Rehydration Process of YugabyteDB Nodes Running on GCP

Purpose

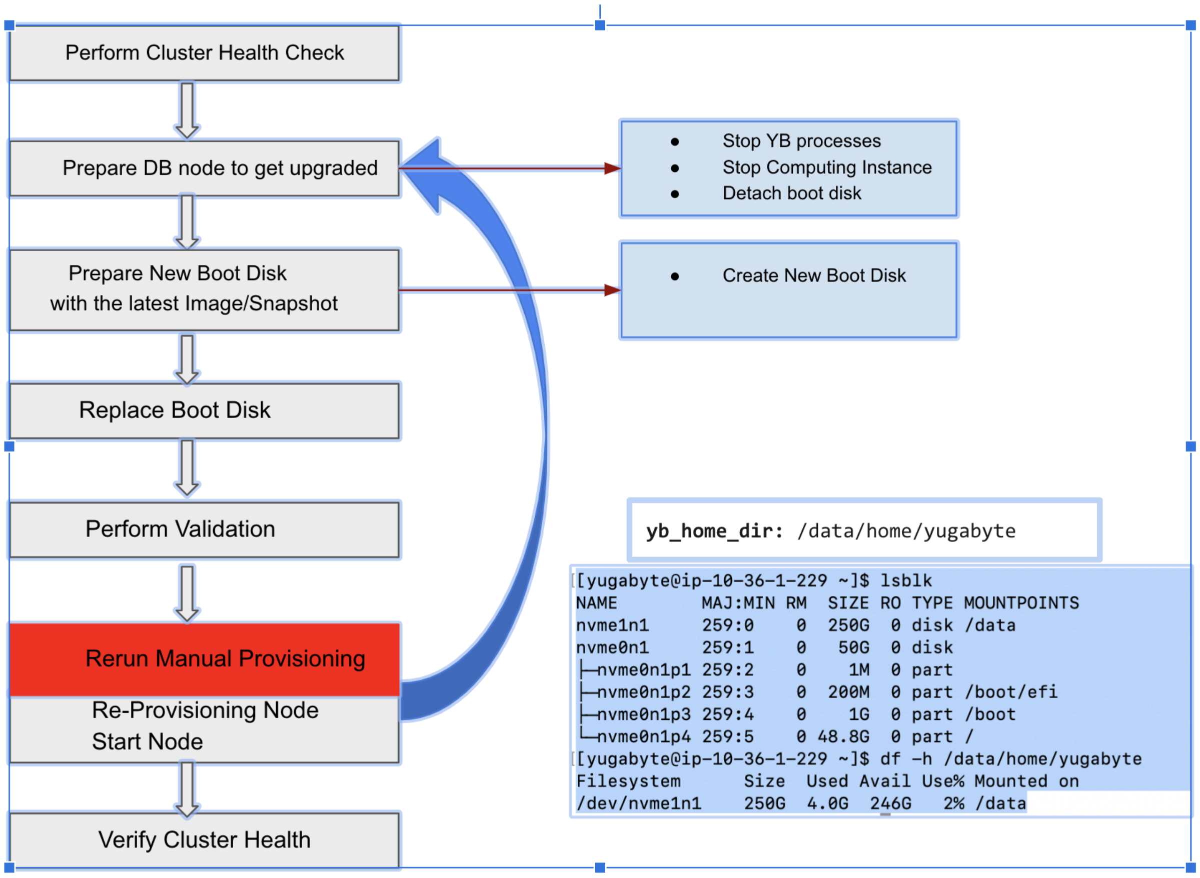
This runbook defines a fully automated, step-by-step workflow for rehydrating YugabyteDB universe nodes running on GCP by deploying the latest image, while ensuring the preservation of private IP addresses and associated data volumes. In accordance with YugabyteDB operational best practices, each node's rehydration is designed to complete within approximately 15 minutes, minimizing disruption and maintaining cluster availability. Key requirements include:

- Preserve current private IP
- Preserve /Data mount point and data disks
- Complete the entire process of each node under 15 mins

Current Deployment Architecture of the customer on AWS

- Multi-region deployment: us-east-1 (2 azs), us-east-2 (1 az), us-west-2 (2 azs)
- Instance type: r7i.4xlarge
- Node count: 50 nodes (13 on us-east-1a, 13 on us-east-1b, 8 on us-east-2a, 8 on us-west-2a, 8 on us-west-2b); on us-east-1a and us-east1b, 3x2 TB gp3 EBS volumes on each node using one RAID-0 array and 6TB YB filesystem will be mounted on this array. on us-east-2 and us-west, 3x3.2 TB gp3 EBS volumes on each node using one RAID-0 array and 9.6TB YB filesystem will be mounted on this array.
- Total storage will be 78TB for each zone on US-east-1; 76.8TB for each zone on US-east-2 and Us-west-2
- Universe is configured using A Record
    - A record for each EC2 instance
    - A record per region (App team uses Regional A record in smart driver)

YugabyteDB Node Boot Disk Replacement Workflow



```
yb_home_dir: /data/home/yugabyte
```

```
[yugabyte@ip-10-36-1-229 ~]$ lsblk
NAME        MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
nvme1n1     259:0    0  250G  0 disk /data
nvme0n1     259:1    0   50G  0 disk
├─nvme0n1p1 259:2    0    1M  0 part
├─nvme0n1p2 259:3    0  200M  0 part /boot/efi
├─nvme0n1p3 259:4    0    1G  0 part /boot
└─nvme0n1p4 259:5    0 48.8G  0 part /
[yugabyte@ip-10-36-1-229 ~]$ df -h /data/home/yugabyte
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme1n1    250G  4.0G  246G   2% /data
```

Prerequisites:

- Access to GCP project
  - IAM permissions to manage Compute Engine disks and instances
  - Existing GCP instance with: Data disk (e.g., XFS-formatted, mounted at /data)
  - Replacement base image available (e.g., rhel-9-v20240415)
- Access to Yugayte Anywhere Instance and Universes provisioned with on-Prem provider
- /data mount point exists
  - Yugabyte user: /data/home/yugabyte
- Required tools installed:
  - Google Cloud SDK
  - Yugabyte Anywhere CLI
  - Python 3.8+

Execution Steps:

The current flow is designed to process one node at a time. However, it can be efficiently extended to handle multiple instances in parallel—such as those within the same availability zone or fault domain—thereby improving the overall rehydration performance across the universe.

1. Create a Configuration File (yba_config.yaml)

```java
project_id: "xxxx"
instance_type: "n2-standard-4"
zone: "us-central1-a"
image_family: "rhel-8"
image_project: "rhel-cloud"
disk_size: 150
extra_disk_size: 250
new_disk_name: "new-boot-disk"
new_image: "rhel-9-v20240415"
disk_type: "pd-balanced"
sh_user: "sh_user"  # or whatever user you use
ssh_wait_time: 30     # seconds to wait for SSH to be available
yba:
  yba_host: "https://xx.xxx.xx.xx/"
  customer_id: "customer_id"
  yba_api_token: "YBA_API_TOKEN"
  universe_name: "UNIVERSE_NAME"
  node_list:
    - "10.128.15.210": "NODE_NAME"
  yba_cli_path:
"/opt/yugabyte/software/active/yb-platform/yugaware/yba-cli/yba_cli-2024.2.2.2-b2-linux-amd64/yba"
```

2. Stop the Node via YBA CLI

```java
$YBA_CLI_PATH universe node stop \
  -H $YBA_HOST -a $YBA_API_TOKEN \
  --name $UNIVERSE_NAME --node-name $NODE_NAME
```

3. Stop the Computing Instance

```
Java
gcloud compute instances stop "$INSTANCE_NAME" --zone "$ZONE" --project
"$project_id"
```

4. Create the New Boot Disk

```
Java
gcloud compute disks create "$NEW_DISK_NAME" \
  --image "$NEW_IMAGE" \
  --type "$DISK_TYPE" \
  --size "$DISK_SIZE" \
  --zone "$ZONE" \
  --proejct "$PROJECT"
```

5. Detach the Old Boot Disk from the computing instance

```
Java
gcloud compute instances detach-disk "$INSTANCE_NAME" --disk="OLD_BOOT_DISK"
--zone "$ZONE" --proejct "$PROJECT"
```

6. Attach the New Boot Disk to the computing instance

```
Java
gcloud compute instances attach-disk "$INSTANCE_NAME" \
  --disk "$NEW_DISK_NAME" \
  --zone "$ZONE" \
  --boot \
  --device-name=boot-disk \
  --mode=rw \
  --proejct "$PROJECT"
```

7. **Restart the Computing Instance**

```java
gcloud compute instances start "$INSTANCE_NAME" --zone "$ZONE" --proejct
"$PROJECT_ID"
```

8. **Verify Boot & Attachments of the Computing Instance**

```java
gcloud compute instances describe "$INSTANCE_NAME" --zone "$ZONE" \
  --format='get(networkInterfaces[0].networkIP,disks.deviceName)' \
  --proejct "$PROJECT"
```

9. **Start the Computing Instance**

```java
gcloud compute instances stop "$INSTANCE_NAME" --zone "$ZONE" --project
"$PROJECT_ID"
```

10. **Provision Node - Manual provisioning Steps required to provision a node, here are some sample steps**
    - **Mount Data Disk**
        - Identify data disk UUID
        - Create /data directory
        - Mount data disk
        - Update /etc/fstable for persistence

- **Provision Node Agent**
    - Clean up existing yugabyte user if exists
    - Create new yugabyte user with correct home directory
    - Set proper permissions
    - Run node-agent-provision.sh

11. Reprovision & Start Node with YBA CLI

```
Java
$YBA_CLI_PATH universe node reprovision \
  -H $YBA_HOST -a $YBA_API_TOKEN \
  --name $UNIVERSE_NAME --node-name $NODE_NAME

$YBA_CLI_PATH universe node start \
  -H $YBA_HOST -a $YBA_API_TOKEN \
  --name $UNIVERSE_NAME --node-name $NODE_NAME
```

Expected Outcome:

- The computing instance is running with the new boot disk
- Private IP and external volumes are unchanged
- It joined the universe successfully
- Old boot disk (if detached) can now be deleted or archived

Validation:

- Check node status in YBA UI and verify node is healthy
- Verify data disk is mounted
- Ensure `/data` is populated: `ls -l /data`
- Confirm `/etc/fstab` entry is present
- Check `node-agent-provision.sh` logs for success
- Check newly created disk info:

```
Java
gcloud compute instances describe INSTANCE_NAME \
  --zone=ZONE \
  --format="value(disks[0].source)"
```

```
gcloud compute disks describe DISK_NAME --zone=ZONE
```

Troubleshooting Matrix

Issue	Possible Causes	Solution
Node fails to start in YBA	- Node agent issues- SSH access problems- Permission issues	- Check node agent logs- Verify SSH access- Check user permissions
Data disk not mounting	- Incorrect UUID- fstab issues- Permission problems	- Verify disk UUID- Check /etc/fstab- Verify mount permissions
SSH access issues	- Instance status- Firewall rules- SSH key configuration	- Check instance status- Verify firewall rules- Check SSH key configuration

Script:

```java
yba_config.yaml
```

```
instance_count: 4
instance_type: e2-standard-4
zone: us-central1-a
image_family: rhel-8
image_project: rhel-cloud
disk_size: 250
new_disk_name: new-boot-disk
new_image: projects/rhel-cloud/global/images/rhel-8-v20240423
disk_type: pd-balanced


//rehydration_full_lifecycle.py
#!/usr/bin/env python3
import subprocess
import logging
import time
import json
import yaml
from datetime import datetime

# Load configuration
with open("yba_config.yaml", "r") as f:
    config = yaml.safe_load(f)

node_map = config["yba"]["node_list"][0]
INSTANCE_IP = list(node_map.keys())[0]
INSTANCE_NAME = list(node_map.values())[0]
NODE_NAME = INSTANCE_NAME

ZONE = config["zone"]
PROJECT = config["project_id"]
NEW_IMAGE = config["new_image"]
IMAGE_PROJECT = config["image_project"]
DISK_TYPE = config["disk_type"]
DISK_SIZE = str(config["disk_size"]) + "GB"
SSH_USER = config["sh_user"]
WAIT_TIME = config["ssh_wait_time"]
YBA_CLI = config["yba"]["yba_cli_path"]
YBA_HOST = config["yba"]["yba_host"]
YBA_TOKEN = config["yba"]["yba_api_token"]
YBA_UNIVERSE = config["yba"]["universe_name"]

log_file = f"rehydration_{datetime.now().strftime('%Y%m%d_%H%M%S')}.log"
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[logging.StreamHandler(), logging.FileHandler(log_file)]
)

def run(cmd, capture_output=False):
    logging.info(f"Running: {' '.join(cmd)}")
    result = subprocess.run(cmd, text=True, capture_output=capture_output,
check=True)
    return result.stdout.strip() if capture_output else None

def resolve_instance_name_and_zone(ip):
    result = run([
        "gcloud", "compute", "instances", "list",
        "--filter", f"networkInterfaces[0].networkIP={ip}",
        "--format", "json",
        "--project", PROJECT
    ], capture_output=True)
    data = json.loads(result)
    if not data:
        raise RuntimeError(f"No instance found for IP {ip}")
    return data[0]["name"], data[0]["zone"].split("/")[-1]

def stop_yugabyte_processes():
    try:
        run([
            YBA_CLI, "universe", "node", "stop",
            "-H", YBA_HOST, "-a", YBA_TOKEN,
            "--name", YBA_UNIVERSE,
            "--node-name", NODE_NAME
        ])
        logging.info(f"✅ Stopped node {NODE_NAME} using YBA CLI")
    except subprocess.CalledProcessError as e:
        logging.warning(f"⚠️ Failed to stop node using YBA CLI: {e}")

def reprovision_and_start_node():
    run([
        YBA_CLI, "universe", "node", "reprovision",
        "-H", YBA_HOST, "-a", YBA_TOKEN,
        "--name", YBA_UNIVERSE,
        "--node-name", NODE_NAME
    ])
    time.sleep(10)
    run([
        YBA_CLI, "universe", "node", "start",
        "-H", YBA_HOST, "-a", YBA_TOKEN,
        "--name", YBA_UNIVERSE,
        "--node-name", NODE_NAME
    ])

def get_boot_disk():
    result = run([
        "gcloud", "compute", "instances", "describe", INSTANCE_NAME,
        "--zone", ZONE, "--project", PROJECT, "--format=json"
    ], capture_output=True)
    info = json.loads(result)
    for disk in info["disks"]:
        if disk.get("boot"):
            return disk["source"].split("/")[-1]
    raise RuntimeError("Boot disk not found")

def wait_for_status(expected, timeout=300):
    for _ in range(timeout // 10):
        result = run([
            "gcloud", "compute", "instances", "describe", INSTANCE_NAME,
            "--zone", ZONE, "--project", PROJECT, "--format=json"
        ], capture_output=True)
        if json.loads(result).get("status") == expected:
            return
        time.sleep(10)
    raise TimeoutError(f"Timeout waiting for instance status: {expected}")

def stop_instance():
    run(["gcloud", "compute", "instances", "stop", INSTANCE_NAME, "--zone",
ZONE, "--project", PROJECT])
    wait_for_status("TERMINATED")

def start_instance():
    run(["gcloud", "compute", "instances", "start", INSTANCE_NAME, "--zone",
ZONE, "--project", PROJECT])
    wait_for_status("RUNNING")

def replace_boot_disk():
    old_disk = get_boot_disk()
    new_disk = f"{INSTANCE_NAME}-boot-{int(time.time())}"
    run([
        "gcloud", "compute", "disks", "create", new_disk,
        "--image", NEW_IMAGE, "--image-project", IMAGE_PROJECT,
        "--size", DISK_SIZE, "--type", DISK_TYPE,
        "--zone", ZONE, "--project", PROJECT
    ])
    run([
        "gcloud", "compute", "instances", "detach-disk", INSTANCE_NAME,
        "--disk", old_disk, "--zone", ZONE, "--project", PROJECT
    ])
    run([
        "gcloud", "compute", "instances", "attach-disk", INSTANCE_NAME,
        "--disk", new_disk, "--zone", ZONE, "--project", PROJECT,
        "--boot"
    ])

def get_data_disk_uuid():
    cmd = [
        "gcloud", "compute", "ssh", f"{SSH_USER}@{INSTANCE_NAME}",
        "--zone", ZONE, "--project", PROJECT,
        "--internal-ip", "--command",
        "lsblk -o NAME,FSTYPE,UUID | grep xfs | grep -v sda"
    ]
    output = run(cmd, capture_output=True)
    device, _, uuid = output.strip().split()
    return f"/dev/{device}", uuid

def mount_data_disk(uuid):
    cmd = " && ".join([
        "sudo mkdir -p /data",
        f"sudo mount UUID={uuid} /data",
        "sudo chmod 777 /data",
        f"grep -q UUID={uuid} /etc/fstab || echo 'UUID={uuid} /data xfs
defaults,nofail 0 2' | sudo tee -a /etc/fstab"
    ])
    run([
        "gcloud", "compute", "ssh", f"{SSH_USER}@{INSTANCE_NAME}",
        "--zone", ZONE, "--project", PROJECT,
        "--internal-ip", "--command", cmd
    ])


def provision_node_agent():
    cmd = " && ".join([
        "echo '🛠️ Reprovisioning yugabyte user and running
node-agent-provision.sh...'",
        "sudo pkill -u yugabyte || true",
        "id yugabyte && sudo userdel -r yugabyte || true",
        "getent group yugabyte && (getent passwd | awk -F: '$4 == \"$(getent
group yugabyte | cut -d: -f3)\"' | grep -q . || sudo groupdel yugabyte) ||
true",
        "sudo useradd -m -d /data/home/yugabyte -s /bin/bash yugabyte",
        "sudo mkdir -p /data/home/yugabyte",
        "sudo chown -R yugabyte:yugabyte /data/home/yugabyte",
        "sudo chmod 755 /data/home/yugabyte",
        "cd /data/2024.2.2.2-b2/scripts",
        "sudo ./node-agent-provision.sh"
    ])
    run([
        "gcloud", "compute", "ssh", f"{SSH_USER}@{INSTANCE_NAME}",
        "--zone", ZONE, "--project", PROJECT,
        "--internal-ip", "--command", cmd
    ])
def main():
    global INSTANCE_NAME, ZONE
    INSTANCE_NAME, ZONE = resolve_instance_name_and_zone(INSTANCE_IP)
    stop_yugabyte_processes()
    stop_instance()
    replace_boot_disk()
    start_instance()
    time.sleep(WAIT_TIME)
    _, uuid = get_data_disk_uuid()
    mount_data_disk(uuid)
    provision_node_agent()
    reprovision_and_start_node()
    logging.info("✅ Rehydration complete with home bind-mount fix.")

if __name__ == "__main__":
    main()


//execution
//chmod +x rehydration_full_lifecycle.py | ./rehydration_full_lifecycle.py
```

Testing Environment

- https://35.184.240.7/

- Access ysqlsh:
    1. ssh to one of nodes:
       ```
       gcloud compute ssh --zone "us-central1-a" "wwang-tu-n1" --project "yuga-rjw"
       ```
    2. `Sudo su - yugabyte`
    3. `./tserver/bin/ysqlsh -h 10.128.15.206`