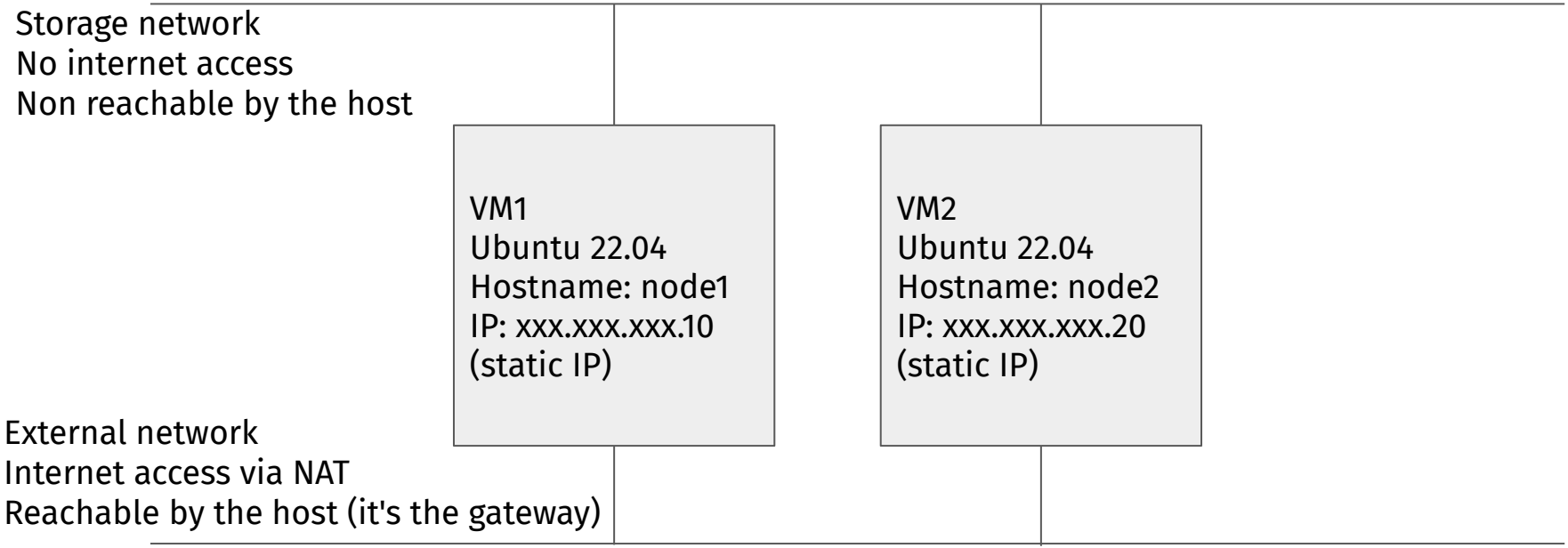
Università di Genova
DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Tasks

Operations that you need to perform

VMs and network setup

1. Setup two VMs and attach to each of them two networks as follows



IP ranges:
`192.168.50.0/24` for the external net
`10.255.255.0/24` for the storage net

NFS share

2. Setup on *node1* an NFS share at /data, allow connections only from *node2*'s storage network IP
3. Mount from *node2* the NFS share so that it persists at every boot
   - Make the task so that multiple nodes after node2 might mount the NFS share
4. Leverage the *ansible.posix* module (1.4.0) action *mount*
   https://docs.ansible.com/ansible/latest/collections/ansible/posix/mount_module.html

Tips:

You can find a node IP in the variable *ansible_default_ipv4['address']*.
You can read another node IP in the variable *hostvars[node_name]['ansible_default_ipv4']['address']*
We did a lesson on this exact operation

Docker

5. Install Docker from their repositories
   https://docs.docker.com/engine/install/ubuntu
6. Use ansible.builtin.apt, ansible.builtin.apt_repository, ansible.builtin.apt_key

Docker swarm

7. Initialize Docker Swarm on the first node
8. Join Docker Swarm on the other node
- Make so the task allows more than one worker node to join
9. Leverage the *community.docker* (version 3.2.1) action *docker_swarm*
   https://docs.ansible.com/ansible/latest/collections/community/docker/docker_swarm_module.html

Tips:

Look at the examples of the module, and on the first node save the result with the “register” directive, on the other nodes access that saved result to get the token at *swarm_facts['JoinTokens']['Worker']*
https://stackoverflow.com/questions/41122199/ansible-how-can-i-access-a-variable-of-other-host
https://stackoverflow.com/questions/48936489/use-variable-from-another-host

Docker swarm

Do not use externally forwarded ports

Docker registry

10. Install a Docker registry via Swarm so that it's reachable by both nodes, saving files on the NFS share at /data/docker-registry
11. Use the Docker image distribution/distribution:2.8.1
12. Use Ansible to reconfigure the nodes to trust your registry
    Tip: https://docs.docker.com/registry/insecure/
13. Setup authentication for the registry and login from the nodes
    Tip: community.docker.docker_login

Tips:

Configuration is at https://github.com/distribution/distribution/blob/v2.8.1/docs/configuration.md
If you have more than one replica, you might need to set something under "http"

Docker registry cache

- In order to avoid getting ratelimited, setup a cache for Docker Hub (called mirror in the documentation), saving files on the NFS share at /data/docker-registry-cache
- Use the Docker image distribution/distribution:2.8.1
- Use Ansible to reconfigure the nodes to trust the registry and use it as a mirror
  Tip: https://docs.docker.com/registry/insecure/
- Setup authentication for the registry and login from the nodes
  Tip: community.docker.docker_login

PostgreSQL

14. Use PostgreSQL as database for the services, storing the database data inside of /data/postgresql
15. Use the Docker image postgres:15.1
16. Choose either one db for all or one db per service (See next slide)

- Setup a primary-secondary cluster
  Tip: https://www.postgresql.org/docs/15/high-availability.html

PostgreSQL (choose one)

DB per service

Each service has its own database

Single shared DB

One DB is shared between services.

Question: Why choose one? Why chose another?

PostgreSQL

Do not configure database usernames, passwords, and tables via Ansible

Either:

- Leverage the default entrypoint to run SQL statements upon first startup (such as in the examples)
- Leverage the default entrypoint environment variables
  (this does not support the *single shared DB architecture*)

SSO (Keycloak)

17. Deploy a Keycloak instance using the “quay.io/keycloak/keycloak:20.0.1” Docker image
18. Use the db from before as your backend
19. Deploy a realm called “vcc”
20. Deploy a client under the “vcc” realm called “nextcloud”
21. Deploy a sample user (ensure that you can login with that on the service)

Tip:
We accidentally left a script performing these operations automatically in examples/nextcloud-keycloak-integrator

We also accidentally left an example on how to provision the server in the Makefile inside examples/keycloak

Service (Nextcloud)

22. Deploy a Nextcloud instance using the “nextcloud:23.0-apache” Docker image
23. Persist its data on NFS in /data/nextcloud
    - Deploy 2 replicas, ensuring that it works correctly
24. Install the “oidc_login” application
25. Configure Nextcloud to use Keycloak as its login source

Tip:
We accidentally left a script performing these operations automatically in examples/nextcloud-keycloak-integrator

External gateway (Traefik)

26. Deploy a Traefik instance using the “traefik:v2.9.6” Docker image
27. Expose Nextcloud on “cloud.localdomain”
28. Expose Keycloak on “auth.localdomain”
29. Expose Grafana (see later) on “mon.localdomain”
30. Redirect automatically from HTTP to HTTPS
31. Leverage Docker swarm label configuration discovery

Tip:
Take a look at Enrico’s lessons.
To connect to a fictitious domain, you need to modify your hosts file so that the IP of the VM is associated with the name.

Monitoring & Logging

Node agent

32. Deploy on each node FluentBit using “cr.fluentbit.io/fluent/fluent-bit:2.0.6”
- Use it for
    33. Sending Docker logs to Loki (see following)
    34. Exposing metrics of the node using the “Node Exporter Metrics” plugin
    35. Sending metrics to Prometheus (see following) in some way
        - Getting logs/metrics from as many things as possible (e.g. systemd service logs)

Tips:
Fluent Bit example configuration for sending Docker logs
Example configuration for sending metrics to a Prometheus clone
Cloud service example configuration for logs and metrics

Monitoring & Logging

Metrics server

36. Deploy “prom/prometheus:v2.40.6”
37. Persist its data in /data/prometheus
38. Set its retention period to 7 days
    - Use it for
        39. Receiving / Scraping the metrics from FluentBit and other components
            - Monitoring containers with cadvisor
40. Ensure that metrics belonging to each node and service are present

Tips:
Prometheus guide on monitoring Docker Swarm

Monitoring & Logging

Log storage server

41. Deploy “grafana/loki:2.7.1”
42. Persist its data in /data/loki
    - Use it for
        43. Receiving logs from FluentBit
44. Ensure that logs belonging to each node and service are present
45. Ensure that the container names / labels are meaningful

Tips:
Fluent Bit documentation for Loki
Loki documentation for FluentBit

Monitoring & Logging

Dashboarding

46. Deploy “grafana/grafana:9.3.1”
    - Automatically provision
        47. Loki and Prometheus as data sources
            - At least two dashboards
                48. One showing metrics (e.g. Node CPU, disk, RAM usage, ...)
                49. One showing logs (e.g. Traefik Access Log, Keycloak authentication log, ...)
50. Enable authentication
    - Automatically provision and use a “grafana” client in keycloak for authentication

Tips:
Grafana Docker setup guide
Automatically provisioning Grafana
Integrating Grafana with Keycloak