Running Legacy VM's along with containers in Kubernetes

Delusion or Reality?

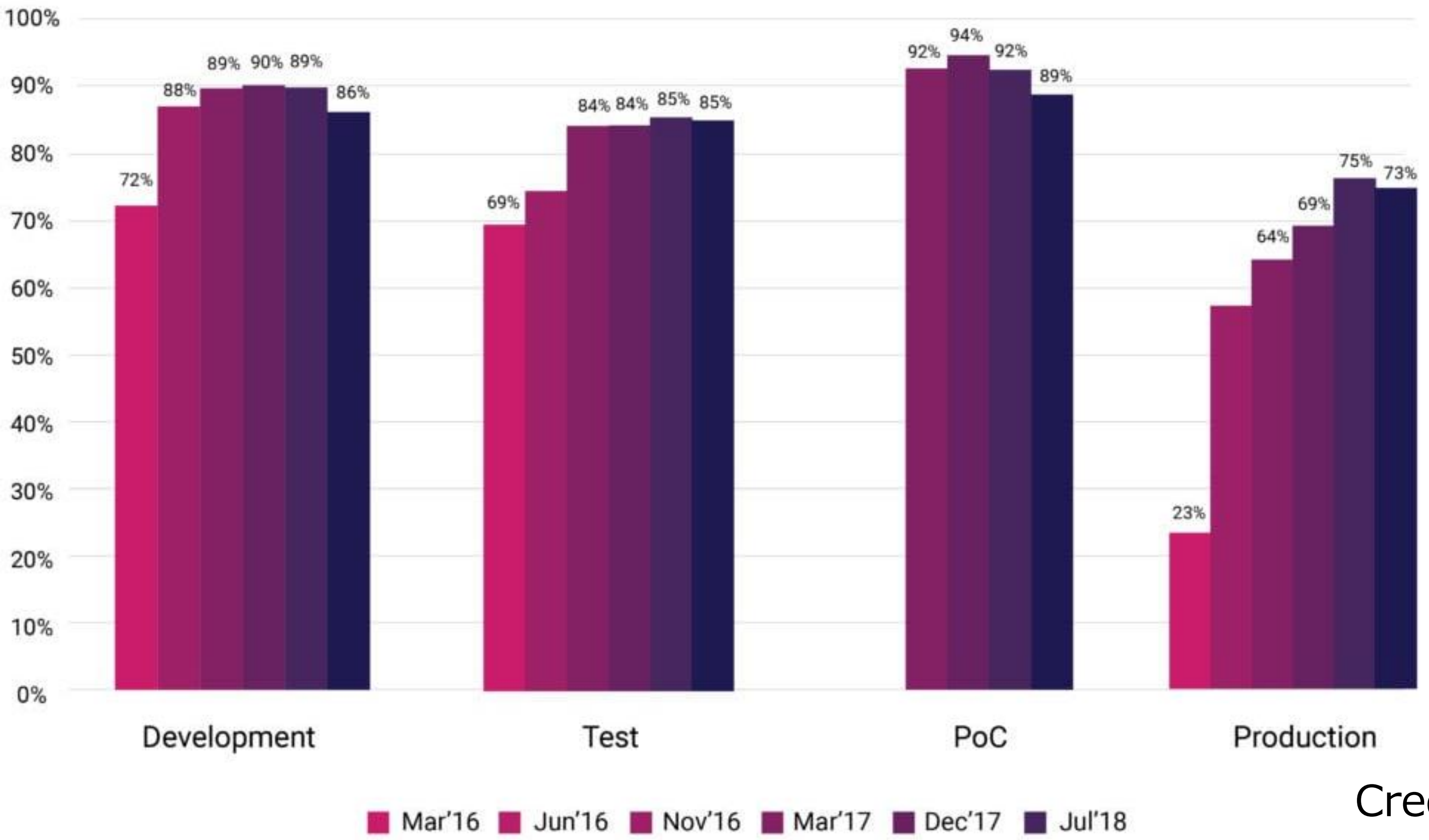
Kunal Kushwaha
NTT Open Source Software Center

About me

- Work @ NTT Open Source Software Center
- Collaborator (Core developer) for libpod (podman)
- Contributor KubeVirt, buildkit and other related projects
- Docker Community Leader @ Tokyo Chapter

Growth of Containers in Companies

Adoption of containers in production has significantly increased

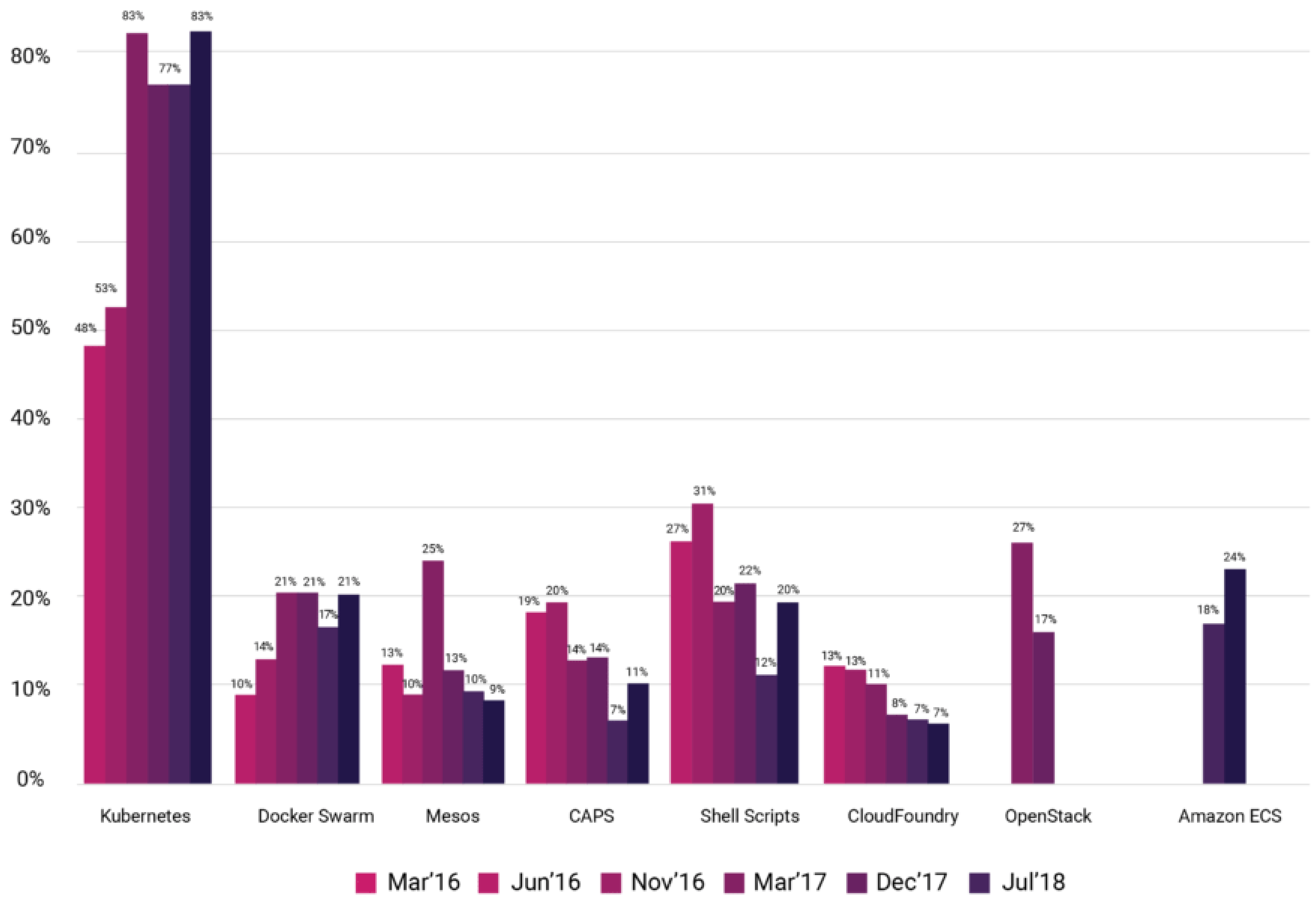


Credits: CNCF website

Growth of Container Orchestration usage

Innovative R&D by NTT

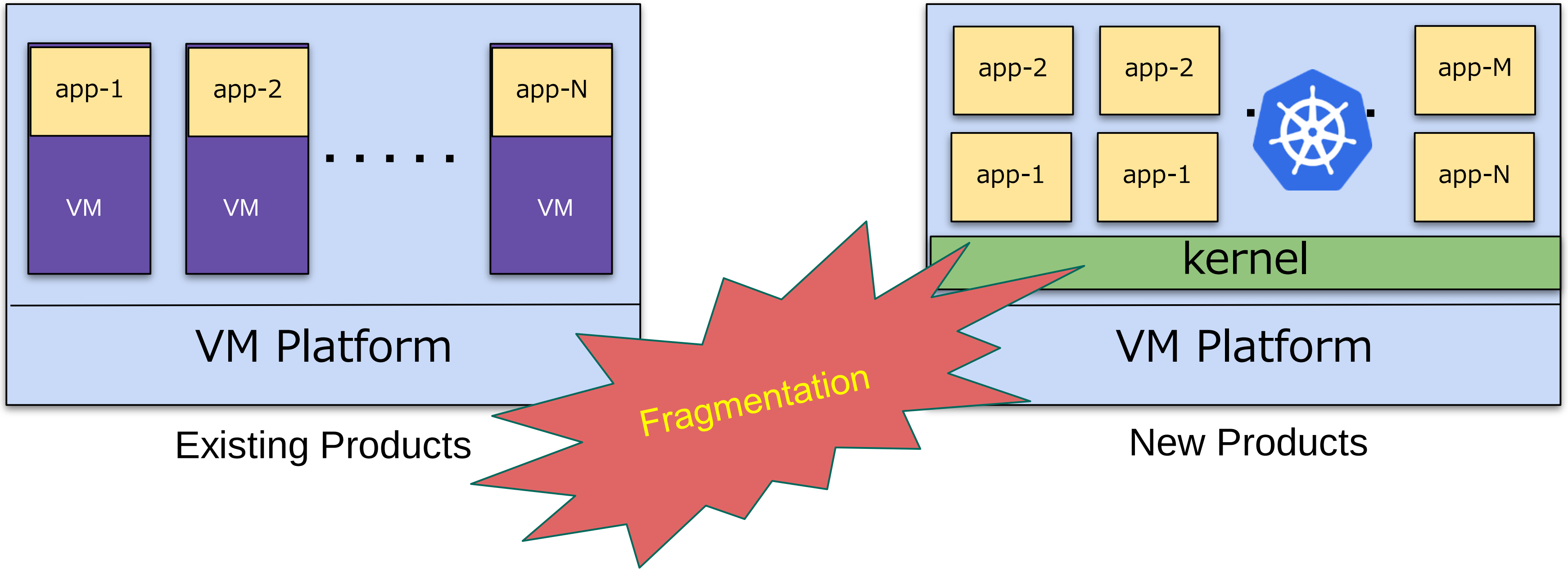
Adoption of container orchestrator like Kubernetes have also increased significantly on public as well private clouds.



Credits: CNCF website

NTT

Infrastructure landscape



- The application infrastructure is fragmented as most of old application still running on traditional infrastructure.
- Fragmentation means more work & increase in cost

What keeps applications away from Containers

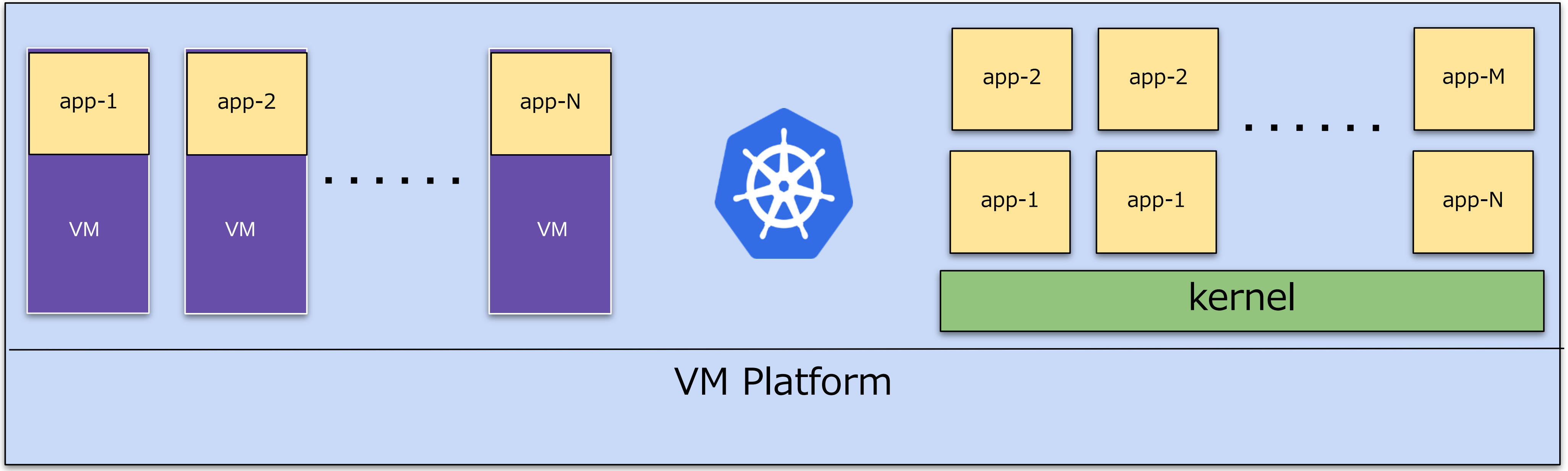
- Lack of knowledge / Too complex to migrate in containers.
- Dependency on custom kernel parameters.
- Application designed for a custom kernel.
- Application towards the end of life.

Companies prefer to re-write application, rather than directly migrating them to containers.

DZone

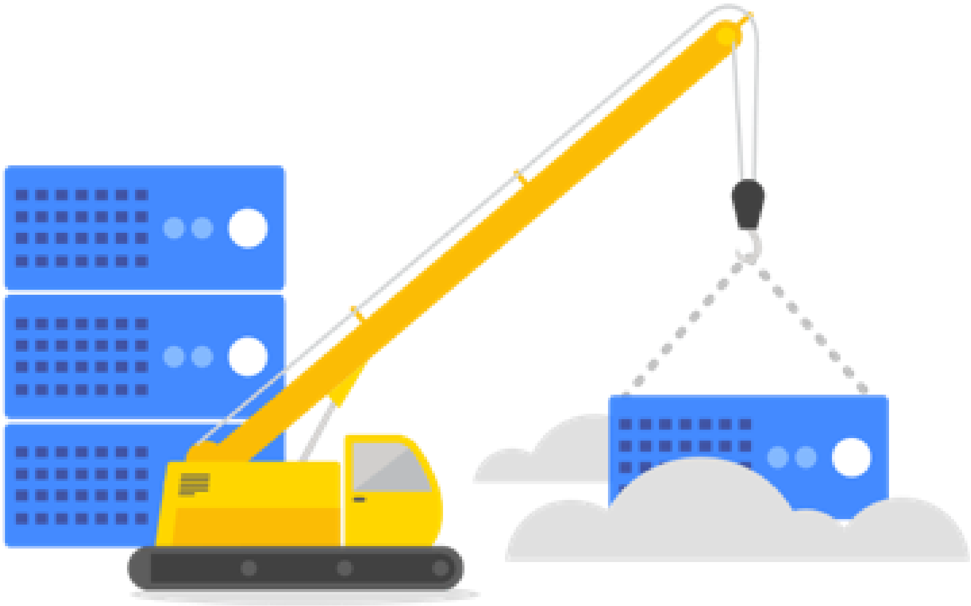
https://dzone.com/guides/containers-orchestration-and-beyond

Ideal World



- Applications in VM and containers can be managed with same control plane
- Management/ Governance Policies like RBAC, Network etc. can same for all application
- Intercommunication between application over containers and VM possible.

“Lift & Shift” Strategy of Migration



- Original terminology coined for migrating in-house application to Cloud.
- Also known as **re-hosting** application.
- The lift and shift migration approach is about migrating your application and associated data to the target platform with minimal or no changes.
- Making VMs part of Kubernetes infrastructure along with containers, will help Lift & Shift strategy for migrating applications running in VMs to Kubernetes.

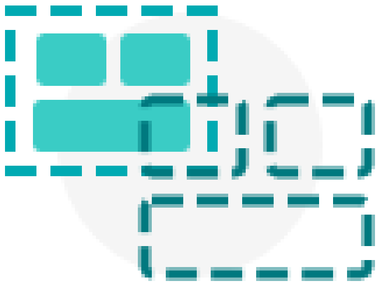
*Pic credit: Google Cloud

KubeVirt Overview

- KubeVirt extends Kubernetes by adding resource types for VMs through Kubernetes Custom Resource Definitions API
- It enables to run VMs along with containers on existing Kubernetes nodes
- VMs run inside regular Kubernetes pods, where they have access to standard pod networking and storage, and managed using standard Kubernetes tools such as kubectl
- Build on mature technology like KVM, qemu, libvirtd, Kubernetes

KubeVirt Goals

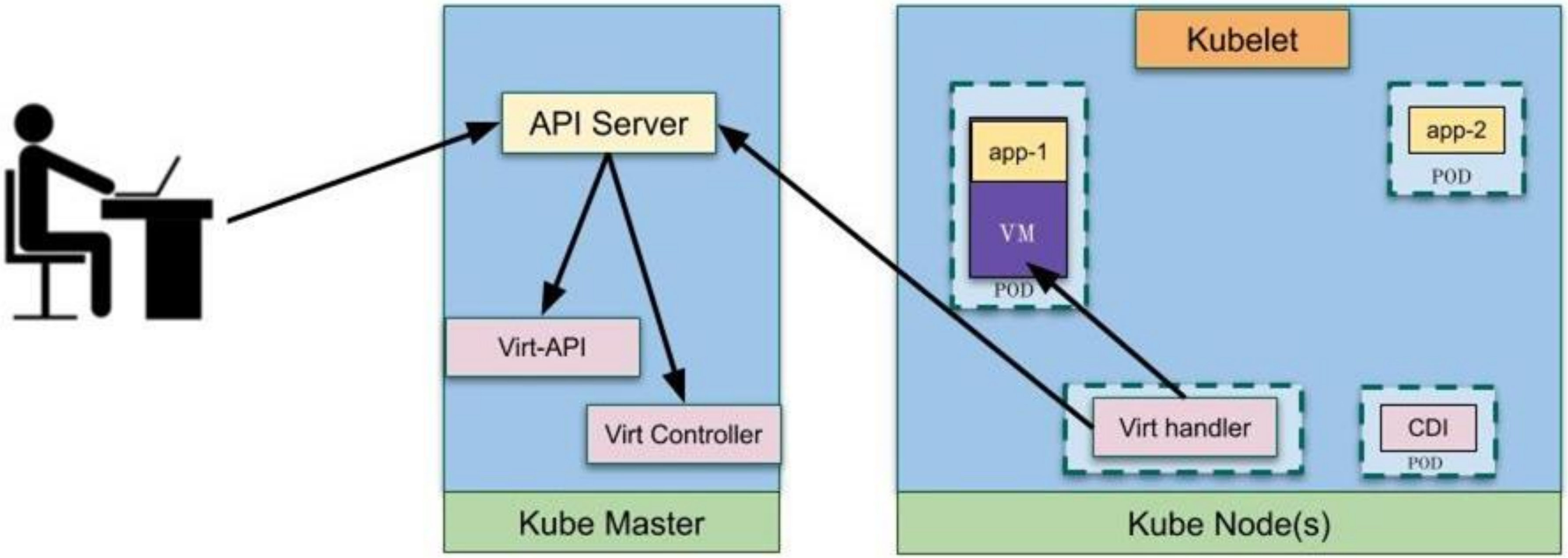
Leverage KubeVirt and Kubernetes to manage virtual machines for impractical-to-containerize apps.



Combine existing virtualized workloads with new container workloads on the one platform.

Support development of new micro-service applications in containers that interact with existing virtualized applications.

KubeVirt Control Flow & Architecture



- Virt-API and Virt-Controller are added to Master Node.
- Virt-Handler is on each worker node, responsible to launch VM in a pod.
- Containerized-data-importer prepare persistent Volumes

Important Features of KubeVirt

- KubeVirt features
  - Can be installed and removed in existing k8s cluster.
  - Supports multiple network and storage options, suitable for migration
  - VMs run as part of pod, so utilize all other k8s components like DNS, RBAC, Network Policies etc.

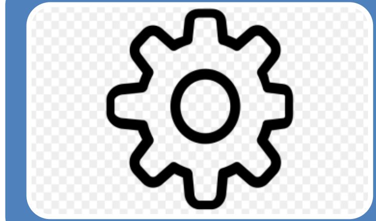
- VM capabilities
  - Run VM with images in qemu qcow2 format, same as in OpenStack
  - latest device support
    - Q35 machine support.

KubeVirt Evaluation Process

Evaluation Viewpoint

VM to K8s Image migration

- Import into k8s PV or Container Image
- Understand problems/limitations of system



Configuration & Deployment

- Design VM to match original requirements / environment
- Understand problems/limitations /workarounds

Operational & Functional Validation

- Service creation
- App functionality/ accessibility / restriction

Reliability

- Time to recover from failure
- Maintenance downtime/disruption

Important KubeVirt Objects

- VirtualMachine (VM) :

  represents a virtual machine in the runtime environment of Kubernetes.

- VirtualMachineInstanceReplicaSet (VMRS) :

  Tries to ensures that a specified number of virtual machine replicas are running at any time.

- DataVolume :

  Data Volumes(DV) are an abstraction on top of Persistent Volume Claims(PVC) and the Containerized Data Importer(CDI)

- ContainerRegistryDisk :

  local ephemeral disk for booting VMI. Any changes to disk are not persisted across reboot.

Migration of VM to KubeVirt

App in VM

- Prepare VM for Migration
- Consistent data state

Export & Build Image

- Export the VM Disk & convert in qcow2 format
- Import in Persistent Volume (PV)
  Or
- Build Docker image

Prepare k8s Manifest

- Prepare yaml file for VM Definition in KubeVirt

Deploy

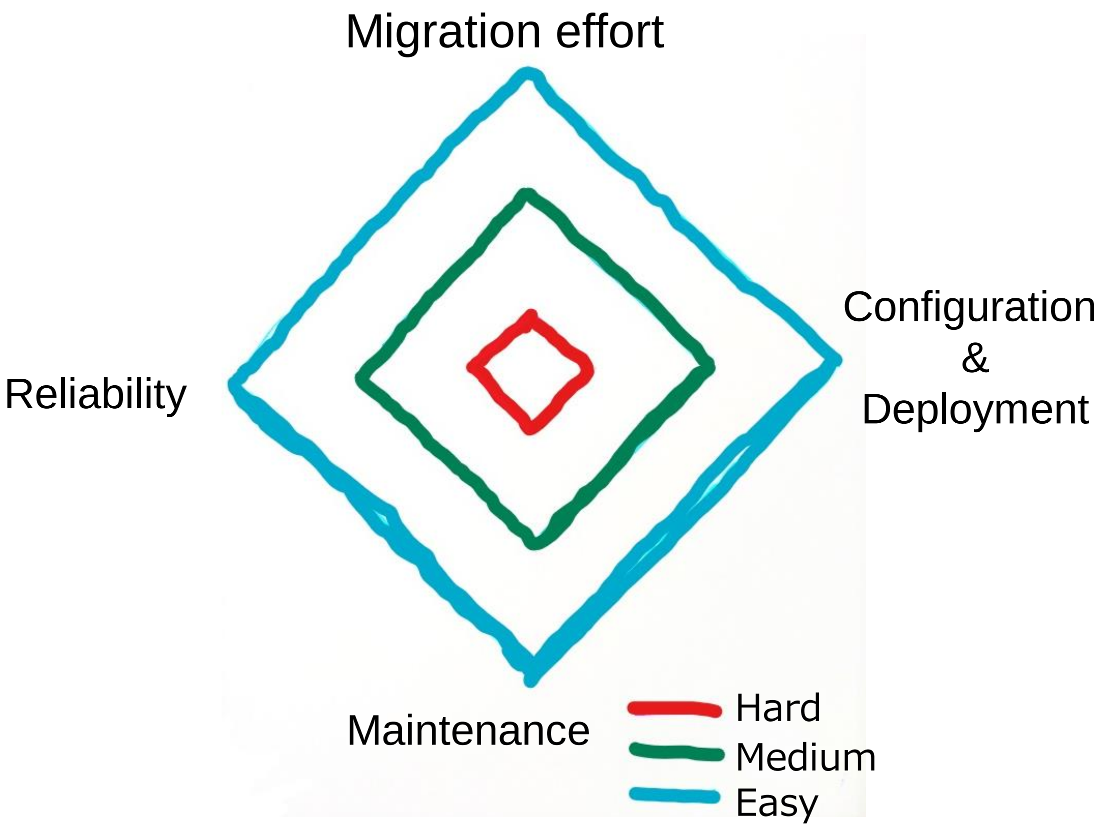
- Deploy application with kubectl apply

Expose Service

- Create Service
- Expose the service to outer network

Measuring Parameters

- Image Migration
- Configuration & Deployment
- Maintenance
- Reliability of service



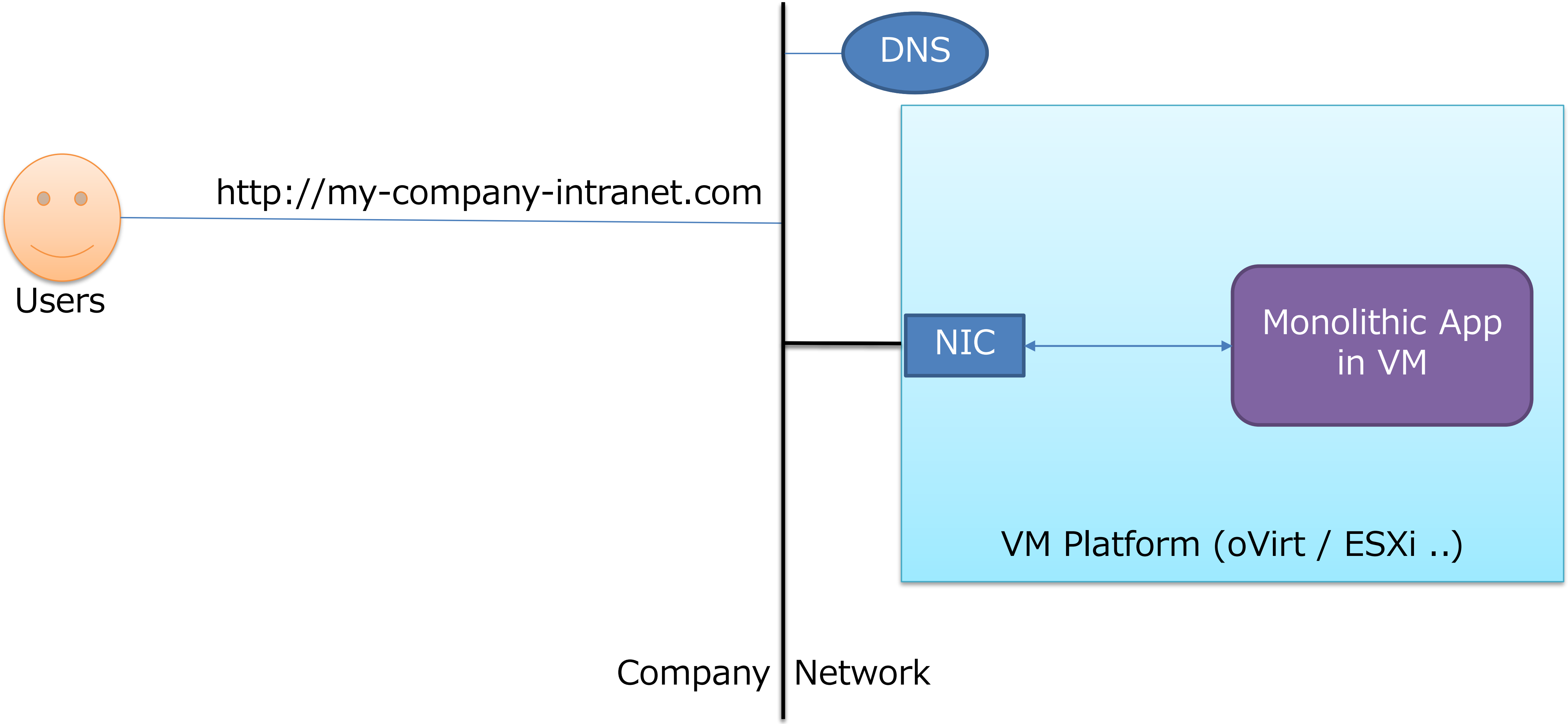
Use Cases

Evaluation Use Case

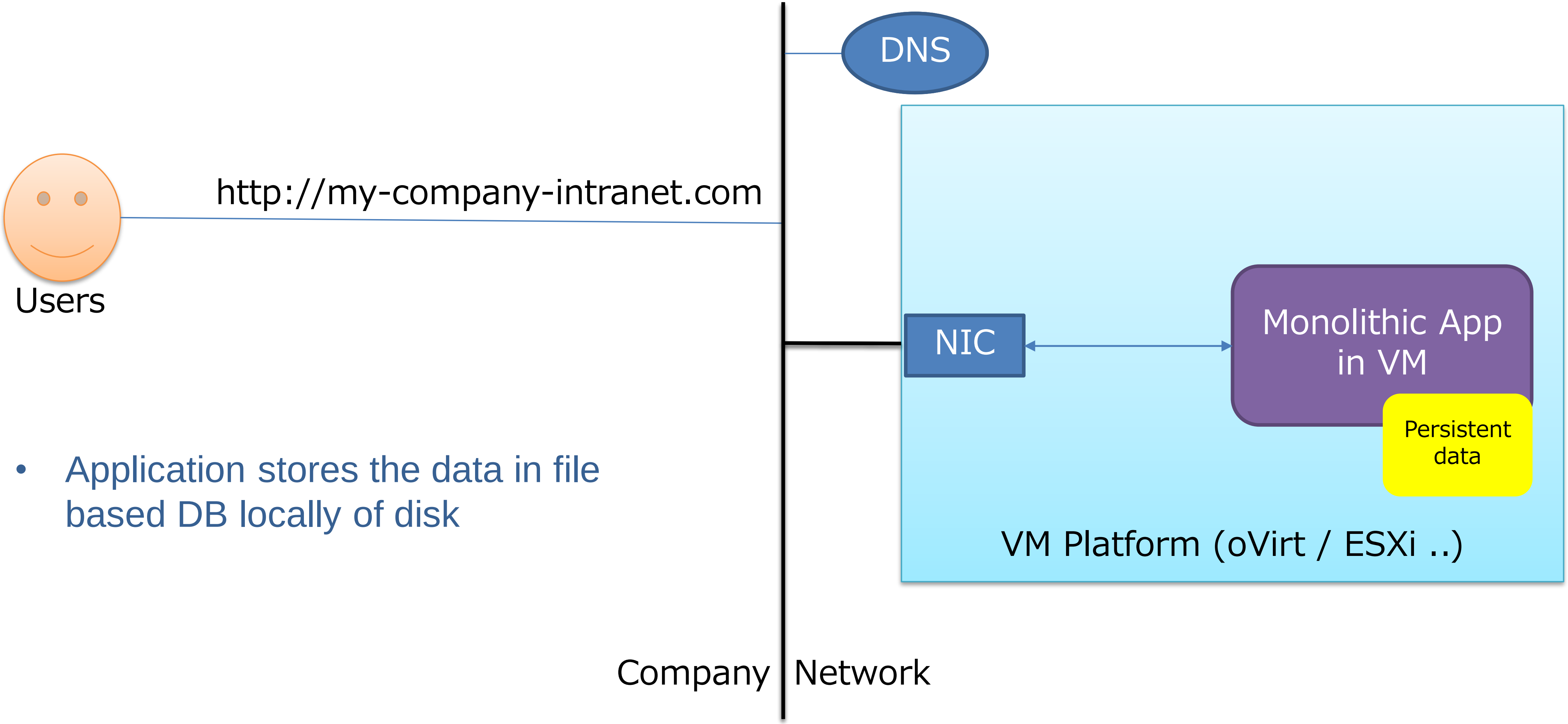
- Monolithic Application (Single VM)
- 3 Tier Web Application (Multiple VM)
- HA with multi network Architecture

Monolithic Application

Monolithic application

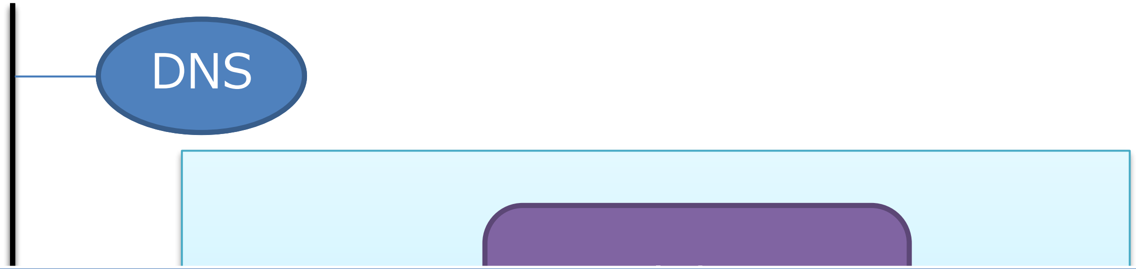


Monolithic application



- Application stores the data in file based DB locally of disk

Monolithic application



Application Type	Standalone application with file based DB.
Requirements	• Persistent Storage • Networking • Volume Backup
Policies	• No auto re-creation of VM • Health Check

Image Migration is simple process
- Depending on disk size, it may be time consuming.

- Converting vm-disk to kubevirt compatible format
  - img, qcow2, iso etc are supported formats *
  - Conversion can be done with any v2v or p2v tools
- Importing disk to KubeVirt (Kubernetes)

```
$ qemu-img convert –f vdi monolithic.vdi –O qcow2 mono.qcow2


$ virtctl image-upload –pvc-name-monolithic-vm-disk \
  --pvc-size=64Gi\
  --image-path=/home/kunal/images/mono.qcow2 \
  --uploadproxy-url=https://172.20.20.51:5002
```

*github.com/kubevirt/containerized-data-importer/blob/master/doc/supported_operations.md

Migration process: VM definition

- Depending on original VM configuration, writing VM yaml file could be tough.[1]
- Translation of old VM configuration to new VM yaml is done manually.

```
apiVersion: kubevirt.io/v1alpha3
kind: VirtualMachine
metadata:
  labels:
    kubevirt.io/vm: monolithic-app
  name: monolithic-app
spec:
  runStrategy: manual
```

```
template:
    spec:
      terminationGracePeriodSeconds: 30
      domain:
         devices:
          disks:
          - disk:
             bus: virtio
             name: pvcdisk
      volumes:
       - name: pvcdisk
         persistentVolumeClaim:
           claimName: monolithic-vm-disk
       networks:
         - name: default
           pod: {}
```

- Key definitions
  - run strategy : defines vm state after object creation (running, manual etc)
  - Volume
  - Network

[1] : OpenShift supports KubeVirt templates, which is helpful

Migration process: Service Definition

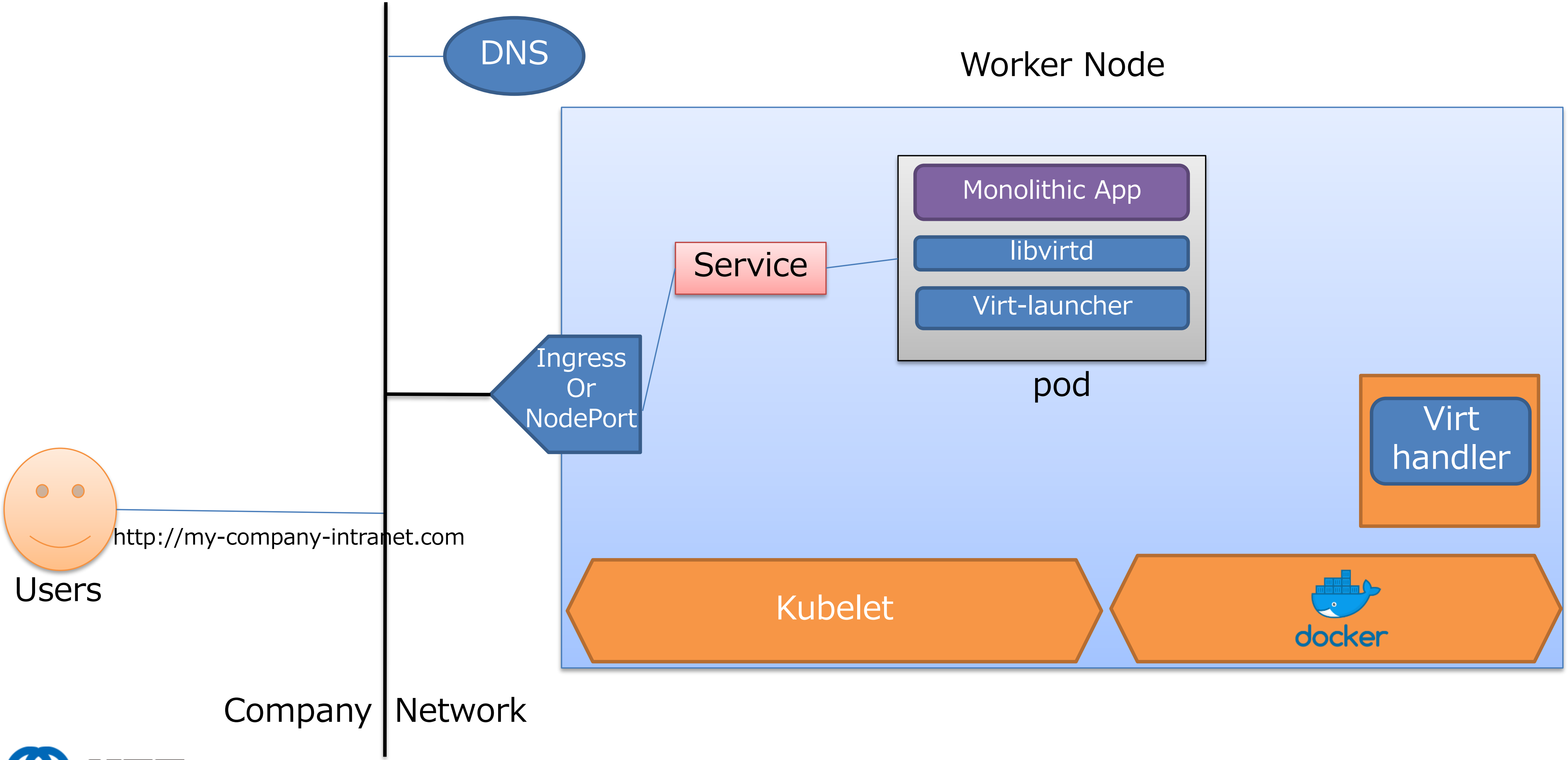
Common to Kubernetes

- All solutions of Service Discovery of Kubernetes shall work with KubeVirt VMs too.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: intranet-services
spec:
  ports:
  - name: nodeport
    nodePort: 30000
    port: 27017
    targetPort: 80
  selector:
    kubevirt.io/vm: monolithic-app
  type: NodePort
```

Sample service definition

After Migration: Monolithic application



Migration process: Maintenance

Kubernetes/KubeVirt do not add much value for maintenance phase for this kind of application

- Backup/snapshot management.
  - PersistentVolume (PV) is provided by K8s storage providers.
  - Managed in similar way as PersistentVolume of K8s.

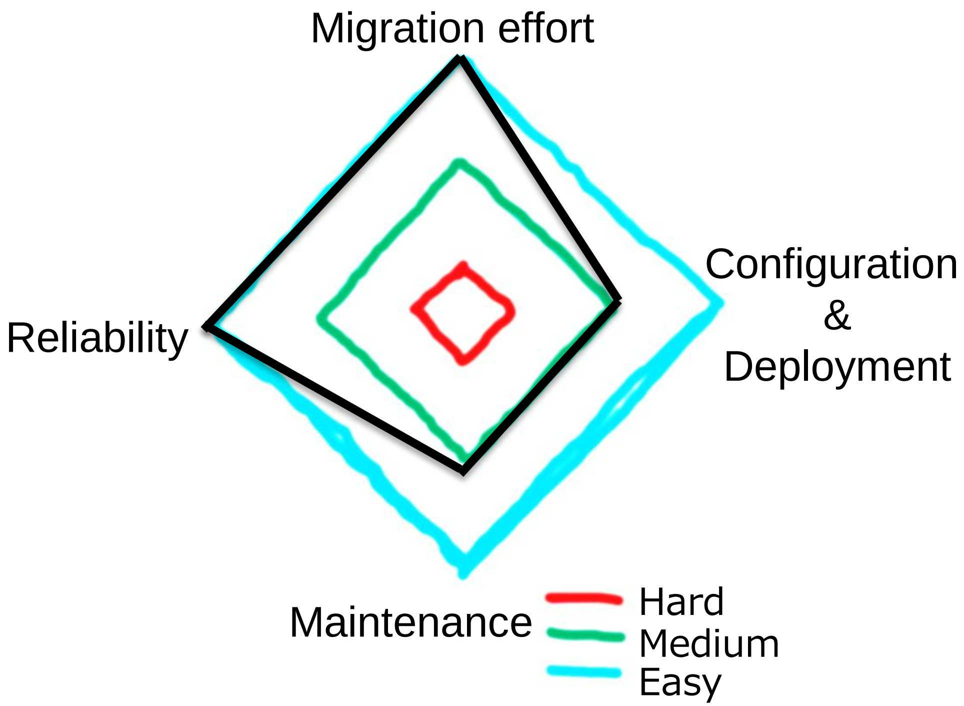
- Patch management/VM upgrade
  - Traditional way (ssh / config manager)

- On failure
  - Depending on Run strategy, action can be defined.

Conclusion: Monolithic application migration

- Easy to migrate and maintain application in Kubernetes

- Migration process :	Easy.
- online migration :	No.
- Security :	Good
- As good as Kubernetes	
- RBAC policies	
- Network policies	
- Maintenance:	Medium
- Reliability with Kubernetes :	Good

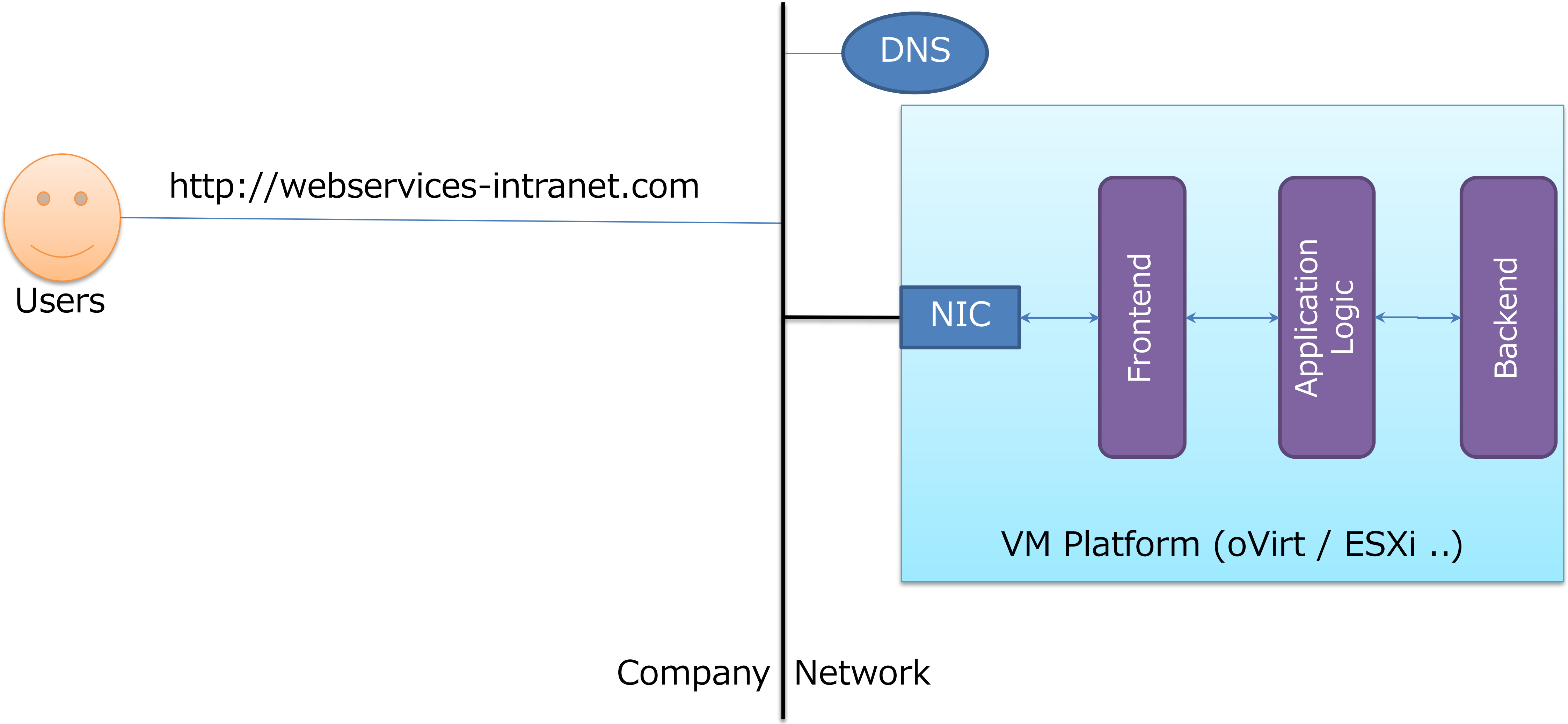


Lesson learnt

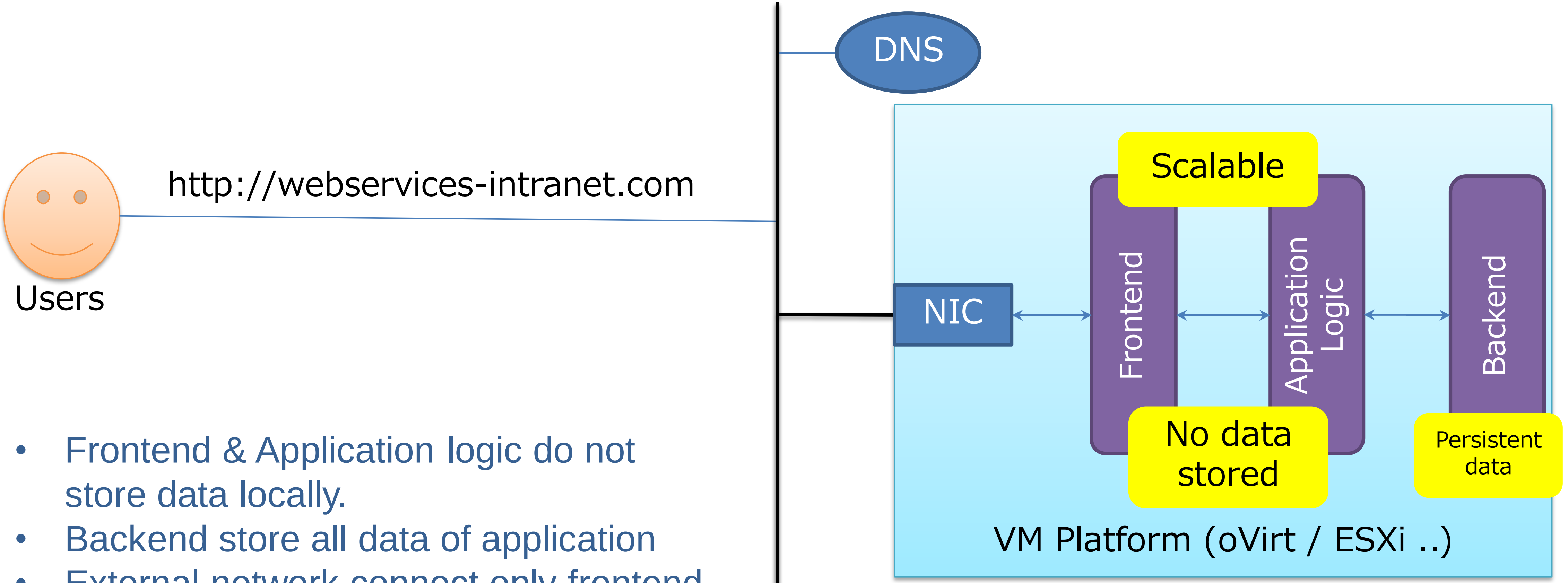
- VM maintenance changes w.r.t. Kubernetes.
- Be expert in Kubernetes.

3-Tire Web Application

3 Tier Web Application



3 Tier Web Application



- Frontend & Application logic do not store data locally.
- Backend store all data of application
- External network connect only frontend

Company Network

NTT

3 Tier Web Application

DNS

U

Application Type	3 tier web architecture.
Requirements	• Application and Frontend should be scalable. • Persistent Storage for Backend • Networking • Inter-VM & external communication • Volume Backup
Policies	• Auto re-create of Application & Frontend VM • No auto re-creation of VM for Backend • Health Check

NTT

Migration process: Image Migration

- ContainerDisk type suites better for immutable application types.
  - Extra temporary storage can be provided using EmptyDisk type.
- PersistentVolume(PV) for storing persistent data in application.

- Frontend and Application VM imported as ContainerDisk

- ContainerDisk is created using Dockerfile with special Base Image provided by KubeVirt.

```
$ cat Dockerfile
FROM kubevirt/container-disk-v1alpha
ADD frontend-disk.qcow2 /disk

$ docker build –t kunalkushwaha/frontend-disk:v1
```

Migration process: VM Definition

Frontend and Application logic are created as VMReplicaSet

- To make Frontend and Application scalable, defined as VMReplicaSet(VMRS).
- Though VMs created using ContainerDisk are not compatible with live-migration.
- Data/Configuration can be passed to application in VM using cloudInit or ConfigMap during VM creation.

```yaml
spec:
   replica: 1
   devices:
          disks:
          - disk:
            bus: virtio
            name: containerdisk
          - disk:
            bus: virtio
            name: configdisk
   volumes:
       - name: containerdisk
         containerDisk:
           image: kunalkushwaha/frontend-vm-disk:v1
       - name: configdisk
         cloudInitNoCloud:
           userDataBase64: $(cat app-scripts.sh | base64 -w0)
.
```

Sample VMReplicaSet definition

Migration process: Service Definition

- Hostname of old topology system becomes service name
- Frontend exposed with NodePort
- Application and Backend as ClusterIP (accessed within Cluster)

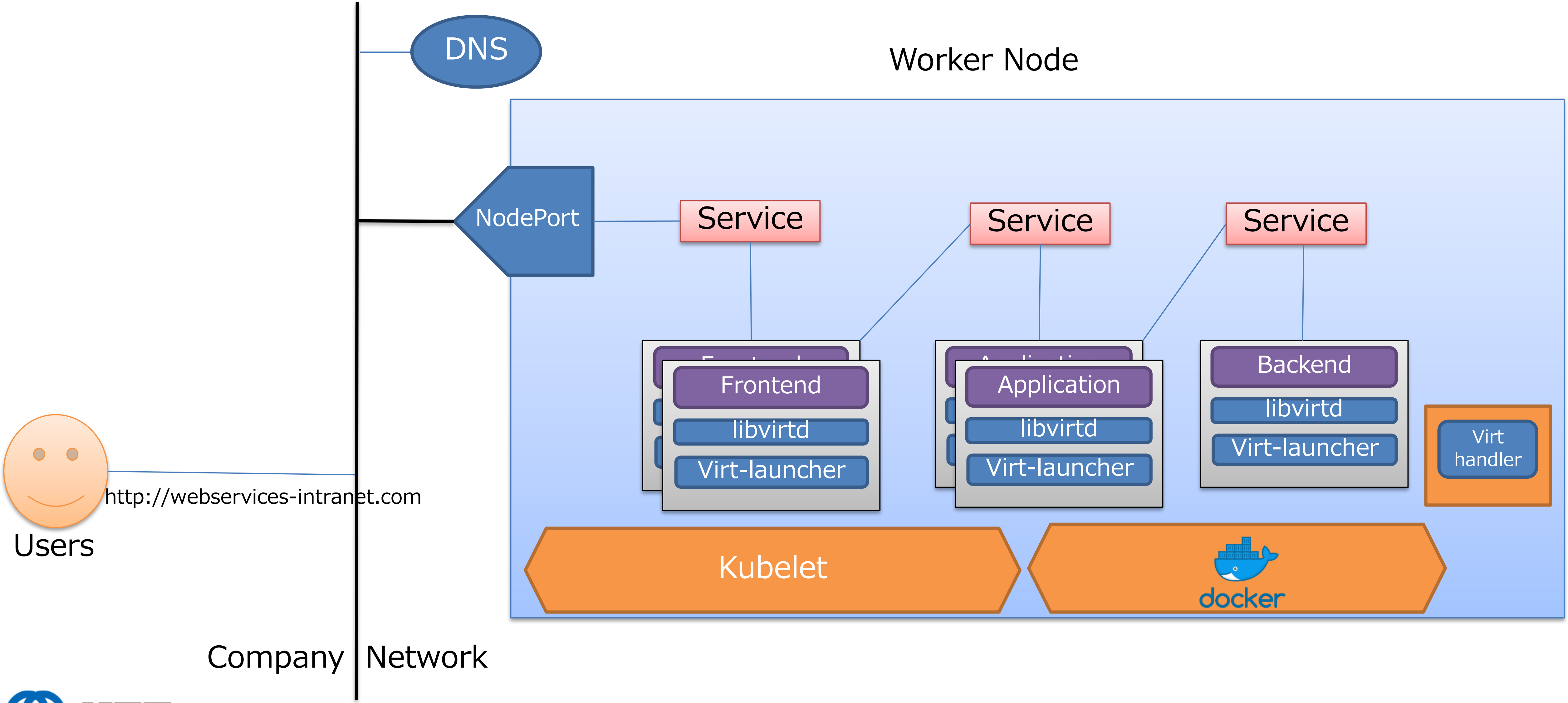
Frontend

```
apiVersion: v1
kind: Service
metadata:
  name: web-server
spec:
  ports:
  - name: nodeport
    nodePort: 30002
    port: 27019
    targetPort: 80
  selector:
    kubevirt.io/vm: frontend-app
  type: NodePort
```

Application & Backend

```
apiVersion: v1
kind: Service
metadata:
  name: application-server
spec:
  ports:
  - name: clusterip
    port: 27021
    targetPort: 80
  selector:
    kubevirt.io/vm: application-app
  type: ClusterIP
```

After Migration: 3 Tier Web Application



Migration process: Maintenance

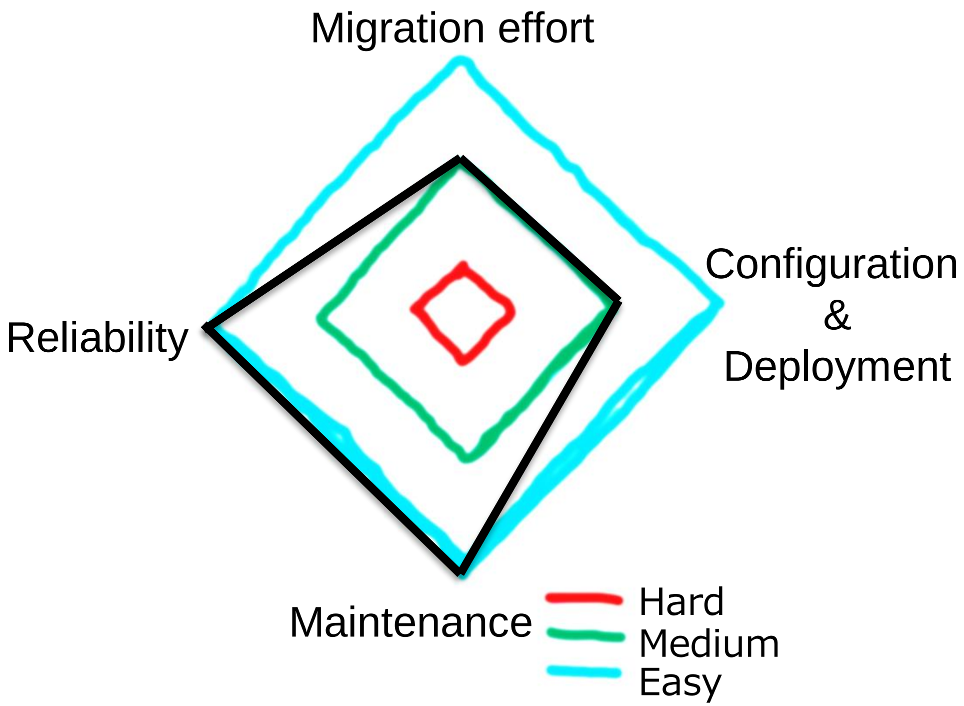
VMReplicaSet are easy to scale, same as Pod replicaset, But no rolling updates supported.

- Blue-Green deployment for updating immutable VMs outside of KubeVirt.
  - Scale with updated image.
  - Delete old image instances
  - Scale down
- Use traditional approach for updating Stateful VM instances.
  - ssh, config management

Conclusion: 3 Tier Web Application

- Maintaining & scaling stateless VMs becomes very easy.

- Migration process : Medium
- Online migration : No
- Maintenance : Good
- Reliability with Kubernetes : Good



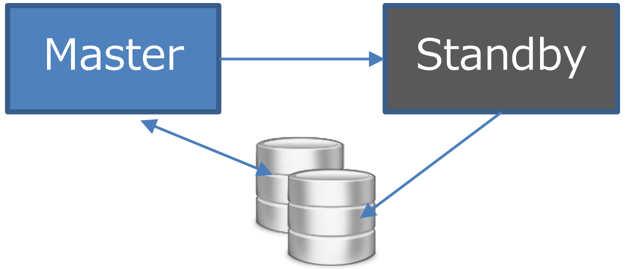
Lesson learnt

- Name resolution/ Fixed IP reference in application config, do not work.
- Hostname of VMs will be services of VM instance.
- Be expert in Kubernetes.

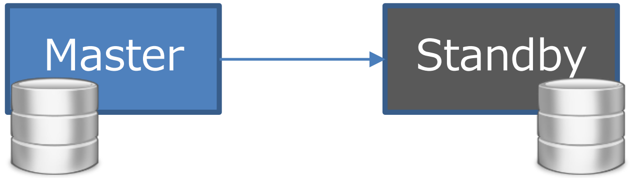
HA Architecture

HA Architecture Patterns

- Active-Standby with Shared Disk

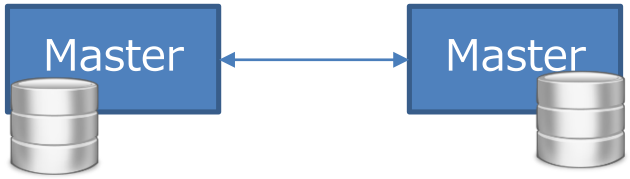


- Active-Standby with Shared nothing

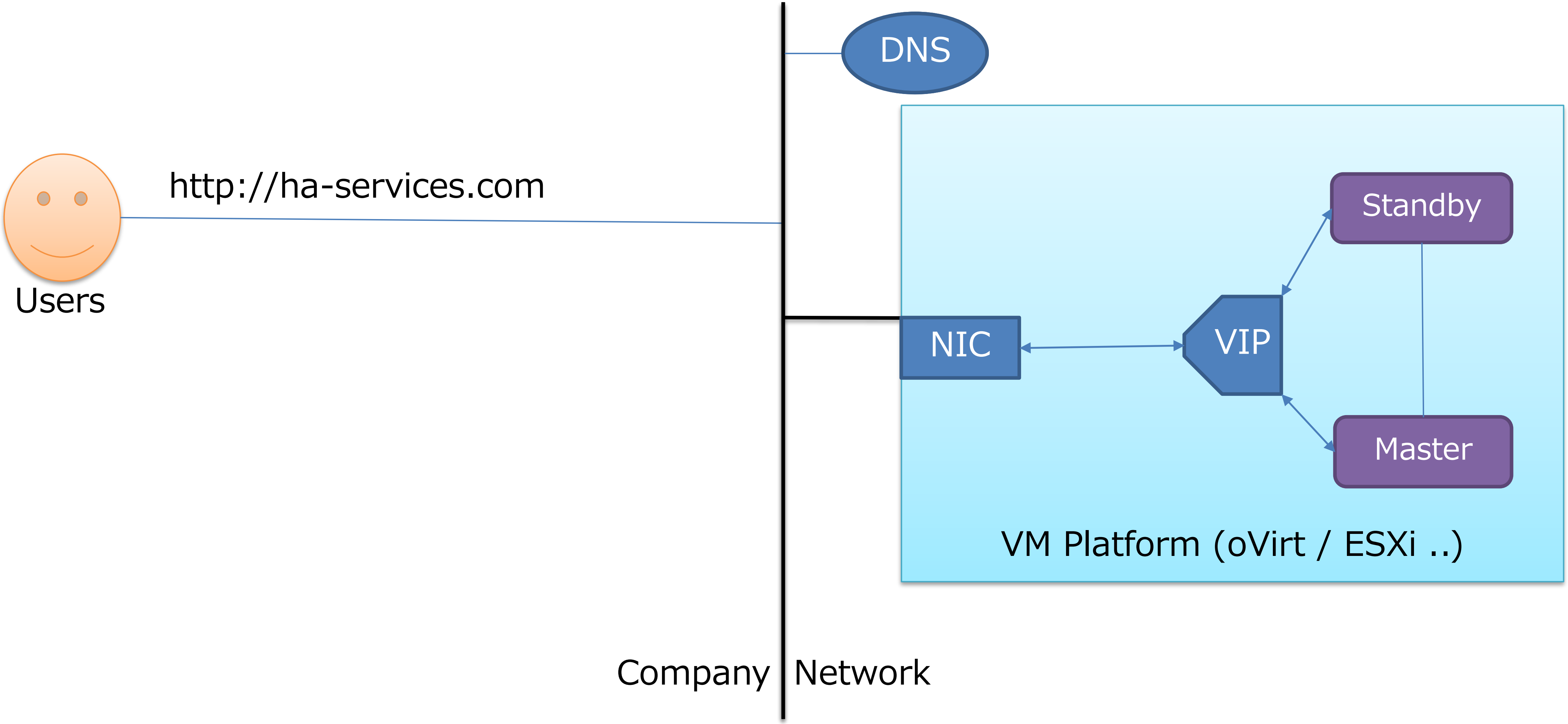


- Active-Active with Shared nothing*

*Please see appendices



HA Architecture (Active-Standby)

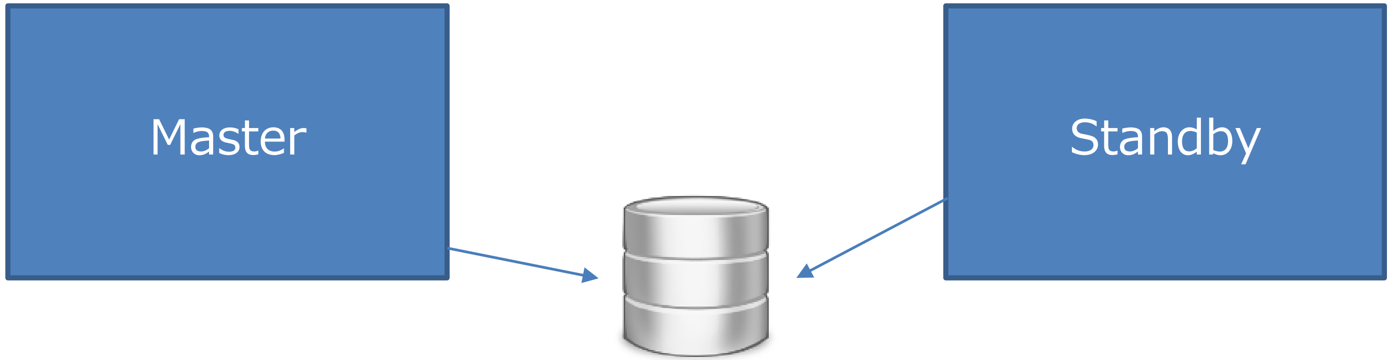


Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)



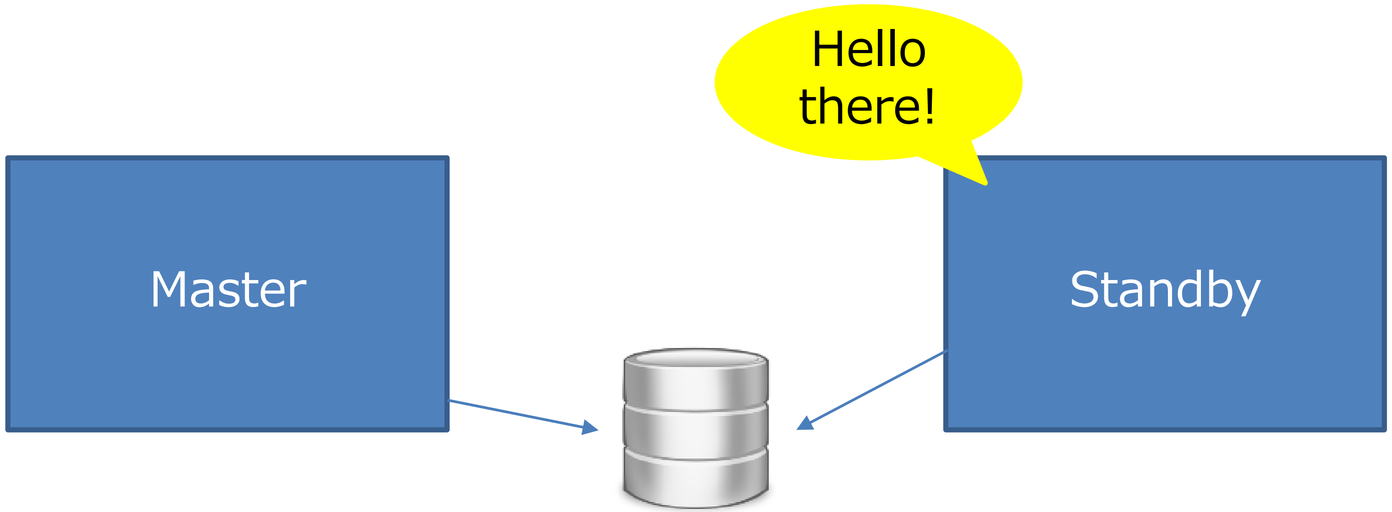
NTT

Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)

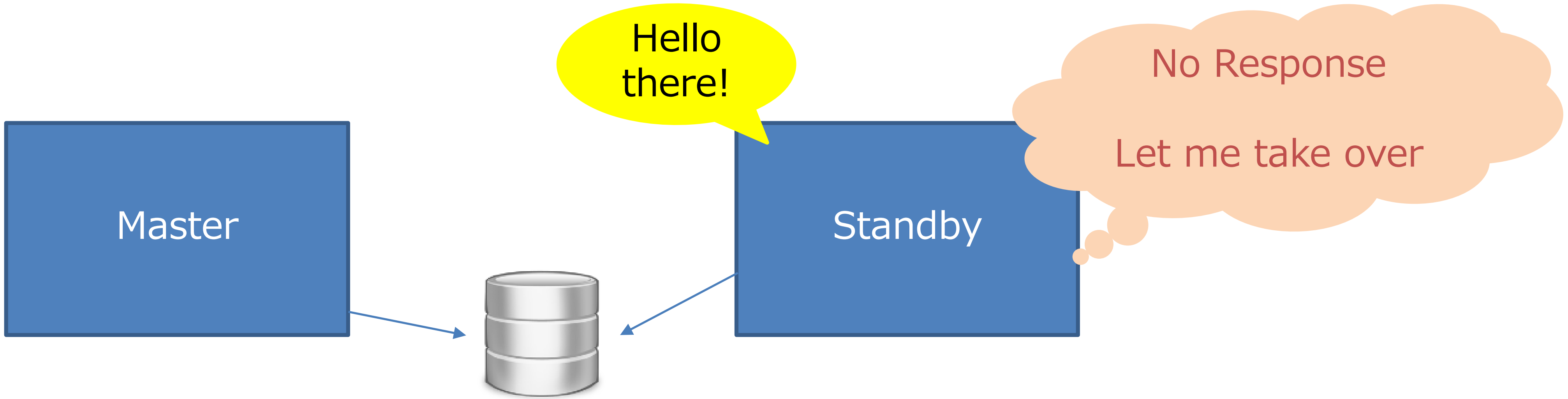


Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)

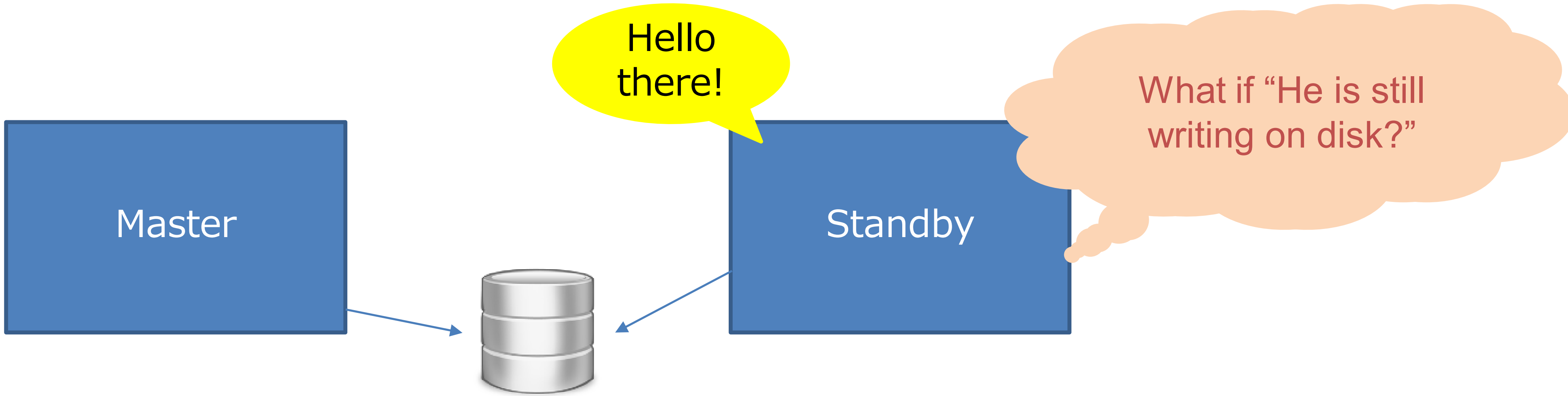


Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)

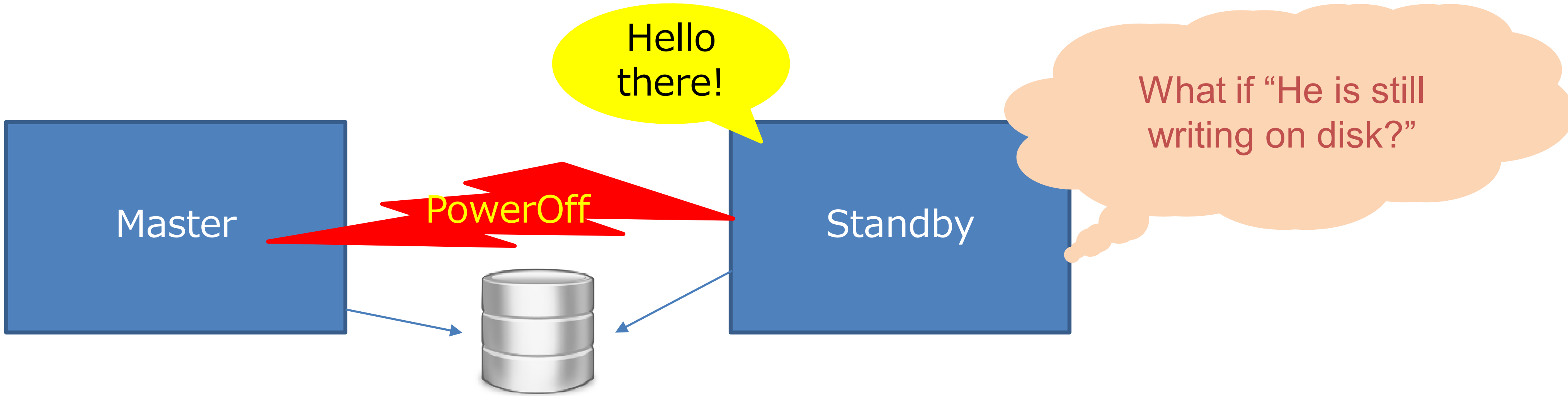


Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)



Active-Standby with Shared Disk

- Data consistency is hard to achieve with this architecture in KubeVirt /Kubernetes
- Fencing mechanism like STONITH, not available in Kubernetes/KubeVirt yet.

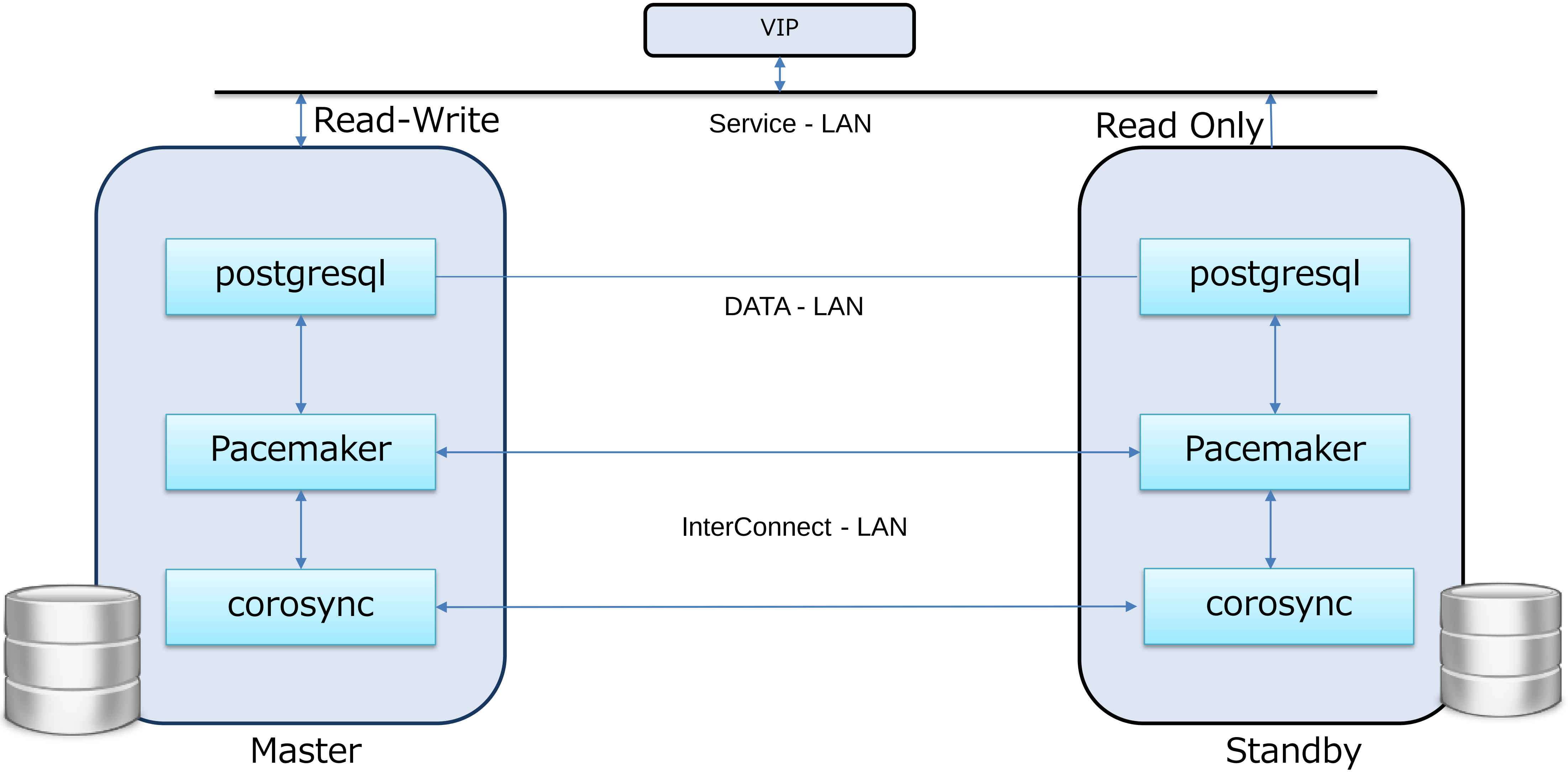
When one node become unresponsive. How it can be ensured if it is not updating disk / Corrupting data?

- Shoot The Other Node In The Head (STONITH)

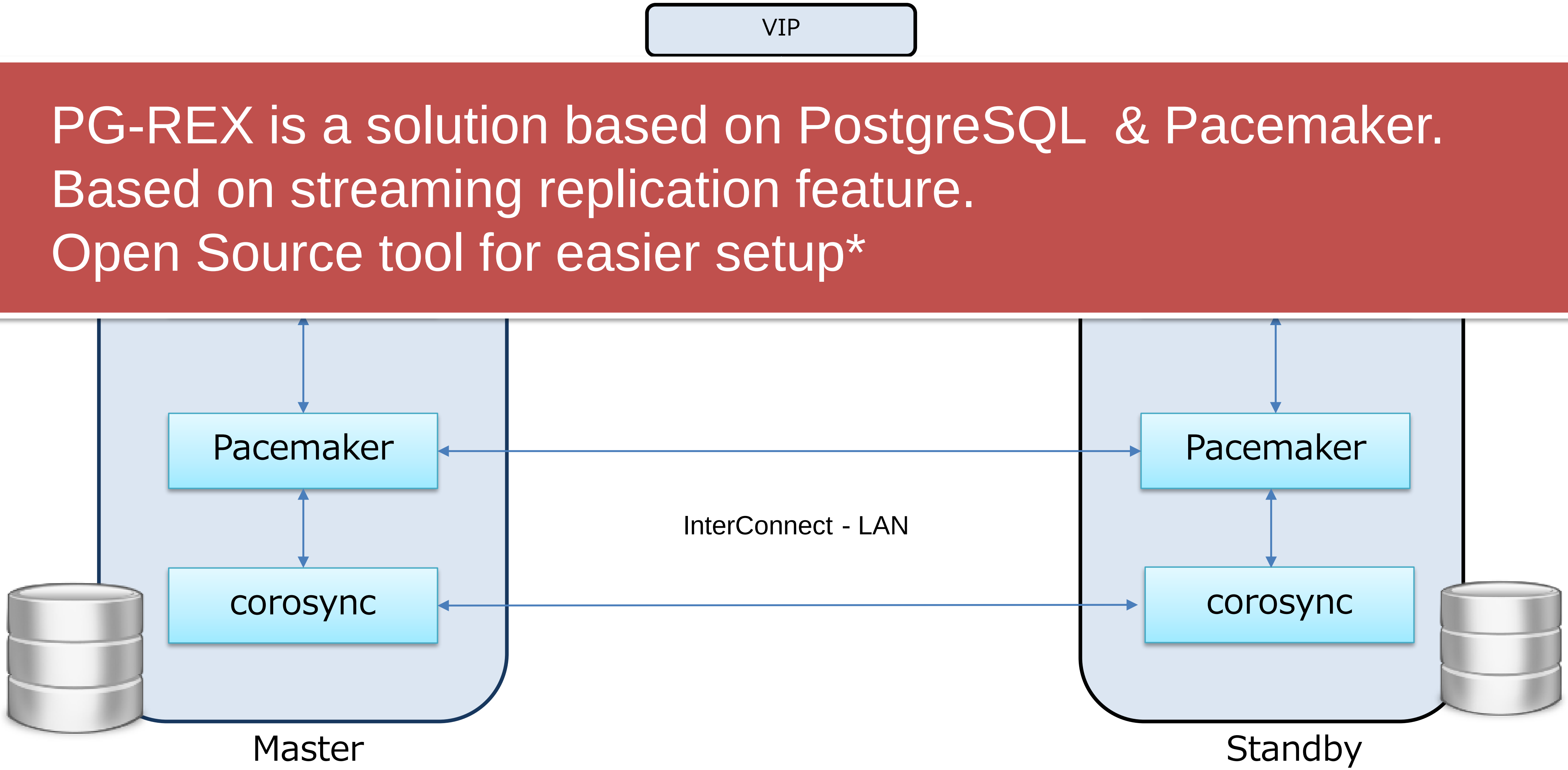
Hello

Lack of fencing mechanism, restrict migration of applications implemented with STONITH like solution

Active-Standby without Shared Disk



Active-Standby without Shared Disk



- PG-REX is a solution based on PostgreSQL & Pacemaker.
- Based on streaming replication feature.
- Open Source tool for easier setup*

*https://osdn.net/projects/pg-rex/

Migration process: VM Definition of HA models

- Multus (a meta CNI plugin) used for providing multiple network interfaces to VMs of KubeVirt.

- Uses NetworkAttachment (CNI CRD) for implementing multiple networks.

- Apart from Persistent Volume, this use case requires multiple Network segments.

- Preparation of network is required before using them in VM Definition i.e. defining NetworkAttchmentDefinition.

Migration process: VM Definition of HA models

- Multus (a meta CNI plugin) used for providing multiple network interfaces to VMs of KubeVirt.

- Uses NetworkAttachment (CNI CRD) for implementing multiple networks.

- Apart from Persistent Volume, this use cas

- Preparation of network is required before u NetworkAttchmentDefinition.

pgrex-s-lan.yaml

```
apiVersion: v1
kind: NetworkAttachmentDefinition
metadata:
  name: pgrex-s-lan
spec:
  config:'{
           "cniVersion": "0.3.0",
           "type": "macvlan",
           "master":"enp0s8",
            .
            .
            . < CNI plugin >
            .
            .
           }'
```

Migration process: VM Definition for PG-REX

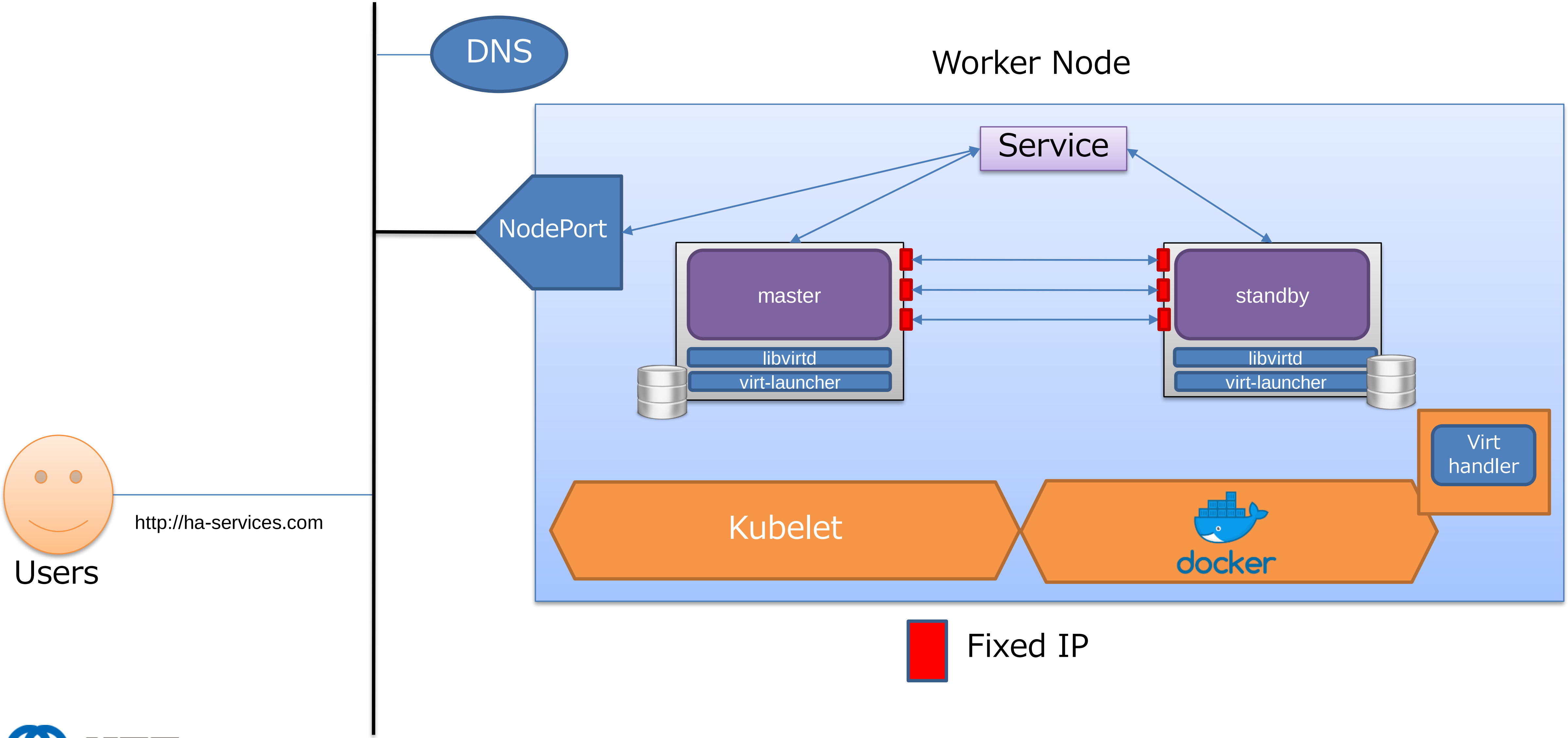
- With private hosted Kubernetes, its hard to get Fixed IP over cluster.
    - Service cannot have custom ClusterIP in different segment.
- Migration in KubeVirt is possible with hackish solution.

- Works on fixed IP address, *but troubleshooting is hard*.
- Using macvlan network, network with narrow range of IP is be created for all segments.
- HA components communicate with VM IP's instead of services.
- Extra logic required to ensure user request goes to Master VM only.*
- Need reconfiguration, if VM's moved from current node.

*Leader election to mark Active VM. (https://kubernetes.io/blog/2016/01/simple-leader-election-with-kubernetes/)

NTT

After Migration: Active-Standby without Shared Disk



Migration process: Maintenance

Maintenance approach of Application VMs do not change much, though little added complexity in connecting the VMs

- Backup/snapshot management.
  - PersistentVolume (PV) is provided by K8s storage providers.
  - Managed in similar way as PersistentVolume of K8s.

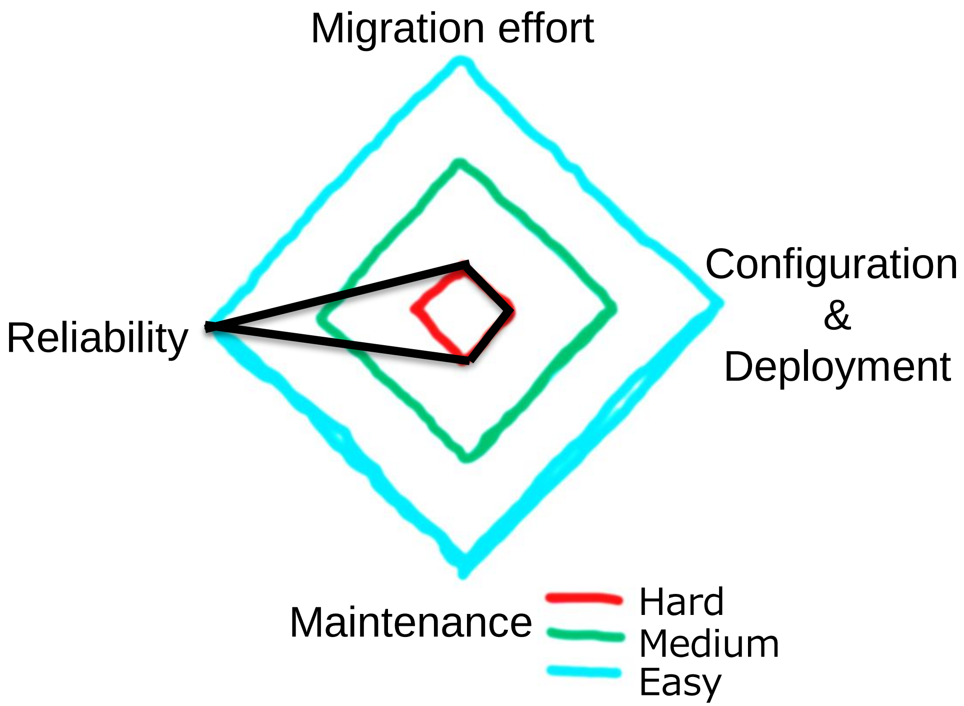
- Patch management/VM upgrade
  - Traditional way (ssh / config manager)

- On failure
  - Application logic of smooth failover works.

Conclusion: HA Architecture

Simply Lift & Shift do not work for application with complex topology

- Migration process : Hard
- Online migration : No
- Maintenance : Medium
- Reliability with Kubernetes : Good



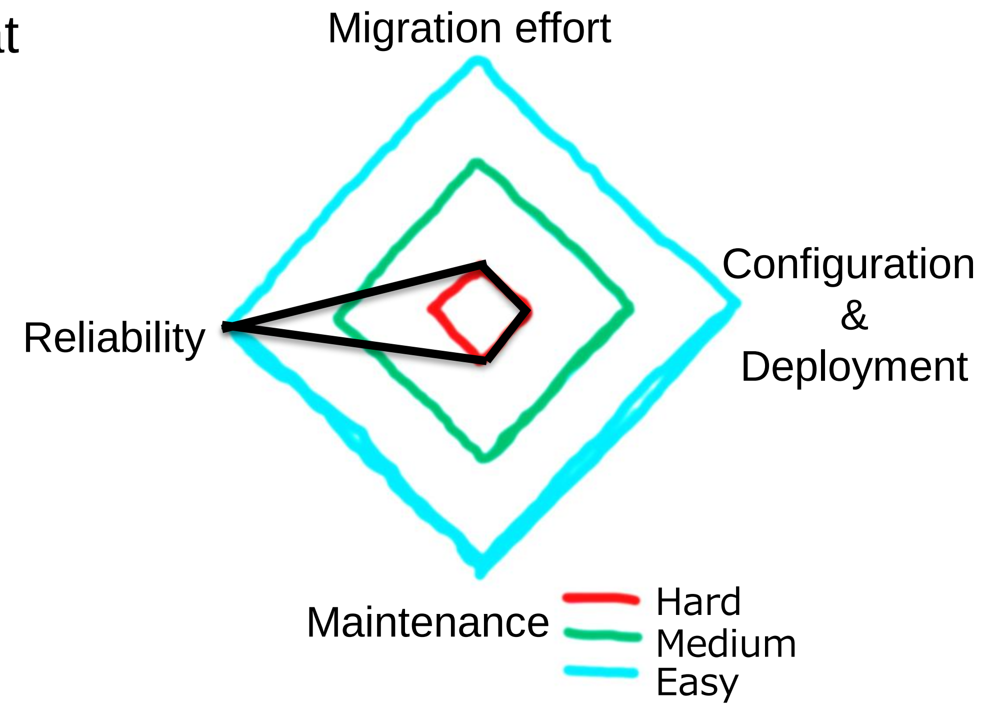
Lesson learnt

- Configuration changes are not apparent.
- Look beyond standard Kubernetes pod communication channels
- Be expert in Kubernetes.

Conclusion: HA Architecture (cont'd)

No perfect solution for migrating DB VMs to Kubernetes.

- Migrating shared disk DB Cluster might not be wise at this moment.
  - Data consistency need to be maintained by application only.
- Particularly for DB, shared nothing kind of configuration there are few solution which works on KubeVirt like environment.
  - PG-REX
    - Works with hack
  - Crunchy
    - A Kubernetes Operator based PostgreSQL solution.
    - Not for migrating existing DB nodes.



Overall Conclusion

- KubeVirt works including multiple networks.

- Migration steps can be automated for VM Definition;
  But IP addresses aren’t portable.

- HA is currently tough; it requires non-standard(*hackish*) configuration.

- Challenges
  - Reliable fencing mechanism
  - Support for service IP other than default network segment

- Future work
  - VM Definition generator from old VM configuration e.g. OVA file.

- Virtlet
  - Project with similar goal, but implemented as Container Runtime Interface(CRI) instead of CRD.
  - KubeVirt is more active project compared to Virtlet.

- Kata Container runtime?
  - Not an alternative.
  - Though it uses VM level isolation, but designed to run docker/container type workload (Single application)

Running Legacy VM’s along with containers in Kubernetes

Delusion or Reality?

- Yes, it is possible in near future.
- It will not be simple Lift & Shift, but shall be less expensive than rewriting or restructuring in containers.
- Automating migration will be daunting task.
  - Application specific details are unique
  - Kubernetes/KubeVirt specific changes could be automated with some declarative objects.

Thank you

Appendices

Evaluation Environment

Kubernetes Master

```
Architecture:    x86_64
Model name:      Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz
Hypervisor :     KVM
Virtualization: full
Kernel:          4.18
OS:              Fedora Server 29
Memory :         4GB
```

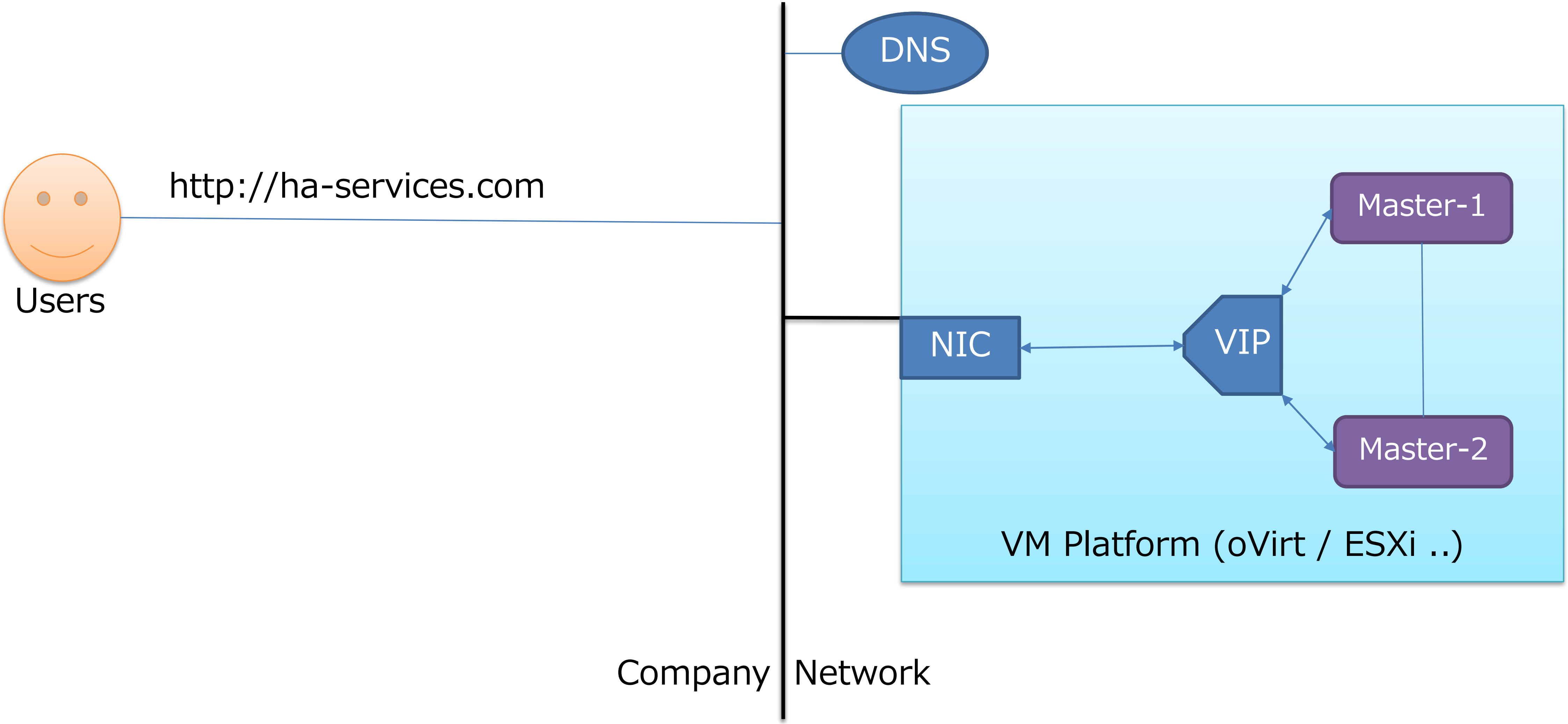
Kubernetes Worker Node x 2

```
Architecture:    x86_64
Model name:      Intel(R) Xeon(R) W-2123 CPU @ 3.60GHz
Hypervisor :     KVM
Virtualization: full
Kernel:          4.18
OS:              Fedora Server 29
Memory :         12GB
```

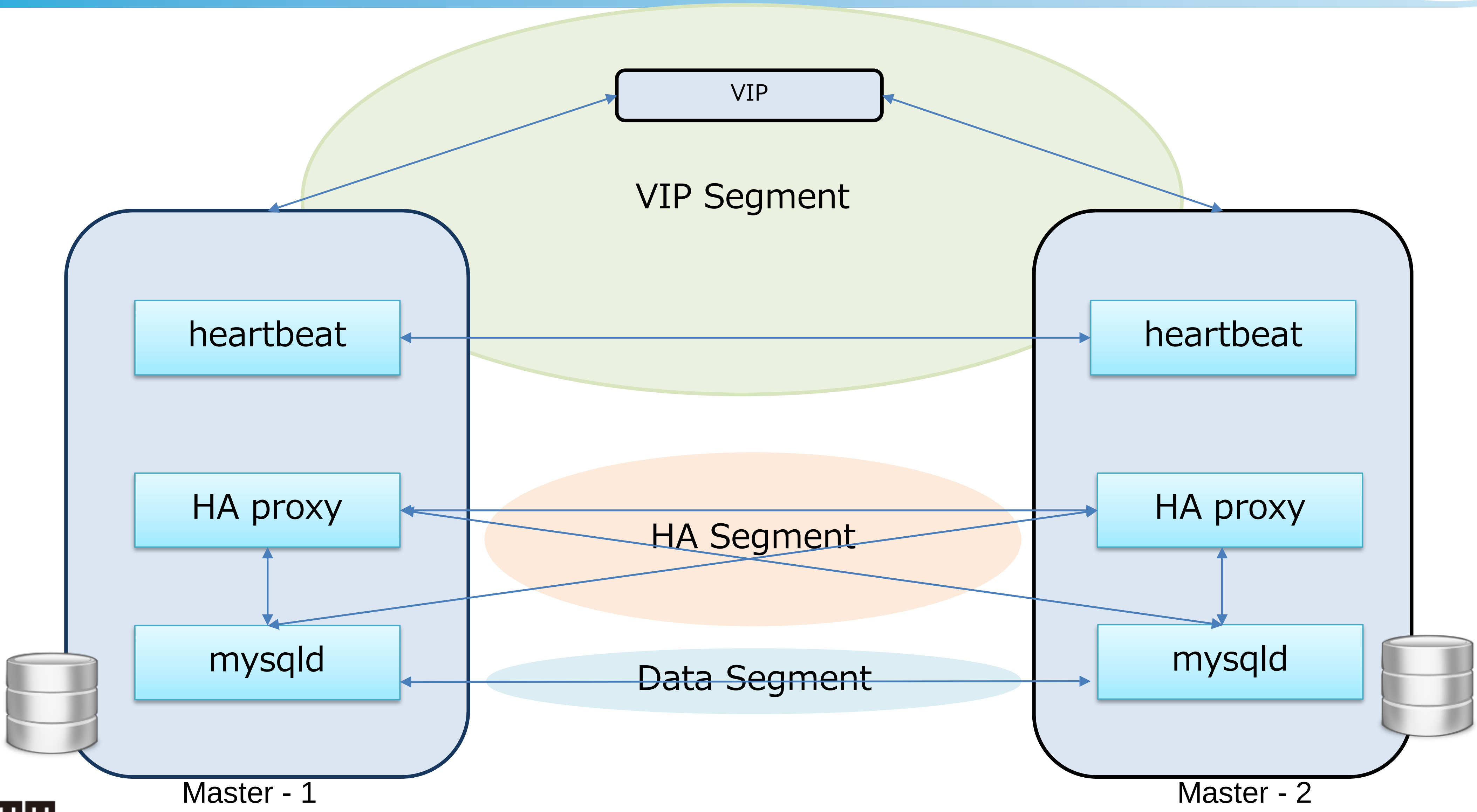
Software version

```
Kubernetes version : v1.12.2
KubeVirt Version   : v0.17.0
CDI version        : v1.9.0
```

HA Architecture (Active-Active without Shared Disk)



Active-Active without Shared Disk



Migration Process: VM Definition for MySQL Active-Active

- Defining multiple network VMs is same as pods using meta CNI plugins like multus.
- Using cloudInit, its easy to make and try changes in application configuration

- Define network for each segment.
- Define ports for each segment too.

Migration Process: VM Definition for MySQL Active-Active

- Defining multiple network VMs … ugins like multus.
- Using cloudInit, its easy to ma … nfiguration

- Define network for each segm
- Define ports for each segment

VM.yaml

```yaml
    interfaces:
    -  bridge:{}
       name: default
    -  bridge {}
       name: green-net
       ports:
       - name: heartbeat
         port: 694
    -  bridge: {}
       name: orange-net
       ports:
      .
      .
      .

networks:
    - name: default
      pod:{}
    - multus:
        networkName: green-network
        name: green-net
```

Migration Process: Service Definition

- Configuration changes required in original VM
  - e.g. Bind of host instead of specific interface (IP)
  - Firewall rules requires to be updated
- Changes makes VM less secure.

- Traditionally application services are bind to particular NIC.
  - These setting required to bind on hostname (or all NICs e.g. 0.0.0.0)
- Firewall rules need to ease out the restriction as static network is missing.
  - These security settings move out of VM i.e. Network Policy for k8s.

After Migration: Active-Active without Shared Disk