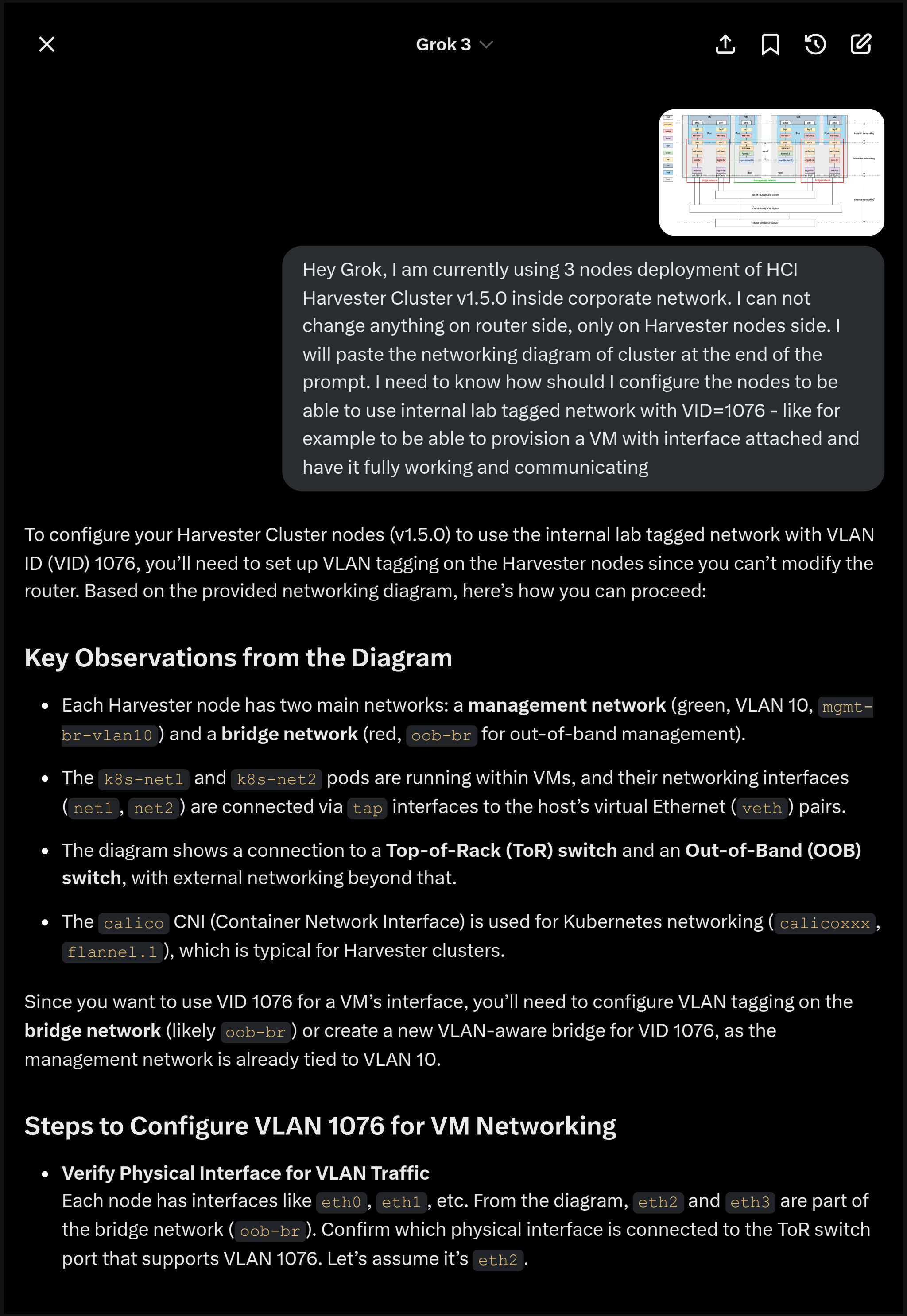


Hey Grok, I am currently using 3 nodes deployment of HCI Harvester Cluster v1.5.0 inside corporate network. I can not change anything on router side, only on Harvester nodes side. I will paste the networking diagram of cluster at the end of the prompt. I need to know how should I configure the nodes to be able to use internal lab tagged network with VID=1076 - like for example to be able to provision a VM with interface attached and have it fully working and communicating

To configure your Harvester Cluster nodes (v1.5.0) to use the internal lab tagged network with VLAN ID (VID) 1076, you'll need to set up VLAN tagging on the Harvester nodes since you can't modify the router. Based on the provided networking diagram, here's how you can proceed:

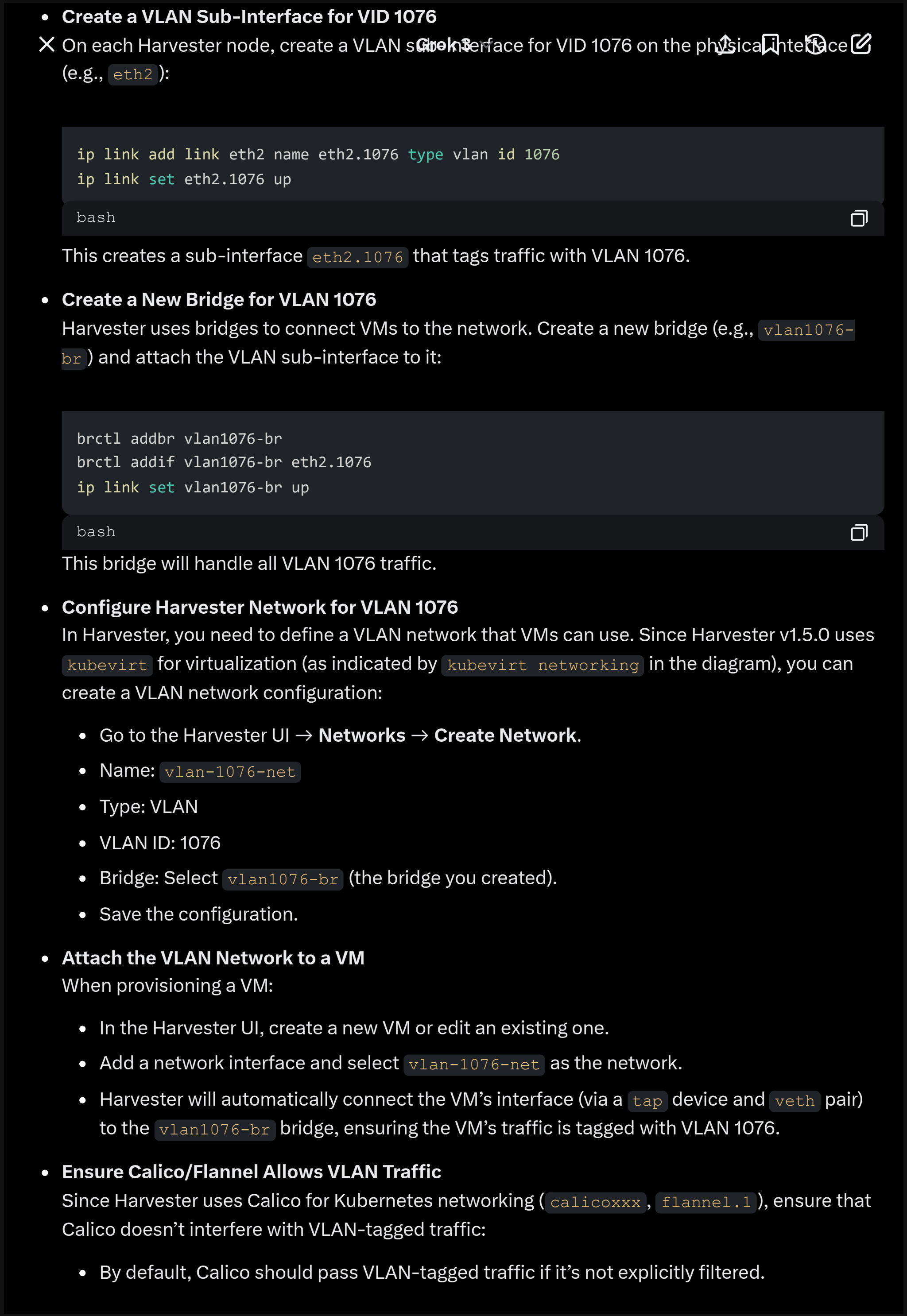
Key Observations from the Diagram

- Each Harvester node has two main networks: a **management network** (green, VLAN 10, `mgmt-br-vlan10`) and a **bridge network** (red, `oob-br` for out-of-band management).
- The `k8s-net1` and `k8s-net2` pods are running within VMs, and their networking interfaces (`net1`, `net2`) are connected via `tap` interfaces to the host's virtual Ethernet (`veth`) pairs.
- The diagram shows a connection to a **Top-of-Rack (ToR) switch** and an **Out-of-Band (OOB) switch**, with external networking beyond that.
- The `calico` CNI (Container Network Interface) is used for Kubernetes networking (`calicoxxx`, `flannel.1`), which is typical for Harvester clusters.

Since you want to use VID 1076 for a VM's interface, you'll need to configure VLAN tagging on the **bridge network** (likely `oob-br`) or create a new VLAN-aware bridge for VID 1076, as the management network is already tied to VLAN 10.

Steps to Configure VLAN 1076 for VM Networking

- **Verify Physical Interface for VLAN Traffic**
  Each node has interfaces like `eth0`, `eth1`, etc. From the diagram, `eth2` and `eth3` are part of the bridge network (`oob-br`). Confirm which physical interface is connected to the ToR switch port that supports VLAN 1076. Let's assume it's `eth2`.



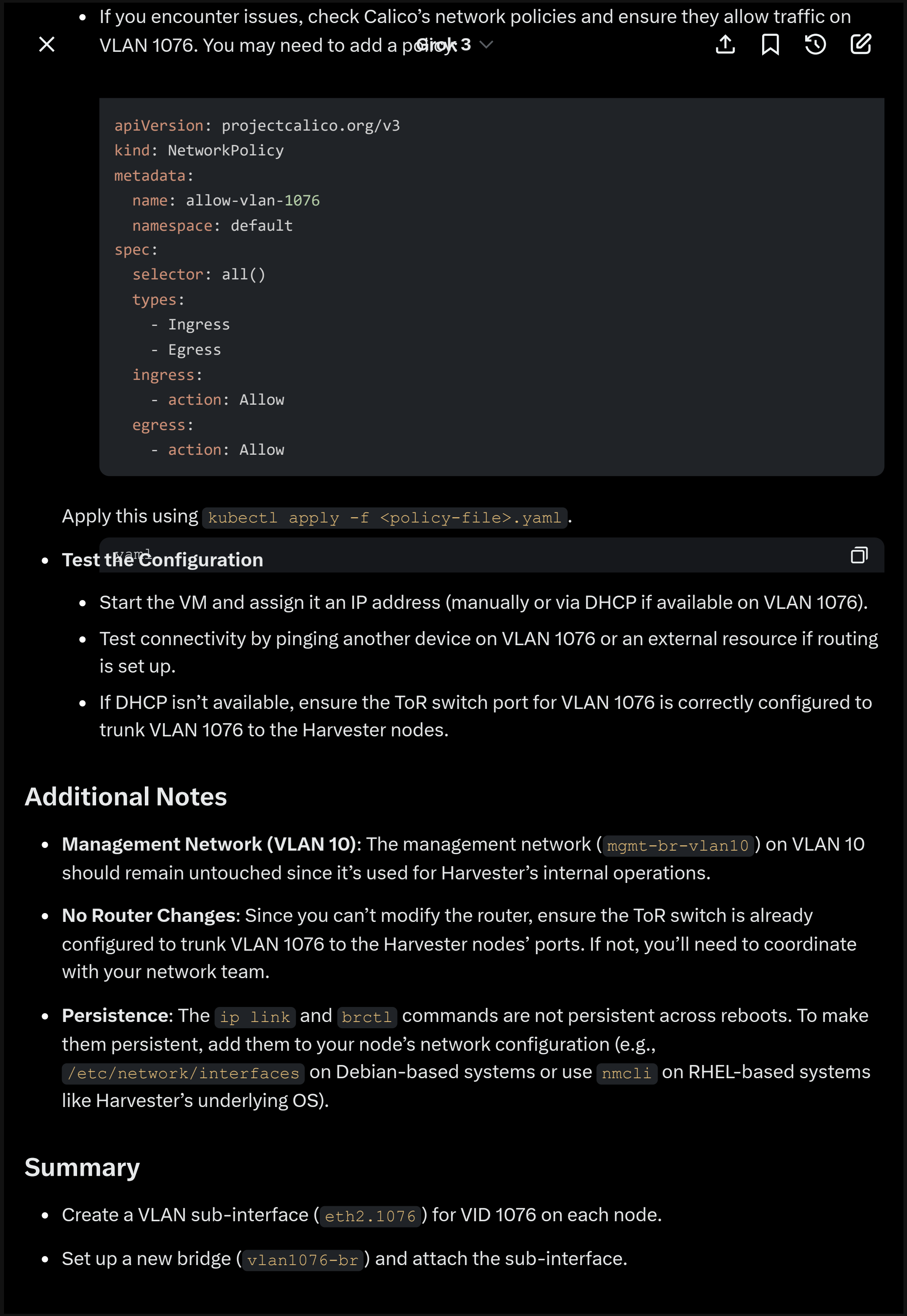
- **Create a VLAN Sub-Interface for VID 1076**
  On each Harvester node, create a VLAN sub-interface for VID 1076 on the physical interface (e.g., `eth2`):

```bash
ip link add link eth2 name eth2.1076 type vlan id 1076
ip link set eth2.1076 up
```

  This creates a sub-interface `eth2.1076` that tags traffic with VLAN 1076.
- **Create a New Bridge for VLAN 1076**
  Harvester uses bridges to connect VMs to the network. Create a new bridge (e.g., `vlan1076-br`) and attach the VLAN sub-interface to it:

```bash
brctl addbr vlan1076-br
brctl addif vlan1076-br eth2.1076
ip link set vlan1076-br up
```

  This bridge will handle all VLAN 1076 traffic.
- **Configure Harvester Network for VLAN 1076**
  In Harvester, you need to define a VLAN network that VMs can use. Since Harvester v1.5.0 uses `kubevirt` for virtualization (as indicated by `kubevirt networking` in the diagram), you can create a VLAN network configuration:
  - Go to the Harvester UI → **Networks** → **Create Network**.
  - Name: `vlan-1076-net`
  - Type: VLAN
  - VLAN ID: 1076
  - Bridge: Select `vlan1076-br` (the bridge you created).
  - Save the configuration.
- **Attach the VLAN Network to a VM**
  When provisioning a VM:
  - In the Harvester UI, create a new VM or edit an existing one.
  - Add a network interface and select `vlan-1076-net` as the network.
  - Harvester will automatically connect the VM's interface (via a `tap` device and `veth` pair) to the `vlan1076-br` bridge, ensuring the VM's traffic is tagged with VLAN 1076.
- **Ensure Calico/Flannel Allows VLAN Traffic**
  Since Harvester uses Calico for Kubernetes networking (`calicoxxx`, `flannel.1`), ensure that Calico doesn't interfere with VLAN-tagged traffic:
  - By default, Calico should pass VLAN-tagged traffic if it's not explicitly filtered.



- If you encounter issues, check Calico's network policies and ensure they allow traffic on VLAN 1076. You may need to add a p

```yaml
apiVersion: projectcalico.org/v3
kind: NetworkPolicy
metadata:
  name: allow-vlan-1076
  namespace: default
spec:
  selector: all()
  types:
    - Ingress
    - Egress
  ingress:
    - action: Allow
  egress:
    - action: Allow
```

Apply this using `kubectl apply -f <policy-file>.yaml`.

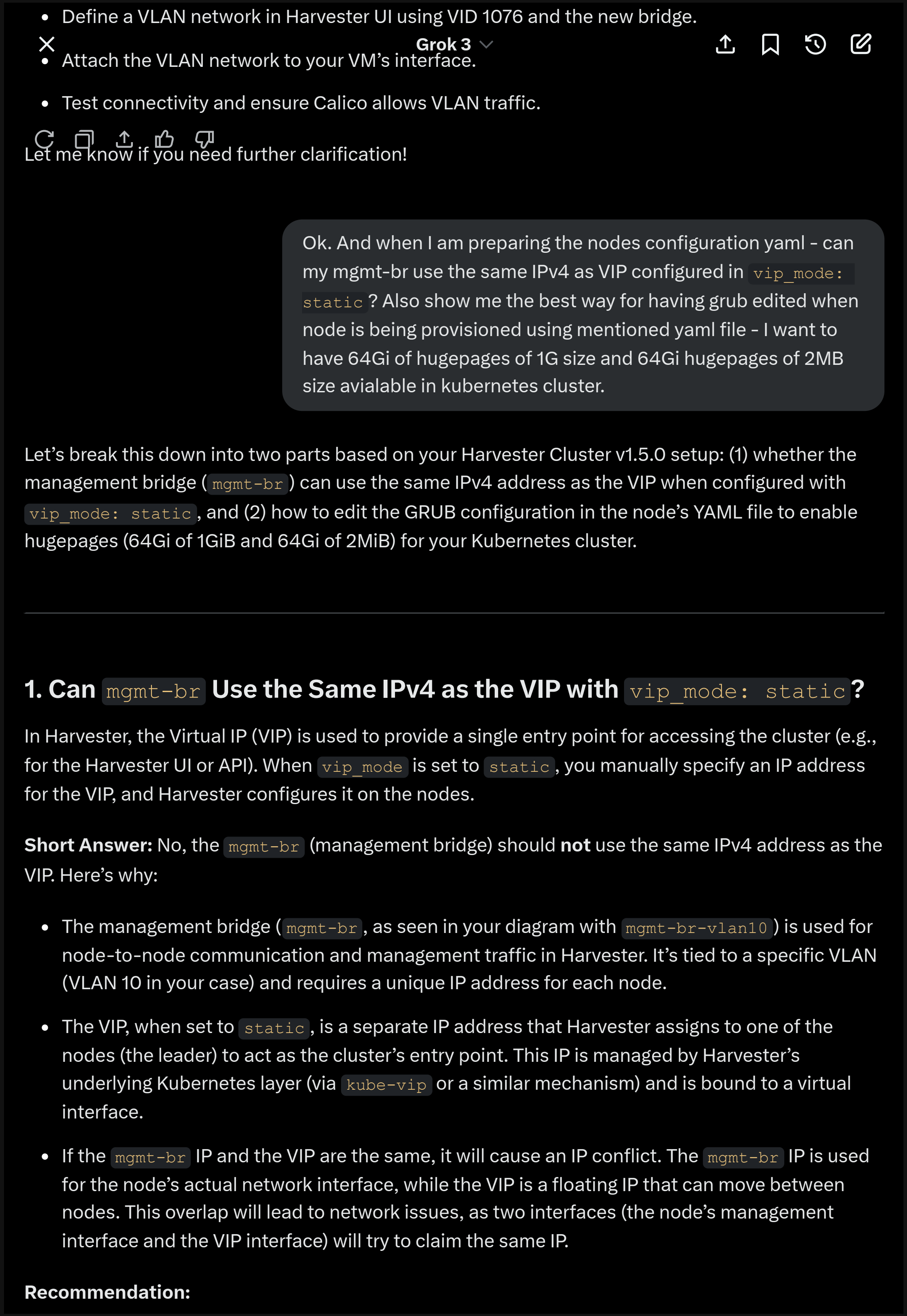
- **Test the Configuration**
  - Start the VM and assign it an IP address (manually or via DHCP if available on VLAN 1076).
  - Test connectivity by pinging another device on VLAN 1076 or an external resource if routing is set up.
  - If DHCP isn't available, ensure the ToR switch port for VLAN 1076 is correctly configured to trunk VLAN 1076 to the Harvester nodes.

Additional Notes

- **Management Network (VLAN 10)**: The management network (`mgmt-br-vlan10`) on VLAN 10 should remain untouched since it's used for Harvester's internal operations.
- **No Router Changes**: Since you can't modify the router, ensure the ToR switch is already configured to trunk VLAN 1076 to the Harvester nodes' ports. If not, you'll need to coordinate with your network team.
- **Persistence**: The `ip link` and `brctl` commands are not persistent across reboots. To make them persistent, add them to your node's network configuration (e.g., `/etc/network/interfaces` on Debian-based systems or use `nmcli` on RHEL-based systems like Harvester's underlying OS).

Summary

- Create a VLAN sub-interface (`eth2.1076`) for VID 1076 on each node.
- Set up a new bridge (`vlan1076-br`) and attach the sub-interface.



- Define a VLAN network in Harvester UI using VID 1076 and the new bridge.
- Attach the VLAN network to your VM's interface.
- Test connectivity and ensure Calico allows VLAN traffic.

Let me know if you need further clarification!

Ok. And when I am preparing the nodes configuration yaml - can my mgmt-br use the same IPv4 as VIP configured in `vip_mode: static`? Also show me the best way for having grub edited when node is being provisioned using mentioned yaml file - I want to have 64Gi of hugepages of 1G size and 64Gi hugepages of 2MB size avialable in kubernetes cluster.

Let's break this down into two parts based on your Harvester Cluster v1.5.0 setup: (1) whether the management bridge (`mgmt-br`) can use the same IPv4 address as the VIP when configured with `vip_mode: static`, and (2) how to edit the GRUB configuration in the node's YAML file to enable hugepages (64Gi of 1GiB and 64Gi of 2MiB) for your Kubernetes cluster.

---

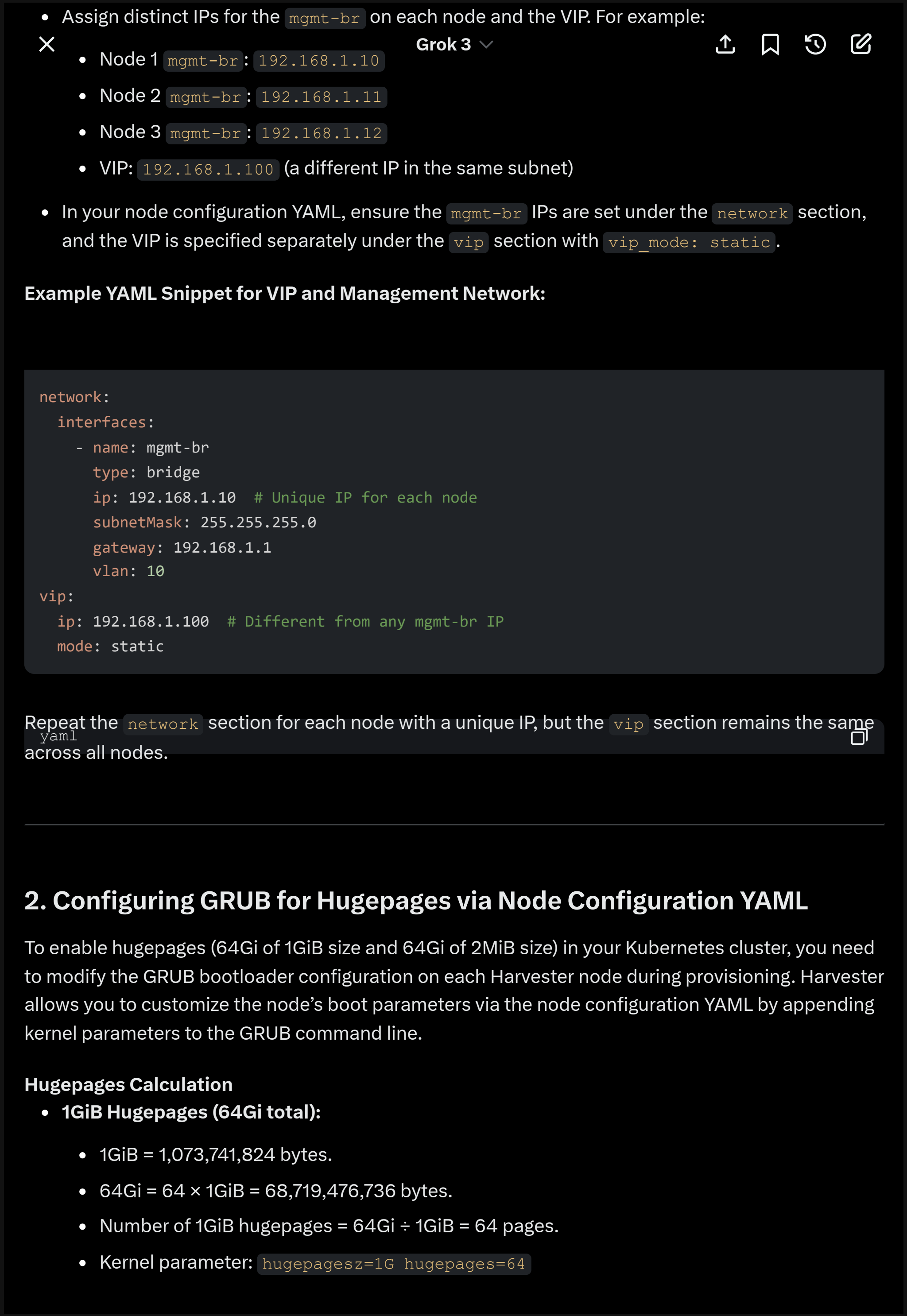
1. Can `mgmt-br` Use the Same IPv4 as the VIP with `vip_mode: static`?

In Harvester, the Virtual IP (VIP) is used to provide a single entry point for accessing the cluster (e.g., for the Harvester UI or API). When `vip_mode` is set to `static`, you manually specify an IP address for the VIP, and Harvester configures it on the nodes.

Short Answer: No, the `mgmt-br` (management bridge) should **not** use the same IPv4 address as the VIP. Here's why:

- The management bridge (`mgmt-br`, as seen in your diagram with `mgmt-br-vlan10`) is used for node-to-node communication and management traffic in Harvester. It's tied to a specific VLAN (VLAN 10 in your case) and requires a unique IP address for each node.
- The VIP, when set to `static`, is a separate IP address that Harvester assigns to one of the nodes (the leader) to act as the cluster's entry point. This IP is managed by Harvester's underlying Kubernetes layer (via `kube-vip` or a similar mechanism) and is bound to a virtual interface.
- If the `mgmt-br` IP and the VIP are the same, it will cause an IP conflict. The `mgmt-br` IP is used for the node's actual network interface, while the VIP is a floating IP that can move between nodes. This overlap will lead to network issues, as two interfaces (the node's management interface and the VIP interface) will try to claim the same IP.

Recommendation:



- Assign distinct IPs for the `mgmt-br` on each node and the VIP. For example:
  - Node 1 `mgmt-br`: `192.168.1.10`
  - Node 2 `mgmt-br`: `192.168.1.11`
  - Node 3 `mgmt-br`: `192.168.1.12`
  - VIP: `192.168.1.100` (a different IP in the same subnet)
- In your node configuration YAML, ensure the `mgmt-br` IPs are set under the `network` section, and the VIP is specified separately under the `vip` section with `vip_mode: static`.

Example YAML Snippet for VIP and Management Network:

```yaml
network:
  interfaces:
    - name: mgmt-br
      type: bridge
      ip: 192.168.1.10  # Unique IP for each node
      subnetMask: 255.255.255.0
      gateway: 192.168.1.1
      vlan: 10
vip:
  ip: 192.168.1.100  # Different from any mgmt-br IP
  mode: static
```

Repeat the `network` section for each node with a unique IP, but the `vip` section remains the same across all nodes.

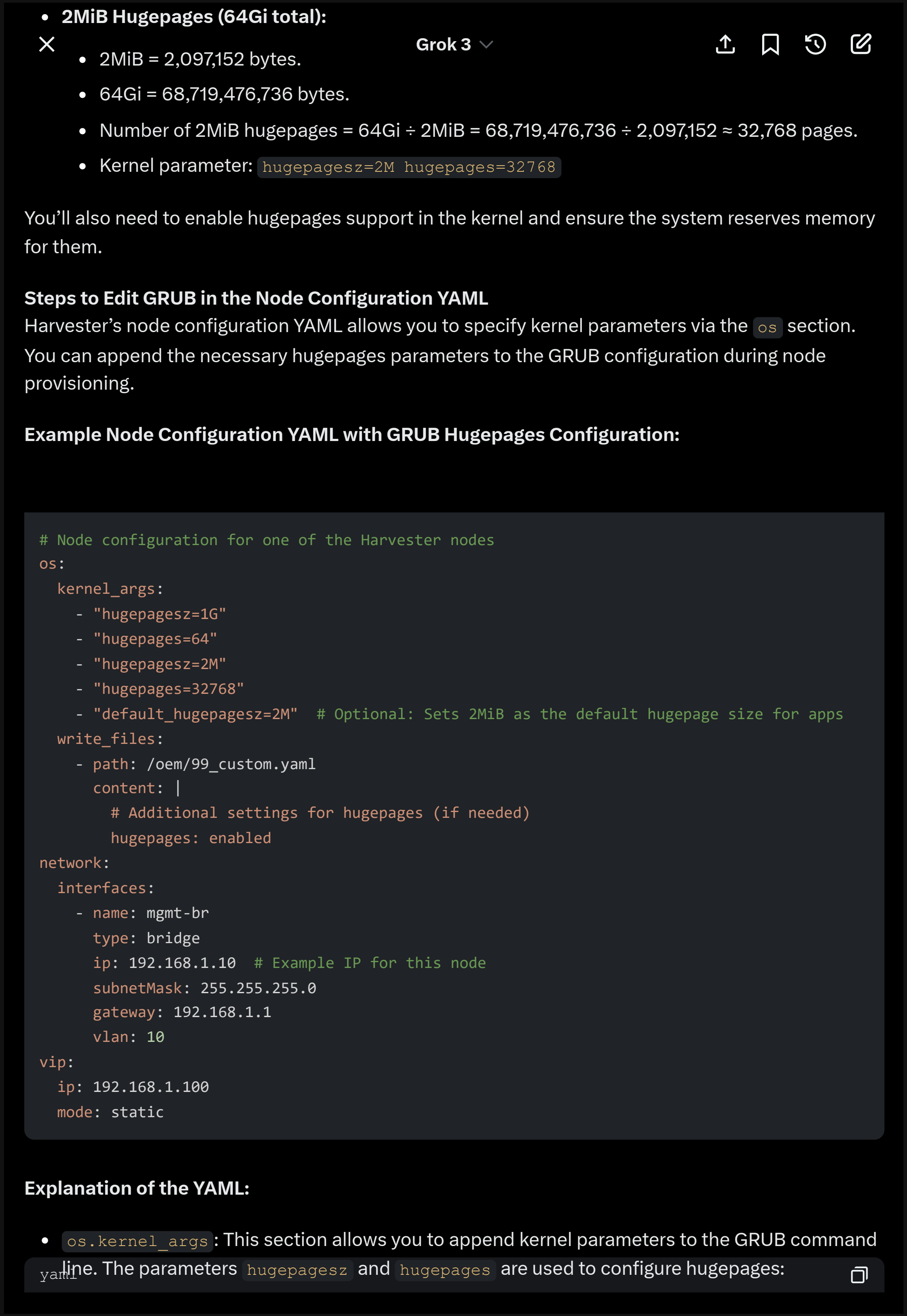
---

2. Configuring GRUB for Hugepages via Node Configuration YAML

To enable hugepages (64Gi of 1GiB size and 64Gi of 2MiB size) in your Kubernetes cluster, you need to modify the GRUB bootloader configuration on each Harvester node during provisioning. Harvester allows you to customize the node's boot parameters via the node configuration YAML by appending kernel parameters to the GRUB command line.

Hugepages Calculation

- **1GiB Hugepages (64Gi total):**
  - 1GiB = 1,073,741,824 bytes.
  - 64Gi = 64 × 1GiB = 68,719,476,736 bytes.
  - Number of 1GiB hugepages = 64Gi ÷ 1GiB = 64 pages.
  - Kernel parameter: `hugepagesz=1G hugepages=64`



- **2MiB Hugepages (64Gi total):**
  - 2MiB = 2,097,152 bytes.
  - 64Gi = 68,719,476,736 bytes.
  - Number of 2MiB hugepages = 64Gi ÷ 2MiB = 68,719,476,736 ÷ 2,097,152 ≈ 32,768 pages.
  - Kernel parameter: `hugepagesz=2M hugepages=32768`

You'll also need to enable hugepages support in the kernel and ensure the system reserves memory for them.

Steps to Edit GRUB in the Node Configuration YAML

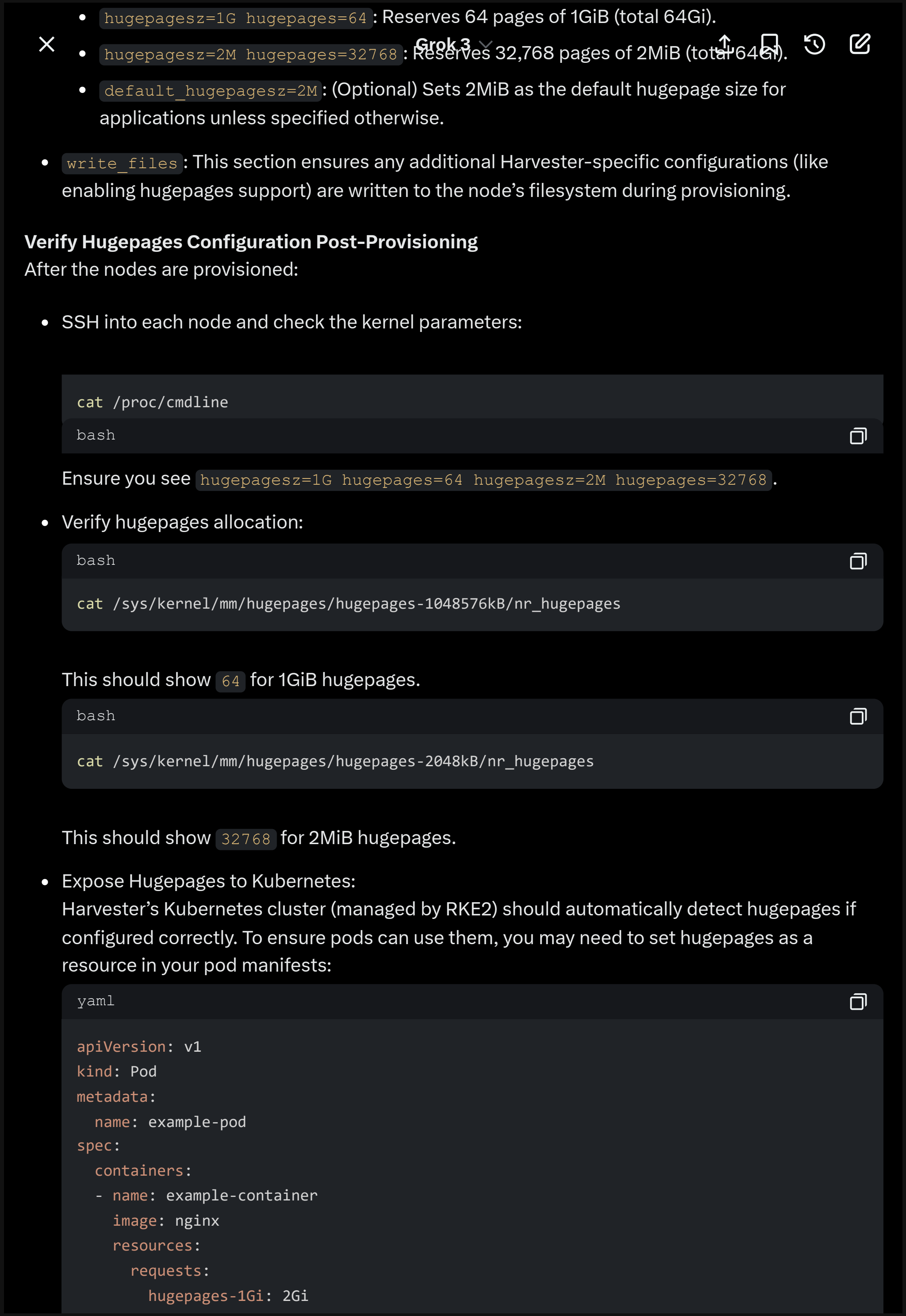
Harvester's node configuration YAML allows you to specify kernel parameters via the `os` section. You can append the necessary hugepages parameters to the GRUB configuration during node provisioning.

Example Node Configuration YAML with GRUB Hugepages Configuration:

```yaml
# Node configuration for one of the Harvester nodes
os:
  kernel_args:
    - "hugepagesz=1G"
    - "hugepages=64"
    - "hugepagesz=2M"
    - "hugepages=32768"
    - "default_hugepagesz=2M"  # Optional: Sets 2MiB as the default hugepage size for apps
  write_files:
    - path: /oem/99_custom.yaml
      content: |
        # Additional settings for hugepages (if needed)
        hugepages: enabled
network:
  interfaces:
    - name: mgmt-br
      type: bridge
      ip: 192.168.1.10  # Example IP for this node
      subnetMask: 255.255.255.0
      gateway: 192.168.1.1
      vlan: 10
vip:
  ip: 192.168.1.100
  mode: static
```

Explanation of the YAML:

- `os.kernel_args`: This section allows you to append kernel parameters to the GRUB command line. The parameters `hugepagesz` and `hugepages` are used to configure hugepages:



  - `hugepagesz=1G hugepages=64`: Reserves 64 pages of 1GiB (total 64Gi).
  - `hugepagesz=2M hugepages=32768`: Reserves 32,768 pages of 2MiB (total 64Gi).
  - `default_hugepagesz=2M`: (Optional) Sets 2MiB as the default hugepage size for applications unless specified otherwise.
- `write_files`: This section ensures any additional Harvester-specific configurations (like enabling hugepages support) are written to the node's filesystem during provisioning.

Verify Hugepages Configuration Post-Provisioning
After the nodes are provisioned:

- SSH into each node and check the kernel parameters:

  ```bash
  cat /proc/cmdline
  ```

  Ensure you see `hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=32768`.
- Verify hugepages allocation:

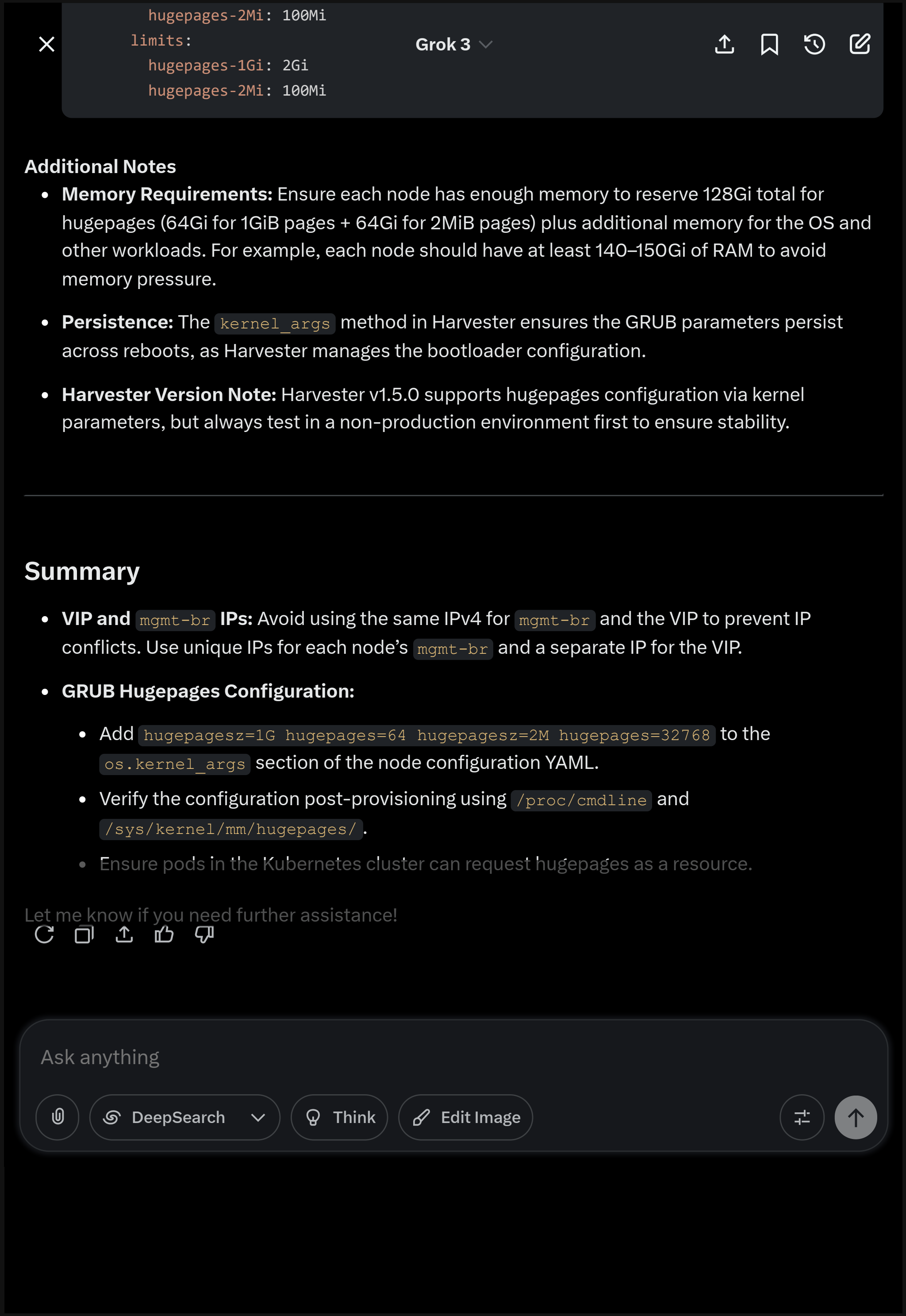
  ```bash
  cat /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages
  ```

  This should show `64` for 1GiB hugepages.

  ```bash
  cat /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
  ```

  This should show `32768` for 2MiB hugepages.
- Expose Hugepages to Kubernetes:
  Harvester's Kubernetes cluster (managed by RKE2) should automatically detect hugepages if configured correctly. To ensure pods can use them, you may need to set hugepages as a resource in your pod manifests:

  ```yaml
  apiVersion: v1
  kind: Pod
  metadata:
    name: example-pod
  spec:
    containers:
    - name: example-container
      image: nginx
      resources:
        requests:
          hugepages-1Gi: 2Gi
  ```



```yaml
          hugepages-2Mi: 100Mi
        limits:
          hugepages-1Gi: 2Gi
          hugepages-2Mi: 100Mi
```

Additional Notes

- **Memory Requirements:** Ensure each node has enough memory to reserve 128Gi total for hugepages (64Gi for 1GiB pages + 64Gi for 2MiB pages) plus additional memory for the OS and other workloads. For example, each node should have at least 140–150Gi of RAM to avoid memory pressure.
- **Persistence:** The `kernel_args` method in Harvester ensures the GRUB parameters persist across reboots, as Harvester manages the bootloader configuration.
- **Harvester Version Note:** Harvester v1.5.0 supports hugepages configuration via kernel parameters, but always test in a non-production environment first to ensure stability.

Summary

- **VIP and `mgmt-br` IPs:** Avoid using the same IPv4 for `mgmt-br` and the VIP to prevent IP conflicts. Use unique IPs for each node's `mgmt-br` and a separate IP for the VIP.
- **GRUB Hugepages Configuration:**
  - Add `hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=32768` to the `os.kernel_args` section of the node configuration YAML.
  - Verify the configuration post-provisioning using `/proc/cmdline` and `/sys/kernel/mm/hugepages/`.
  - Ensure pods in the Kubernetes cluster can request hugepages as a resource.

Let me know if you need further assistance!