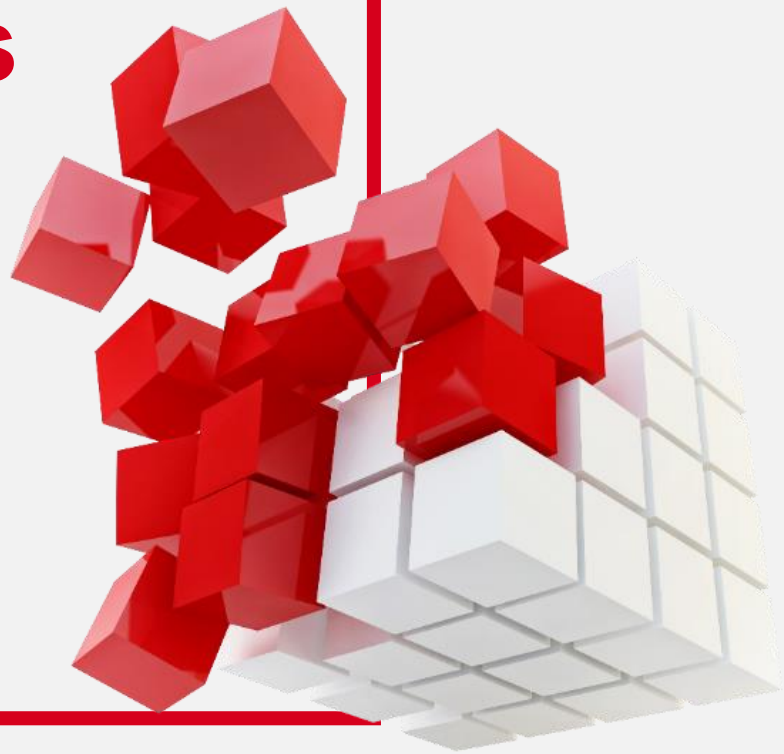
Kubernetes for DevOps



Nguyen Canh Hat

June 2023

Nash Tech.

Agenda

1. **Kubernetes Overview**
   - Benefits of Kubernetes
   - What is Kubernetes?
2. **Kubernetes Architecture**
3. **Primary Kubernetes Objects**
   - Deployment, StatefulSet, DaemonSet
   - Probes, Deployment Strategies
   - Resource Request and Limit
   - CronJob
   - Network
   - Configuration data
   - Persistent Storage
4. **Deployment Strategies**
5. **Kubernetes User Interaction**
   - K8s CLI
   - External tool

Kubernetes Overview

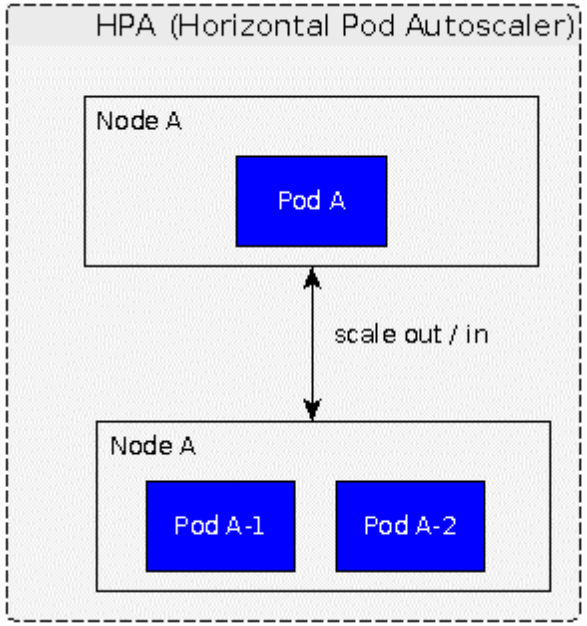
Benefits of Kubernetes – High Availability (1/3)

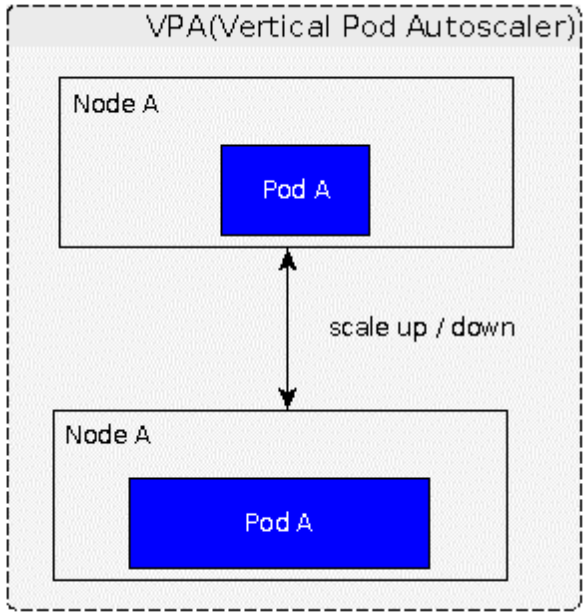
Docker

- NOT able to manage auto-scaling itself

Kubernetes – auto scaling

- Provide HPA, VPA, CA





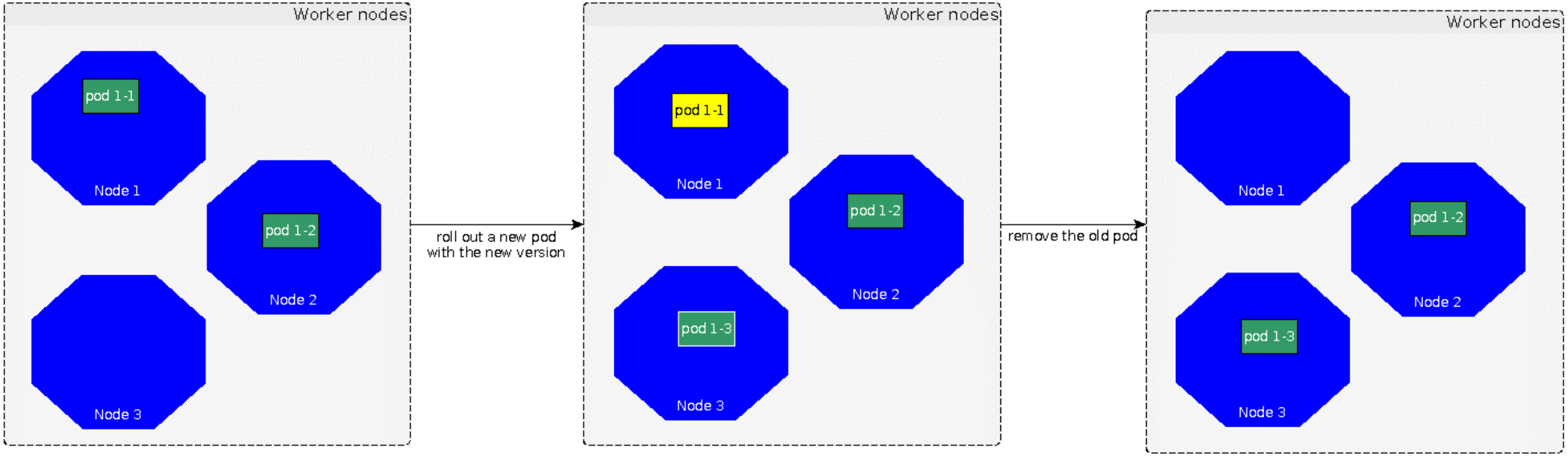
Benefits of Kubernetes – High Availability (2/3)

Docker

- Introduce a downtime where there is an update

Kubernetes – zero downtime deployment

- Deployment strategies: rolling update, recreate



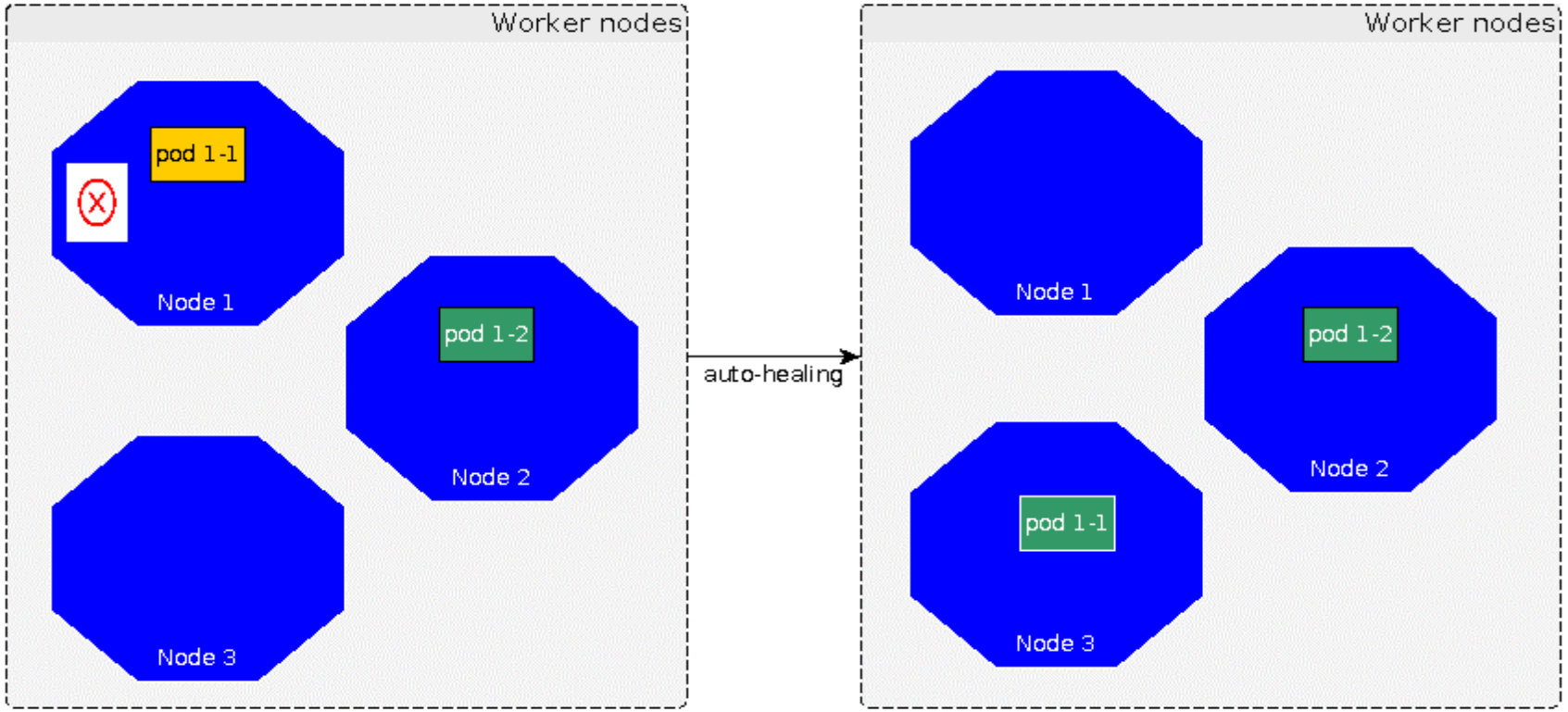
Benefits of Kubernetes – High Availability (3/3)

Docker

- NOT able to get failed applications back

Kubernetes – auto healing capability

- Probes: detect failure and recover applications



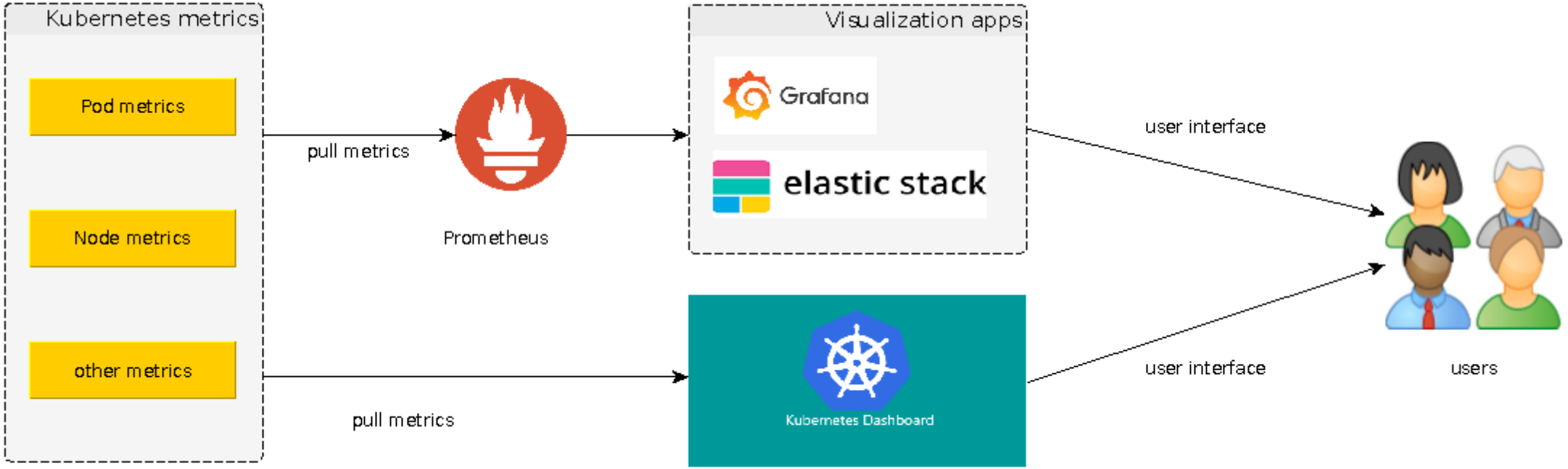
Benefits of Kubernetes – Monitoring

Docker

- Challenges of setting up monitoring tools

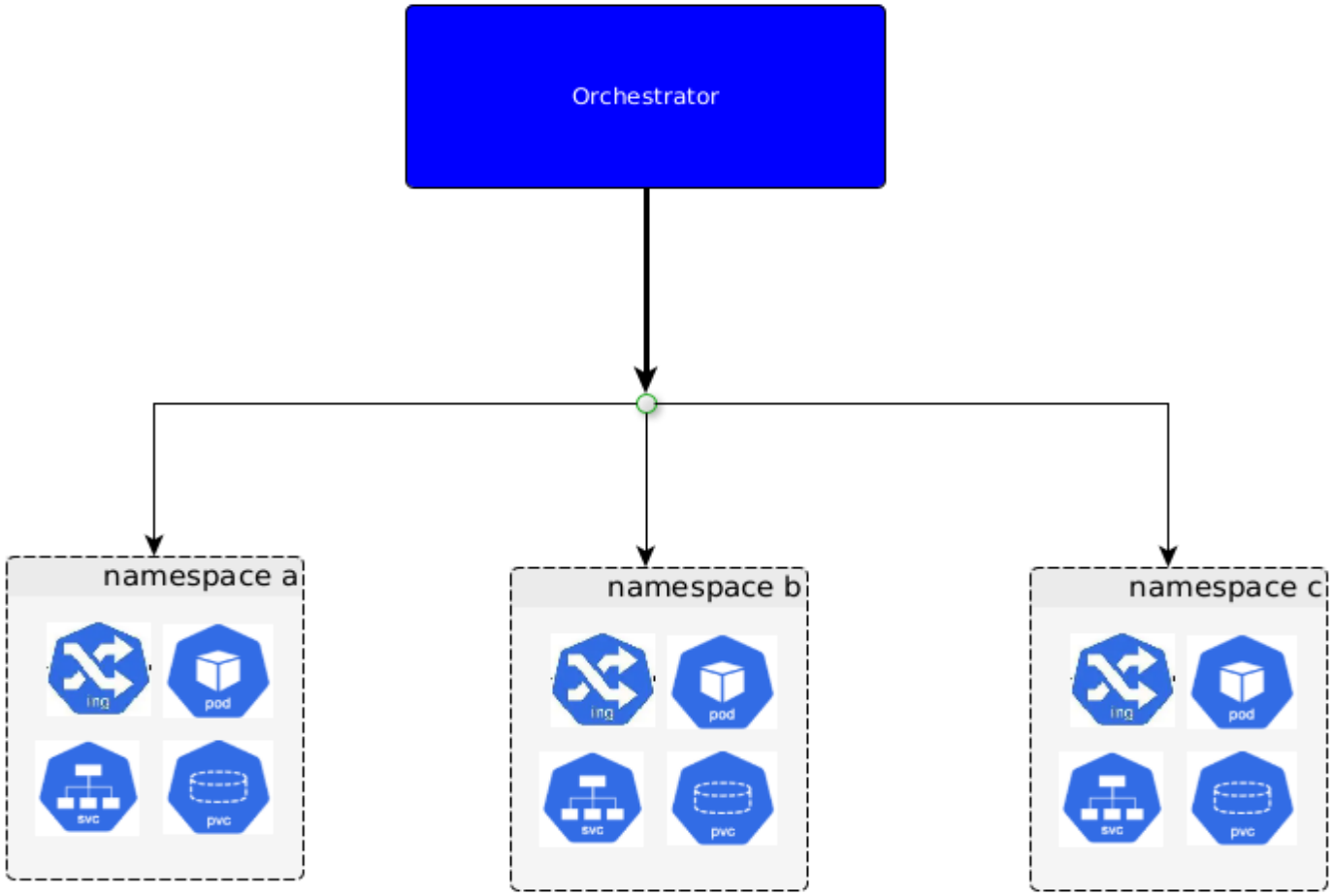
Kubernetes – Kubernetes APIs

- Provide metrics and built in functions to integrate with 3rd tools



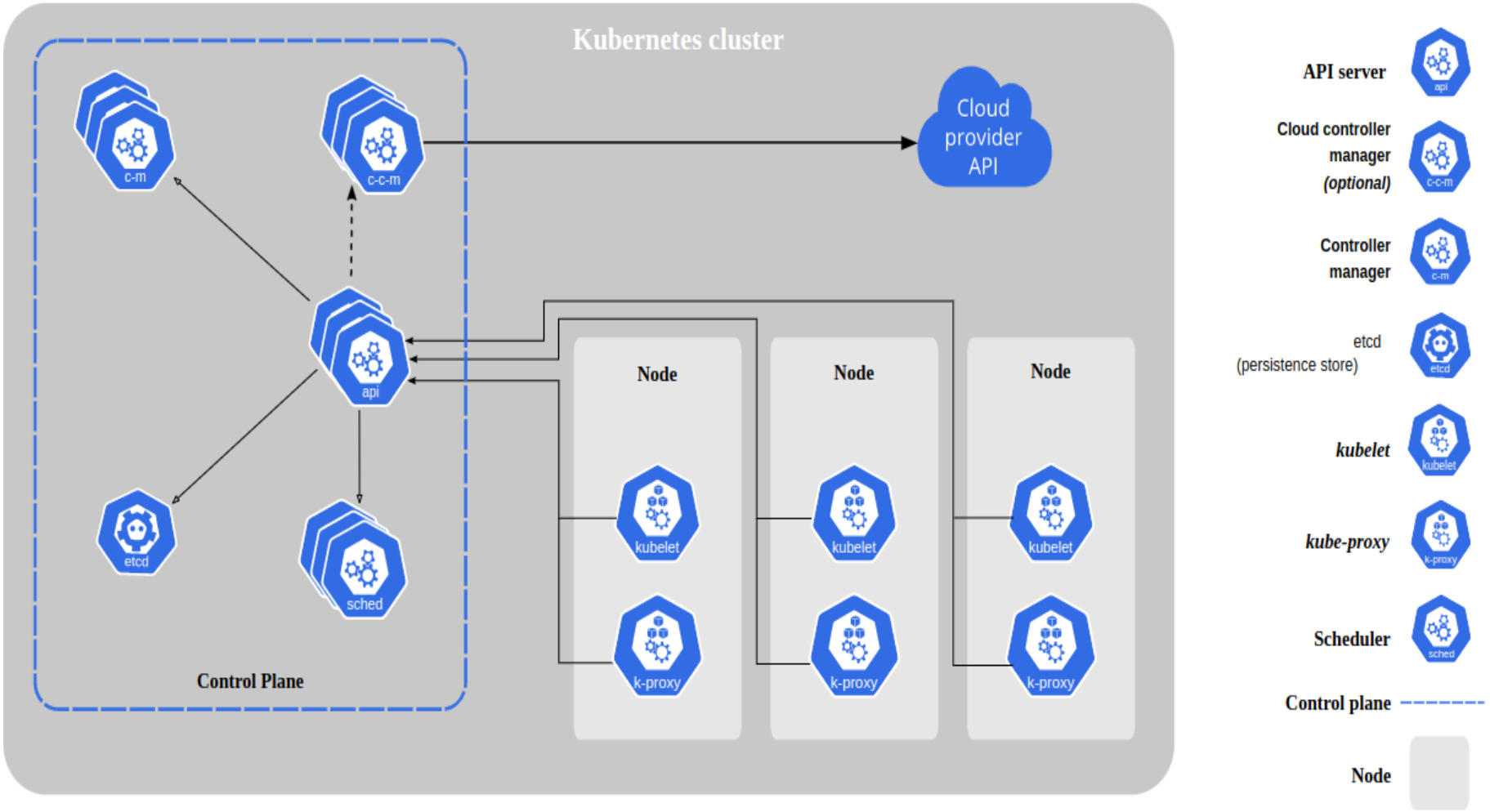
What is Kubernetes?

- Found by Google
- Open source
- Container orchestrator
- Automate manual processes (deployment, scaling, management of containerized applications



Kubernetes Architecture

- Control plane (master)
- Nodes (worker)

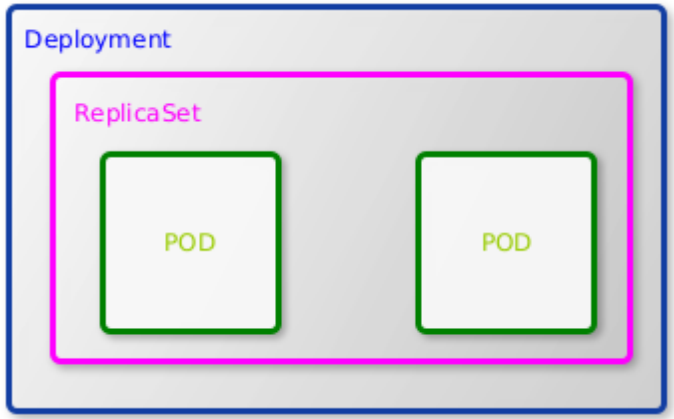


02

Primary Kubernetes Objects

Deployment

- A declarative template for pods and replicaSet
- Used for stateless application




```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

StatefulSet

Similar to “deployment” but:

- Used for stateful application
- There are 3 sections that can be updated on the fly:
  - replicas
  - template
  - updateStrategy
- Ordered pod creation/deletion
- Pod can be accessed by its own service

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: postgres-sts
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgresql
          image: postgresql:latest
          imagePullPolicy: "IfNotPresent"
          env:
            - name: POSTGRES_USER
              value: "postgres"
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: postgresql-password
            - name: POSTGRES_DB
              value: "postgres"
          ports:
            - name: postgresql
              containerPort: 5432
              protocol: TCP
          volumeMounts:
            - name: data-vct
              mountPath: /var/lib/postgresql/data
```

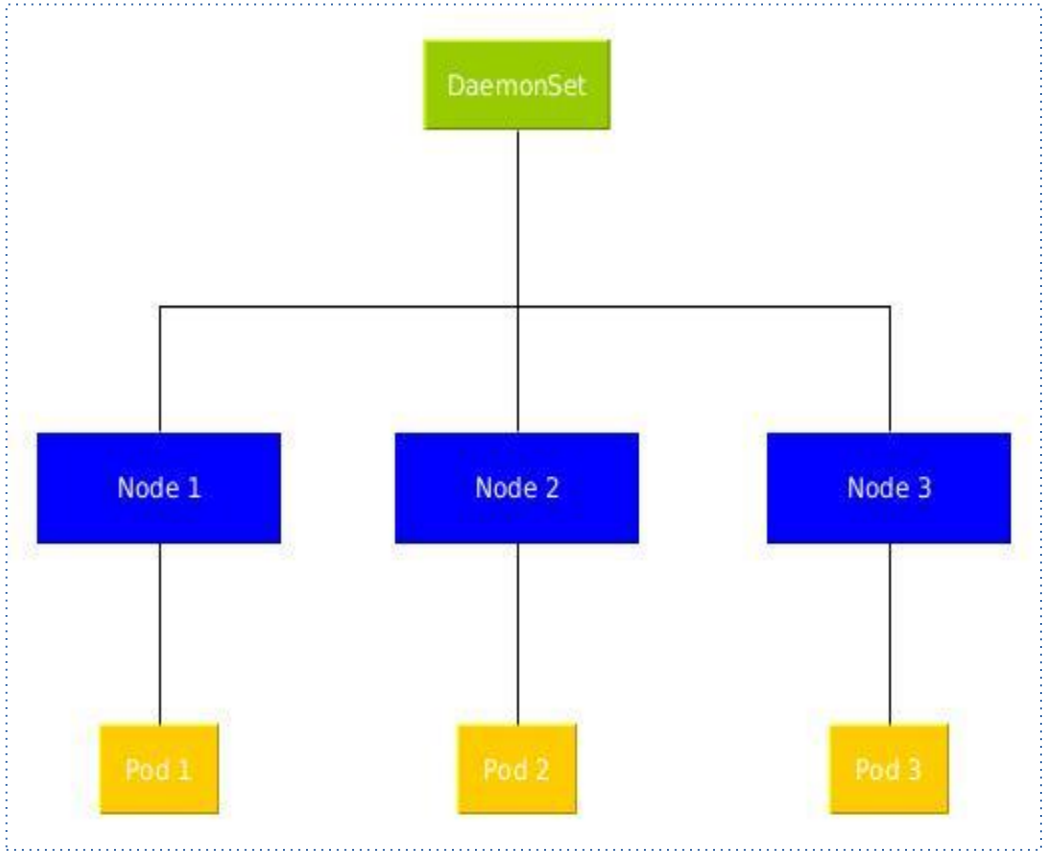
DeamonSet

Similar to “deployment” but:

- Every node has one pod (master & worker)
- Each Pod lifetime depends on its node

Use cases:

- Install a monitoring agent per node
- Install a log connector on each node



Probes

Probes are health checks that can monitor a container's status and act on it.

- StartupProbes: Indicate when the pod is started.

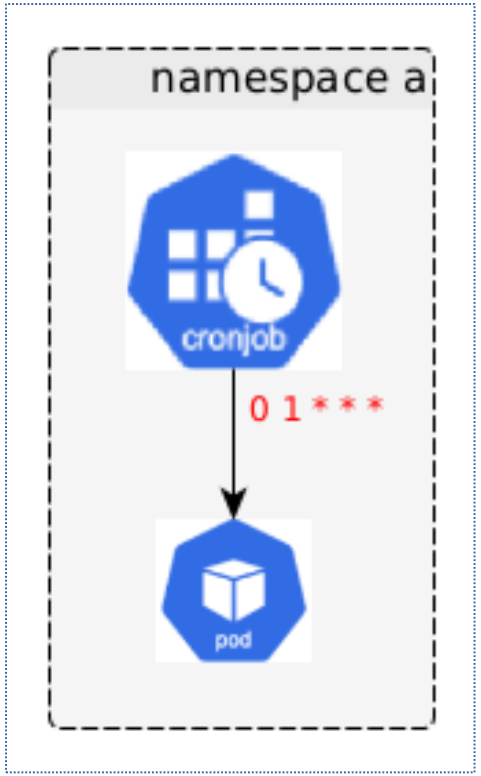
- ReadinessProbes: indicate when to start serving

- LivenessProbes: indicate when to restart the pod

```yaml
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
    livenessProbe:
      httpGet:
        path: /
        port: 80
        httpHeaders:
      initialDelaySeconds: 5
      timeoutSeconds: 1
      periodSeconds: 10
      successThreshold: 1
      failureThreshold: 3
```

Cronjob

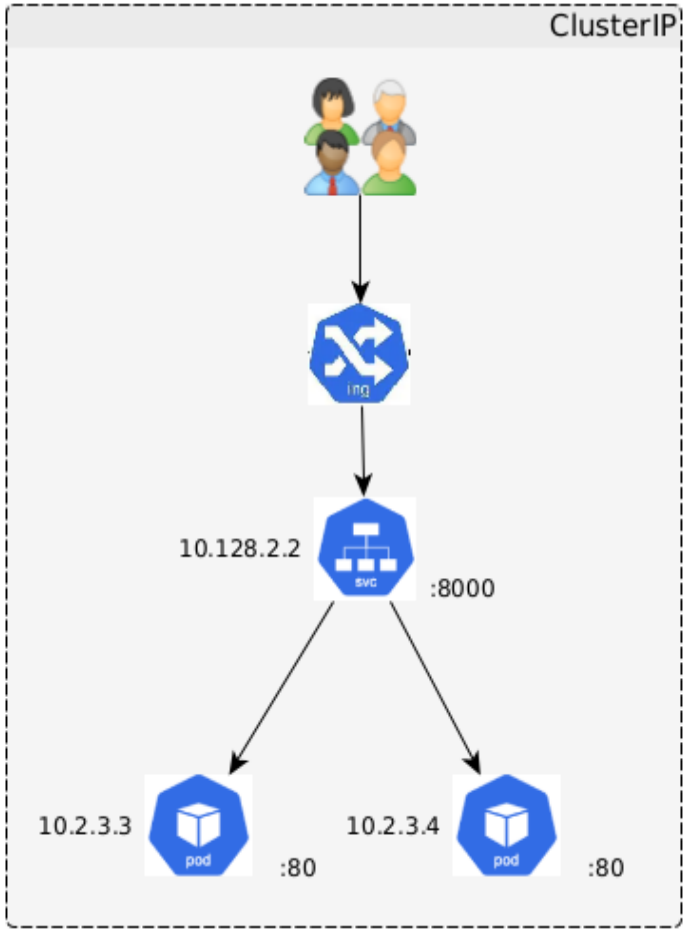
- Automatic action triggered in every scheduled time




```yaml
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello-friends
spec:
  schedule: "0 1 * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox:1.28
            imagePullPolicy: IfNotPresent
            command:
            - /bin/sh
            - -c
            - date; echo Hello friends
          restartPolicy: OnFailure
```

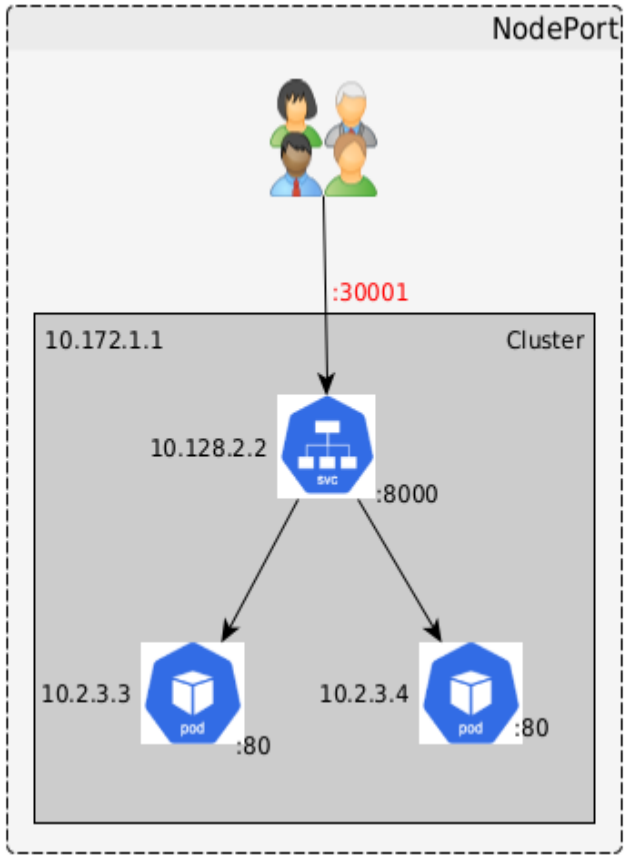
Network – Service (1/2)

- A logical collection of pods
- Roles:
  - Enable network access to pods
  - Load balancing
- Types:
  - ClusterIP: provide internal access inside cluster only
  - NodePort:
    - Provide a unique fixed cluster node port for external access
    - Not recommended due to security (expose access to cluster)
  - LoadBalacer: Integrate with cloud providers for external access

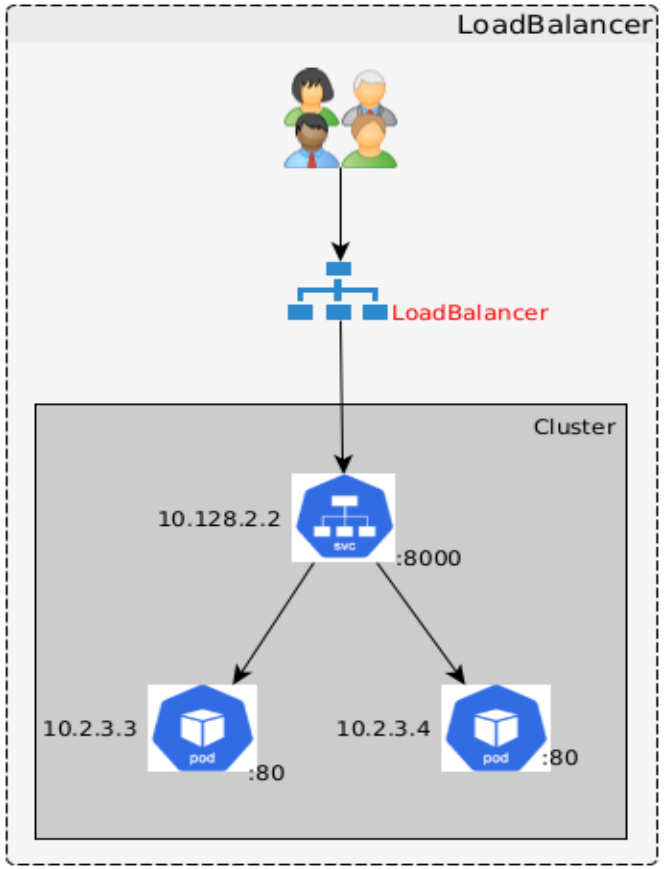



```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
```

Network – Service (2/2)



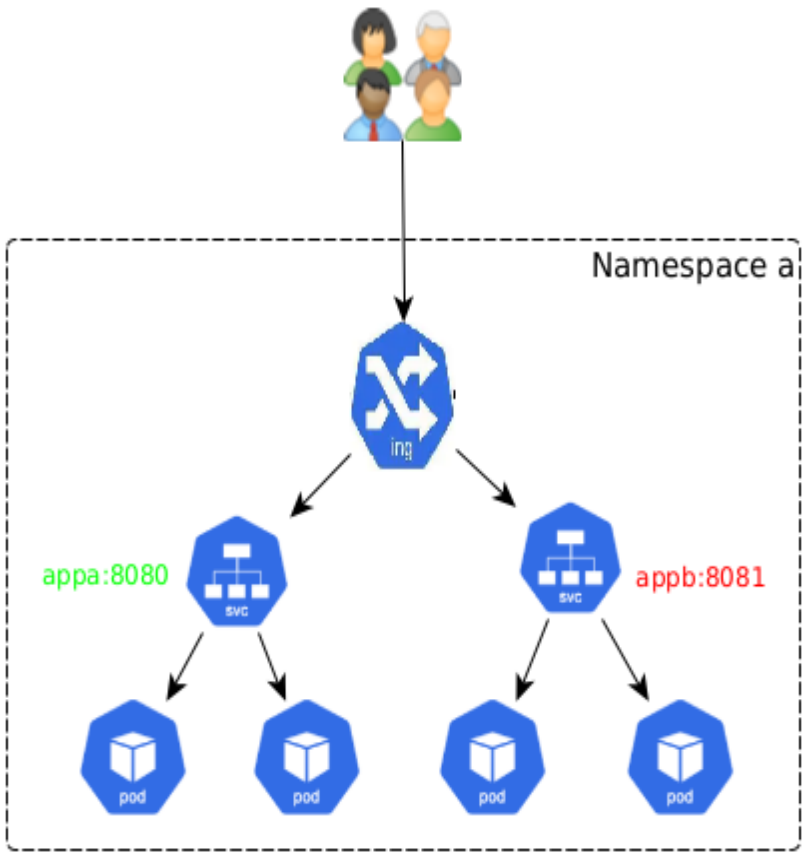

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: nginx
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
    nodePort: 30001
```




```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: LoadBalancer
  selector:
    app: nginx
  ports:
  - port: 8000
    targetPort: 80
    protocol: TCP
```

Network - Ingress

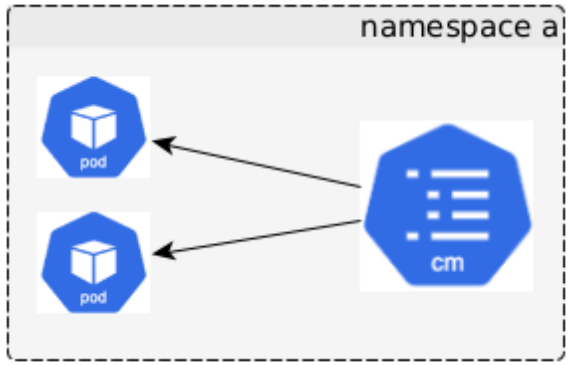
- Provide access to application by providing readable human URL with http/https protocols and routing rules.




```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: app-ingress
  annotations:
    nginx.ingress.kubernetes.io/ssl-redirect: "false"
    nginx.ingress.kubernetes.io/force-ssl-redirect: "false"
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  tls:
  - hosts:
    - abc.com
    secretName: tls-secret
  rules:
  - host: abc.com
    http:
      paths:
        - path: /appa
          backend:
            serviceName: appa-service
            servicePort: 8080
        - path: /appb
          backend:
            serviceName: appb-service
            servicePort: 8081
```

Configuration data - ConfigMap

- Key-value pairs of data
- Manage pods configuration data
  - Environment variables
  - Configuration files
- Used for non-sensitive data (plain text)

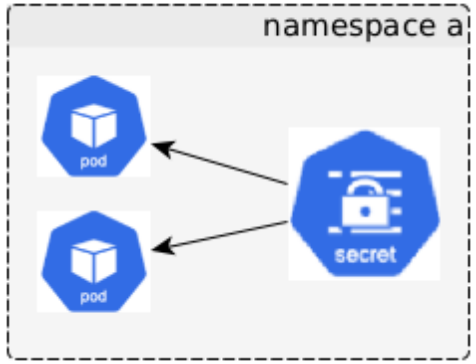



```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
data:
  proxy-connect-timeout: "10s"
  proxy-read-timeout: "10s"
  client-max-body-size: "2m"
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        envFrom:
         - configMapRef:
             name: nginx-config
```

Configuration data - Secret

- Key-value pairs of data
- Manage pods configuration data
  - Environment variables
  - Secret files
- Used for sensitive data (base64–encoded)

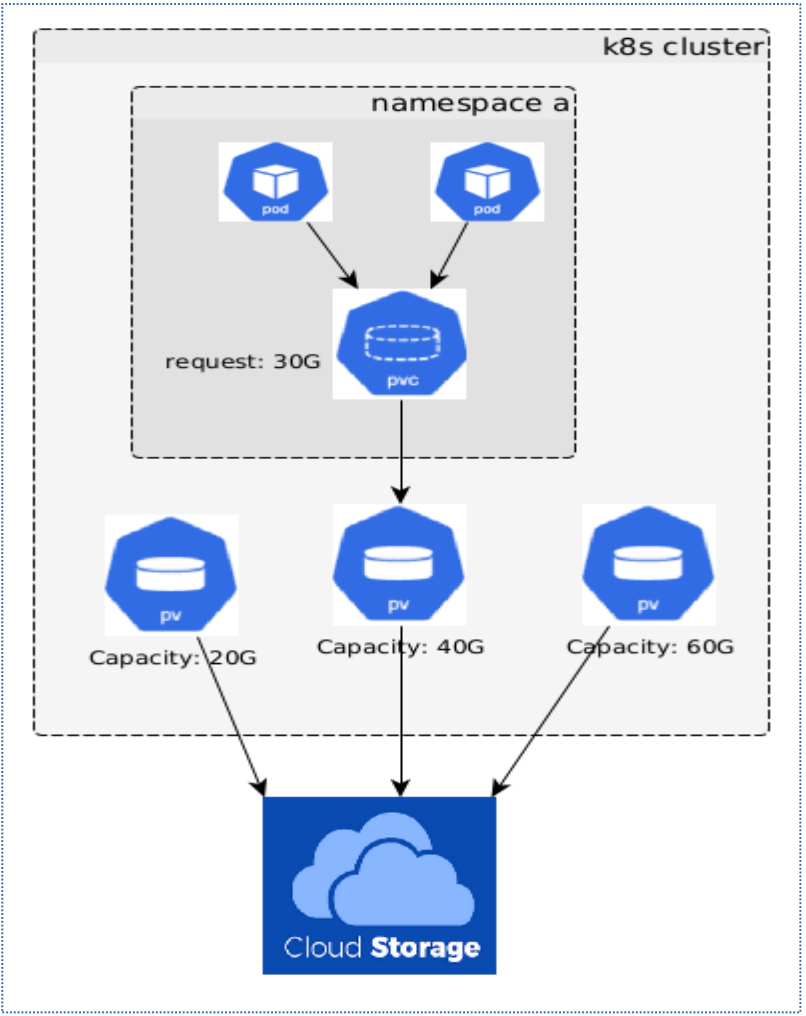



```yaml
apiVersion: v1
kind: Secret
metadata:
  name: nginx-secrets
type: Opaque
data:
  PASSWORD: cGFzc3dvcmQK
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
         - containerPort: 80
        envFrom:
        - configMapRef:
            name: nginx-config
        - secretRef:
            name: nginx-secrets
```

Persistent Storage – Persistent Volume, Persistent Volume Claim

- Persistent volume:
  - Static storage allocated in a cluster
  - Independent from pod's lifecycle
  - Provisioned by cluster admin
- Persistent volume claim:
  - Dynamic storage request
  - Depending on pod's lifecycle
  - Asked by pods



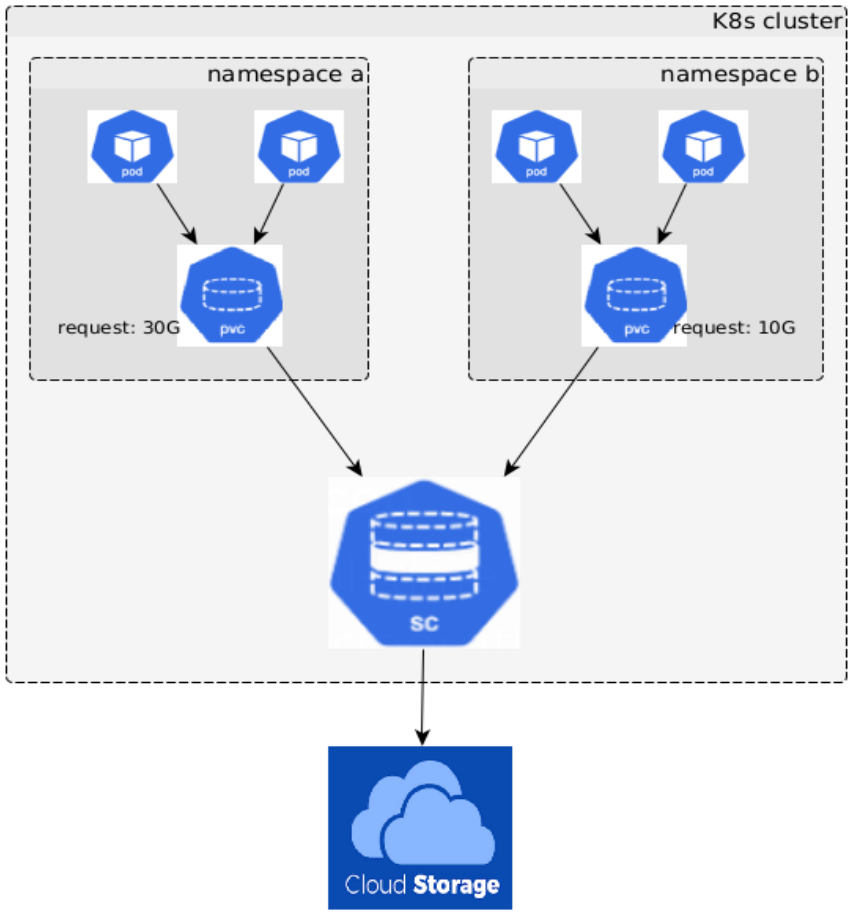

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: efs-pv
spec:
  capacity:
    storage: 5Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: “”
  csi:
    driver: efs.csi.aws.com
    volumeHandle: [FileSystemId]::[AccessPointId]
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: efs-claim
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: ""
  resources:
    requests:
      storage: 5Gi
```

Persistent Storage – Persistent Volume, Persistent Volume Claim

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        envFrom:
        - configMapRef:
            name: nginx-config
        - secretRef:
            name: nginx-secrets
        volumeMounts:
        - mountPath: /var/www/html
          name: data-volume
        ports:
        - containerPort: 80
      volumes:
      - name: data-volume
        persistentVolumeClaim:
          claimName: efs-claim
```

Persistent Storage – Storage Class

- Similar to PV, but providing a way to dynamically provision PV in a cluster




```yaml
apiVersion: storage.k8s.io/v1
kind: Storage-class
metadata:
  name: ebs-sc
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
volumeBindingMode: WaitForFirstConsumer
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: ebs-pvc
spec:
  storageClassName: ebs-sc
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

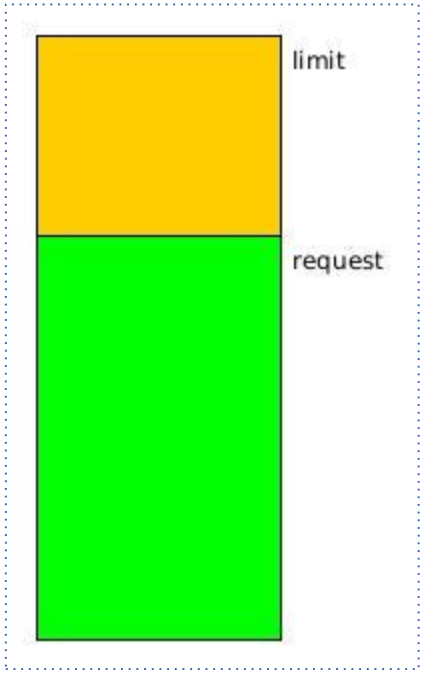
Persistent Storage – Volume Claim Template

- Similar to PVC but:
- Used for stateful app (DB)
- Each pod can have its own allocated PVC

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: postgres-sts
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: postgres
    spec:
      containers:
        - name: postgresql
          image: postgresql:latest
          imagePullPolicy: "IfNotPresent"
          env:
            - name: POSTGRES_USER
              value: "postgres"
            - name: POSTGRES_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: postgres-secrets
                  key: postgresql-password
            - name: POSTGRES_DB
              value: "postgres"
          ports:
            - name: postgresql
              containerPort: 5432
              protocol: TCP
          volumeMounts:
            - name: data-vct
              mountPath: /var/lib/postgresql/data
  volumeClaimTemplates:
    - metadata:
        name: data-vct
      spec:
        storageClassName: ebs-sc
        accessModes:
          - "ReadWriteOnce"
        resources:
          requests:
            storage: "2Gi"
```

Resource Request & Limit

- There are 2 main types of resources: CPU & memory
- Request: Ensure the minimum number of resources required for the container
- Limit: Is the maximum number of resources that Kubernetes will allow the container to use



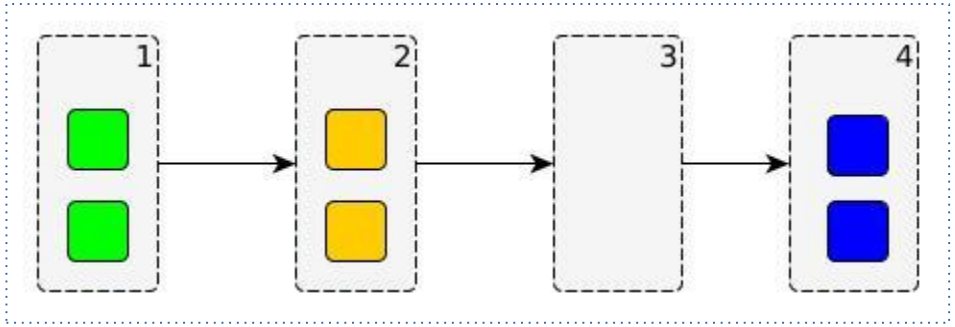

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
        resources:
            limits:
              memory: 300Mi
              cpu: 1
            requests:
              memory: 100Mi
              cpu: 100m
```

03

Deployment Strategies

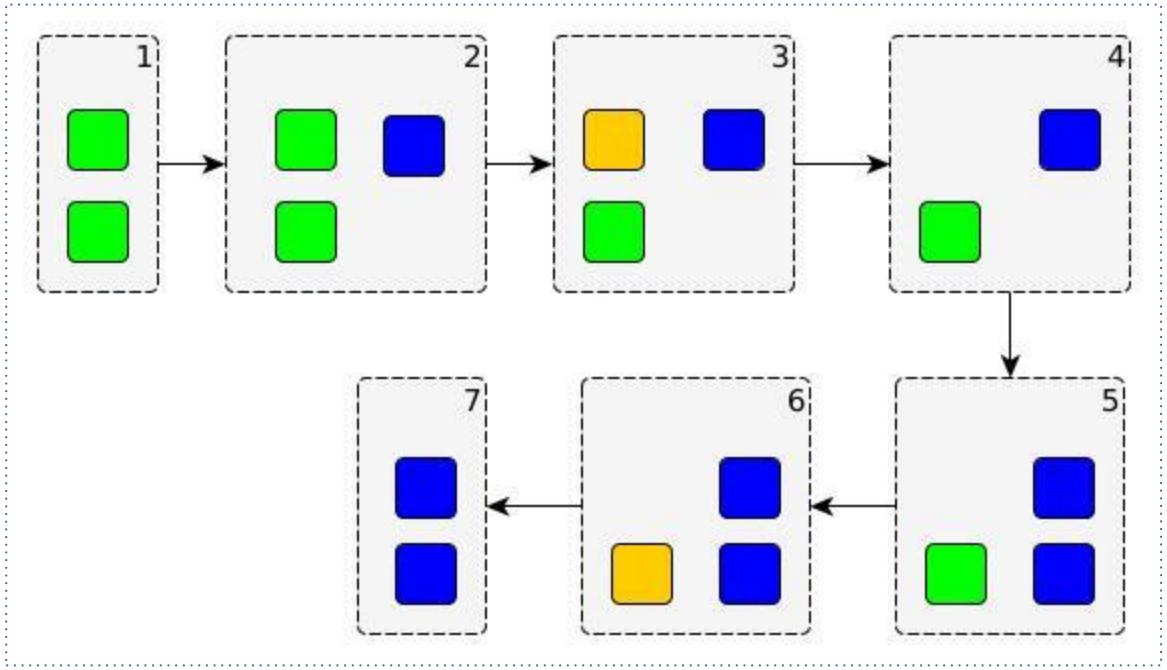
Recreate

- Terminate all current pods, create new ones
- Use case:
  - Not able to have 2 different versions of app installed at the same time (the big update)
- Cons:
  - Introduce a downtime period.
- Pros:
  - Quick update



RollingUpdate

- Create a few of new pods, then terminate old ones correspondingly
- Use case:
  - There is no conflict between old and new versions
- Cons:
  - There is a period when the old and new application versions serve users at the same time
- Pros:
  - No downtime



04

Kubernetes User Interaction

Command line - kubectl

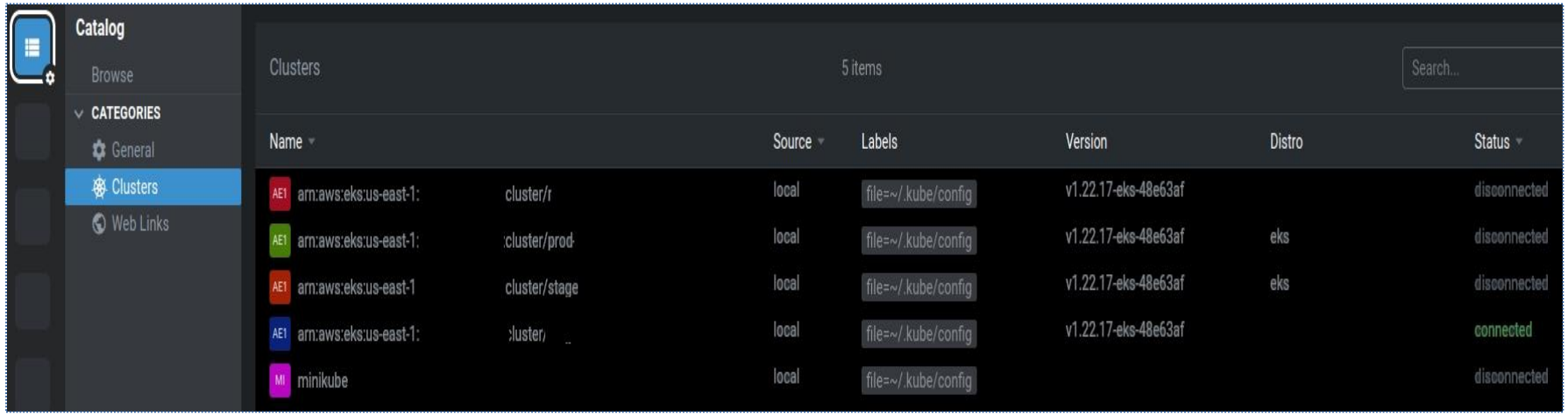
- Communicate with K8s resource
- Command line tool (refer to “install kubectl” for the installation”

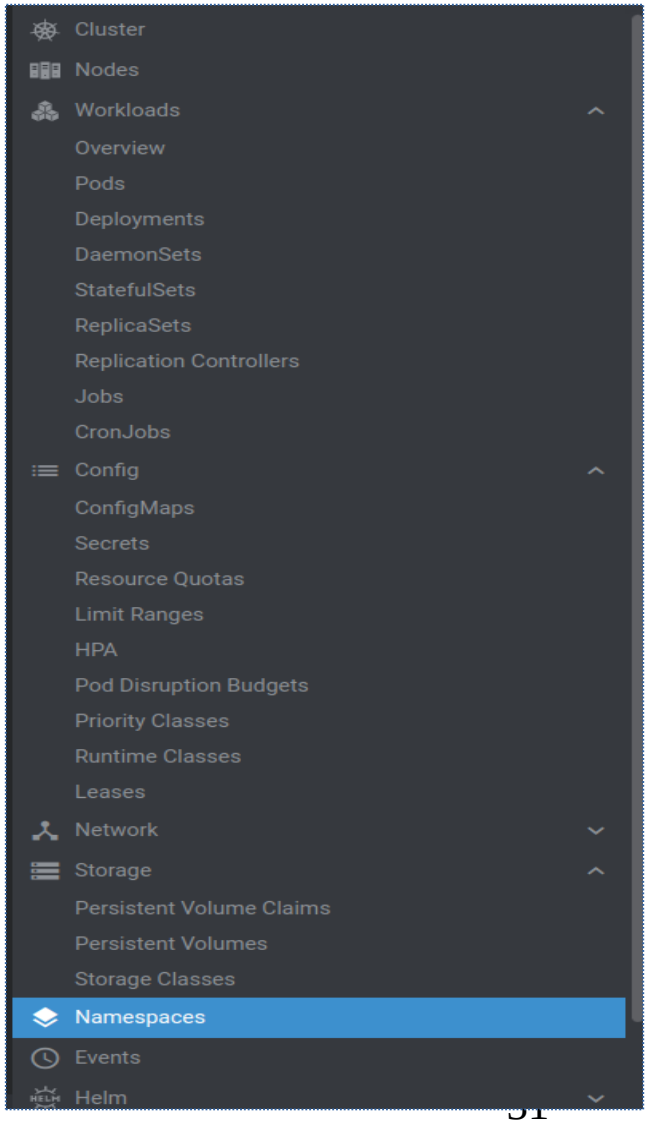
```
Kubectl [action] [resource]  [options]
kubectl get       pod        -o wide
kubectl get       deployment -o wide


kubectl [action]  [object]    [object-name]
kubectl  describe  pod        [pod-name]
kubectl  describe  deployment [deployment-name]
kubectl  delete    deployment [deployment-name]
```

User interface - Lens

- Communicate with K8s resource
- User interface tool (refer to “install lens” for the installation)





Thank you

Nash Tech.