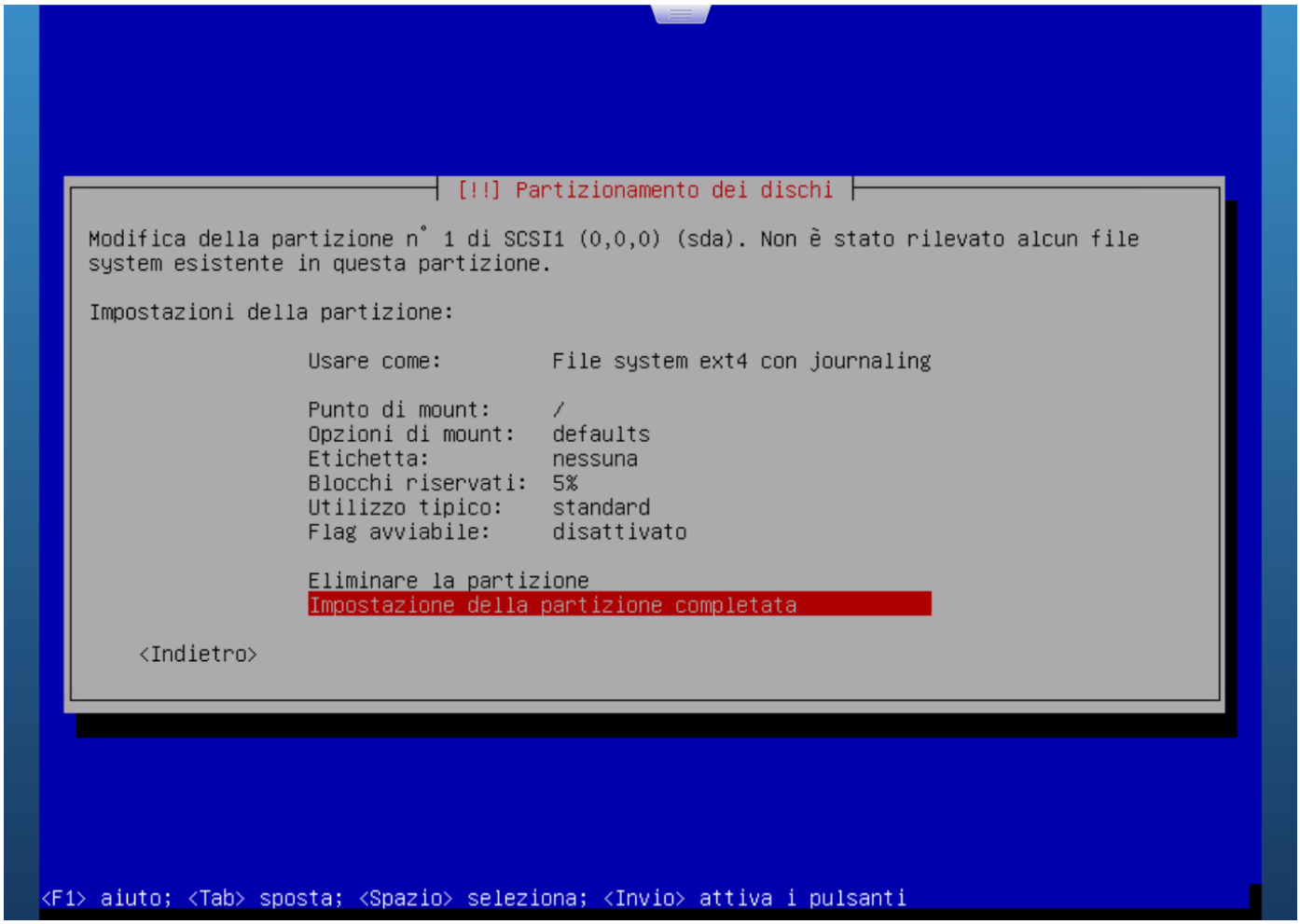
Impostare la password di root: HFbird3692.

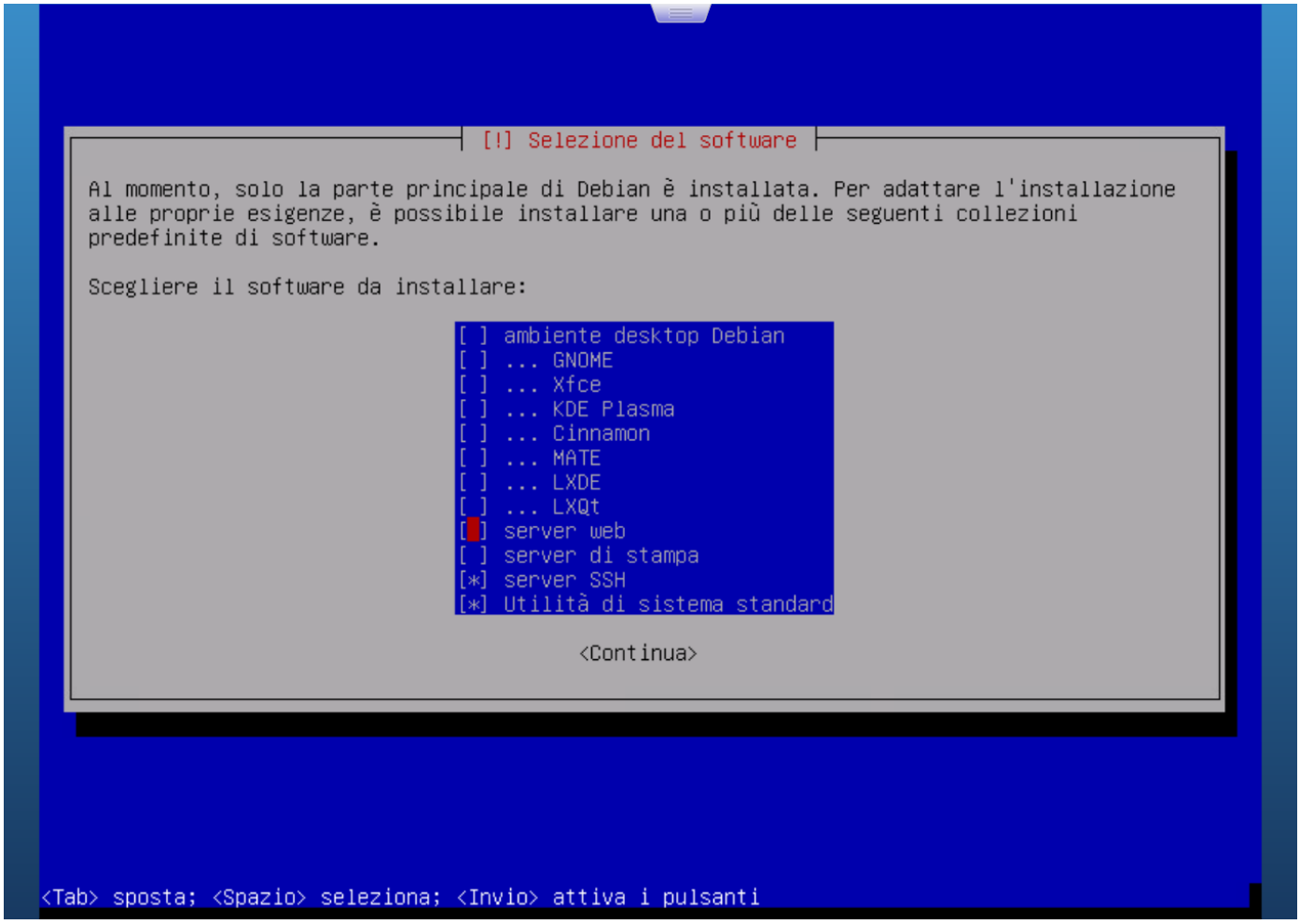
Creare un nuovo utente con username hfadmin e password 7745HFlion

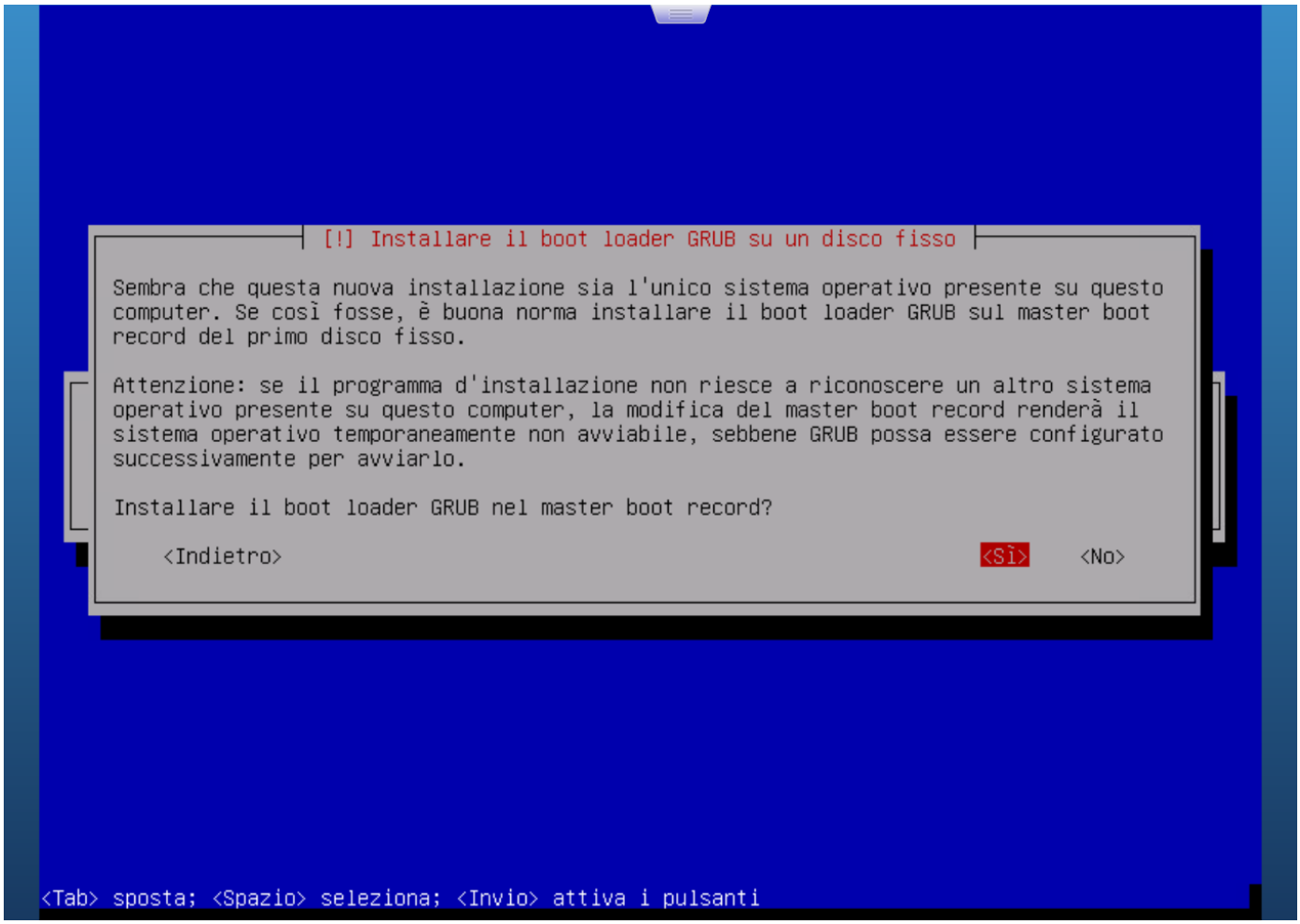
Partizionamento

Per il partizionamento, scegliere il partizionamento manuale e impostare una sola partizione ext4 come /. Kubernetes non necessita di partizione di swap.


[!!] Partizionamento dei dischi
Il programma d'installazione può guidare nel partizionare un disco o, se si preferisce, è possibile procedere manualmente. Anche usando la procedura guidata si potranno successivamente vedere i risultati e adattarli alle proprie esigenze.
Scegliendo il partizionamento guidato per l'intero disco, sarà chiesto il disco da usare.
Metodo di partizionamento:
Guidato - usa l'intero disco
Guidato - usa l'intero disco e imposta LVM
Guidato - usa l'intero disco e imposta LVM cifrato
Manuale
<Indietro>
<Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti


[!!] Partizionamento dei dischi
Modifica della partizione n° 1 di SCSI1 (0,0,0) (sda). Non è stato rilevato alcun file system esistente in questa partizione.
Impostazioni della partizione:
Usare come: File system ext4 con journaling
Punto di mount: /
Opzioni di mount: defaults
Etichetta: nessuna
Blocchi riservati: 5%
Utilizzo tipico: standard
Flag avviabile: disattivato
Eliminare la partizione
Impostazione della partizione completata
<Indietro>
<F1> aiuto; <Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti


[!] Selezione del software
Al momento, solo la parte principale di Debian è installata. Per adattare l'installazione alle proprie esigenze, è possibile installare una o più delle seguenti collezioni predefinite di software.
Scegliere il software da installare:
[ ] ambiente desktop Debian
[ ] ... GNOME
[ ] ... Xfce
[ ] ... KDE Plasma
[ ] ... Cinnamon
[ ] ... MATE
[ ] ... LXDE
[ ] ... LXQt
[ ] server web
[ ] server di stampa
[*] server SSH
[*] Utilità di sistema standard
<Continua>
<Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti


[!] Installare il boot loader GRUB su un disco fisso
Sembra che questa nuova installazione sia l'unico sistema operativo presente su questo computer. Se così fosse, è buona norma installare il boot loader GRUB sul master boot record del primo disco fisso.
Attenzione: se il programma d'installazione non riesce a riconoscere un altro sistema operativo presente su questo computer, la modifica del master boot record renderà il sistema operativo temporaneamente non avviabile, sebbene GRUB possa essere configurato successivamente per avviarlo.
Installare il boot loader GRUB nel master boot record?
<Indietro>
<Sì>
<No>
<Tab> sposta; <Spazio> seleziona; <Invio> attiva i pulsanti

Per il mirror dei pacchetti scegliere l'impostazione predefinita.

Nella scelta dei pacchetti selezionare:

```
- utilità standard
- server ssh
```

Rispondere Si all'installazione del boot loader e scegliere il disco *sda*.

Avviare l'installazione dei VMWARE tools. Come utente root:

```
apt-get install open-vm-tools
```

Al termine riavviare la macchina e disabilitare il device CD/DVD nelle preferenze della VM (flag connected).

Configurazione locale

Eseguire il comando

```
sudo dpkg-reconfigure locales
```

scegliendo l'opzione

```
it_IT.UTF-8
```

Per il timezone

```
sudo dpkg-reconfigure tzdata
```

Scegliendo le opzioni

- Europa
- Roma

Modificare il servizio ssh editando il file di configurazione col comando:

```
nano /etc/ssh/sshd_config
```

Rispetto alla configurazione standard sono stati impostati i seguenti parametri:

```
X11Forwarding no
PermitRootLogin no
PasswordAuthentication no
AllowUsers hfadmin
```

Successivamente è stato riavviato il servizio ssh sulla macchina:

```
systemctl restart sshd.service
```

Accedere con utente normale e configurare una chiave pubblica

```
mkdir ~/.ssh
chmod 700 ~/.ssh
touch ~/.ssh/authorized_keys
cat id_rsa.pub >> ~/.ssh/authorized_keys
chmod 600 ~/.ssh/authorized_keys
```

Pacchetti opzionali

```
sudo apt install unattended-upgrades unzip bash-completion curl git jq net-tools
dnsutils
```

Installazione Docker

https://docs.docker.com/engine/install/debian/

```
# Set up the Docker daemon
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

mkdir -p /etc/systemd/system/docker.service.d

# Restart Docker
systemctl daemon-reload
systemctl stop docker
systemctl enable docker.service

# enable current user
sudo usermod -aG docker ${USER}
```

Installazione Kubernetes

https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF

sudo sysctl --system


sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg
https://packages.cloud.google.com/apt/doc/apt-key.gpg
echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

Init del master

```
sudo kubeadm init

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version |
base64 | tr -d '\n')"
```

Per aggiungere un nodo usare il comando restituito ad kubeadm. Ad esempio:

```
# replace tokens
kubeadm join 192.168.1.7:6443 --token imz2kp.u4bqs5q4sllra1dj \
  --discovery-token-ca-cert-hash
sha256:3739b5e1c6027b03b8c0e528fe258c2b7c694af3af721e5f5ed939d26394811b
```

Configurazione del numero max di pod per nodo

Si modifica l'attributo *maxPods* nel file di configurazione **/var/lib/kubelet/config.yaml**

Info:

https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/
https://kubernetes.io/docs/tasks/administer-cluster/reconfigure-kubelet/
https://docs.openshift.com/container-platform/4.1/nodes/nodes/nodes-nodes-managing-max-pods.html

Aggiungere la riga seguente al file /var/lib/kubelet/config.yaml e riavviare kubelet

```
maxPods: 20
```

Configurare la gestione di un DNS locale

Installare dnsmask con

```
sudo apt install dnsmasq
```

Edit /etc/dnsmasq.conf aggiungendo:

```
expand-hosts
domain=masterk8s.lan

# Public nameservers
server=8.8.8.8
server=8.8.4.4

# Private nameservers
address=/masterk8s.lan/192.168.1.7
```

Riavviare dnsmasq

```
sudo systemctl restart dnsmasq
```

Aggiungere il nuovo dns server tramite i comandi

```
sudo apt install resolvconf
sudo systemctl start resolvconf.service
sudo systemctl enable resolvconf.service
sudo systemctl status resolvconf.service

sudo nano /etc/resolvconf/resolv.conf.d/head
```

Aggiungere questa riga e riavviare il servizio

```
nameserver 192.168.1.7
```

cioè l'ip del master. A questo punto si può utilizzare come dominio del master: masterk8s.lan

Info https://www.tecmint.com/set-permanent-dns-nameservers-in-ubuntu-debian/

Configurazione in Kubernetes

In kubernetes serve modificare il servizio dns per gestire il nuovo dns server.

Creare un file custom_domain.yml con il seguente contenuto (modifica il dominio del dns server locale)

```
apiVersion: v1
data:
  Corefile: |
    .:53 {
        errors
        health {
           lameduck 5s
        }
        ready
        kubernetes cluster.local in-addr.arpa ip6.arpa {
           pods insecure
           fallthrough in-addr.arpa ip6.arpa
           ttl 30
        }
        prometheus :9153
        forward . /etc/resolv.conf {
           max_concurrent 1000
        }
        cache 30
        loop
        reload
        loadbalance
    }
    masterk8s.lan:53 {
        errors
        cache 30
        forward . 192.168.1.7
    }
    masterk8s.lan.homenet.telecomitalia.it:53 {
      errors
      cache 1
      forward . 192.168.1.7
    }
kind: ConfigMap
metadata:
  name: coredns
  namespace: kube-system
```

Pubblicare la modifica

```
kubectl delete -f custom_domain.yml ; kubectl apply -f custom_domain.yml
kubectl delete pods -n kube-system -l k8s-app=kube-dns
```

Installazione e Configurazione di Docker Registry versione 2.0

Prima di poter distribuire un docker *registry*, è necessario installare *Docker* sull'host. Un docker registry è un'istanza dell'immagine *registry*, disponibile su Docker Hub, che viene eseguito all'interno di *Docker*.

Questo documento fornisce informazioni di base sull'installazione e configurazione di un docker *registry*. Di seguito sono elencati i passi di installazione dell'ultima versione stabile (ver. 2.0) e le relative configurazioni effettuate sulla macchina di sviluppo a partire da un'installazione base del sistema operativo Debian 10.

Requisiti preliminari

Per installare un'istanza di Docker Registry versione 2.0, è necessario aver precedentemente installato e configurato *Docker*.

Creazione di un certificato self signed

Impostare un dominio che risolve l'ip del master. Generare i certificati con questo script e usare masterk8s.lan come dominio

```
#!/usr/bin/env bash
echo
CgogICBfX19fXyAgICAgICAgICBfICAgX19fX18gICAgICAgICAgICAgICAgXyAgICAgCiAgLyBfX19ffCA
gICAgICAgfCB8IHwgIF9fIFwgICAgICAgICAgICAgIHwgfCAgICAKIHwgfCAgICAgX19fIF8gX198IHxffC
B8X18pIHwgX18gX19fICAgICAgX19ffCB8X18gIAogfCB8ICAgIC8gXyBcICdfX3wgX198ICBfX18vICdfXy8gX
yBcICAgIC8gX198ICdfIFwgCiB8IHxfX198ICBfXy8gfCAgfCB8X3wgfCAgIHwgfCB8IChfKSB8ICBfXF9fIFwg
fCB8IHwKICBcX19fX19cX19ffF98ICAgXF9ffF98ICAgfF98ICBcX19fLyAgKF8pX19fL198IHxffAogICAgICA
gICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAgCiAgICAgICAgICAgICAgIC
AgICAgICAgICAgICAgICAgICAgICAgICAgICAgICAKCg== | base64 --decode

echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo Welcome to CertPro Automated Certificate Generator!
echo WARNING! If you did not run this script as superuser, it will fail. Restart if
needed.
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Creating Cert Authority Private Key.--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
#openssl genrsa -des3 -out ca.key 2048
openssl genrsa -out ca.key 2048
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Done Creating Key. Creating Certificate Authority Root--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
openssl req -x509 -new -nodes -key ca.key -sha256 -days 1825 -out ca.pem
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Done creating CA Root. Creating Server Private Key--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
openssl genrsa -out nginx.key 2048
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
echo --Done creating Server Key. Creating Server CSR--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
openssl req -new -key nginx.key -out nginx.csr
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Done creating CSR. Creating extension file--
read -p 'You will now need to enter the FQDN of your server: ' fqdn
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo "authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1=$fqdn">>nginx.ext
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Done Creating SAN. Now creating Server Certificate.--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
openssl x509 -req -in nginx.csr -CA ca.pem -CAkey ca.key -CAcreateserial \
-out nginx.crt -days 18250 -sha256 #-extfile nginx.ext
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
echo --Done. Get files you need in the directory you ran this script in.--
echo @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

Copiare i file seguenti

```
sudo mkdir -p /etc/docker/certs.d/masterk8s.lan:5000/
sudo cp ca.pem /etc/docker/certs.d/masterk8s.lan\:5000/cert.crt
sudo cp nginx.key /opt/docker/registry-security/masterk8s.lan.key
sudo cp nginx.crt /opt/docker/registry-security/masterk8s.lan.crt
```

Creazione manuale del certificato:

```
openssl req -x509 -newkey rsa:4096 -subj '/CN=masterk8s.lan' -nodes -keyout
masterk8s.lan.key -out masterk8s.lan.crt -days 3365
```

Copiare il certificato *.crt nella directory /etc/docker/certs.d/$REGISTRY_FQDN/cert.crt

```
sudo mkdir -p /etc/docker/certs.d/masterk8s.lan:5000/
sudo cp cert.crt /etc/docker/certs.d/masterk8s.lan:5000/cert.crt
```

DA ESEGUIRE SU TUTTI I NODI

```
sudo tar zcvf docker_certs.tgz /etc/docker/certs.d/
sudo chmod 666 docker_certs.tgz
scp -r docker_certs.tgz  hfplatform-worker:/home/hfadmin

ssh hfplatform-worker

# nel nodo
tar xf docker_certs.tgz
sudo mv -v etc/docker/certs.d/ /etc/docker/
sudo systemctl restart docker.service
```

Installazione e configurazione del docker registry

Al momento della stesura del presente documento, docker *registry* 2.0 è l'ultima versione stabile disponibile. Scarichiamo l'immagine base di docker *registry* versione 2.0 da Docker Hub, mediante il comando sotto riportato.

```
sudo docker pull registry:latest
```

Come e quando inseriamo le immagini nel *registry*, esso memorizzerà i dati. Si vuole garantire che i nostri dati siano sicuri, anche se *Docker*, all'interno del quale è eseguito il *registry*, dovesse avere problemi durante l'esecuzione. Il modo più semplice per farlo è montare una directory dall'host al contenitore che memorizzerà i dati. Quindi, creiamo la directory mediante il seguente comando:

```
sudo mkdir -p /opt/docker/registry-data
```

Esecuzione del docker registry

Il registry è configurato in modalità sicura con il certificato ssl utilizzato da nginx (vedi documentazione .md)

Installare il pacchetto apache2-utils

```
sudo apt-get install apache2-utils
```

Creare la directory per i certificati e copiare i certificati creati in precedenza

```
sudo mkdir -p /opt/docker/registry-security

sudo chmod 640 /opt/docker/registry-security/*
```

Creare le credenziali di autenticazione

> Sostituire USERNAME ed inserire la password quando richiesto

> Username: userregistry
> Password: pwregistry

```
sudo touch /opt/docker/registry-security/htpasswd
sudo htpasswd -B /opt/docker/registry-security/htpasswd USERNAME
```

Eseguiamo un'istanza del docker *registry* attraverso il comando:

```
docker run -d \
  -p 5000:5000 \
  -v /opt/docker/registry-data:/var/lib/registry \
  -v /opt/docker/registry-security:/etc/security \
  -e REGISTRY_HTTP_TLS_CERTIFICATE=/etc/security/masterk8s.lan.crt \
  -e REGISTRY_HTTP_TLS_KEY=/etc/security/masterk8s.lan.key \
  -e REGISTRY_AUTH=htpasswd \
  -e REGISTRY_AUTH_HTPASSWD_PATH=/etc/security/htpasswd \
  -e REGISTRY_AUTH_HTPASSWD_REALM="Registry Realm" \
  --restart=always \
  --name registry \
  registry:latest
```

Verifichiamo il contenitore *registry* sia effettivamente in esecuzione. Eseguiamo, il comando:

```
sudo docker ps
```

Se l'installazione e configurazione è andata a buon fine l'output mostrato assomiglia al seguente:

```
CONTAINER ID        IMAGE               COMMAND                  CREATED
STATUS              PORTS                    NAMES
f58d7ad85985        registry:latest     "/entrypoint.sh /etc…"   10 seconds ago      Up
9 seconds        0.0.0.0:5000->5000/tcp   registry
```

Avvio del registry e update del certificato

Per l'avvio del registry e l'aggiornamento del certificato utilizzare lo script in

```
/opt/docker/mystart-registry.sh
```

Stop a local registry

Per interrompere il servizio registry, utilizzare il comando:

```
docker container stop registry
```

Per eliminare il container del registry dalla cache locale:

```
docker container stop registry && docker container rm -v registry
```

Test local registry

Login al registry locale

```
docker login --username userregistry masterk8s.lan:5000
```

Per pubblicare l'immagine creata sul registry

```
docker tag webapp_example:0.1 masterk8s.lan:5000/webapp_example:0.1
```

Per pubblicare l'immagine con il comando

```
docker push masterk8s.lan:5000/webapp_example:0.1
```

L'immagine caricata sarà memorizzata nella directory

```
tree /opt/docker/registry-data
```

Pubblicazione dell'immagine ELK

Partendo dal file .tar di un'immagine precaricata:

```
docker load < elkhf_20210315.tar
docker tag elkhf:latest masterk8s.lan:5000/elkhf:1.0
docker push masterk8s.lan:5000/elkhf:1.0
```

Eliminare immagini dal docker registry

Per eliminare un'immagine dal registry (modificare il nome dell'immagine e l'url del registry):

```
registry='localhost:5000'
name='my-image'
curl -v -sSL -X DELETE "http://${registry}/v2/${name}/manifests/$(
    curl -sSL -I \
        -H "Accept: application/vnd.docker.distribution.manifest.v2+json" \
        "http://${registry}/v2/${name}/manifests/$(
            curl -sSL "http://${registry}/v2/${name}/tags/list" | jq -r '.tags[0]'
        )" \
    | awk '$1 == "Docker-Content-Digest:" { print $2 }' \
    | tr -d $'\r' \
)"
```

Output atteso:

```
* About to connect() to localhost port 5000 (#0)
*   Trying 127.0.0.1...
* Connected to localhost (127.0.0.1) port 5000 (#0)
> DELETE /v2/my-
image/manifests/sha256:14f6ecba1981e49eb4552d1a29881bc315d5160c6547fdd100948a9e30a90dff
HTTP/1.1
> User-Agent: curl/7.29.0
> Host: localhost:5000
> Accept: */*
>
< HTTP/1.1 202 Accepted
< Docker-Distribution-Api-Version: registry/2.0
< X-Content-Type-Options: nosniff
< Date: Wed, 15 Nov 2017 23:25:30 GMT
< Content-Length: 0
< Content-Type: text/plain; charset=utf-8
<
* Connection #0 to host localhost left intact
```

Garbage cleanup

dopo aver eliminato un'immagine, è necessario invocare la funzione garbage:

```
docker exec -it registry bin/registry garbage-collect /etc/docker/registry/config.yml
```

Test registry with curl

Elenco delle immagini nel registry

```
curl -X GET https://masterk8s.lan:5000/v2/_catalog -k -u username:password
```

Per eliminare le vecchie immagini:

```
docker container stop registry && docker container rm -v registry

rm -r /opt/docker/registry-data/*

sudo bash /opt/docker/mystart-registry.sh
```

Verificare che una versione di un'immagine esista nel registry

```
curl -X GET https://masterk8s.lan:5000/v2/gateway_service/manifests/1.5 -k -u
username:password
```

in caso di errore si ha

```
{"errors":[{"code":"MANIFEST_UNKNOWN","message":"manifest unknown","detail":
{"Tag":"1.6"}}]}
```

Deploy di ELK

Pubblicazione del secret per il registry

Solo la prima volta, pubblicare il secret per accedere al registry.

Prerequisito per il deploy è l'autenticazione con il registry.

Per recuperare le credenziali di accesso si può utilizzare la configurazione memorizzata nel file config.json di Docker

```
cat ~/.docker/config.json | base64 -w0
```

Poi si crea un file YAML *docker_registry_secret.yml* con questo contenuto

```
apiVersion: v1
kind: Secret
metadata:
  name: registrypullsecret
data:
  .dockerconfigjson: <base-64-encoded-json-here>
type: kubernetes.io/dockerconfigjson
```

E creare il secret con:

```
kubectl create -f docker_registry_secret.yml && kubectl get secrets
```

Creazione dell'immagine docker

Per creare l'immagine docker, nella directory con il *Dockerfile* eseguire il comando:

```
docker build -t elkhf .
```

Caricamento dell'immagine nel registry

```
docker load < elkhf_20210315.tar
docker tag elkhf:latest masterk8s.lan:5000/elkhf:1.0
docker push masterk8s.lan:5000/elkhf:1.0
```

Labeling del nodo

Per stampare la lista dei nodi del cluster:

```
kubectl get nodes
```

Dall'output del comando selezionare il nome del nodo dedicato ad elk, poi eseguire il comando seguente per assegnare la label

```
kubectl label nodes <node-name> pltns=elknode

# quindi:
kubectl label nodes hfplatform-worker pltns=elknode
```

Per visualizzare le label assegnate ai nodi:

```
kubectl get nodes --show-labels
```

> i nodi per ELK hanno label pltns=elknode

Deploy dell'applicazione

Nella dir ~/hf_k8s_config, creare un file kustomization.yml con contenuto

```
generatorOptions:
  disableNameSuffixHash: true

secretGenerator:
- name: elk-env
  literals:
  - xpack.security.enabled=true
  - xpack.security.audit.enabled=true
  - ELASTIC_PASSWORD=Pa9snCu9ExT8mxAdkEJ9
  - ELASTICSEARCH_USERNAME=elastic
  - ELASTICSEARCH_PASSWORD=Pa9snCu9ExT8mxAdkEJ9
```

Aggiornare le configurazioni condivise. Nella directory che contiene il file kustomization.yml

```
kubectl delete -v1 -k . --ignore-not-found=true
kubectl apply -v1 -k .
```

Solo la prima volta pubblicare il deployment del volume con i dati

```
kubectl apply -f 1_elk_volumes.yml
```

Pubblicare il servizio ELK

```
kubectl apply -f 2_elasticsearch.k8s.yml
kubectl apply -f 3_kibana.k8s.yml

# per rimuovere
kubectl delete -f *.yml
```

Per fare il deploy di logstash

```
cd ./logstash
kubectl apply -k .

# per rimuovere
kubectl delete -k .
```

> Se si usa la versione con imaagine docker personalizzata di logstash
>
> kubectl apply -f 4_logstash.k8s.yml
>
> Modificare la versione dell'immagine docker in accordo a quella pubblicata nel registry

Per testare l'installazione collegarsi al master sulle porte dedicate a ELK

```
  ports:
  - name: kibana
    protocol: TCP
    port: 5061
    nodePort: 31001
  - name: elastic
    protocol: TCP
    port: 9200
    nodePort: 31002
  - name: logstash
    protocol: TCP
    port: 5044
    nodePort: 31003
```