Deploy K8s Cluster with DeepOps and Kubespray

Full Documentation:

- Read full Kubernetes on DeepOps documentation here.
- Read full Kubespray Getting Started documentation here.

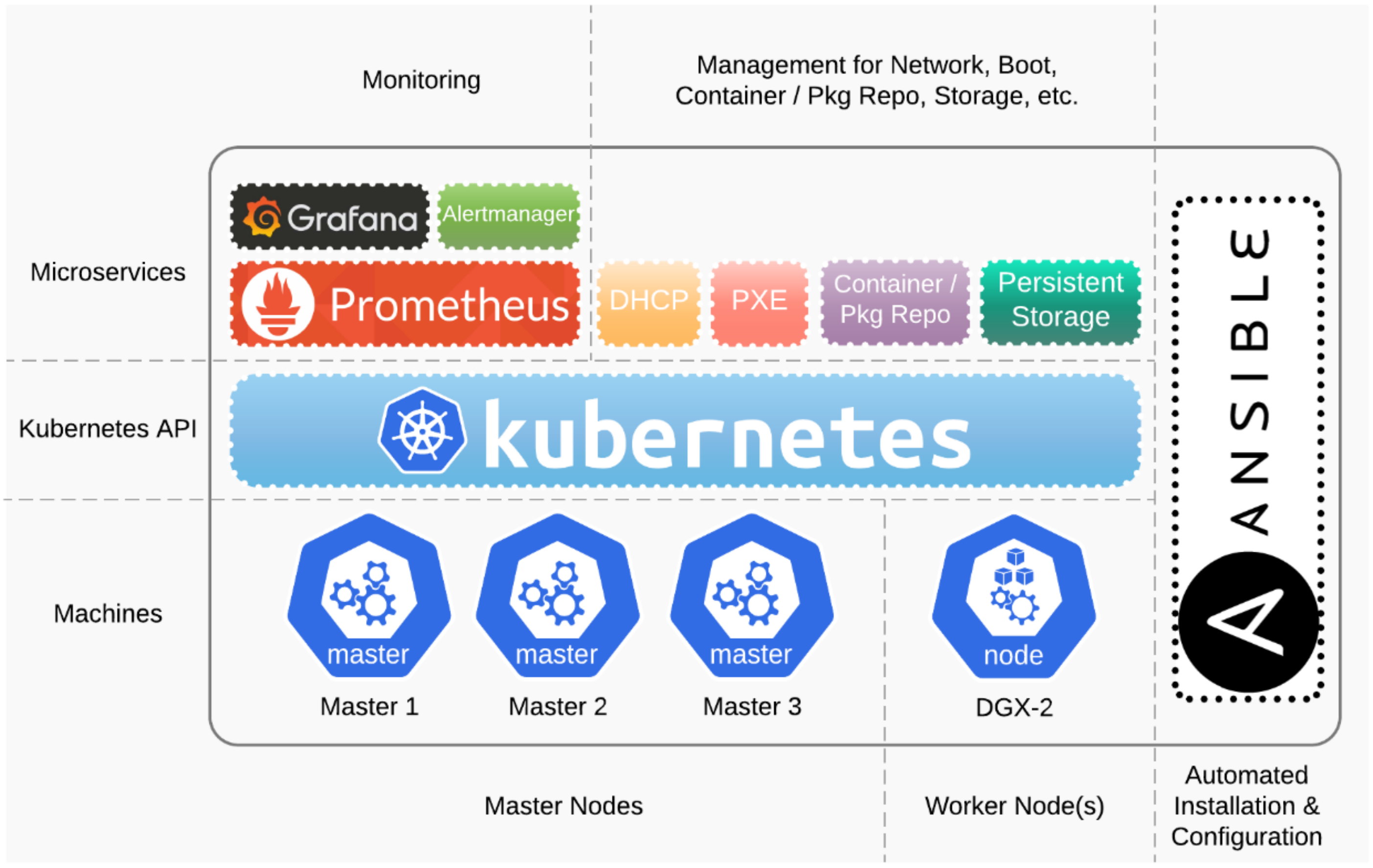
Minimum Specifications:

3 Master Nodes (VM's) soft limits

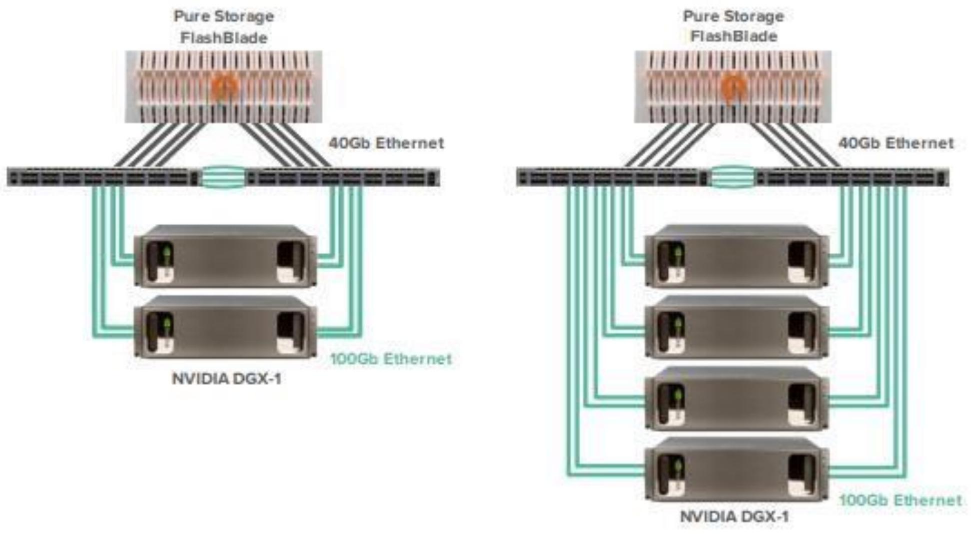
- CPU: 4 CPU cores
- Memory: 32G
- Disk: 100G
- Network: 2 NIC's
- OS: Ubuntu 18.04 LTS

Table of Contents:

DeepOps Architecture Overview:

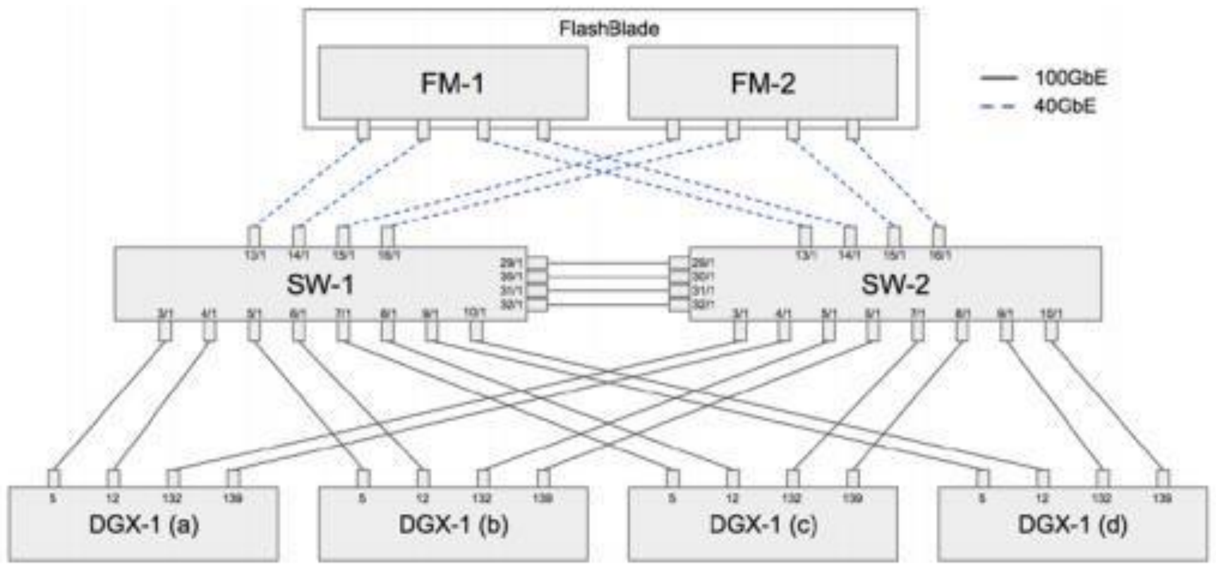


Possible System Architecture (Cisco):



Source: AIRI Reference Architecture with Cisco Nexus 9300 Ethernet Switch

Possible Network Architecture (Cisco):



Source: AIRI Reference Architecture with Cisco Nexus 9300 Ethernet Switch

Create Cluster:

Worker Nodes:

- Add SSH keys from master node(s) to worker node(s) with the command:

  `ssh-copy-id -i ~/.ssh/id_rsa.pub user@host`

Main Master Node:

- Clone DeepOps repo and enter directory.
- To setup provisioning machine, run: `./scripts/setup.sh`.
- Run `cp -r config.example config` and edit `config/inventory` to add nodes.
- Format like: `node01      ansible_host=127.0.0.1      ip=127.0.0.1`
  - See Appendix B for an example.
- Create cluster via Ansible/Kubespray with the command: `ansible-playbook -i config/inventory playbooks/k8s-cluster.yml -k -K`

Troubleshooting:

- During installation:
  - Error: Failed to lock apt for exclusive operation:
    - Continue anyways, it should work regardless.
  - Error: Could not get lock /var/lib/dpkg/lock-frontend - open (11: Resource temporarily unavailable)
    - See this AskUbuntu page if the error causes problems with the installation. Even though it says nodes failed, the cluster should come up without a problem.
- Post-installation:
  - `coredns` won't deploy: "0/X nodes are available: X node(s) didn't match pod affinity/anti-affinity, X node(s) didn't satisfy existing pods anti-affinity rules"
    - The deployment is likely trying to create X+1 pods, where X is the number of nodes available. Scale the deployment down to the number of available nodes, and the error should go away.
  - `tiller-deploy` won't deploy: "0/X nodes are available: X node(s) didn't match node selector, X node(s) had taints that the pod didn't tolerate"
    - Remove the taint on one of your master nodes with `kubectl taint node <node-name> node-role.kubernetes.io/master:NoSchedule-`

Install Cluster Add-ons:

Main Master Node:

- Install GPU drivers with `ansible-playbook -i config/inventory playbooks/k8s-gpu-plugin.yml -K -k`
  - **Troubleshooting:** If you cannot schedule pods with GPU resources, check the log of the `nvidia-device-plugin-daemonset` pod on the GPU node to find the error.
    - If the log says "Failed to initialize NVML: could not load NVML library":
      - Check to make sure `/etc/docker/daemon.json` on your GPU node is configured to use the nvidia-container-runtime.
      - If the docker runtime is configured correctly and you still cannot schedule GPU resources, try the following commands on your GPU node:
        - `systemctl daemon-reload`
        - `systemctl restart docker`
    - If you still cannot schedule pods with GPU resources:
      - Try running `ansible-playbook -i config/inventory playbooks/nvidia-driver.yml -K -k` to manually install the Nvidia GPU drivers.
      - If all else fails, remove the GPU node or tear down the entire cluster and rebuild it.
- Install K8s dashboard with `./scripts/k8s_deploy_dashboard_user.sh`
  - Script will:
    - Install dashboard and create any resources required (including deployments, services, service accounts, etc.)
    - Automatically expose an endpoint for the service (port changes per script run).
    - Output a URL to open in web browser.
    - Output a token to access K8s dashboard login page.
- Install monitoring with `./scripts/k8s_deploy_monitoring.sh`
  - Script will:
    - Install dashboard and create any resources required (including deployments, services, service accounts, etc.)
    - Automatically expose an endpoint for the service (ports are set).
    - Output URLs to open in web browser to access dashboards.
  - Services can be reached from:
    - Grafana: http://mgmt:30200

        - Prometheus: http://mgmt:30500
        - Alertmanager: http://mgmt:30400

- Install persistent storage with `./scripts/k8s_deploy_rook.sh`
    - Script will:
        - Install Rook using Ceph as a backend.
        - Deploy Ceph file and block storage.
        - Deploy a Ceph cluster manager and toolbox container.
        - Deploy the Ceph dashboard.
        - Use `./services/rook-cluster.yml` as the deployment configuration – make any desired changes here.
    - **Troubleshooting:** If you've previously installed rook-ceph and your rook-ceph pods are not deploying, try deleting everything in the default data directory with `sudo rm -rf /var/lib/rook` on all nodes.
        - To remove the Rook deployment, run `./scripts/rmrook.sh`

- Install Kubeflow with `./scripts/k8s_deploy_kubeflow.sh`
    - Full Kubeflow documentation can be read here.
    - Script will:
        - Install dashboard and create any resources required (including deployments, services, service accounts, etc.)
        - Automatically expose an endpoint for the service (port changes per script run).
        - Output a URL to open in web browser to access Kubeflow dashboard.

Scale Cluster:

- Recreate inventory file using the inventory builder tool as seen here under Quick Start (see Appendix C for an example).
- Make sure to only have Kubernetes resources specified here such as `[all]`, `[kube-master]`, `[kubenode]`, `[etcd]`, and `[k8s-cluster:children]`.
- Additionally, make sure these resources match the previous inventory file.
- In `kubespray/` directory:

  Add node:

  - Add new node to `inventory/mycluster/hosts.yml` (as well as `../config/inventory`).
  - Run `ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user=root scale.yml -K -k`

  Remove node:

  - Run `ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user=root remove-node.yml --extra-vars "node=node-to-remove" -K -k`
  - Remove node from inventory files.

Delete Cluster:

- Recreate inventory file using the inventory builder tool like before.
- Additionally, make sure these resources match the previous inventory file.
- Script will:
  - Fully tear down cluster.
  - Uninstall kubectl, kubeadm, kubelet from nodes.
- In `kubespray/` directory:
  - Run `ansible-playbook -i inventory/mycluster/hosts.yml --become --become-user=root reset.yml -K -k`

Note:

- If during Cluster deletion or Node removal, the Ansible script hangs on "TASK [reset : reset | unmount kubelet dirs", try restarting docker service on all nodes via `systemctl restart docker.service`.

Appendix A: Compatibility

DeepOps Version: 19.07

Software	Version
Ansible	2.8.11
Kubespray	v2.10.4
Kubernetes	v1.14.3
Docker	18.09.6
Rook	v1.0.2
Ceph	v13 (v13.2.6-20190604)
Slurm	19.05

Appendix B: Example Inventory File

`deepops/config/inventory`

```
# Server Inventory File
#
# Uncomment and change the IP addresses in this file to match your environment
# Define per-group or per-host configuration in group_vars/*.yml

######
# ALL NODES
######
[all]
mgmt-node01     ansible_host=127.0.0.1 ip=127.0.0.1
worker-node01   ansible_host=127.0.0.2 ip=127.0.0.2
worker-node02   ansible_host=127.0.0.3 ip=127.0.0.3
######
# KUBERNETES
######
[kube-master]
mgmt-node01     ansible_host=127.0.0.1 ip=127.0.0.1

[etcd]
mgmt-node01     ansible_host=127.0.0.1 ip=127.0.0.1

[kube-node]
worker-node01   ansible_host=127.0.0.2 ip=127.0.0.2
worker-node02   ansible_host=127.0.0.3 ip=127.0.0.3

[k8s-cluster:children]
kube-master
kube-node

######
# SLURM
######
[slurm-master]

[slurm-node]

[slurm-cluster:children]
```

```
######
# NFS
######
[nfs-server]

[nfs-clients]

#[nfs-clients:children]

######
# OFFLINE CACHE BUILDER
######
[cache_builder]
localhost   ansible_connection=local

######
# SSH connection configuration
######
[all:vars]
#ansible_user=ubuntu
# Configure SSH bastion/jumpbox for the cluster
#ansible_ssh_common_args='-o ProxyCommand="ssh -W %h:%p -q ubuntu@10.0.0.1"'
```

Appendix C: Example Hosts File

`deepops/kubespray/inventory/mycluster/hosts.yml`

```
[all]
mgmt-node01        ansible_host=127.0.0.1 ip=127.0.0.1
worker-node01      ansible_host=127.0.0.2 ip=127.0.0.2
worker-node02      ansible_host=127.0.0.3 ip=127.0.0.3

[kube-master]
mgmt-node01

[kube-node]
worker-node01
worker-node02

[etcd]
mgmt-node01

[k8s-cluster:children]
kube-node
kube-master
```