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

Realizzazione di un ambiente PostgreSQL distribuito ad alta affidabilità con replica sincrona, backup cloud-based e strumenti di monitoring

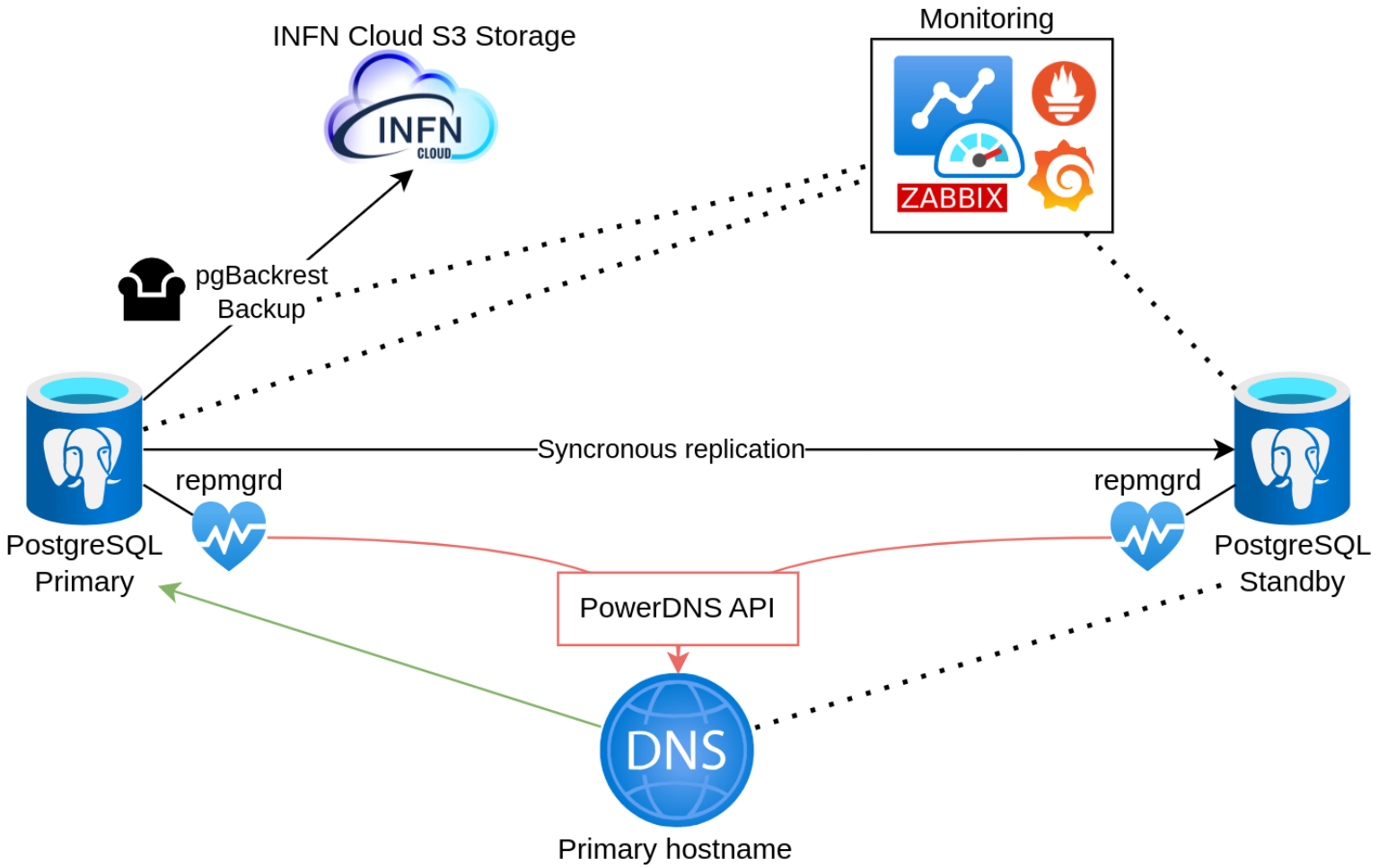
Speaker: **Nadir Marcelli**

Workshop sul Calcolo nell'INFN, Palau 20 – 24 maggio 2024

Sommario

- Architettura dell'ambiente
- Componenti e funzionalità
- Monitoring
- Installazione con Ansible
- Gestione del cluster

Architettura dell'ambiente



PostgreSQL

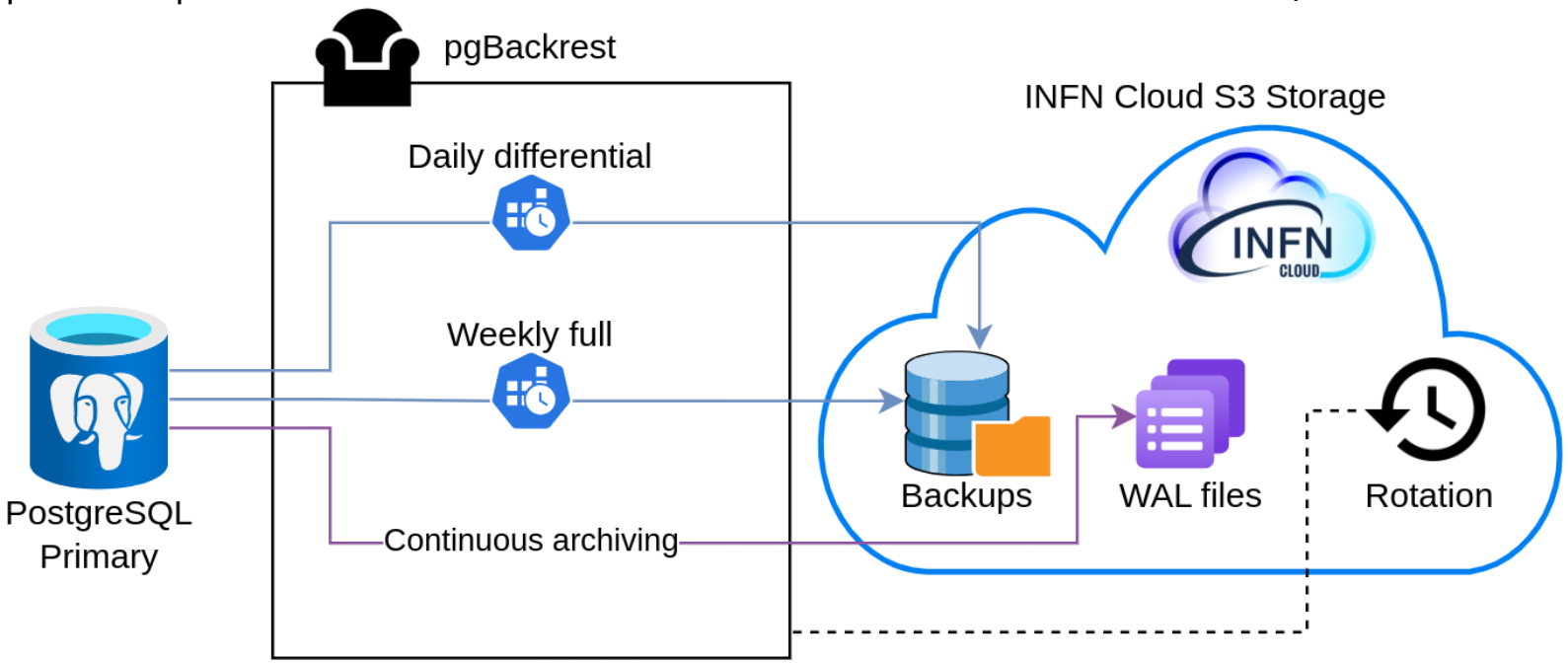
PostgreSQL: Backup

- **SQL dump**
  - Genera la sequenza di query da eseguire per ricreare il database nello stesso stato in cui era al momento del dump
  - PostgreSQL fornisce i comandi *pg_dump* e *pg_dumpall*
- **File system level**
  - Copia manuale dei file usati da PostgreSQL (i.e., $PG_DATA). Richiede l'nterruzione del servizio
  - PostgreSQL fornisce il comando *pg_basebackup* per eseguire questo tipo di backup senza interrompere il servizio
- **Archiviazione continua**
  - PostgreSQL salva costantemente tutte le modifiche fatte al database in dei file chiamati *Write Ahead Log* (WAL)
  - I WAL file si trovano nella sottocartella pg_wal della cartella del cluster e sono segmentati in file da 16 MB (default)
  - Combinati con il file system level backup offrono una terza soluzione per il backup
  - PostgreSQL permette di specificare un comando shell da eseguire per salvare questi file dove e come si vuole

PostgreSQL: pgBackrest

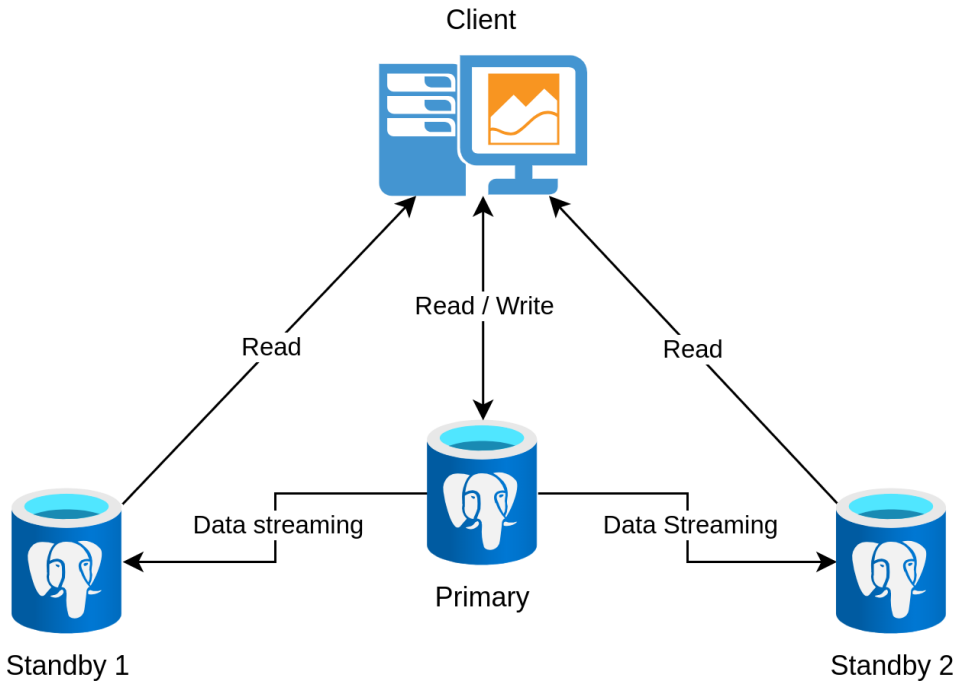
- **Soluzione affidabile per il backup e ripristino che offre diverse funzionalità**

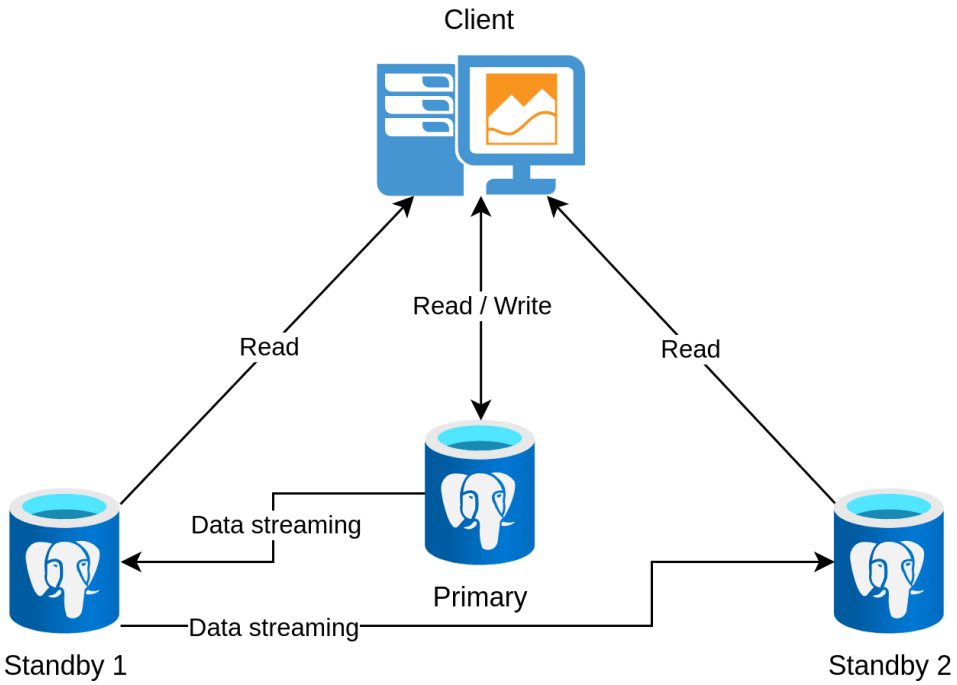
  - Controllo integrità del backup
  - Compressione e crittografia backup e WAL file
  - Supporto per backup cloud-based
  - Backup completi, differenziali e incrementali
  - Archiviazione dei WAL file
  - Rotazione dei backup e scadenza WAL file



PostgreSQL: Ambiente distribuito

- Diversi server collaborano per mantenere il servizio sempre attivo e raggiungibile
- Gli stessi dati sono distribuiti in diversi siti dando la possibilità di bilanciare il carico delle connessioni
- È necessario che i server siano consistenti tra loro

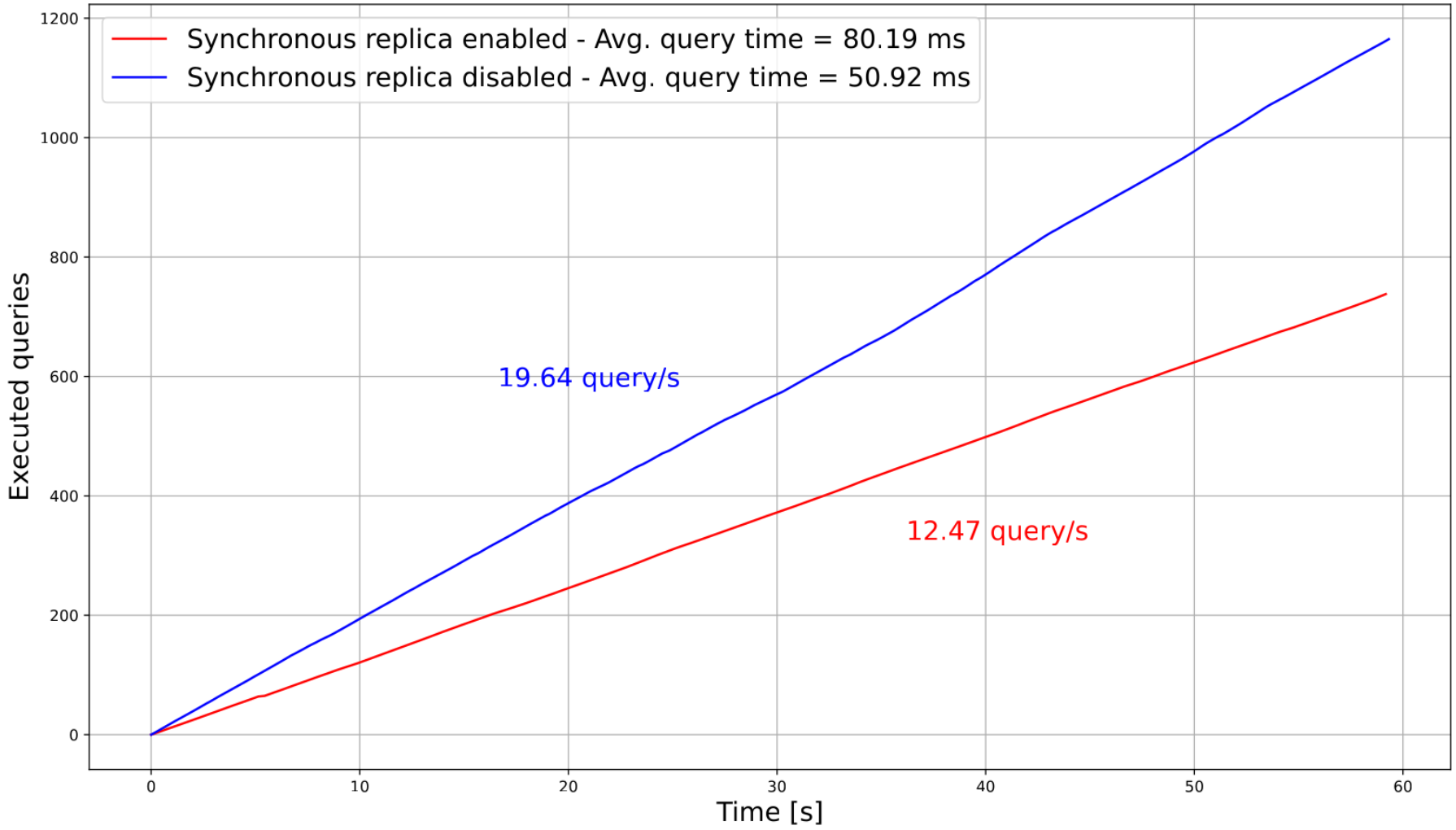




PostgreSQL: Propagazione dei dati

- **Archiviazione continua**
  - I WAL file vengono archiviati solo quando raggiungono una dimensione di 16 MB (default)
  - I server standby si aggiornano solo quando un nuovo WAL file viene archiviato
  - Si possono forzare dei checkpoint ad intervalli regolari per archiviare un WAL file anche se non è completo
  - I WAL file "incompleti" hanno comunque una dimensione di 16 MB
- **Streaming replication**
  - Il server primario stabilisce una connessione con gli standby (oppure in cascata) e trasmette i record dei WAL file mentre vengono generati, senza aspettare che il WAL file venga completamente riempito
  - Questa procedura è asincrona e presenta comunque un piccolo ritardo tra il primario e gli standby
- **Replica sincrona**
  - Ogni transazione di scrittura attende che il commit sia stato scritto nel WAL file sia dal primario che dagli standby
  - Questa modalità introduce una latenza nelle transazioni in scrittura

PostgreSQL: Latenza replica sincrona



Failover management

Failover management: repmgr

Software open-souce per la gestione della replica e del failover in un cluster PostgreSQL

Command line interface

repmgr

- Creazione nuovi server standby
- Promozione di uno standby a primario
- Switchover tra primario e standby
- Monitorare lo stato del cluster

Daemon

repmgrd

- Monitora lo stato e la connettività tra i server del cluster
- Esegue la promozione automatica in caso di failover
- Notifica gli eventi nel cluster
- Le notifiche possono far eseguire un comando custom

Failover management: Notifiche eventi

Ogni azione significativa di **repmgr** o **repmgrd** innesca l'esecuzione di un comando specificato nel file di configurazione di repmgr

```
event_notification_command = '/path/to/some/script %n %e %s'
```

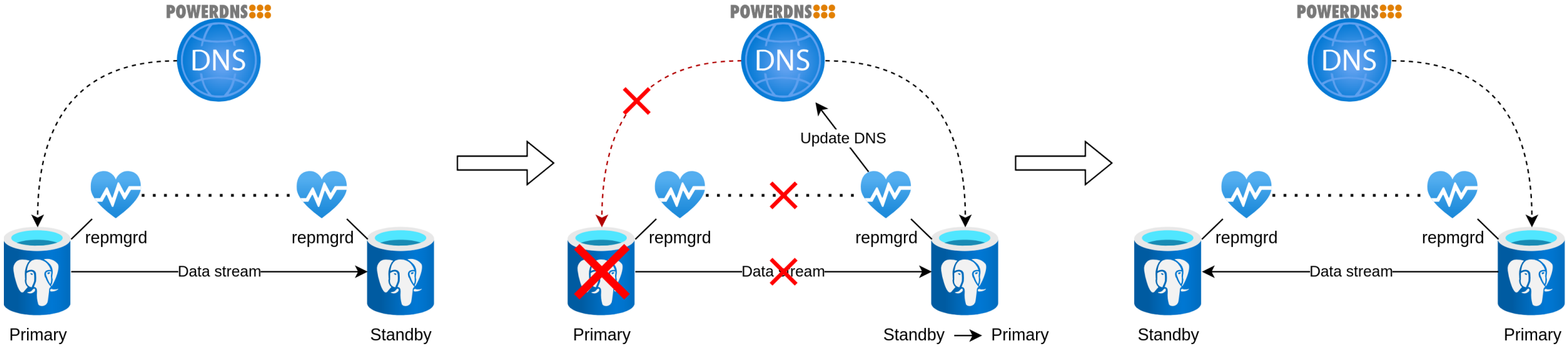
%n = ID del nodo %e = Tipo di evento %s = esito dell'evento

- La promozione automatica di uno standby in caso di failover è gestito di default da repmgr
- Lo script è utile per implementare azioni personalizzate a seguito di determinati eventi (es., aggiornare DNS, fencing del primario)

```
[postgres@psql ~]$ psql -d repmgr -U repmgr -c "SELECT * from repmgr.events;"
 node_id |             event              | successful |        event_timestamp        |                                  details
---------+--------------------------------+------------+-------------------------------+---------------------------------------------------------------------
       1 | cluster_created                | t          | 2024-02-14 09:03:27.457557+01 |
       1 | primary_register               | t          | 2024-02-14 09:03:27.50033+01  |
       1 | repmgrd_start                  | t          | 2024-02-14 09:03:29.278539+01 | monitoring cluster primary "pgsql1" (ID: 1)
       2 | standby_register               | t          | 2024-02-14 09:03:30.649457+01 | standby registration succeeded; upstream node ID is 1
       2 | repmgrd_start                  | t          | 2024-02-14 09:03:32.103127+01 | monitoring connection to upstream node "pgsql1" (ID: 1)
       1 | child_node_new_connect         | t          | 2024-02-14 09:03:35.488536+01 | new standby "pgsql2" (ID: 2) has connected
       2 | standby_unregister             | t          | 2024-02-14 09:29:56.736587+01 |
       1 | primary_unregister             | t          | 2024-02-14 09:31:10.723942+01 | node "pgsql1" (ID: 1) unregistered
       1 | primary_register               | t          | 2024-02-14 12:32:01.702546+01 |
       1 | repmgrd_shutdown               | t          | 2024-02-14 12:32:03.272131+01 | TERM signal received
       1 | repmgrd_start                  | t          | 2024-02-14 12:32:03.361276+01 | monitoring cluster primary "pgsql1" (ID: 1)
       2 | standby_register               | t          | 2024-02-14 12:32:04.685767+01 | standby registration succeeded; upstream node ID is 1
       2 | repmgrd_start                  | t          | 2024-02-14 12:32:06.072197+01 | monitoring connection to upstream node "pgsql1" (ID: 1)
       1 | child_node_new_connect         | t          | 2024-02-14 12:32:09.559795+01 | new standby "pgsql2" (ID: 2) has connected
       2 | standby_promote                | t          | 2024-02-14 12:35:36.236493+01 | server "pgsql2" (ID: 2) was successfully promoted to primary
       2 | standby_switchover             | t          | 2024-02-14 12:35:40.058337+01 | node "pgsql2" (ID: 2) promoted to primary, node "pgsql1" (ID: 1) demoted to standby
```

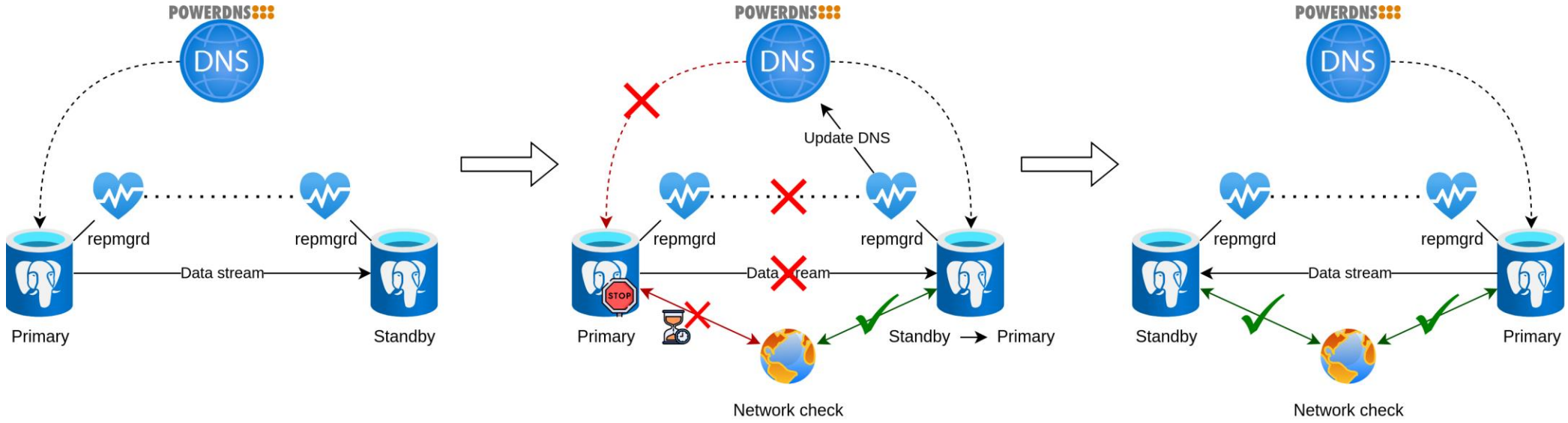
Failover management: Scenari supportati

- **Crash del servizio PostgreSQL o dell'intero nodo primario**
  - Repmgrd notifica di aver perso la connessione col primario
  - Se dopo un certo periodo di tempo la connessione non viene ripristinata, repmgrd invia un segnale di promozione allo standby
  - La promozione del nuovo primario genera una notifica repmgr che esegue lo script e aggiorna il DNS
  - Quando il vecchio primario torna online, repmgrd rileva la presenza di un altro primario nel cluster e avvia il server in standby



Failover management: Prevenzione split-brain scenario

- **Interruzione della rete e *fencing***
  - Il repmgrd del primario notifica di aver perso lo standby e il repmgrd dello standby notifica di aver perso il primario
  - In corrispondenza di questa notifica, entrambi i server effettuano un check della rete tramite un ping
  - Il server isolato dalla rete interrompe il servizio PostgreSQL ed effettua un check della rete finché non torna online
  - Quando il server isolato torna online, se è già presente un primario nel cluster, viene avviato come standby

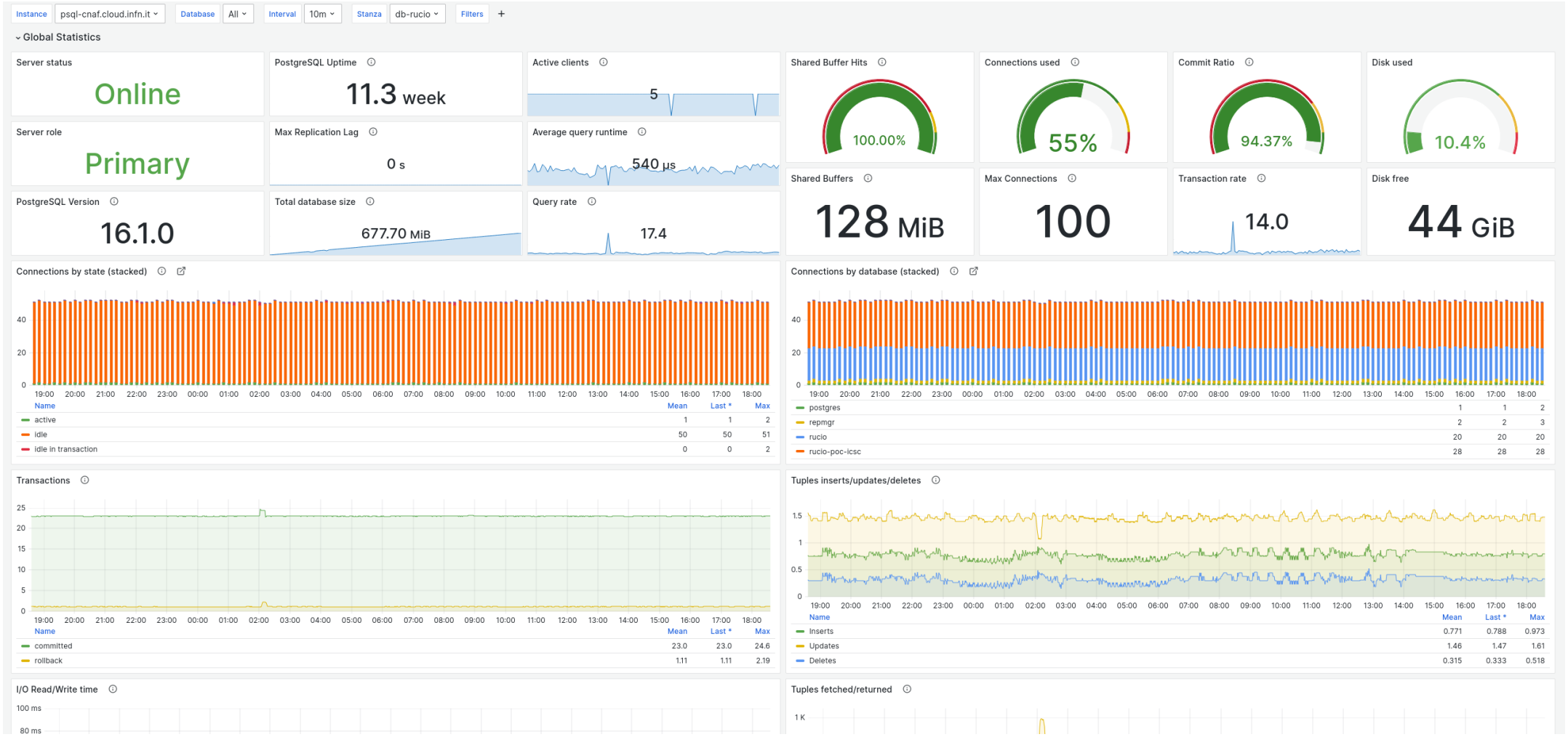


Monitoring

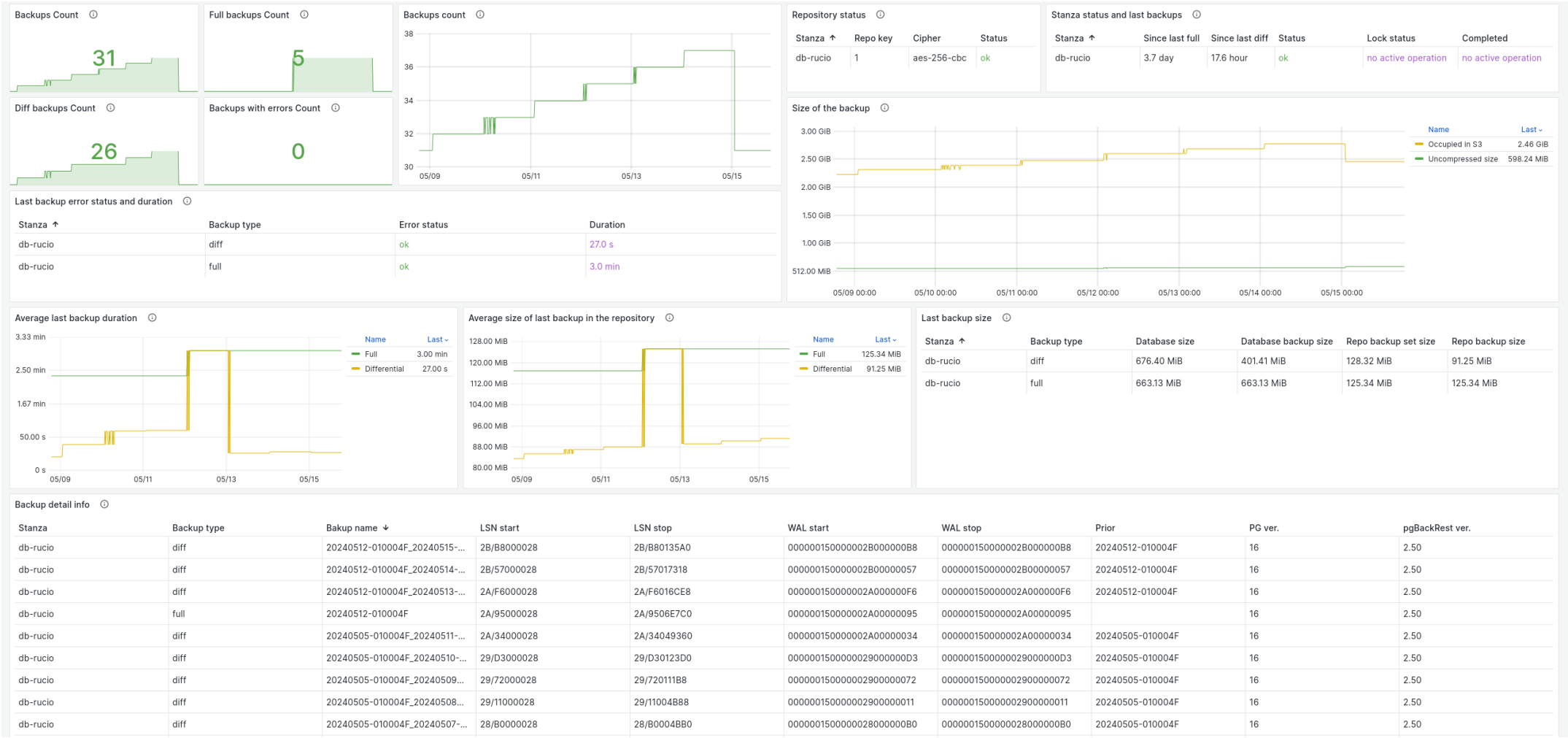
Monitoring: Exporters e server

- **Prometheus exporters**
  - Gli exporter Prometheus sono eseguiti in dei container
  - Node exporter: espone le metriche relative allo stato del nodo (spazio disco, memoria, stato rete, etc.)
  - Postgres exporter: espone le metriche relative allo stato del cluster PostgreSQL
  - pgBackrest exporter: espone le metriche relative allo stato dei backup
- **Agente Zabbix**
  - Viene eseguito un agente Zabbix che interroga il database PostgreSQL ed espone le metriche al server Zabbix
  - La configurazione prevede anche dei trigger per inviare alerts all'amministratore
- **Server Prometheus e Grafana**
  - I server girano in dei container su una macchina specificata nell'inventario Ansible

Monitoring: Dashboard Grafana (PostgreSQL metrics)



Monitoring: Dashboard Grafana (pgBackrest metrics)



Progetto Ansible

Progetto Ansible: Info generali

- **Repository su Baltig**
  - Il progetto può essere clonato dal seguente link
  - La documentazione per l'installazione, configurazione e utilizzo si trova qui
- **Requisiti**
  - Un server per il ruolo del primario e almeno un server per il ruolo di standby
  - Git e Ansible installati sul sistema locale
  - Server di destinazione raggiungibili via ssh passwordless e con privilegi sudo

```
postgres-ansible
├── compose_monitoring
│   ├── compose.yml
│   ├── grafana
│   │   └── dashboards
│   │       └── PostgreSQL_Exporter-Dashboard.json
│   ├── prometheus
│   │   └── prometheus.yml
│   └── traefik
│       └── tls.toml
├── inventory
│   └── pgsql_inventory.yml
├── playbooks
│   ├── setup-cronjobs.yml
│   ├── setup-everything.yml
│   ├── setup-monitoring.yml
│   ├── setup-pgbackrest.yml
│   ├── setup-postgresql.yml
│   ├── setup-post-installation.yml
│   ├── setup-replica.yml
│   └── setup-repmgr.yml
├── README.md
└── templates
    ├── cronjob_backup.j2
    ├── pgbackrest.j2
    ├── pgsql_logrotate.j2
    ├── prevent_split_brain.j2
    ├── prometheus.j2
    └── repmgr_event_action.j2
```

Progetto Ansible: Inventario

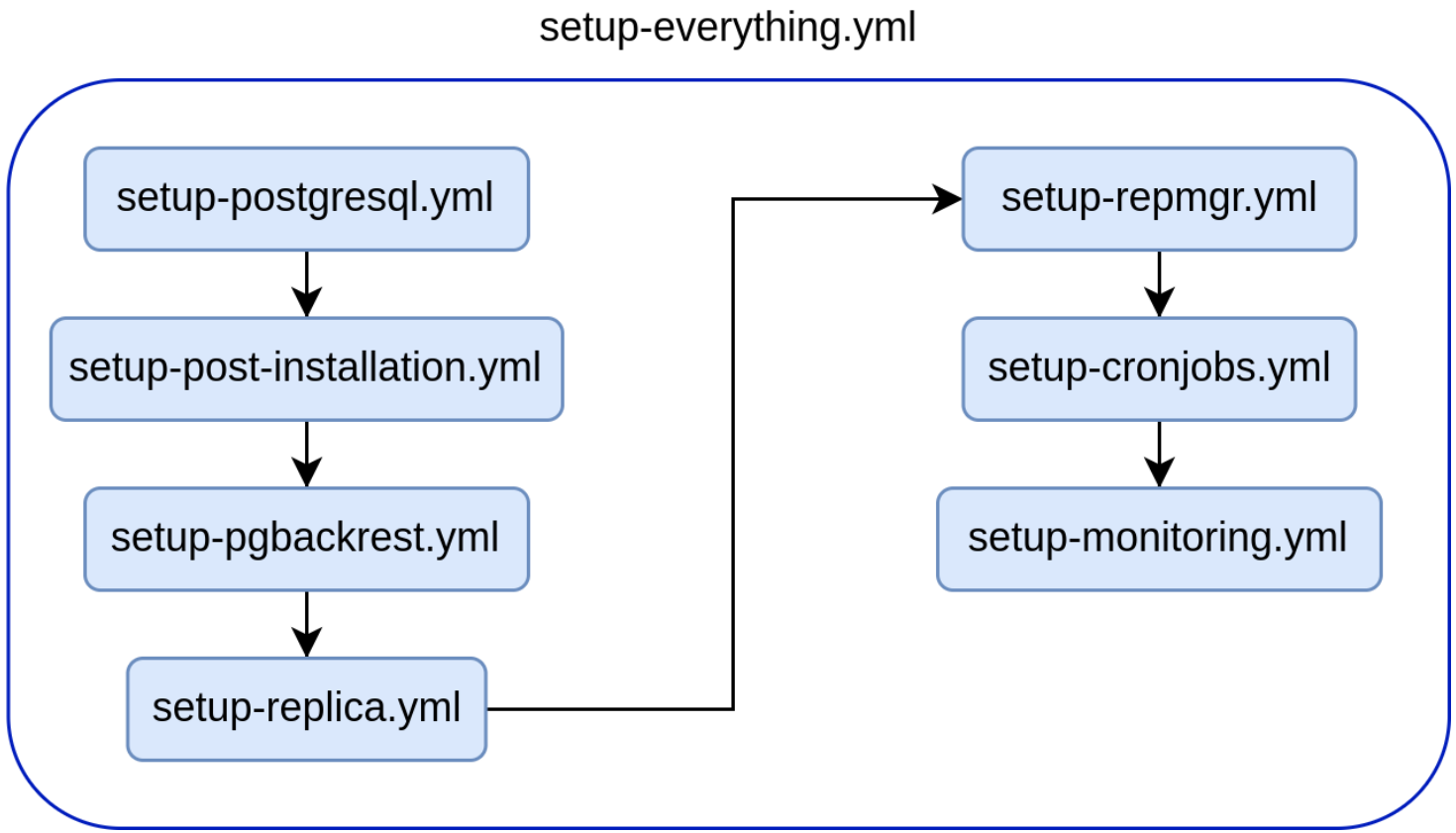
```
postgres_primary:
    hosts:
        pgsql-cnaf:
            app_name: pgsql1
            fqdn: psql-cnaf.cloud.infn.it

postgres_replica:
    hosts:
        pgsql-bari:
            app_name: pgsql2
            fqdn: psql-bari.cloud.infn.it

postgres_monitoring:
    hosts:
        pgsql-bari:
            fqdn: psql-bari.cloud.infn.it
```

```
postgres_servers:
    children:
        postgres_primary:
        postgres_replica:
    vars:
        pgsql_home: /usr/pgsql-16
        pgsql_bin: "{{ pgsql_home }}/bin/"
        pgsql_data_dir: "{{ pgsql_home }}/data"
        pgsql_script_dir: "{{ pgsql_home }}/scripts/"
        pgsql_logdir: /var/log/pgsql
        pgsql_logfile: "{{ pgsql_logdir }}/logfile"
        pgsql_cert_dir: "{{ pgsql_home }}/certificates"
        app_names: "{{ groups['postgres_replica'] |
                       map('extract', hostvars, 'app_name') |
                       list | join(',') }}"
        pg_user_pass: p05tgr35
        replica_user: replicator
        replica_pass: r3pl1c4t0r
        wal_archive_dir: /psql-db-backup/wal_archive
        base_backup_dir: /psql-db-backup/base_backup
        keystore_pass: k3yst0r3p455
        repmgr_pass: f41lm4n4g3r
        zbx_user_pass: Z6X.D3f4ulT
        stanza_name: db-rucio
        s3_key: secretkey
        s3_key_secret: verysecretkey
```

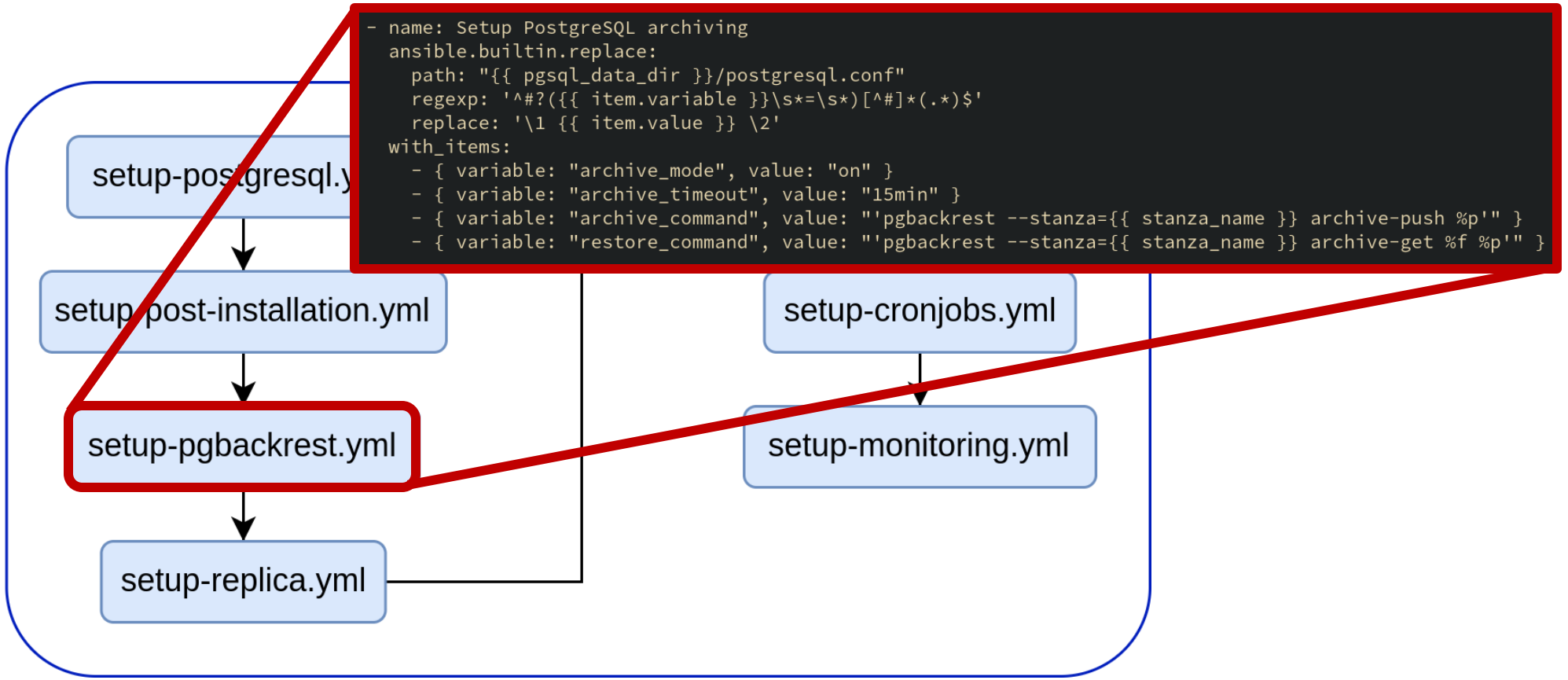
Progetto Ansible: Templates

- **pgBackrest**
  - Di default configura un bucket su INFN Cloud S3 storage
  - Documentazione file di configurazione pgBackrest
- **repmgr**
  - Il template *repmgr_event_action.j2* genera lo script che gestisce le azioni in corrispondenza di eventi significativi
- **Compose monitoring**
  - Il docker compose per i server Prometheus e Grafana va adattato alle proprie esigenze
  - Se non necessari, basta commentare i task nel gruppo "Setup monitoring server" in *playbook/setup-monitoring.yml*

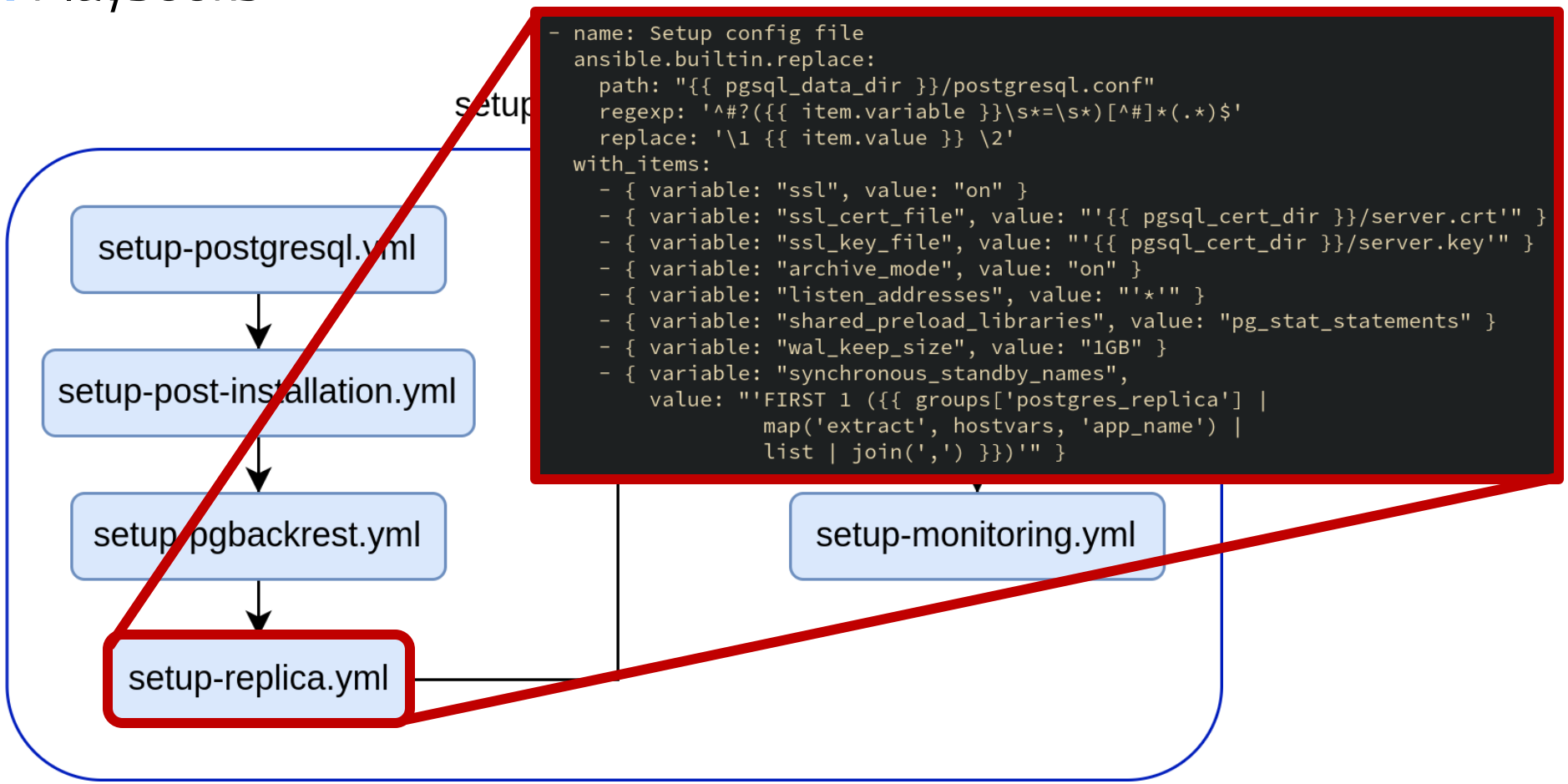
Progetto Ansible: Playbooks



Progetto Ansible: Playbooks



Progetto Ansible: Playbooks



Progetto Ansible: Playbooks


```
- name: Setup repmgr properties file
  ansible.builtin.replace:
    path: /etc/repmgr/16/repmgr.conf
    regexp: '^#?({{ item.variable }}\s*=\s*)[^#]*(.*)$'
    replace: '\1 {{ item.value }} \2'
  with_items:
    - { variable: "node_id", value: "{{ groups['postgres_servers'].index(inventory_hostname) + 1 }}"}
    - { variable: "node_name", value: "{{ hostvars[inventory_hostname].app_name }}"}
    - { variable: "conninfo", value: "'host={{ hostvars[inventory_hostname].fqdn }}
                                      user=repmgr
                                      application_name={{ hostvars[inventory_hostname].app_name }}
                                      sslmode=require
                                      connect_timeout=20'"}
    - { variable: "data_directory", value: "'{{ pgsql_data_dir }}'"}
    - { variable: "replication_user", value: "{{ replica_user }}"}
    - { variable: "location", value: "INFN-Cloud"}
    - { variable: "ssh_options", value: "'-q -o ConnectTimeout=10'"}
    - { variable: "log_file", value: "'/var/log/repmgr/repmgrd.log'"}
    - { variable: "event_notification_command", value: "'{{ pgsql_script_dir }}repmgr_event_action.sh %n %e %s'"}
    - { variable: "child_nodes_connected_min_count", value: "1"}
    - { variable: "child_nodes_disconnect_command", value: "'{{ pgsql_script_dir }}repmgr_event_action.sh 0 isolated_primary 0'"}
    - { variable: "service_start_command", value: "'sudo systemctl start postgresql-16'"}
    - { variable: "service_stop_command", value: "'sudo systemctl stop postgresql-16'"}
    - { variable: "service_restart_command", value: "'sudo systemctl restart postgresql-16'"}
    - { variable: "service_reload_command", value: "'sudo systemctl reload postgresql-16'"}
    - { variable: "reconnect_interval", value: "5"}
    - { variable: "failover", value: "automatic"}
    - { variable: "promote_command", value: "'/bin/repmgr standby promote --log-to-file'"}
    - { variable: "follow_command", value: "'/bin/repmgr standby follow --log-to-file --upstream-node-id=%n'"}
    - { variable: "primary_visibility_consensus", value: "true"}
    - { variable: "repmgrd_service_start_command", value: "'sudo systemctl start repmgr-16'"}
    - { variable: "repmgrd_service_stop_command", value: "'sudo systemctl stop repmgr-16'"}
    - { variable: "monitoring_history", value: "yes"}
```

Gestione cluster

Gestione cluster: pgBackrest

- **Stato dei backup**

```
[postgres@psql ~]$ pgbackrest info
stanza: db-rucio
    status: ok
    cipher: aes-256-cbc

    db (current)
        wal archive min/max (16): 000000150000001F000000FB/000000150000002C0000009B

        full backup: 20240414-010003F
            timestamp start/stop: 2024-04-14 01:00:03+02 / 2024-04-14 01:02:20+02
            wal start/stop: 000000150000001F000000FB / 000000150000001F000000FB
            database size: 528.1MB, database backup size: 528.1MB
            repo1: backup set size: 95.9MB, backup size: 95.9MB

        diff backup: 20240414-010003F_20240415-010003D
            timestamp start/stop: 2024-04-15 01:00:03+02 / 2024-04-15 01:00:20+02
            wal start/stop: 0000001500000020000005C / 0000001500000020000005C
            database size: 532.5MB, database backup size: 264.4MB
            repo1: backup set size: 96.9MB, backup size: 60.5MB
            backup reference list: 20240414-010003F

        diff backup: 20240414-010003F_20240416-010004D
            timestamp start/stop: 2024-04-16 01:00:04+02 / 2024-04-16 01:00:20+02
            wal start/stop: 0000001500000020000000BD / 0000001500000020000000BD
            database size: 536.9MB, database backup size: 268.8MB
            repo1: backup set size: 97.9MB, backup size: 61.5MB
            backup reference list: 20240414-010003F

        diff backup: 20240414-010003F_20240417-010004D
            timestamp start/stop: 2024-04-17 01:00:04+02 / 2024-04-17 01:00:21+02
```

- **Backup manuale**
  - pgbackrest --type=full --stanza=stanza-name backup
  - Type può essere: full, diff, incr
- **Disaster recovery**
  1. systemctl stop postgresql-16
  2. rm -rf /usr/pgsql-16/data/*
  3. pgbackrest --stanza=stanza-name restore
  4. systemctl start postgresql-16

Gestione cluster: repmgr

- **Stato del cluster**

```
[postgres@psql ~]$ repmgr cluster show
 ID | Name   | Role    | Status    | Upstream | Location   | Priority | Timeline | Connection string
----+--------+---------+-----------+----------+------------+----------+----------+-----------------------------------------------------------------------------------------------------------
 1  | pgsql1 | primary | * running |          | INFN-Cloud | 100      | 21       | host=psql-cnaf.cloud.infn.it user=repmgr application_name=pgsql1 sslmode=require connect_timeout=20
 2  | pgsql2 | standby |   running | pgsql1   | INFN-Cloud | 100      | 21       | host=psql-bari.cloud.infn.it user=repmgr application_name=pgsql2 sslmode=require connect_timeout=20
```

- **Switchover**
  - repmgr standby switchover
  - Eseguire con l'opzione --dry-run per verificare se i requisiti per lo switchover sono soddisfatti
- **Creazione nuovo standby**
  - repmgr -h hostname-primario -U repmgr -d repmgr standby clone

Grazie

Nadir Marcelli
nadir.marcelli@roma1.infn.it