****

**Namespaces**

Namespaces (like environment dev,test,prod)

Namespaces provide for a scope of Kubernetes objects. You can think of it as a workspace you’re sharing with other users. Many objects such as pods and services are namespaced, As a developer you’d usually simply use an assigned namespace, however admins may wish to manage them, for example to set up access control or resource quotas.

Namespaces are intended for use in environments with many users spread across multiple teams, or projects. For clusters with a one to tens of users, you should not need to create or think about namespaces at all.

Namespace provides an additional qualification to a resource name. This is helpful when multiple teams are using the same cluster and there is a potential of name collision. It can be as a virtual wall between multiple clusters.

Namespaces (like environment dev,test,prod)

Namespace can be used in multi-tenant environment to split up the resources like production,testing,development.

Namespaces provide a scope for names

Namespaces are a way to divide cluster resources between multiple users

\*kubernetes will list the default namespace only

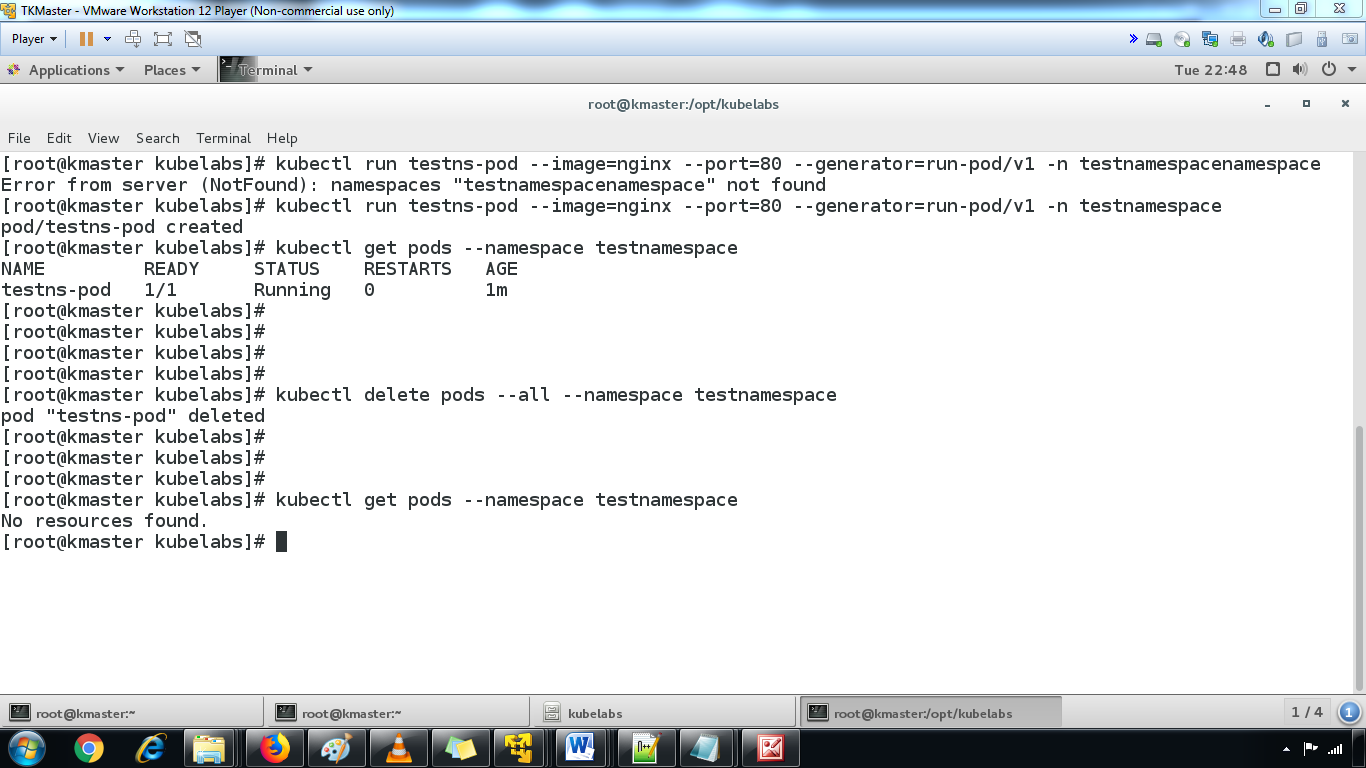
## Functionality of Namespace

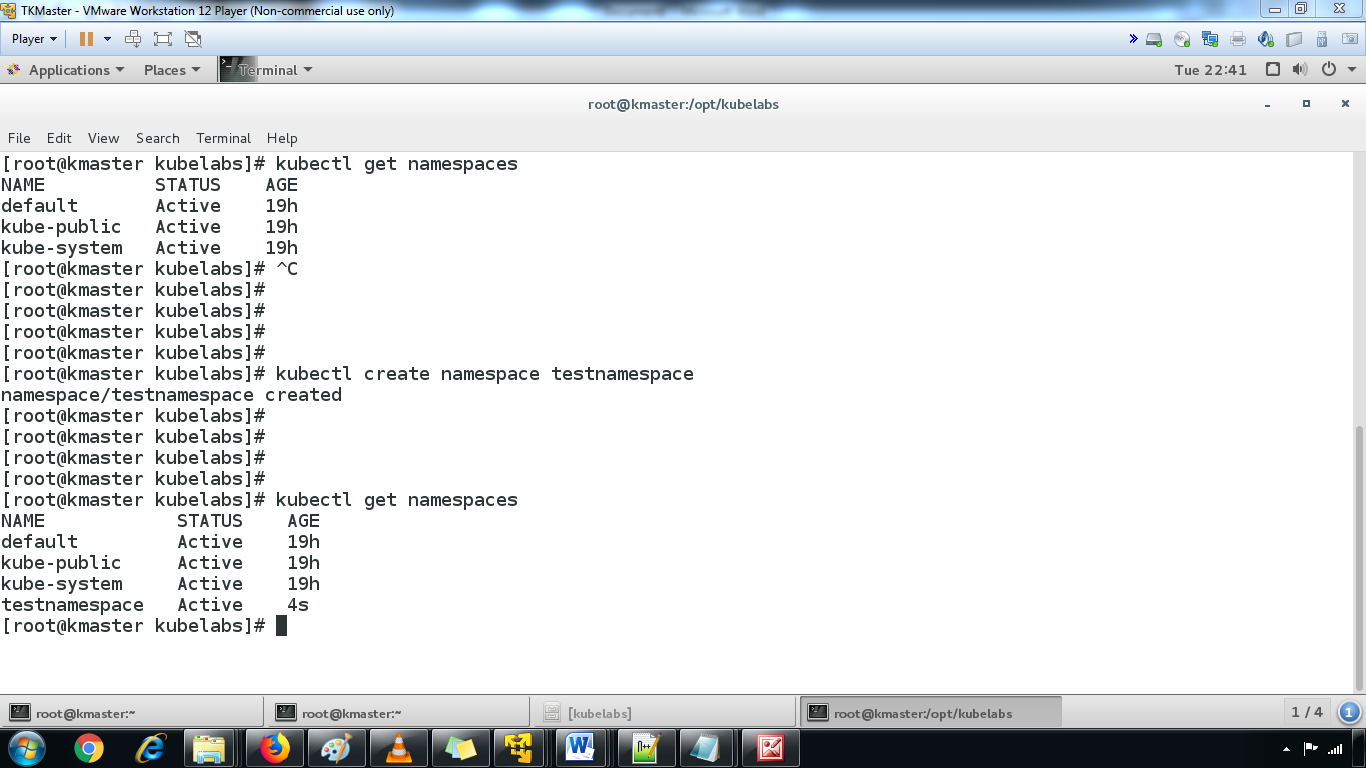
Following are some of the important functionalities of a Namespace in Kubernetes

* Namespaces help pod-to-pod communication using the same namespace.
* Namespaces are virtual clusters that can sit on top of the same physical cluster.
* They provide logical separation between the teams and their environments.
* To create a namespace, use kubectl create command.

**Syntax:**kubectl create namespace <namespace name>

**Example:**kubectl create namespace testnamespace





* To delete a namespace.

**Syntax:**kubectl delete namespace <namespace name>

**Example:**kubectl delete namespace testnamespace

### Deployments

Deployments are intended to replace Replication Controllers.  They provide the same replication functions (through Replica Sets) and also the ability to rollout changes and roll them back if necessary.

The StrategyType is RollingUpdate. This value can also be set to Recreate.

By default we have a minReadySeconds value of 0; we can change that value if we want pods to be up and running for a certain amount of time — say, to load resources — before they’re truly considered “ready”.

The RollingUpdateStrategy shows that we have a limit of 1 maxUnavailable — meaning that when we’re updating the Deployment, we can have up to 1 missing pod before it’s replaced, and 1 maxSurge, meaning we can have one extra pod as we scale the new pods back up

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

name: soaktest

spec:

replicas: 5

template:

metadata:

labels:

app: soaktest

spec:

containers:

- name: soaktest

image: nickchase/soaktest

ports:

- containerPort: 80