Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Federazione delle risorse di ICSC dell'INFN di Milano con INFN DataCloud: stato e prospettive del cluster Kubernetes/RKE2 su bare metal

L. Carminati, **F. Dalessandro**, C. Marcon, D. Rebatto, F. Prelz

Workshop sul Calcolo nell'I.N.F.N.
La Biodola - Isola d'Elba, 26 - 30 maggio 2025

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Composizione del cluster Kubernetes/RKE2 (bare metal)

- 3 Control Plane (bare metal) (*)

2 Socket per nodo — 4 Core per socket
2 Thread per core — 48 GB RAM per nodo
→ 6 GB RAM/Core

- 20 Worker Node (bare metal)

2 Socket per nodo — 24 Core per socket
2 Thread per core — 512 GB RAM per nodo
→ 10,7 GB RAM/Core

- 4 Worker Node HPC Bubble (bare metal)

2 Socket per nodo — 96 Core per socket
2 Thread per core — 1,5 TB RAM per nodo
→ 7,8 GB RAM/Core

Complessivamente per i WN

→9,4GB RAM/Core

Core totali 1728
Thread totali 3456
RAM totale 16,24 TB

Cluster di TEST (VM su 4 nodi Proxmox)

→ cluster IPv4 only (3CP - 2WN)
→ cluster IPv6 only (3CP - 2WN)
→ cluster IPv4/IPv6 testing (3CP - 2WN)
→ cluster IPv4/IPv6 sperimental (3CP - 2WN)
→ cluster IPv4/IPv6 per un cluster geografico (3CP – 2WN)

(*) Scheduling non consentito

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Non solo cluster Kubernetes/RKE2

A supporto del cluster per aumentare la disponibilità del servizio

- 4 nodi Proxmox (bare metal)
- 3 VM TheForeman (Debian + AlmaLinux) + 1 TheForeman su Debian (bare metal)
- 3 VM Puppet server (AlmaLinux) + 1 Puppet server su AlmaLinux (bare metal)
- 2 repository per AlmaLinux [Debian + AlmaLinux] (in fase di configurazione anche per Debian) che fungono anche da piccoli storage NFS per test (bare metal)
- 2 Proxmox Backup Server per il backup delle VM (bare metal)
- 5 nodi per il backup hardware (3 per i control plane [AlmaLinux] – 2 per il resto [Debian + AlmaLinux]) (bare metal)
- 2 nodi per il backup dei dati [Debian + AlmaLinux] (bare metal)
- 2 nodi per il monitoring (AlmaLinux [Prometheus + Grafana] – Debian [InfluxDB + Telegraf]) in R&D (bare metal)
- 1 storage da 48 TB NFS
- 3 sottoreti IPv4 per test

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Stato della federazione delle risorse di ICSC

- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC
- Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC
- Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack
- Load balancer con MetalLB
- Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2
- Spegnimento, accensione ed aggiornamento di un cluster RKE2
- Procedure di ripristino nello scenario in cui tutto è andato distrutto

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

#/etc/rancher/rke2/config.yaml

```
kube-apiserver-arg:
 - --oidc-issuer-url=https://iam.cloud.infn.it/
 - --oidc-client-id=test-cp-01-aud
 - --oidc-username-claim=preferred_username
 - --oidc-groups-claim=groups
```

In alternativa, se si sta usando il modulo wp1/rke2, nel Puppet server, basta editare opportunamente il template corrispondente:

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

Nel secondo caso, non ci sarà bisogno di riavviare rke2-server manualmente, in quanto penserà a tutto puppet.

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

```
#/etc/rancher/rke2/config.yaml
    kube-apiserver-arg:
     - --oidc-issuer-url=https://iam.cloud.infn.it/
     - --oidc-client-id=test-cp-01-aud
     - --oidc-username-claim=preferred_username
     - --oidc-groups-claim=groups
```

In alternativa, se si sta usando il modulo wp1/rke2, nel Puppet server, basta editare opportunamente il template corrispondente:

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

Nel secondo caso, non ci sarà bisogno di riavviare rke2-server manualmente, in quanto penserà a tutto puppet.

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

#/etc/rancher/rke2/config.yaml

```
kube-apiserver-arg:
 - --oidc-issuer-url=https://iam.cloud.infn.it/
 - --oidc-client-id=test-cp-01-aud
 - --oidc-username-claim=preferred_username
 - --oidc-groups-claim=groups
```

In alternativa, se si sta usando il modulo wp1/rke2, nel Puppet server, basta editare opportunamente il template corrispondente:

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

Nel secondo caso, non ci sarà bisogno di riavviare rke2-server manualmente, in quanto penserà a tutto puppet.

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

ID_Token=$(oidc-token --aud=test-cp-01-aud test-cp-01.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml #ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

ID_Token=$(oidc-token --aud=test-cp-01-aud test-cp-01.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml #ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

ID_Token=$(oidc-token --aud=test-cp-01-aud test-cp-01.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml #ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

ID_Token=$(oidc-token --aud=test-cp-01-aud test-cp-01.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml #ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

```
ID_Token=$(oidc-token --aud=test-cp-01-
aud test-cp-01.mi.infn.it)
```

```
echo $ID_Token | jq -R 'split(".") | .[1] |
@base64d | fromjson'
```

```
kubectl --kubeconfig=/$HOME/.kube/config-
rke2-milano.yaml --token "$ID_Token"
apply -f /$HOME/test.yaml #ma non siamo
ancora autorizzati
```


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/iam-users-binding created

#iam-users-auth.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch",
"delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: iam-users-binding
subjects:
  - kind: Group
    name: users
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/iam-users-binding created

#iam-users-auth.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch",
"delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: iam-users-binding
subjects:
  - kind: Group
    name: users
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/iam-users-binding created

#iam-users-auth.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch",
"delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: iam-users-binding
subjects:
  - kind: Group
    name: users
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

```
kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
```

Creazione della risorsa:

```
pod/dnsutils created
```

Utilizzo della risorsa:

```
kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it
```

Visualizzazione del risultato:

```
google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.
```

Cancellazione della risorsa:

```
kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml
```

Visualizzazione della cancellazione:

```
pod "dnsutils" deleted
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e RBAC

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

#/etc/rancher/rke2/config.yaml

```
kube-apiserver-arg:
 - --oidc-issuer-url=https://iam-icsc.cloud.infn.it/ #← IAM di ICSC
 - --oidc-client-id=icsc-test-cp-02-aud
 - --oidc-username-claim=preferred_username
 - --oidc-groups-claim=wlcg.groups #← IAM di ICSC
```

In alternativa, se si sta usando il modulo wp1/rke2, nel Puppet server, basta editare opportunamente il template corrispondente:

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

Nel secondo caso, non ci sarà bisogno di riavviare rke2-server manualmente, in quanto penserà a tutto puppet.

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

#/etc/rancher/rke2/config.yaml

```
kube-apiserver-arg:
 - --oidc-issuer-url=https://iam-icsc.cloud.infn.it/ #← IAM di ICSC
 - --oidc-client-id=icsc-test-cp-02-aud
 - --oidc-username-claim=preferred_username
 - --oidc-groups-claim=wlcg.groups #← IAM di ICSC
```

In alternativa, se si sta usando il modulo wp1/rke2, nel Puppet server, basta editare opportunamente il template corrispondente:

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

Nel secondo caso, non ci sarà bisogno di riavviare rke2-server manualmente, in quanto penserà a tutto puppet.

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

```
ID_Token=$(oidc-token --aud=icsc-test-cp-02-aud test-cp-02.mi.infn.it)
```

```
echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'
```

```
kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
#ma non siamo ancora autorizzati
```


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

ID_Token=$(oidc-token --aud=icsc-test-cp-02-aud test-cp-02.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
#ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

ID_Token=$(oidc-token --aud=icsc-test-cp-02-aud test-cp-02.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
#ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

```
ID_Token=$(oidc-token --aud=icsc-test-cp-02-aud test-cp-02.mi.infn.it)
```

```
echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'
```

```
kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
#ma non siamo ancora autorizzati
```


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

ID_Token=$(oidc-token --aud=icsc-test-cp-02-aud test-cp-02.mi.infn.it)

echo $ID_Token | jq -R 'split(".") | .[1] | @base64d | fromjson'

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
#ma non siamo ancora autorizzati


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

```
kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/icsc-iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/icsc-iam-users-binding created
```

```
#icsc-iam-users-auth.yaml

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: icsc-iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: icsc-iam-users-binding
subjects:
  - kind: Group
    name: /test #parametro che va opportunamente modificato
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/icsc-iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/icsc-iam-users-binding created

#icsc-iam-users-auth.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: icsc-iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch",
"delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: icsc-iam-users-binding
subjects:
  - kind: Group
    name: /test #parametro che va opportunamente modificato
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

kubectl apply -f iam-users-auth.yaml

clusterrole.rbac.authorization.k8s.io/icsc-iam-users-roles
createdclusterrolebinding.rbac.authorization.k8s.io/icsc-iam-users-binding created

#icsc-iam-users-auth.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: icsc-iam-users-roles
rules:
  - apiGroups: [""]
    resources: ["pods", "pods/exec", "services"]
    verbs: ["get", "list", "watch", "create", "update", "patch",
"delete"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: icsc-iam-users-binding
subjects:
  - kind: Group
    name: /test #parametro che va opportunamente modificato
roleRef:
  kind: ClusterRole
  name: iam-users-roles
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml

Creazione della risorsa:

pod/dnsutils created

Utilizzo della risorsa:

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it

Visualizzazione del risultato:

google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.

Cancellazione della risorsa:

kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml

Visualizzazione della cancellazione:

pod "dnsutils" deleted

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di ICSC e RBAC

```
kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" apply -f /$HOME/test.yaml
```

Creazione della risorsa:

```
pod/dnsutils created
```

Utilizzo della risorsa:

```
kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" exec -i -t dnsutils -- host google.it
```

Visualizzazione del risultato:

```
google.it has address 142.250.180.131
google.it has IPv6 address 2a00:1450:4002:402::2003
google.it mail is handled by 0 smtp.google.com.
```

Cancellazione della risorsa:

```
kubectl --kubeconfig=/$HOME/.kube/icsc-config-rke2-milano.yaml --token "$ID_Token" delete -f /$HOME/test.yaml
```

Visualizzazione della cancellazione:

```
pod "dnsutils" deleted
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e dallo IAM di ICSC con RBAC

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
<% if @role == 'server' -%>
kube-apiserver-arg:
 - --oidc-username-claim=preferred_username
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-issuer-url=https://iam-icsc.cloud.infn.it/ #IAM di ICSC
 - --oidc-client-id=icsc-test-cp-02-aud #IAM di ICSC
<% else -%>
 - --oidc-issuer-url=https://iam.cloud.infn.it/ #IAM di DataCloud
 - --oidc-client-id=test-cp-01-aud #IAM di DataCloud
<% end -%>
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-groups-claim=wlcg.groups #IAM di ICSC
<% else -%>
 - --oidc-groups-claim=groups #IAM di DataCloud
<% end -%>
<% end -%>
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e dallo IAM di ICSC con RBAC

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
<% if @role == 'server' -%>
kube-apiserver-arg:
 - --oidc-username-claim=preferred_username
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-issuer-url=https://iam-icsc.cloud.infn.it/ #IAM di ICSC
 - --oidc-client-id=icsc-test-cp-02-aud #IAM di ICSC
<% else -%>
 - --oidc-issuer-url=https://iam.cloud.infn.it/ #IAM di DataCloud
 - --oidc-client-id=test-cp-01-aud #IAM di DataCloud
<% end -%>
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-groups-claim=wlcg.groups #IAM di ICSC
<% else -%>
 - --oidc-groups-claim=groups #IAM di DataCloud
<% end -%>
<% end -%>
```

Autenticazione ed autorizzazione tramite API Server con ID Token rilasciato dallo IAM di DataCloud e dallo IAM di ICSC con RBAC

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
<% if @role == 'server' -%>
kube-apiserver-arg:
 - --oidc-username-claim=preferred_username
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-issuer-url=https://iam-icsc.cloud.infn.it/ #IAM di ICSC
 - --oidc-client-id=icsc-test-cp-02-aud #IAM di ICSC
<% else -%>
 - --oidc-issuer-url=https://iam.cloud.infn.it/ #IAM di DataCloud
 - --oidc-client-id=test-cp-01-aud #IAM di DataCloud
<% end -%>
<% if ("192.168.0.1" == @facts['networking']['ip']) -%>
 - --oidc-groups-claim=wlcg.groups #IAM di ICSC
<% else -%>
 - --oidc-groups-claim=groups #IAM di DataCloud
<% end -%>
<% end -%>
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    server: https://192.168.X.X:6443
#    server: https://192.134.X.X:6443
#    server: https://192.135.X.X:6443
#    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    server: https://192.168.X.X:6443
#    server: https://192.134.X.X:6443
#    server: https://192.135.X.X:6443
#    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
    server: https://192.168.X.X:6443
#    server: https://192.134.X.X:6443
#    server: https://192.135.X.X:6443
#    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
#    server: https://192.168.X.X:6443
    server: https://192.134.X.X:6443 #config-rke2-milano.yaml
#    server: https://192.135.X.X:6443
#    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
#    server: https://192.168.X.X:6443
#    server: https://192.134.X.X:6443
    server: https://192.135.X.X:6443 #icsc-config-rke2-milano.yaml
#    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

File di configurazione per diversi endpoint

```
#Configurando opportunamente il file di configurazione è possibile accedere all'API Server del cluster tramite:
#IP locale
#IPv4 pubblico
#IPv6 pubblico

#/etc/rancher/rke2/rke2.yaml
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
#    server: https://192.168.X.X:6443
#    server: https://192.134.X.X:6443
#    server: https://192.135.X.X:6443
    server: https://[2001:X:X:X::X]:6443
  name: default
contexts:
- context:
    cluster: default
    user: default
  name: default
current-context: default
kind: Config
preferences: {}
```

ID Token a confronto (IAM di DataCloud e IAM di ICSC)

ID Token - IAM di DataCloud


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```


ID Token - IAM di ICSC


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson_enti
e email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

ID Token a confronto (IAM di DataCloud e IAM di ICSC)

ID Token - IAM di DataCloud


```
{
  "sub": "[redacted]",
  "iss": "https://iam.cloud.infn.it/",
  "groups": [
    "users",
    "end-users-catchall",
    "users/s3",
    "users/naas",
    "users/catchall",
    "admins"
  ],
  "preferred_username": "[redacted]",
  "organisation_name": "infn-cloud",
  "client_id": "[redacted]",
  "aud": "test-cp-01-aud",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "email": "[redacted]"
}
```


ID Token - IAM di ICSC


```
{
  "sub": "[redacted]",
  "iss": "https://iam-icsc.cloud.infn.it/",
  "preferred_username": "[redacted]",
  "client_id": "[redacted]",
  "wlcg.ver": "1.0",
  "aud": "icsc-test-cp-02-aud",
  "nbf": [redacted],
  "scope": "entitlements address openid profile eduperson_enti
e email wlcg.groups",
  "name": "Francesco Dalessandro",
  "exp": [redacted],
  "iat": [redacted],
  "jti": "[redacted]",
  "wlcg.groups": [
    "/test",
    "/test/k8s-test-milano"
  ],
  "email": "[redacted]"
}
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

```
#ip a
    1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
    2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
    1000
       link/ether 00:11:22:33:44:f4 brd ff:ff:ff:ff:ff:ff
       altname enp0s18
       inet 192.168.0.1/24 brd 192.168.0.255 scope global noprefixroute ens18
          valid_lft forever preferred_lft forever
       inet6 fe80::211:22ff:fe33:44f4/64 scope link
          valid_lft forever preferred_lft forever
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

```
#ip a
    1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
       link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
       inet 127.0.0.1/8 scope host lo
          valid_lft forever preferred_lft forever
       inet6 ::1/128 scope host
          valid_lft forever preferred_lft forever
    2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen
    1000
       link/ether 00:11:22:33:44:f4 brd ff:ff:ff:ff:ff:ff
       altname enp0s18
       inet 192.168.0.1/24 brd 192.168.0.255 scope global noprefixroute ens18
          valid_lft forever preferred_lft forever
       inet6 fe80::211:22ff:fe33:44f4/64 scope link
          valid_lft forever preferred_lft forever
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

#nmcli con mod ens18 ipv6.addresses 2001:X:X:X::X01/64 ipv6.gateway 2001:X:X:X::1 ipv6.dns 2001:X:X:X::3 ipv6.dns-search mi.infn.it ipv6.method manual connection.autoconnect yes

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:11:22:33:44:f4 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.0.1/24 brd 192.168.0.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet6 2001:X:X:X::X01/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::211:22ff:fe33:44f4/64 scope link
       valid_lft forever preferred_lft forever
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

#nmcli con mod ens18 ipv6.addresses 2001:X:X:X::X01/64 ipv6.gateway 2001:X:X:X::1 ipv6.dns 2001:X:X:X::3 ipv6.dns-search mi.infn.it ipv6.method manual connection.autoconnect yes

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:11:22:33:44:f4 brd ff:ff:ff:ff:ff:ff
    altname enp0s18
    inet 192.168.0.1/24 brd 192.168.0.255 scope global noprefixroute ens18
       valid_lft forever preferred_lft forever
    inet6 2001:X:X:X::X01/64 scope global noprefixroute
       valid_lft forever preferred_lft forever
    inet6 fe80::211:22ff:fe33:44f4/64 scope link
       valid_lft forever preferred_lft forever
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

Ping in modalità IPv6

```
[root@localhost ~]# ping -6 google.com
PING google.com(123.example.net (2a00:1450:4002:402::200e)) 56 data bytes
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=1 ttl=117 time=0.776 ms
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=2 ttl=117 time=0.686 ms
```

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
cluster-cidr: "10.42.0.0/16,fd00:42::/56"
service-cidr: "10.43.0.0/16,fd00:43::/112"
```

kubectl get nodes -o=custom-columns="NAME:.metadata.name,ADDRESSES:.status.addresses[?(@.type=='InternalIP')].address,PODCIDRS:.spec.podCIDRs[*]"

NAME	ADDRESSES	PODCIDRS
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::e	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::f	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::0	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::1	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::2	10.42.[redacted].0/24,fd00:[redacted]/64

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

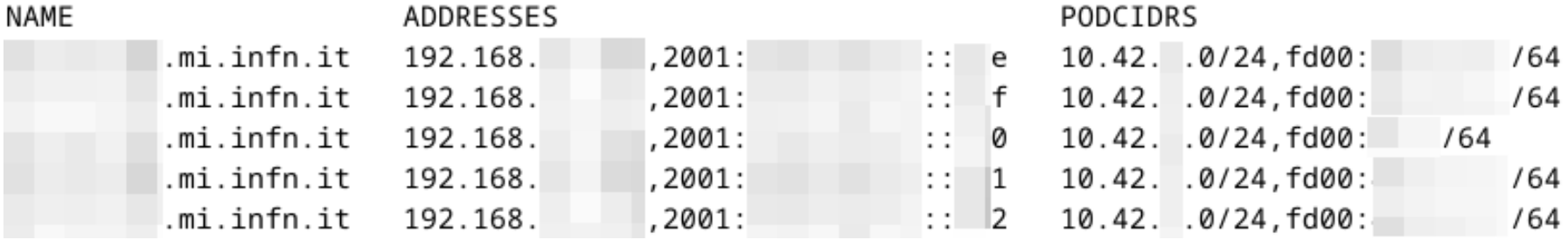
Ping in modalità IPv6

```
[root@localhost ~]# ping -6 google.com
PING google.com(123.example.net (2a00:1450:4002:402::200e)) 56 data bytes
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=1 ttl=117 time=0.776 ms
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=2 ttl=117 time=0.686 ms
```

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
cluster-cidr: "10.42.0.0/16,fd00:42::/56"
service-cidr: "10.43.0.0/16,fd00:43::/112"
```

kubectl get nodes -o=custom-columns="NAME:.metadata.name,ADDRESSES:.status.addresses[?(@.type=='InternalIP')].address,PODCIDRS:.spec.podCIDRs[*]"



NAME	ADDRESSES	PODCIDRS
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::e	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::f	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::0	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::1	10.42.[redacted].0/24,fd00:[redacted]/64
[redacted].mi.infn.it	192.168.[redacted],2001:[redacted]::2	10.42.[redacted].0/24,fd00:[redacted]/64

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

Ping in modalità IPv6

```
[root@localhost ~]# ping -6 google.com
PING google.com(123.example.net (2a00:1450:4002:402::200e)) 56 data bytes
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=1 ttl=117 time=0.776 ms
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=2 ttl=117 time=0.686 ms
```

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
cluster-cidr: "10.42.0.0/16,fd00:42::/56"
service-cidr: "10.43.0.0/16,fd00:43::/112"
```

kubectl get nodes -o=custom-columns="NAME:.metadata.name,ADDRESSES:.status.addresses[?(@.type=='InternalIP')].address,PODCIDRS:.spec.podCIDRs[*]"

```
NAME                 ADDRESSES                         PODCIDRS
[illegible].mi.infn.it   192.168.[illegible],2001:[illegible]::[illegible]e   10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it   192.168.[illegible],2001:[illegible]::[illegible]f   10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it   192.168.[illegible],2001:[illegible]::[illegible]0   10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it   192.168.[illegible],2001:[illegible]::[illegible]1   10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it   192.168.[illegible],2001:[illegible]::[illegible]2   10.42.[illegible].0/24,fd00:[illegible]/64
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

Ping in modalità IPv6

```
[root@localhost ~]# ping -6 google.com
PING google.com(123.example.net (2a00:1450:4002:402::200e)) 56 data bytes
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=1 ttl=117 time=0.776 ms
64 bytes from 123.example.net (2a00:1450:4002:402::200e): icmp_seq=2 ttl=117 time=0.686 ms
```

/etc/puppetlabs/code/environments/production/modules/rke2/templates/config.yaml.erb

```
cluster-cidr: "10.42.0.0/16,fd00:42::/56"
service-cidr: "10.43.0.0/16,fd00:43::/112"
```

kubectl get nodes -o=custom-columns="NAME:.metadata.name,ADDRESSES:.status.addresses[?(@.type=='InternalIP')].address,PODCIDRS:.spec.podCIDRs[*]"

NAME	ADDRESSES	PODCIDRS
[illegible].mi.infn.it	192.168.[illegible],2001:[illegible]::[illegible]e	10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it	192.168.[illegible],2001:[illegible]::[illegible]f	10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it	192.168.[illegible],2001:[illegible]::[illegible]0	10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it	192.168.[illegible],2001:[illegible]::[illegible]1	10.42.[illegible].0/24,fd00:[illegible]/64
[illegible].mi.infn.it	192.168.[illegible],2001:[illegible]::[illegible]2	10.42.[illegible].0/24,fd00:[illegible]/64

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345   1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345   1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                      READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345    1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345   1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345   1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Installazione e configurazione del cluster in modalità IPv4/IPv6 dual-stack

kubectl apply -f web.yaml

```
[root@localhost ~]# kubectl get pod
NAME                     READY   STATUS    RESTARTS      AGE
httpd-99c6f8dabc-12345   1/1     Running   1 (8d ago)    12d
```

Descrizione del pod:

kubectl describe pod httpd-99c6f8dabc-12345

Modalità dual-stack IPv4/IPv6:

```
Status:         Running
IP:             10.42.X.X
IPs:
  IP:           10.42.X.X
  IP:           fd00:42:X:X::X
```

Da un nodo del cluster, possiamo dare i seguenti comandi:

```
[root@localhost ~]# curl 10.42.X.X
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# curl [fd00:42:X:X::X]
<html><body><h1>It works!</h1></body></html>
```

```
Name:             httpd-
Namespace:        default
Priority:         0
Service Account:  default
Node:             .mi.infn.it/192.168.
Start Time:       Tue, 29 Apr 2025 08:31:35 +0200
Labels:           app=httpd-on-k8s
                  pod-template-hash=
                  tier=httpd
Annotations:      cni.projectcalico.org/containerID:
                  cni.projectcalico.org/podIP: 10.42.    /32
                  cni.projectcalico.org/podIPs: 10.42.    /32,fd00:42:
Status:           Running
IP:               10.42.
IPs:
  IP:             10.42.
  IP:             fd00:42:
Controlled By:    ReplicaSet/httpd-
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.9/config/manifests/metallb-native.yaml

#kubectl apply -f metallb-config.yaml

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: rke2-ipaddresspool
  namespace: metallb-system
spec:
  addresses:
  - 192.1.X.1/32
  - 192.1.X.2/32
  - 192.1.X.3/32
  - 192.1.X.4/32
  - 2001:1:1:1::Y:1/128
  - 2001:1:1:1::Y:2/128
  - 2001:1:1:1::Y:3/128
  - 2001:1:1:1::Y:4/128
  autoAssign: true
```

#kubectl apply -f metallb-l2advertisement.yamll

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: rke2-l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - rke2-ipaddresspool
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.9/config/manifests/metallb-native.yaml

#kubectl apply -f metallb-config.yaml

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: rke2-ipaddresspool
  namespace: metallb-system
spec:
  addresses:
  - 192.1.X.1/32
  - 192.1.X.2/32
  - 192.1.X.3/32
  - 192.1.X.4/32
  - 2001:1:1:1::Y:1/128
  - 2001:1:1:1::Y:2/128
  - 2001:1:1:1::Y:3/128
  - 2001:1:1:1::Y:4/128
  autoAssign: true
```

#kubectl apply -f metallb-l2advertisement.yamll

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: rke2-l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - rke2-ipaddresspool
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.9/config/manifests/metallb-native.yaml

#kubectl apply -f metallb-config.yaml

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: rke2-ipaddresspool
  namespace: metallb-system
spec:
  addresses:
  - 192.1.X.1/32
  - 192.1.X.2/32
  - 192.1.X.3/32
  - 192.1.X.4/32
  - 2001:1:1:1::Y:1/128
  - 2001:1:1:1::Y:2/128
  - 2001:1:1:1::Y:3/128
  - 2001:1:1:1::Y:4/128
  autoAssign: true
```

#kubectl apply -f metallb-l2advertisement.yamll

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: rke2-l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - rke2-ipaddresspool
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.14.9/config/manifests/metallb-native.yaml

#kubectl apply -f metallb-config.yaml

```
apiVersion: metallb.io/v1beta1
kind: IPAddressPool
metadata:
  name: rke2-ipaddresspool
  namespace: metallb-system
spec:
  addresses:
  - 192.1.X.1/32
  - 192.1.X.2/32
  - 192.1.X.3/32
  - 192.1.X.4/32
  - 2001:1:1:1::Y:1/128
  - 2001:1:1:1::Y:2/128
  - 2001:1:1:1::Y:3/128
  - 2001:1:1:1::Y:4/128
  autoAssign: true
```

#kubectl apply -f metallb-l2advertisement.yamll

```
apiVersion: metallb.io/v1beta1
kind: L2Advertisement
metadata:
  name: rke2-l2advertisement
  namespace: metallb-system
spec:
  ipAddressPools:
  - rke2-ipaddresspool
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f web.yaml

type: LoadBalancer

ipFamilyPolicy: PreferDualStack
ipFamilies:
- IPv4
- IPv6

Per IPv6, usando RKE2, non scritte delle regole di iptables.

Pull request per correggere il bug all'indirizzo https://github.com/rancher/rke2/pull/8029 di cui ne è stato fatto il merge.

Test fatto sul codice di RKE2 modificato, prima che fosse stato fatto il merge della pull request.

```
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP                    PORT(S)        AGE
httpd        LoadBalancer   10.43.         192.13 ,2001:                  80:30301/TCP   16d
kubernetes   ClusterIP      10.43.         <none>                         443/TCP        41d

root@ :~#  curl 192.13
<html><body><h1>It works!</h1></body></html>
root@ :~# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f web.yaml

type: LoadBalancer

ipFamilyPolicy: PreferDualStack
ipFamilies:
- IPv4
- IPv6

Per IPv6, usando RKE2, non scritte delle regole di iptables.

Pull request per correggere il bug all'indirizzo https://github.com/rancher/rke2/pull/8029 di cui ne è stato fatto il merge.

Test fatto sul codice di RKE2 modificato, prima che fosse stato fatto il merge della pull request.

```
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP                    PORT(S)        AGE
httpd        LoadBalancer   10.43.         192.13 ,2001:                  80:30301/TCP   16d
kubernetes   ClusterIP      10.43.         <none>                         443/TCP        41d

root@ :~#  curl 192.13
<html><body><h1>It works!</h1></body></html>
root@ :~# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f web.yaml

type: LoadBalancer

ipFamilyPolicy: PreferDualStack
ipFamilies:
- IPv4
- IPv6

Per IPv6, usando RKE2, non scritte delle regole di iptables.

Pull request per correggere il bug all'indirizzo https://github.com/rancher/rke2/pull/8029 di cui ne è stato fatto il merge.

Test fatto sul codice di RKE2 modificato, prima che fosse stato fatto il merge della pull request.

```
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP                    PORT(S)        AGE
httpd        LoadBalancer   10.43.         192.13 ,2001:                  80:30301/TCP   16d
kubernetes   ClusterIP      10.43.         <none>                         443/TCP        41d

root@ :~#  curl 192.13
<html><body><h1>It works!</h1></body></html>
root@ :~# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f web.yaml

type: LoadBalancer

ipFamilyPolicy: PreferDualStack
ipFamilies:
- IPv4
- IPv6

Per IPv6, usando RKE2, non scritte delle regole di iptables.

Pull request per correggere il bug all'indirizzo https://github.com/rancher/rke2/pull/8029 di cui ne è stato fatto il merge.

Test fatto sul codice di RKE2 modificato, prima che fosse stato fatto il merge della pull request.

```
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP    EXTERNAL-IP                PORT(S)        AGE
httpd        LoadBalancer   10.43.        192.13 ,2001:              80:30301/TCP   16d
kubernetes   ClusterIP      10.43.        <none>                     443/TCP        41d

root@ :~#  curl 192.13
<html><body><h1>It works!</h1></body></html>
root@ :~# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
```

Load balancer con MetalLB (modalità Layer2)

kubectl apply -f web.yaml

type: LoadBalancer

ipFamilyPolicy: PreferDualStack
  ipFamilies:
  - IPv4
  - IPv6

Per IPv6, usando RKE2, non scritte delle regole di iptables.

Pull request per correggere il bug all'indirizzo https://github.com/rancher/rke2/pull/8029 di cui ne è stato fatto il merge.

Test fatto sul codice di RKE2 modificato, prima che fosse stato fatto il merge della pull request.

```
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP     EXTERNAL-IP                    PORT(S)        AGE
httpd        LoadBalancer   10.43.         192.13 ,2001:                  80:30301/TCP   16d
kubernetes   ClusterIP      10.43.         <none>                         443/TCP        41d
root@ :~#  curl 192.13
<html><body><h1>It works!</h1></body></html>
root@ :~# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
```

Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2

- Creati due server, uno basato su AlmaLinux e l'altro su Debian per il backup, tramite rsync, dei nodi più importanti (3 Control Plane, DNS, TheForeman e Puppet server su BM, ecc.)
- Create procedure per aggiungere, rimuovere e sostituire un nodo

```
[root@xxxx ~]# kubectl cordon <node-name>
node/<node-name> cordoned

[root@xxxx ~]# kubectl drain <node-name> --force --ignore-daemonsets --delete-emptydir-data
node/<node-name> already cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/rke2-canal-hmj6c, kube-system/rke2-ingress-nginx-controller-zrwxd
node/<node-name> drained

[root@xxxx ~]# kubectl delete node <node-name>
node "<node-name>" deleted
```

Fondamentale non far perdere il quorum. Procedura testata per la sostituzione di un CP senza spegnere il cluster.

Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2

- Creati due server, uno basato su AlmaLinux e l'altro su Debian per il backup, tramite rsync, dei nodi più importanti (3 Control Plane, DNS, TheForeman e Puppet server su BM, ecc.)
- Create procedure per aggiungere, rimuovere e sostituire un nodo

```
[root@xxxx ~]# kubectl cordon <node-name>
node/<node-name> cordoned

[root@xxxx ~]# kubectl drain <node-name> --force --ignore-daemonsets --delete-emptydir-data
node/<node-name> already cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/rke2-canal-hmj6c, kube-system/rke2-ingress-nginx-controller-zrwxd
node/<node-name> drained

[root@xxxx ~]# kubectl delete node <node-name>
node "<node-name>" deleted
```

Fondamentale non far perdere il quorum. Procedura testata per la sostituzione di un CP senza spegnere il cluster.

Copia, ripristino, aggiunta, rimozione e sostituzione di un nodo di un cluster RKE2

- Creati due server, uno basato su AlmaLinux e l'altro su Debian per il backup, tramite rsync, dei nodi più importanti (3 Control Plane, DNS, TheForeman e Puppet server su BM, ecc.)
- Create procedure per aggiungere, rimuovere e sostituire un nodo

```
[root@xxxx ~]# kubectl cordon <node-name>
node/<node-name> cordoned

[root@xxxx ~]# kubectl drain <node-name> --force --ignore-daemonsets –-delete-emptydir-data
node/<node-name> already cordoned
Warning: ignoring DaemonSet-managed Pods: kube-system/rke2-canal-hmj6c, kube-system/rke2-ingress-nginx-controller-zrwxd
node/<node-name> drained

[root@xxxx ~]# kubectl delete node <node-name>
node "<node-name>" deleted
```

Fondamentale non far perdere il quorum. Procedura testata per la sostituzione di un CP senza spegnere il cluster.

Spegnimento, accensione ed aggiornamento di un cluster RKE2

- Create procedure per lo spegnimento e l'accensione del cluster

  Fondamentale non far perdere il quorum

- Create procedure per l'aggiornamento del SO e della versione di RKE2 (Rolling update)

  Versione iniziale → v1.26.13+rke2r1

  v1.26.14+rke2r1 #minor release del ramo v.1.26X
  v1.26.15+rke2r1 #minor release del ramo v.1.26X
  v1.27.16+rke2r2 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.27X
  v1.28.15+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.28X
  v1.29.14+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.29X
  v1.30.10+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.30X
  v1.31.6+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.31X
  v1.32.3+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.32X

  Versione finale → v1.32.3+rke2r1

Spegnimento, accensione ed aggiornamento di un cluster RKE2

- Create procedure per lo spegnimento e l'accensione del cluster

  Fondamentale non far perdere il quorum

- Create procedure per l'aggiornamento del SO e della versione di RKE2 (Rolling update)

  Versione iniziale → v1.26.13+rke2r1

  v1.26.14+rke2r1 #minor release del ramo v.1.26X
  v1.26.15+rke2r1 #minor release del ramo v.1.26X
  v1.27.16+rke2r2 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.27X
  v1.28.15+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.28X
  v1.29.14+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.29X
  v1.30.10+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.30X
  v1.31.6+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.31X
  v1.32.3+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.32X

  Versione finale → v1.32.3+rke2r1

Spegnimento, accensione ed aggiornamento di un cluster RKE2

- Create procedure per lo spegnimento e l'accensione del cluster

  Fondamentale non far perdere il quorum

- Create procedure per l'aggiornamento del SO e della versione di RKE2 (Rolling update)

  Versione iniziale → v1.26.13+rke2r1

  v1.26.14+rke2r1 #minor release del ramo v.1.26X
  v1.26.15+rke2r1 #minor release del ramo v.1.26X
  v1.27.16+rke2r2 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.27X
  v1.28.15+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.28X
  v1.29.14+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.29X
  v1.30.10+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.30X
  v1.31.6+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.31X
  v1.32.3+rke2r1 #nel momento in cui si scrive è l'ultima minor release del ramo v.1.32X

  Versione finale → v1.32.3+rke2r1

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Simulazione di scenari catatrofici

Un cluster RKE2 con 24 worker node e 3 control plane in 4 ore (bare metal)

60 minuti -> Installazione di TheForeman + DHCP + TFTP

20 minuti -> TheForeman pronto all'uso (sottorete, SO, dispositivo di installazione ed associazione template)

30 minuti -> Creazione di 2 repository

30 minuti -> Installazione e configurazione del Puppet server (SO + Puppet server)

28 minuti (circa 30 minuti) -> Creazione di 27 host in TheForeman

27 minuti (circa 30 minuti) -> Riavvio dei worker node, tramite iDRAC, per l'installazione da scheda di rete

10 minuti -> Attesa per il completamento dell'installazione dell'ultimo nodo

15 minuti circa (30 secondi x 27 nodi) -> Riavvio dei worker node per la corretta configurazione di Puppet agent e di altre impostazioni dei worker node/control plane

5 minuti -> Assegnazione degli indirizzi IPv6 su tutti i nodi, in modo automatico, tramite il lancio di uno script fatto appositamente

5 minuti -> Richiesta del catalogo, da parte degli agent, al Puppet server e creazione del cluster RKE2 (*)

(*) Indirizzi PODCIDR assegnati in modo non ordinato

Sostituzione di un control plane di un cluster RKE2 in 30 minuti (bare metal)

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Sviluppi futuri

→ Assegnazione, in modo automatico, di un FQDN ai deploymet degli utenti che ne fanno richiesta

→ Sviluppare altri metodi di autorizzazione (individuate diverse modalità)

→ Estensione geografica del Cluster

→ Ospitare un carico di lavoro HTCondor (Poster C. Marcon)

→ Integrazione con INFN DataCloud

Anche le sezioni di ROMA1 e BARI stanno facendo un lavoro simile a quello svolto dalla sezione di MILANO

Rolling update (SO + RKE2) al 23/05/2025 + patch per IPv6 in distribuzione

```
root@ :~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
.mi.infn.it   Ready    control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@  ~]# kubectl get nodes
NAME          STATUS   ROLES                       AGE     VERSION
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@  ~]#
```

```
root@ :~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready    control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@  ~]# hostnamectl
 Static hostname:  .mi.infn.it
       Icon name: computer-desktop
         Chassis: desktop 🖥️
      Machine ID:
         Boot ID:
Operating System: AlmaLinux 9.6 (Sage Margay)
     CPE OS Name: cpe:/o:almalinux:almalinux:9::baseos
          Kernel: Linux 5.14.0-  .x86_64
    Architecture:
 Hardware Vendor:
  Hardware Model:
Firmware Version:
[root@  ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP   EXTERNAL-IP          PORT(S)        AGE
httpd        LoadBalancer   10.43.       192.13  ,2001:       80:30338/TCP   35d
kubernetes   ClusterIP      10.43.       <none>               443/TCP        35d
[root@  ~]# curl [2001:  ]
<html><body><h1>It works!</h1></body></html>
[root@  ~]#
```

Rolling update (SO + RKE2) al 23/05/2025 + patch per IPv6 in distribuzione

```
root@ :~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
.mi.infn.it   Ready    control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@ ~]# kubectl get nodes
NAME              STATUS   ROLES                       AGE     VERSION
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    <none>                      35d     v1.33.0+rke2r1
.mi.infn.it       Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it       Ready    control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it       Ready    control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@ ~]#
```

```
root@ :~
File  Modifica  Visualizza  Cerca  Terminale  Aiuto
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   <none>                      35d     v1.33.0+rke2r1
.mi.infn.it   Ready   control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready   control-plane,etcd,master   35d     v1.33.0+rke2r1
.mi.infn.it   Ready   control-plane,etcd,master   3h38m   v1.33.0+rke2r1
[root@ ~]# hostnamectl
 Static hostname: .mi.infn.it
       Icon name: computer-desktop
         Chassis: desktop 🖥️
      Machine ID:
         Boot ID:
Operating System: AlmaLinux 9.6 (Sage Margay)
     CPE OS Name: cpe:/o:almalinux:almalinux:9::baseos
          Kernel: Linux 5.14.0- .x86_64
    Architecture:
 Hardware Vendor:
  Hardware Model:
Firmware Version:
[root@ ~]# kubectl get svc
NAME         TYPE           CLUSTER-IP   EXTERNAL-IP        PORT(S)        AGE
httpd        LoadBalancer   10.43.       192.13 ,2001:      80:30338/TCP   35d
kubernetes   ClusterIP      10.43.       <none>             443/TCP        35d
[root@ ~]# curl [2001: ]
<html><body><h1>It works!</h1></body></html>
[root@ ~]#
```

Guide

https://gitlab.com/lamp0/autenticazione-tramite-id-token-ed-autorizzazione-tramite-rbac-su-di-un-cluster-kubernetes-o-rke2

https://gitlab.com/lamp0/autenticazione-ed-autorizzazione-su-di-un-cluster-kubernetes-o-rke2-con-due-o-piu-iam

https://gitlab.com/lamp0/configurazione-di-un-cluster-kubernetes-o-rke2-in-modalita-dual-stack-ipv4-ipv6

https://gitlab.com/lamp0/installazione-e-configurazione-di-un-load-balancer-per-un-cluster-kubernetes-o-rke2-su-bare-metal

https://gitlab.com/lamp0/copia-ripristino-aggiunta-rimozione-e-sostituzione-di-un-nodo-di-un-cluster-rke2

https://gitlab.com/lamp0/spegnimento-accensione-ed-aggiornamento-di-un-cluster-rke2

https://gitlab.com/lamp0/un-cluster-rke2-con-24-worker-node-e-3-control-plane-in-4-ore-e-sostituzione-di-un-control-plane-in-30-minuti

Link esterni

https://openid.net/specs/openid-connect-core-1_0.html#IDToken

https://openid.net/connect/

https://indigo-dc.gitbook.io/oidc-agent/user/oidc-gen/provider/iam

https://github.com/indigo-dc/oidc-agent

https://baltig.infn.it/infn-cloud/wp1/rke2

https://it.wikipedia.org/wiki/IPv6

https://fibra.click/ipv6/

https://www.extraordy.com/guida-allipv6-link-local-address/

https://docs.rke2.io/networking/basic_network_options?CNIplugin=Canal+CNI+Plugin#dual-stack-configuration

https://metallb.io/

https://github.com/rancher/rke2/pull/8029

https://www.theforeman.org/

https://www.puppet.com/

https://docs.rke2.io/

https://kubernetes.io/docs/home/

Finanziato dall'Unione europea
NextGenerationEU

Ministero dell'Università e della Ricerca

Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

ICSC

Centro Nazionale di Ricerca in HPC
Big Data and Quantum Computing

Supercomputing shaping the future