

## Assignment #08: Kubernetes Lab

- What is K8s

The slide features a blue header with the title 'Official definition of Kubernetes' in white. In the top right corner is the Kubernetes logo, a blue hexagon containing a white steering wheel. Below the title, four bullet points are listed in blue text:

- > Open source container orchestration tool
- > Developed by Google
- > Helps you manage containerized applications
- > in different deployment environments

At the bottom left, there are three categories: 'physical', 'cloud', and 'virtual'. To the right of these categories is a video frame showing a woman with dark hair, wearing a pink Hollister California t-shirt, speaking. She is positioned in front of a bookshelf and some green plants.

The slide has a light blue header with the title 'What features do orchestration tools offer?' in white. To the right is the Kubernetes logo. Below the title is a smiling emoji. Three bullet points are listed in blue text:

- > High Availability or no downtime
- > Scalability or high performance
- > Disaster recovery - backup and restore

On the left side of the slide, there is a screenshot of a Google Docs document titled 'Nanthakarn LIMKOOL - Assignment #...'. The document contains a bulleted list of topics related to Kubernetes. At the bottom right of the slide is a decorative image of red roses on a dark background.

## - Main K8s Components

The slide contains a diagram illustrating the main components of a Kubernetes cluster:

- Pod:** Contains two containers: **my-app** and **DB**. Each container has its own IP address.
- Service:** Provides external access to the Pod's IP.
- Ingress:** Manages external traffic entry points.
- ConfigMap:** Stores configuration data.
- Secrets:** Stores sensitive data.

Below the diagram, a list of main K8s components is provided:

- K8s Architecture
- Minikube and kubectl
- Local Setup
- Main Kubectl Commands
- K8s CLI
- K8s YAML Configuration File
- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Stateful Apps with StatefulSet
- K8s Services explained

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

Official definition

my-app

DB

Service

Pod

Container

Service:

- ▶ permanent IP address
- ▶ lifecycle of Pod and Service  
NOT connected

Node 1

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

Ingress

https://my-app.com

Internal service

http://db-service-ip:port

Pod

Container

Ingress

Internal service

Node 1

- K8s Architecture

- Minikube and kubectl

- Local Setup

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

Container

Local Setup

- K8s Architecture
- Minikube and kubectl
- Local Setup
- Main Kubectl Commands
- K8s CLI
- K8s YAML Configuration File
- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Stateful Apps with StatefulSet
- K8s Services explained

**ConfigMap:**

- ▶ external configuration of your application

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

Container

Local Setup

- K8s Architecture
- Minikube and kubectl

**Secret:**

- ▶ used to store secret data
- ▶ base64 encoded

**!** The built-in security mechanism is not enabled by default!

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

https://my-app.com  
can't be reached

User

- K8s Architecture

- Minikube and kubectl  
- Local Setup  
- Main Kubectl Commands  
- K8s CLI  
- K8s YAML Configuration File  
- Demo Project: MongoDB and MongoExpress  
- Organizing your components with K8s Namespaces  
- K8s Ingress explained  
- Helm - Package Manager

Deployment  
Service  
my-app  
DB  
Service  
my-app  
DB  
StatefulSet  
Node 1  
Node 2

Deployment for stateLESS Apps

StatefulSet for stateFUL Apps or Databases

## K8s Architecture

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

User

- K8s Architecture

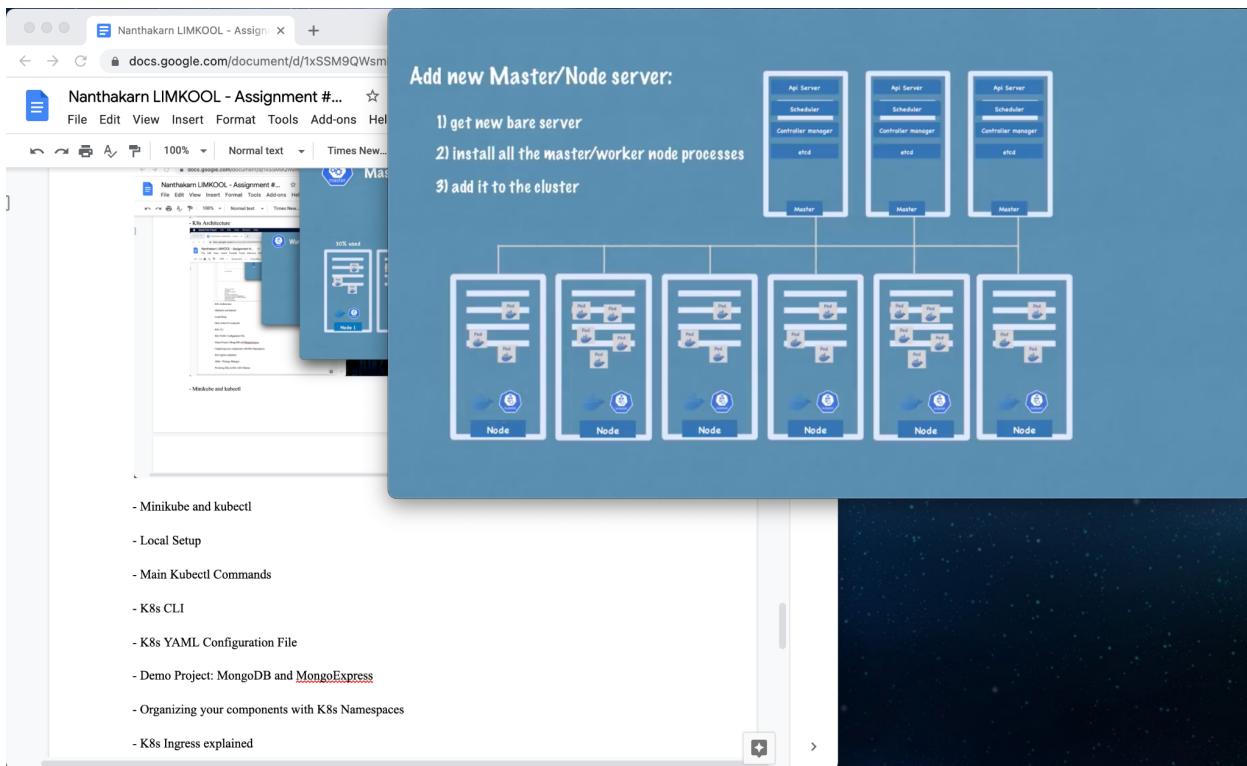
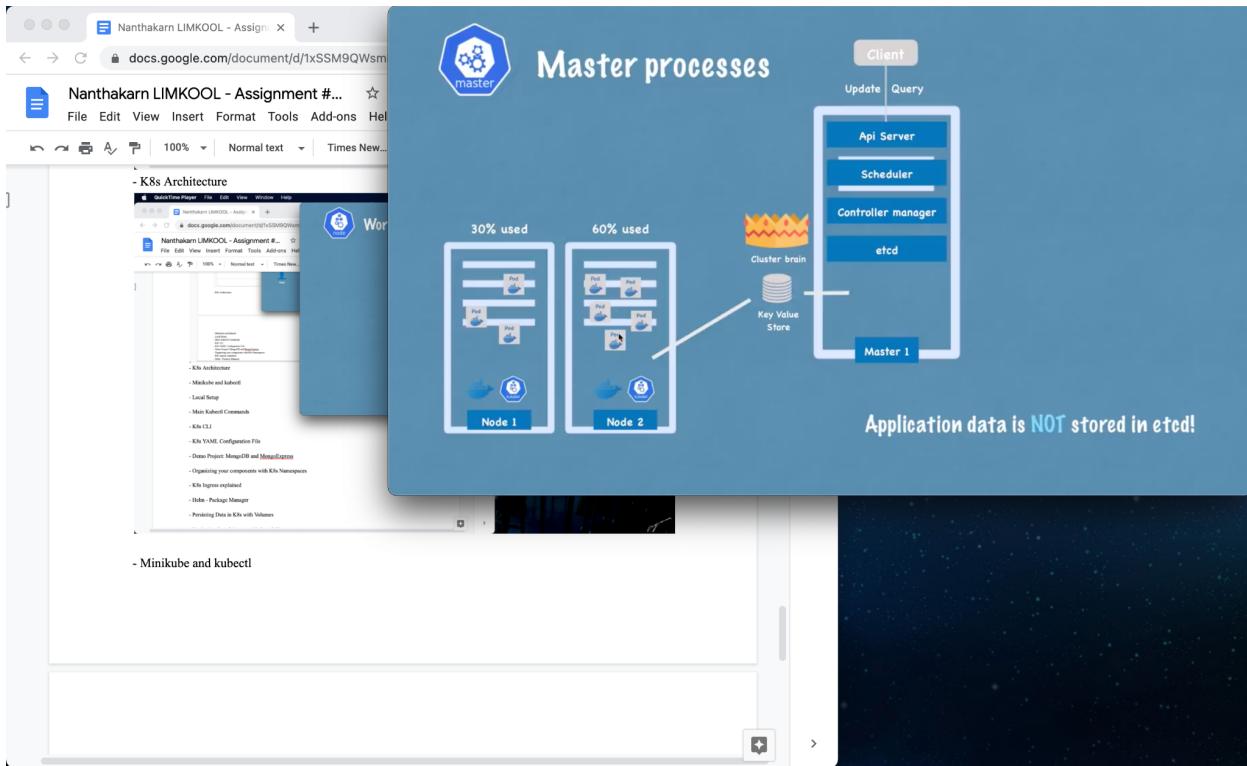
- Minikube and kubectl  
- Local Setup  
- Main Kubectl Commands  
- K8s CLI  
- K8s YAML Configuration File  
- Demo Project: MongoDB and MongoExpress  
- Organizing your components with K8s Namespaces  
- K8s Ingress explained  
- Helm - Package Manager

**Worker machine in K8s cluster**

3 Node processes:

1. Kubelet
2. Kube Proxy
3. Container runtime

my-app  
DB Service  
my-app  
DB Service  
Kubelet  
Kubelet  
Container runtime  
Container runtime  
Node 1  
Node 2



- Minikube and kubectl

**Test/Local Cluster Setup**

**minikube**

- ▶ creates Virtual Box on your laptop
- ▶ Node runs in that Virtual Box
- ▶ 1 Node K8s cluster
- ▶ for testing purposes

- Minikube and kubectl

- Local Setup

- Main Kubectl Commands

- K8s CLI

- K8s YAML Configuration File

- Demo Project: MongoDB and [MongoExpress](#)

- Organizing your components with K8s Namespaces

- K8s Ingress explained

- Helm - Package Manager

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

**What is kubectl?**

**Master processes**

**Worker processes**

enable pods to run on node

**UI**

**API**

**CLI**

**KUBECTL**

The most powerful of 3 clients

**Api Server**

**POD**

**Service**

**Secret**

**ConfigMap**

**Node**

**create pods**

**create services**

**destroy pods**

- Local Setup

Nanthakarn LIMKOOL - Assignment #... docs.google.com/document/d/1xSSM9QWsm...

**Installation**

! Virtualization on your machine needed!

- Local Setup

- Main Kubectl Commands

- K8s CLI
- K8s YAML Configuration File
- Demo Project: MongoDB and [MongoExpress](#)
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Stateful Apps with StatefulSet

Nanthakarn LIMKOOL - Assignment #... docs.google.com/document/d/1xSSM9QWsm...

**Installation**

**Installation Guide Links for your OS:**

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

<https://kubernetes.io/docs/tasks/tools/install-kubectl/>

- Local Setup

- Main Kubectl Commands

- K8s CLI
- K8s YAML Configuration File
- Demo Project: MongoDB and [MongoExpress](#)
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager

## - Main Kubectl Commands

**CRUD commands**

- Create deployment** `kubectl create deployment [name]`
- Edit deployment** `kubectl edit deployment [name]`
- Delete deployment** `kubectl delete deployment [name]`

**Status of different K8s components**

`kubectl get nodes | pod | services | replicaset | deployment`

**Debugging pods**

- Log to console** `kubectl logs [pod name]`
- Get Interactive Terminal** `kubectl exec -it [pod name] -- bin/bash`

- K8s YAML Configuration File

- Demo Project: MongoDB and [MongoExpress](#)

- Organizing your components with K8s Namespaces

- K8s Ingress explained

- Helm - Package Manager

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

## - K8s CLI

**Debugging pods**

- Log to console** `kubectl logs [pod name]`
- Get Interactive Terminal** `kubectl exec -it [pod name] -- bin/bash`
- Get info about pod** `kubectl describe pod [pod name]`

**Use configuration file for CRUD**

**Apply a configuration file** `kubectl apply -f [file name]`

**Delete with configuration file** `kubectl delete -f [file name]`

- K8s CLI

- K8s YAML Configuration File

- Demo Project: MongoDB and [MongoExpress](#)

- Organizing your components with K8s Namespaces

- K8s Ingress explained

## - K8s YAML Configuration File

Each configuration file has 3 parts

- 1) metadata
- 2) specification
- 3) status

Automatically generated and added by Kubernetes!

The screenshot shows a Google Docs slide titled "Nanthakarn LIMKOOL - Assignment #...". The slide content is as follows:

Each configuration file has 3 parts

- 1) metadata
- 2) specification
- 3) status

Automatically generated and added by Kubernetes!

The slide includes a list of topics:

- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager

A small image of a dark, rocky landscape is visible on the right side of the slide.

**Layers of Abstraction**

```

graph TD
    Deployment[Deployment manages a ..] --> RS[ReplicaSet manages a ..]
    RS --> Pod[Pod is an abstraction of ..]
    Pod --> Container[Container]
  
```

The diagram illustrates the layers of abstraction in Kubernetes:

- Deployment** manages a ..
- ReplicaSet** manages a ..
- Pod** is an abstraction of ..
- Container**

**Deployment manage Pods**

Google Docs slide content:

- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Stateful Apps with StatefulSet
- K8s Services explained

**Template**

**has it's own "metadata" and "spec" section**

**applies to Pod**

**blueprint for a Pod**

port? name? image?

```

1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: nginx-deployment
5  > labels:-
6  spec:
7    replicas: 2
8    selector: ...
9    template:
10      metadata:
11        labels:
12          app: nginx
13      spec:
14        containers:
15          - name: nginx
16            image: nginx:1.16
17            ports:
18              - containerPort: 8080
19
20
21
22
  
```

Google Docs slide content:

- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces

The screenshot shows a presentation slide with the title "Deployment". Below the title, there are two code snippets representing Kubernetes resources:

```
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:...
```

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:...
```

The first snippet is for a Deployment resource, and the second is for a Service resource. Both snippets have specific sections highlighted with yellow boxes: the "labels" section under "Deployment" and the "labels" section under "template" for the Deployment, as well as the "selector" and "ports" sections under "Service".

The slide has a light blue background with a decorative sea-themed footer image. At the top right, the title 'Ports in Service and Pod' is displayed in a large, bold, black serif font. To the left of the title is a screenshot of a Google Docs editor showing a YAML file for a deployment named 'nginx-deployment'. The code defines a deployment with two replicas, each running an 'nginx' container. The container's port is mapped to 8080 on the host. A yellow arrow points from the 'targetPort: 8080' line in the code to a callout box labeled 'DB Service'. Another yellow arrow points from the 'containerPort: 8080' line to a callout box labeled 'nginx Service'. A third yellow arrow points from the 'nginx Service' box down to a final callout box labeled 'Pod'.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels: ...
spec:
  replicas: 2
  selector: ...
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.16
          ports:
            - containerPort: 8080
```

DB Service  
port: 80

nginx Service  
targetPort: 8080

Pod

- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Stateful Apps with StatefulSet
- K8s Services explained

- Demo Project: MongoDB and MongoExpress

# Overview of K8s Components

The diagram illustrates the components of a K8s application stack:

- Internal Service:** MongoDB pod
- ConfigMap:** DB Url
- Deployment.yaml:** Env variables
- Secret:** DB User, DB Pwd
- External Service:** Mongo Express pod

URL:

- IP Address of Node
- Port of external Service

Annotations from the slide notes:

- Demo Project: MongoDB and MongoExpress
- Organizing your components with K8s Namespaces
- K8s Ingress explained
- Helm - Package Manager
- Persisting Data in K8s with Volumes
- Deploying Static Apps with Staticfile
- K8s Services explained

Annotations from the browser screenshot:

- Nanthakarn LIMKOOL - Assignment #...
- File Edit View Insert Format Tools Add-ons Help
- 100% Normal text Times New...
- Nanthakarn LIMKOOL - Assignment #...
- https://docs.google.com/document/d/1tXSSM9QWsm...
- 2 Deployment / Pod
- 2 Service
- 1 ConfigMap
- 1 Secret

Nanthakarn LIMKOOL - Assignment #...

- Organizing your components with K8s Namespaces

- K8s Ingress explained

- Helm - Package Manager

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

**Kubernetes Tutorial for Beginners [FULL COURSE in 4 Hours].mp4**

```

1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: mongodb-secret
5  type: Opaque
6  data:
7    mongo-root-username: dXNlc3hbWU=
8    mongo-root-password: cGFzc3dvcmQ=
9

```

**Deployment**

```

env:
  - name: MONGO_INITDB_ROOT_USERNAME
    valueFrom:
      secretKeyRef:
        name: mongodb-secret
        key: mongo-root-username
  - name: MONGO_INITDB_ROOT_PASSWORD
    valueFrom:
      secretKeyRef:
        name: mongodb-secret
        key: mongo-root-password

```

**Secret**

```

apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  mongo-root-username: dXNlc3hbWU=
  mongo-root-password: cGFzc3dvcmQ=

```

The screenshot shows a Google Slides presentation slide. The title of the slide is "Service Configuration File". The content of the slide is organized into two columns. The left column contains a code editor window showing two YAML files: "mongo.yaml" and "mongo-secret.yaml". The "mongo.yaml" file defines a Service object named "mongodb-service" that connects to a Pod labeled "app: mongodb". The "mongo-secret.yaml" file defines a Secret object named "MONGO\_INITDB\_ROOT\_PASSWORD" with a single key-value pair. The right column contains a list of terms and their definitions:

- kind: "Service"
- metadata / name: a random name
- selector: to connect to Pod through label
- ports:
  - port: Service port
  - targetPort: containerPort of Deployment

Below the slide content, there is a watermark or background image of a character from an anime.

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
```

```
name: MONGO_INITDB_ROOT_PASSWORD
key: mongo-root-password
valueFrom:
  secretKeyRef:
    name: mongodb-secret
    key: mongo-root-password
```

- Organizing your components with K8s Namespaces  
- K8s Ingress explained

Nanthakarn LIMKOOL - Assignment #...

```

16 spec:
17   containers:
18     - name: mongo-express
19       image: mongo-express
20       ports:
21         - containerPort: 8081
22     env:
23       - name: ME_CONFIG_MONGO
24         valueFrom:
25           secretKeyRef:
26             name: mongodb-secret
27             key: mongo-root-password
28       - name: ME_CONFIG_MONGO
29         valueFrom:
30           secretKeyRef:
31             name: mongodb-secret
32             key: mongo-root-password
33       - name: ME_CONFIG_MONGO
34         value: |

```

## ConfigMap

- external configuration
- centralized
- other components can use it

Nanthakarn LIMKOOL - Assignment #...

```

1  apiVersion: v1
2  kind: ConfigMap
3  metadata:
4    name: mongodb-configmap
5  data:
6    database_url: mongodb-service

```

## ConfigMap Configuration File

- kind: "ConfigMap"
- metadata / name: a random name
- data: the actual contents - in key-value pairs

Nanthakarn LIMKOOL - Assignment #...

- Organizing your components with K8s Namespaces

```
mongo.yaml mongo-express.yaml
31      name: mongodb-secret
32      key: mongo-root-password
33  - name: ME_CONFIG_MONGODB_SERVER
34    valueFrom:
35      configMapKeyRef:
36        name: mongodb-configmap
37        key: database_url
38
39  apiVersion: v1
40  kind: Service
41  metadata:
42    name: mongo-express-service
43  spec:
44    selector:
45      app: mongo-express
46      type: LoadBalancer
47    ports:
48      - protocol: TCP
49        port: 8081
50        targetPort: 8081
51        nodePort: 
```

**How to make it an External Service?**

- **type:** "Loadbalancer"
- ...**assigns service an external IP address and so accepts external requests**
- **nodePort:** must be between 30000-32767

**Port for external IP address**

**Port you need to put into browser**

## - Organizing your components with K8s Namespaces

Nanthakarn LIMKOOL - Assignment #...

- Organizing your components with K8s Namespaces

- K8s Ingress explained

- Helm - Package Manager

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

**4 Namespaces per Default**

- resources you create are located here

**1.** Resources grouped in Namespaces 😊

The diagram illustrates a **Kubernetes Cluster** represented by a blue rectangle containing four boxes: **Database**, **Monitoring**, **Elastic Stack**, and **Nginx-Ingress**. Each box contains icons representing different types of resources, such as databases, monitoring tools, search engines, and ingress controllers. A smiley face icon is located in the top right corner of the slide.

**2.** Conflicts: Many teams, same application

The diagram shows a **Kubernetes Cluster** with two separate boxes representing **Project A Namespace** and **Project B Namespace**. Both boxes contain a central **my-app deployment** icon with a circular arrow. On the left, a group of people icon is connected to the Project A box, and on the right, another group of people icon is connected to the Project B box. Arrows point from each group of people towards their respective project boxes.

**3. Resource Sharing: Blue/Green Deployment**

No need for a separate cluster

Kubernetes Cluster

Production Blue

Production Green

Nginx-Ingress Controller

Elastic Stack

- K8s Ingress explained  
- Helm - Package Manager

**4. Access and Resource Limits on Namespaces**

Limit: CPU, RAM, Storage per NS

Kubernetes Cluster

Project A Namespace

Project B Namespace

Resource Quota A

Resource Quota B

- K8s Ingress explained  
- Helm - Package Manager  
- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

Nanthakarn LIMKOOL - Assignment #...

Components, which can't be created within a Namespace

You can't create components which live globally in a cluster.

Each NS must define own ConfigMap

Kubernetes Cluster

Project A

Project B

vol

node

- live globally in a cluster  
- you can't isolate them

kubectl api-resources --namespaced=false  
kubectl api-resources --namespaced=true

Nanthakarn LIMKOOL - Assignment #...

You can't access most resources from another Namespace

Each NS must define own ConfigMap

Kubernetes Cluster

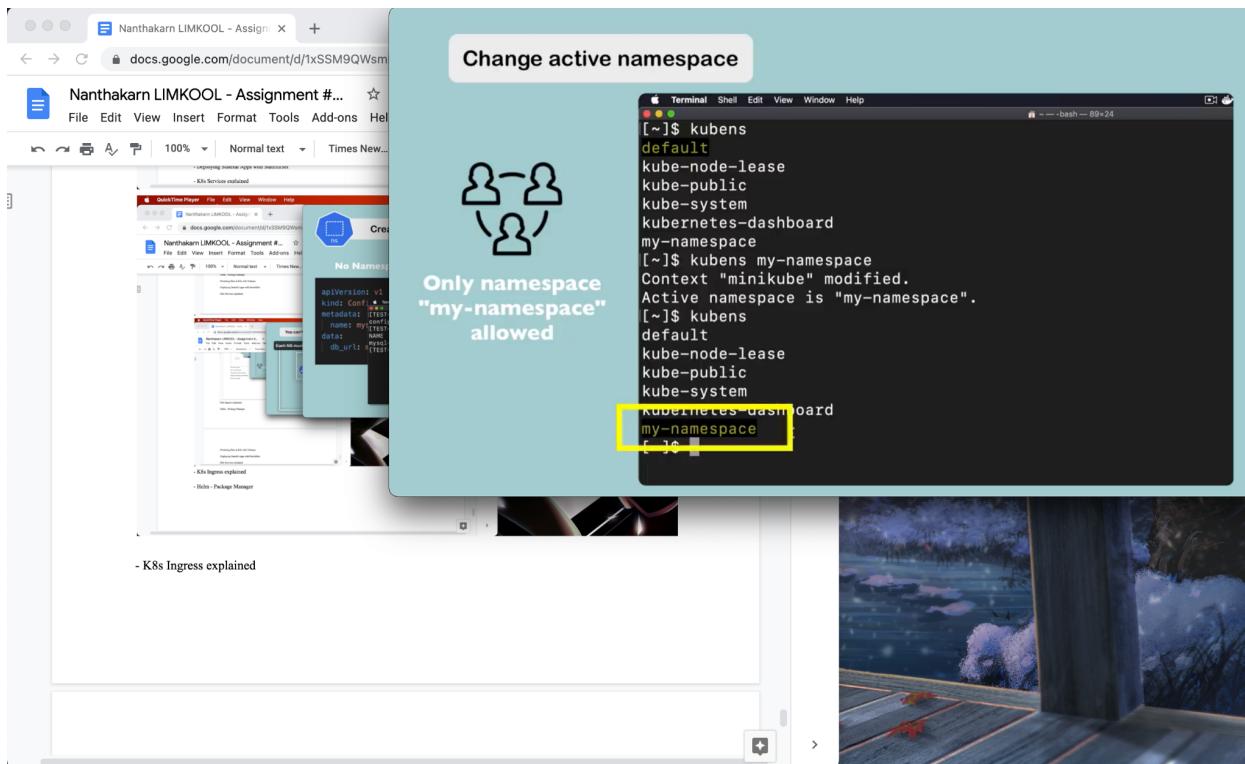
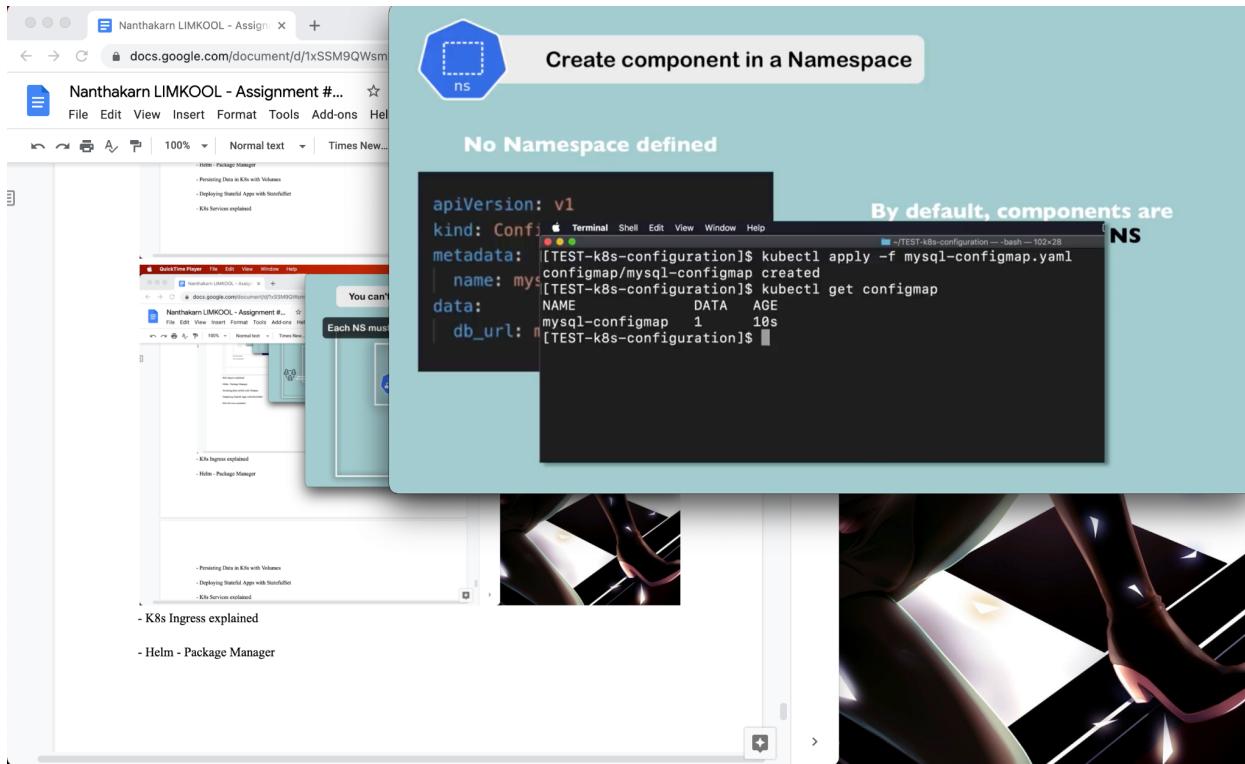
Project A

ConfigMap

Project B

ConfigMap

RB



- K8s Ingress explained

## External Service vs. Ingress

The diagram illustrates the flow of traffic from an external source to a Kubernetes application. An external browser window shows a URL <https://my-app.com>. This traffic enters a **Kubernetes Cluster** via an **Ingress** (represented by a blue hexagon with a double-headed arrow). The Ingress points to a **Service** (a blue hexagon with a hierarchical icon). The Service then points to a **Pod** (a blue hexagon with a cube icon). A green oval labeled "internal" is positioned above the Service and Pod components.

- K8s Ingress explained

- Helm - Package Manager

- Persisting Data in K8s with Volumes

## Ingress and Internal Service configuration

**Ingress:**

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - backend:
              serviceName: myapp-internal-service
              servicePort: 8080
```

**Host:**

- valid domain address
- map domain name to Node's IP address, which is the endpoint's IP address

The diagram shows a cluster of three nodes, each represented by a vertical rectangle with horizontal lines inside. A blue hexagon labeled "Kubernetes Cluster" is connected to these nodes. A white line connects the "host" entry in the Ingress configuration to one of the nodes.

- Helm - Package Manager

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

**What is Ingress Controller?** 🤔

**Ingress Controller Pod**

**Kubernetes Cluster**

**my-app pod** → **my-app service** → **my-app ingress** → **pod**

**internal**

- evaluates all the rules
- manages redirections
- entrypoint to cluster

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: myapp-ingress
spec:
  rules:
    - host: myapp.com
      http:
        paths:
          - backend:
              serviceName: myapp-internal-service
              servicePort: 8080
  
```

- Helm - Package Manager  
- Persisting Data in K8s with Volumes

Note. env = cloud load balancer, server

## Multiple paths for same host

**Ingress API Version:** networking.k8s.io/v1beta1

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: simple-fanout-example
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: myapp.com
    http:
      paths:
        - path: /analytics
          backend:
            serviceName: analytics-service
            servicePort: 3000
        - path: /shopping
          backend:
            serviceName: shopping-service
            servicePort: 8080

```

Note. env = cloud load balancer, server  
- Helm - Package Manager  
- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

## Multiple sub-domains or domains

**Ingress API Version:** networking.k8s.io/v1beta1

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: name-virtual-host-ingress
spec:
  rules:
  - host: analytics.myapp.com
    http:
      paths:
        - path: /analytics
          backend:
            serviceName: analytics-service
            servicePort: 3000
  - host: shopping.myapp.com
    http:
      paths:
        - path: /shopping
          backend:
            serviceName: shopping-service
            servicePort: 8080

```

Note. env = cloud load balancer, server  
- Helm - Package Manager  
- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

**Configuring TLS Certificate - https://**

```

apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
    - hosts:
      - myapp.com
      secretName: myapp-secret-tls
  rules:
    - host: myapp.com
      http:
        paths:
          - path: /
            backend:
              serviceName: myapp-internal-service
              servicePort: 8080

```

```

apiVersion: v1
kind: Secret
metadata:
  name: myapp-secret-tls
  namespace: default
data:
  tls.crt: base64 encoded cert
  tls.key: base64 encoded key
type: kubernetes.io/tls

```

## - Helm - Package Manager

**What is Helm?**

The diagram illustrates the Helm ecosystem components:

- Kubernetes Cluster**: Represented by a blue hexagon icon.
- my-app pod**: Represented by a blue hexagon icon with a cube symbol.
- my-app service**: Represented by a blue hexagon icon with a network-like symbol.
- ELASTIC Stack for Logging**: Represented by a blue hexagon icon with a stack of logs symbol.
- Cloud icon**: Contains a person icon.
- User icons**: Three person icons with laptops below the cloud.

Nanthakarn LIMKOOL - Assignment #... docs.google.com/document/d/1xSSM9QWsm...

Nanthakarn LIMKOOL - Assignment #... File Edit View Insert Format Tools Add-ons Help

Normal text Times New...

Persisting Data in K8s with Volumes

Deploying Stateful Apps with StatefulSet

K8s Services explained

## What is Helm?

### Helm Charts

- Bundle of YAML Files
- Create your own Helm Charts with Helm
- Push them to Helm Repository
- Download and use existing ones

- Persisting Data in K8s with Volumes

## Sharing Helm Charts 😊

Kubernetes Cluster

Need some Deployment?

my-app pod my-app service

helm search <keyword>

or Helm Hub:

Discover & launch great Kubernetes-ready apps

Search charts... 1140 charts ready to deploy

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

Nanthakarn LIMKOO - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New Roman

docs.google.com/document/d/1xSSM9QWsm...

Need some help? [Get help](#)

Kubernetes Cluster

microservice microservice

microservice microservice microservice

Deployment and Service configurations almost the same!

YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

The diagram illustrates a Kubernetes Cluster with several microservices represented by blue cubes. A callout box labeled "Templating Engine" contains a standard YAML configuration for a Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

Below this, another callout box also labeled "Templating Engine" shows a template YAML configuration where placeholder values are highlighted in blue:

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

A large blue button at the bottom right of the second callout box contains the text "{{ .Values... }}".

Nanthakarn LIMKOO - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New Roman

docs.google.com/document/d/1xSSM9QWsm...

Need some help? [Get help](#)

Kubernetes Cluster

microservice microservice

microservice microservice

Template YAML config

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

Template {{ .Values... }}

- Persisting Data in K8s with Volumes

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

The diagram illustrates a Kubernetes Cluster with several microservices represented by blue cubes. A callout box labeled "Templating Engine" contains a standard YAML configuration for a Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app-container
      image: my-app-image
      port: 9001
```

Below this, another callout box also labeled "Templating Engine" shows a template YAML configuration where placeholder values are highlighted in blue:

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

A large blue button at the bottom right of the second callout box contains the text "{{ .Values... }}".

**Templating Engine**

**Template YAML config**

```
apiVersion: v1
kind: Pod
metadata:
  name: {{ .Values.name }}
spec:
  containers:
    - name: {{ .Values.container.name }}
      image: {{ .Values.container.image }}
      port: {{ .Values.container.port }}
```

Object, which is created based on the values defined

- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

**Helm Chart Structure**

**Directory structure:**

```
mychart/
  Chart.yaml
  values.yaml
  charts/
  templates/
  ...
```

Top level mychart folder → name of chart  
Chart.yaml → meta info about chart  
values.yaml → values for the template files  
charts folder → chart dependencies  
templates folder → the actual template files

- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

Nanthakarn LIMKOO - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

Directory structure

mychart/ Chart.yaml values.yaml charts/ charts/ ...

- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

## Values injection into template files

**values.yaml**

```
imageName: myapp
port: 8080
version: 1.0.0
```

**default**

**my-values.yaml**

```
version: 2.0.0
```

**override values**

**result**

```
imageName: myapp
port: 8080
version: 2.0.0
```

**.Values object**

OR on Command Line:

```
helm install --set version=2.0.0
```

## Release Management

Keeping track of all chart executions:

Revision	Request
1	Installed chart
2	Upgraded to v 1.0.0
3	Rolled back to 1

- Changes are applied to existing deployment instead of creating a new one
- Handling rollbacks

- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet



## Downsides of Tiller

- Tiller has too much power inside of K8s cluster
- Security Issue
- Solves the Security Concern ☺

In Helm 3 Tiller got removed!



- Persisting Data in K8s with Volumes  
- Deploying Stateful Apps with StatefulSet



## - Persisting Data in K8s with Volumes

The slide has a blue header bar with the text "What we've covered so far" and a light blue background. Below the header, there is a list of bullet points:

- Volume is directory with some data
- These volumes are accessible in containers in a pod
- How made available, backed by which storage medium
  - defined by specific volume types

Below the text, there is a diagram illustrating storage concepts. It shows three icons: a cloud icon labeled "cloud-storage", a cylinder icon labeled "Storage Requirements", and a cube icon labeled "/var/lib". An arrow points from the "Storage Requirements" icon to the "cloud-storage" icon. Another arrow points from the "Storage Requirements" icon to the "/var/lib" icon.

On the left side of the slide, there is a screenshot of a Google Docs window showing a slide with the title "- Persisting Data in K8s with Volumes". The slide content includes a bulleted list and a diagram similar to the one on the right.

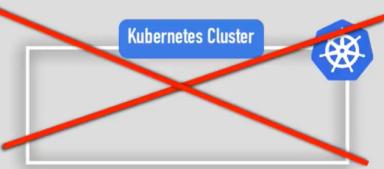
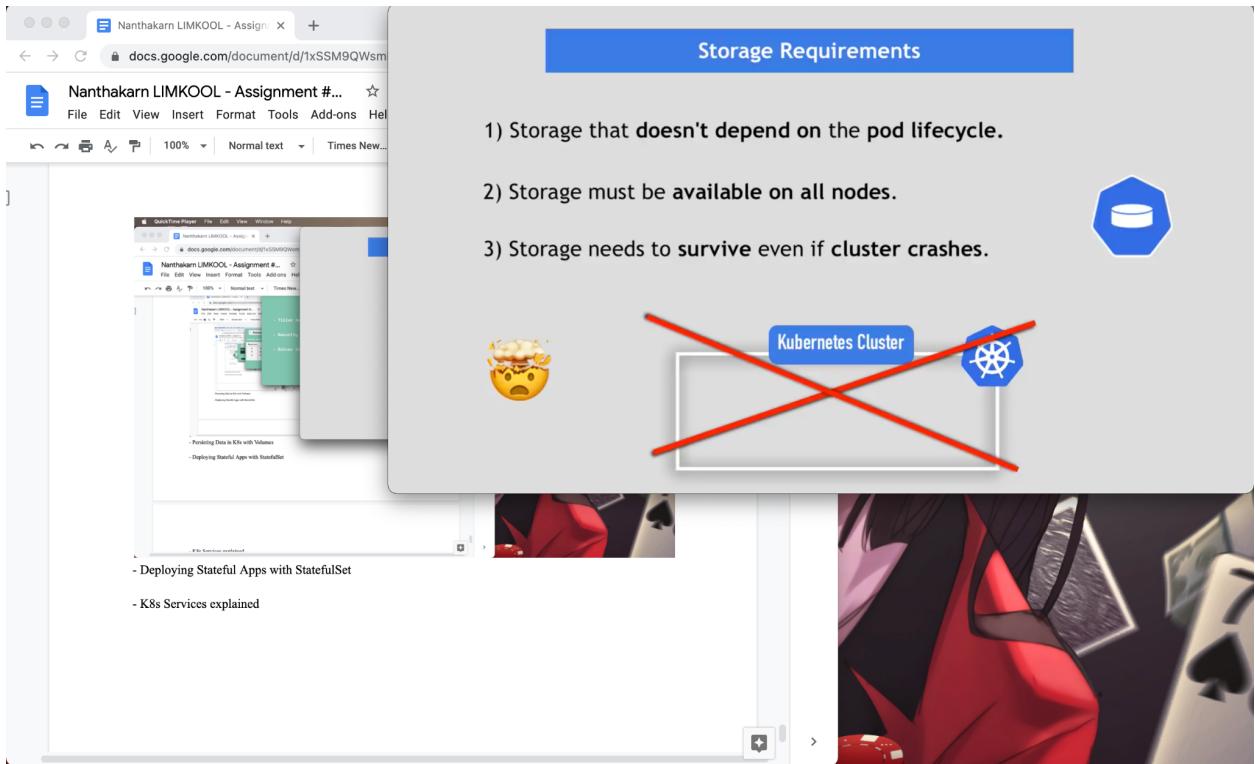
On the right side of the slide, there is a small decorative image of a colorful mural featuring various letters and shapes.

## Storage Requirements

1) Storage that doesn't depend on the pod lifecycle.

2) Storage must be available on all nodes.

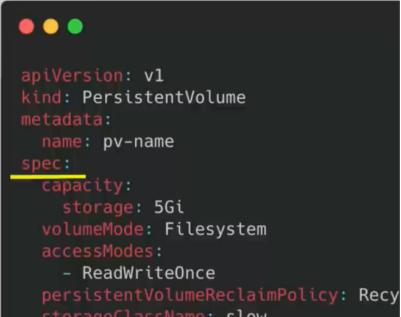
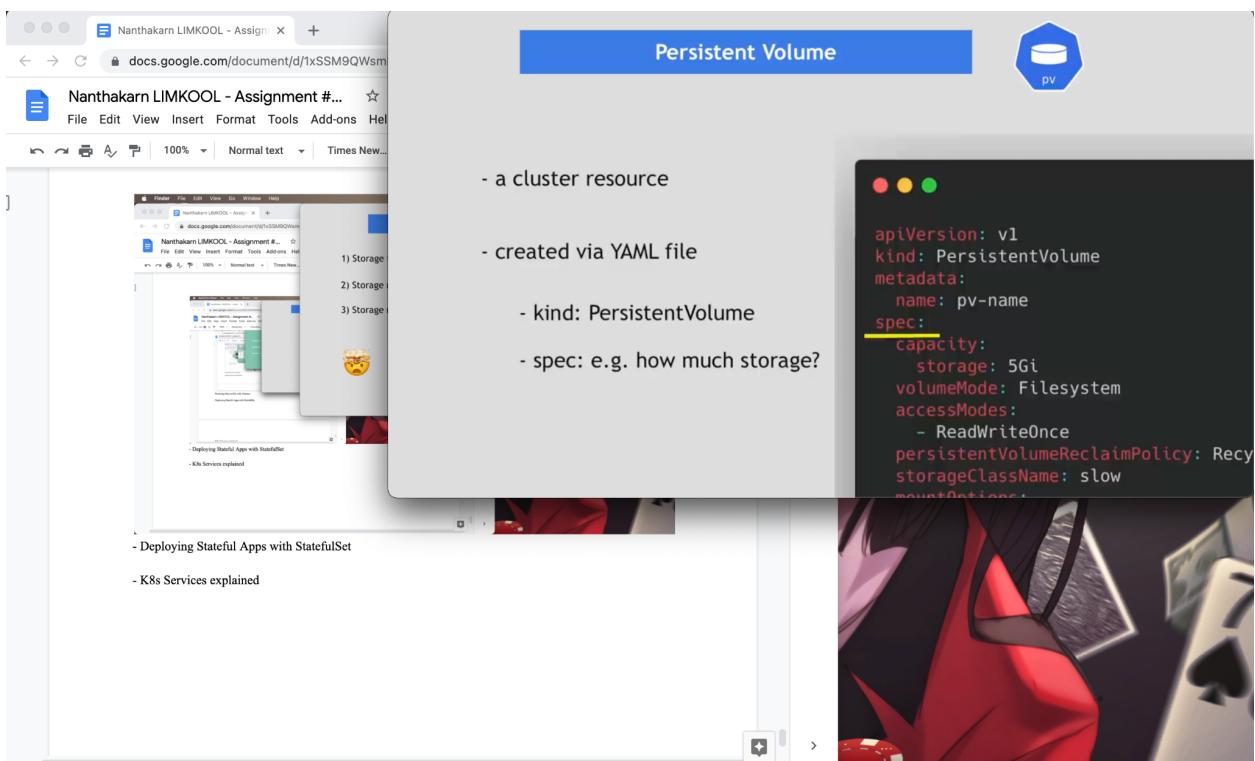
3) Storage needs to survive even if cluster crashes.

## Persistent Volume



- a cluster resource
- created via YAML file
- kind: PersistentVolume
  - spec: e.g. how much storage?

Nanthakarn LIMKOO - Assignment #...

docs.google.com/document/d/1xSSM9QWsm...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

QuickTime Player File Edit View Window

Nanthakarn LIMKOO - Assignment #...

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

Persistent Volume

pv

What type of storage do you need?

You need to **create and manage** them by yourself

external plugin to your cluster

Kubernetes Cluster

local disk

cloud-storage

nfs server

The diagram illustrates the different types of storage available for a Kubernetes cluster. It features a central box labeled 'Kubernetes Cluster' with three arrows pointing to external storage components: 'local disk', 'nfs server', and 'cloud-storage'. A callout box with the text 'You need to create and manage them by yourself' is positioned above the cluster. Another callout box, 'external plugin to your cluster', is located near the 'local disk' icon.

**Persistent Volumes are NOT namespaced**

The diagram illustrates a Persistent Volume (PV) located below two separate namespaces, Namespace A and Namespace B. Each namespace contains a pod icon and a storage icon. Two yellow arrows point from the namespaces up to the PV, indicating that it is accessible to both. The background shows a blurred image of a mural with paint cans labeled 'TAN' and 'KEI'.

PV outside of the namespaces  
Accessible to the whole cluster

Kubernetes Cluster

Namespace A

Namespace B

pv

**Local vs. Remote Volume Types**

Each volume type has its own use case!

Local volume types violate 2. and 3. requirement for data persistence:

- ✗ Being tied to 1 specific node
- ✗ surviving cluster crashes

For DB persistence use remote storage!

The diagram compares local and remote volume types. It states that each type has its own use case and that local volume types violate requirements 2. and 3. for data persistence. It recommends using remote storage for database persistence. The background shows a blurred image of a mural with paint cans labeled 'TAN' and 'KEI'.

Nanthakarn LIMKOO - Assignment #...

PV outside of the namespaces  
Accessible to the whole cluster

- Deploying Stateful Apps with StatefulSet  
- K8s Services explained

**ConfigMap and Secret**

Configuration file for your pod.

Certificate file for your pod.

The diagram illustrates the relationship between a Kubernetes Cluster and a pod named "prometheus-app". Inside the cluster, there are two blue hexagonal icons: one labeled "secret" and another labeled "cm" (ConfigMap). Arrows point from both of these icons to the "prometheus-app" pod, which is represented by a white cube icon.

**ConfigMap and Secret**

- 1) Create ConfigMap and/or Secret component
- 2) Mount that into your pod/container

```

aptVersion: v1
kind: Pod
metadata:
  name: mypod
spec:
  containers:
    - name: busybox-container
      image: busybox
      volumeMounts:
        - name: config-dir
          mountPath: /etc/config
  volumes:
    - name: config-dir
      configMap:
        name: bb-configmap

```

**Storage Class**

StorageBackend is defined in the SC component

- via "provisioner" attribute
- each storage backend has own provisioner
- **internal provisioner** - "kubernetes.io"
- **external provisioner**
- configure **parameters** for storage we want to request for PV

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: storage-class-name
provisioner: kubernetes.io/aws-ebs
parameters:
  type: io1
  iopsPerGB: "10"
  fsType: ext4
```

**Storage Class usage**

Kubernetes Cluster

```

graph TD
    Pod[Pod] --> PVC[PVC]
    PVC --> PV[PV]
    SC[SC] --> PV
  
```

- 1) Pod claims storage via PVC
- 2) PVC requests storage from SC
- 3) SC creates PV that meets the needs of the Claim

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

- Deploying Stateful Apps with StatefulSet

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

**StatefulSet for stateful applications**

**stateful applications**

- examples of stateful applications:



- don't keep record of state

- each request is completely new

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

- Deploying Stateful Apps with StatefulSet

- K8s Services explained

**Stateful and stateless applications**

**node**

HTTP

update / query

mongoDB

doesn't depend on previous data

passthrough for data query/update

- update data based on previous state

- query data

depends on most up-to-date data/state

Nanthakarn LIMKOOL - Assignment #...

File Edit View Insert Format Tools Add-ons Help

100% Normal text Times New...

- K8s Services explained

docs.google.com/document/d/1xSSM9QWsm...

Deployment of stateful and stateless applications

stateless applications

stateful applications

deployed using Deployment

deployed using StatefulSet

Both manage Pods based on container specification!

The diagram illustrates the deployment of stateless and stateful applications in Kubernetes. It features two main sections: 'stateless applications' and 'stateful applications'. Each section contains a blue hexagonal icon with a white symbol: a circular arrow for 'Deployment' and a database with arrows for 'StatefulSet'. Below each icon is a descriptive text: 'deployed using Deployment' for 'stateless applications' and 'deployed using StatefulSet' for 'stateful applications'. A central callout box states 'Both manage Pods based on container specification!'. The background of the slide shows a window titled 'K8s Services explained' containing a screenshot of a web browser displaying a complex diagram of a Node.js application architecture with various components like 'HTTP', 'node', 'up', and 'mon'.

### Deployment vs StatefulSet

The diagram illustrates a deployment setup. Three identical blue cube icons, each labeled "my-app", are connected by arrows to a central blue hexagon icon labeled "SVC". Above the pods, their respective names are listed: "my-app-f5c304e", "my-app-fxc118b", and "my-app-a36b80p". A Java logo is located in the top right corner.

- ▶ identical and interchangeable
- ▶ created in random order with random hashes
- ▶ one Service that load balances to any Pod

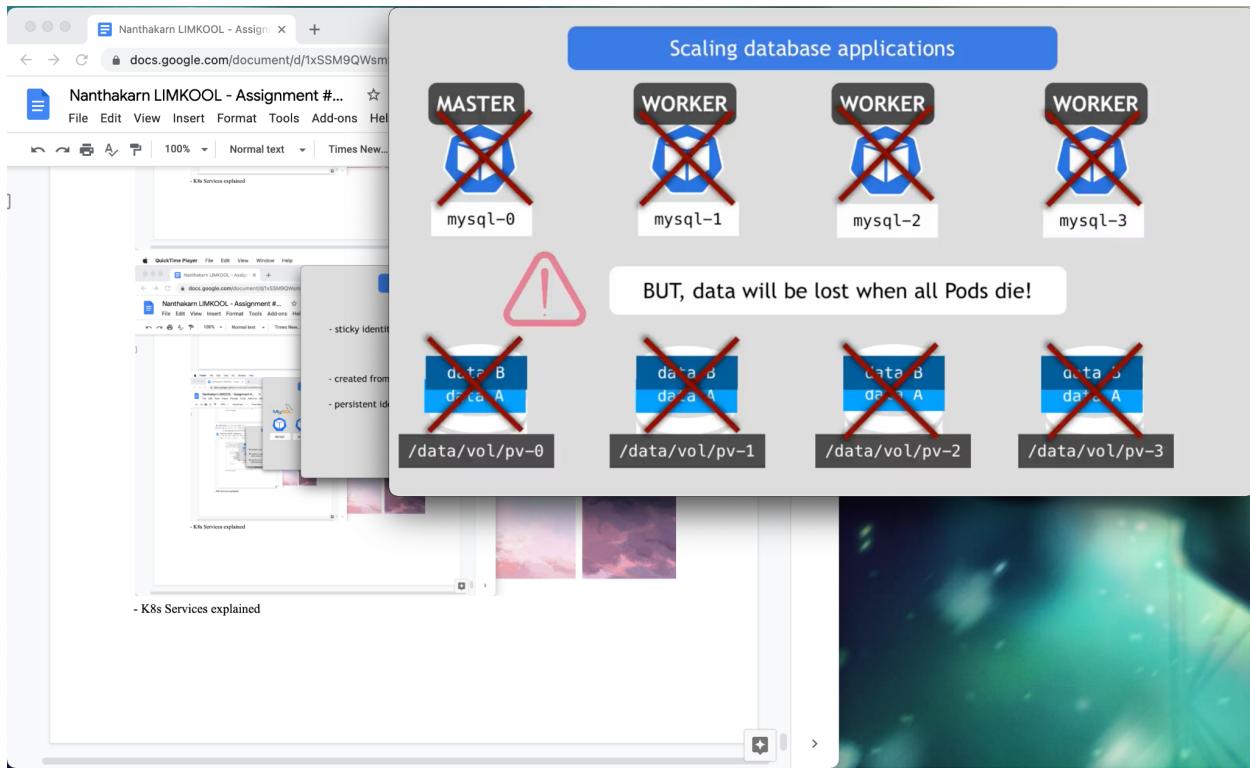
- K8s Services explained

### Deployment vs StatefulSet

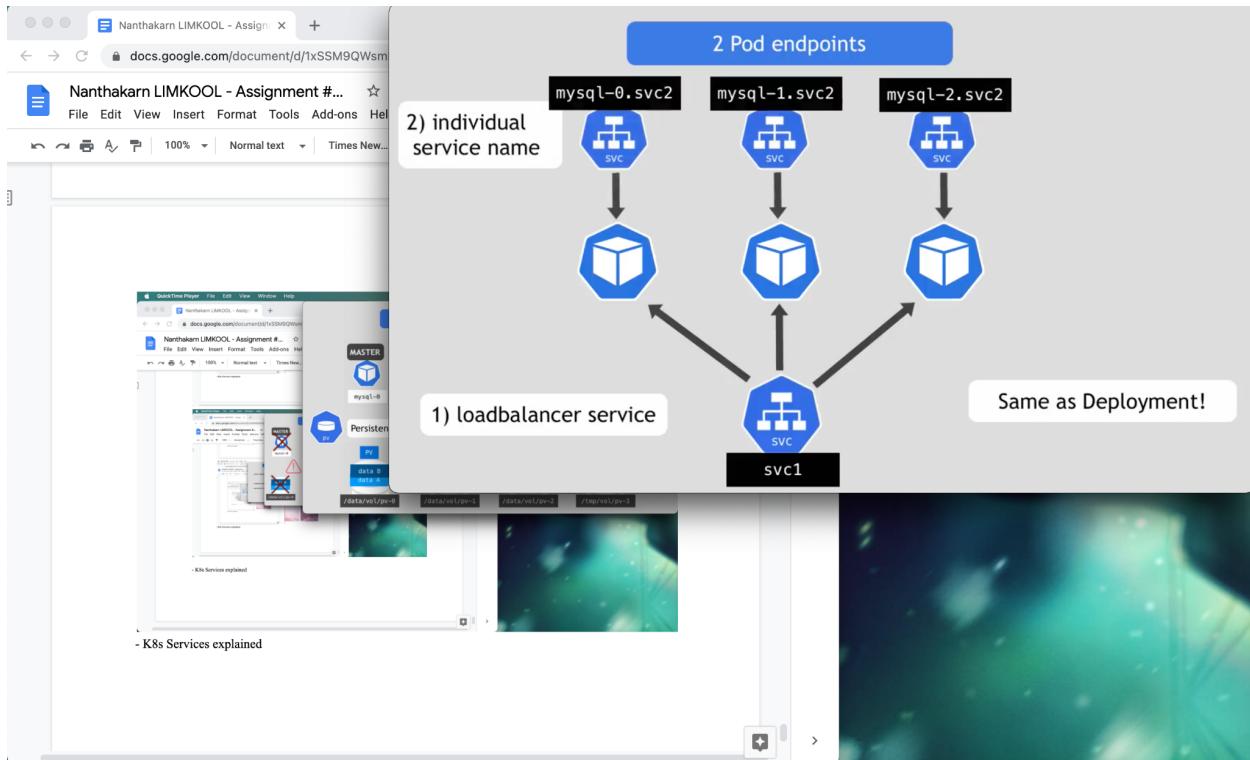
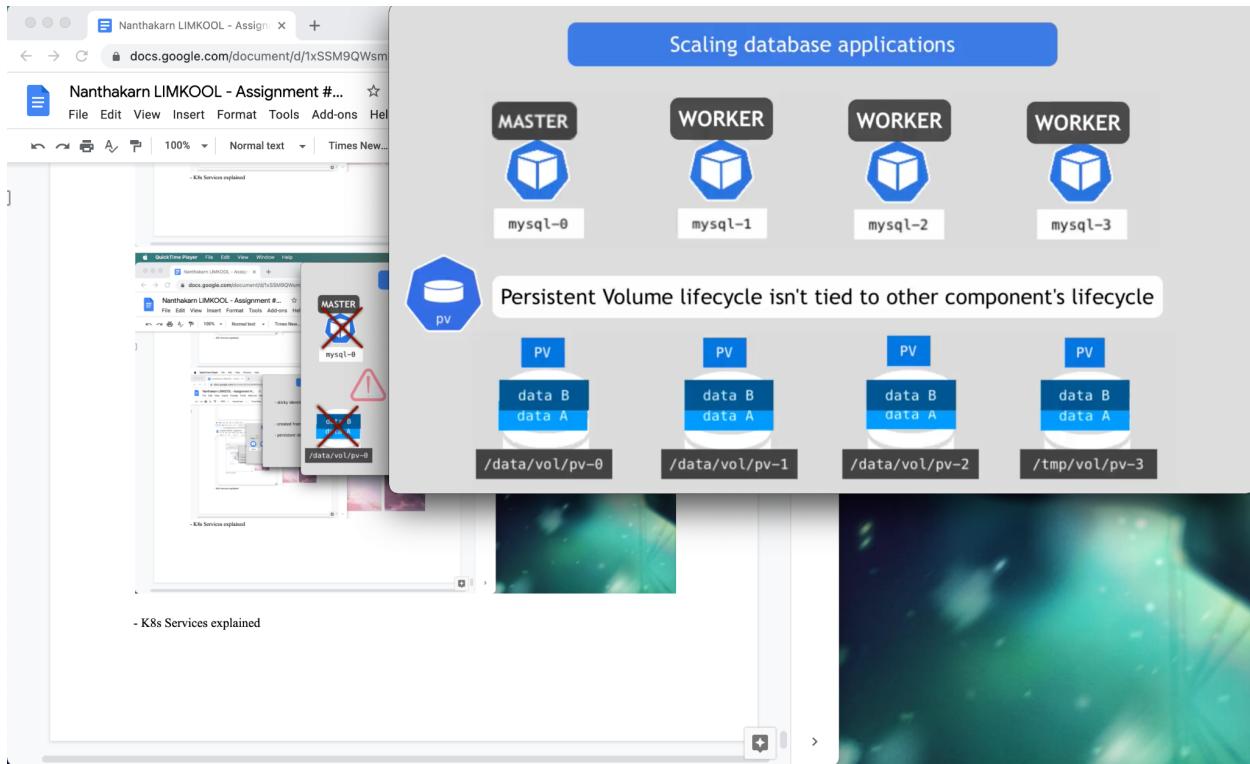
The diagram illustrates a deployment setup. Three identical blue cube icons, each labeled "mysql", are connected by arrows to a central blue hexagon icon. A MySQL logo is located in the top left corner. A speech bubble in the top right corner contains the text "more difficult".

- ▶ can't be created/deleted at same time
- ▶ can't be randomly addressed
- ▶ replica Pods are not identical  
- Pod Identity

- K8s Services explained



After config pod state



Nanthakarn LIMKOOL - Assignment #...

docs.google.com/document/d/1xSSM9QWs...

### Replicating stateful apps

- it's complex
- Kubernetes helps you
- You still need to do a lot:
  - Configuring the cloning and data synchronization
  - Make remote storage available
  - Managing and back-up

- K8s Services explained

#### - K8s Services explained

## Kubernetes Services

- What is a Kubernetes Service and when we need it?
- Different Service types explained
 

	ClusterIP Services		Headless Services
	NodePort Services		LoadBalancer Services
- Differences between them and when to use which one