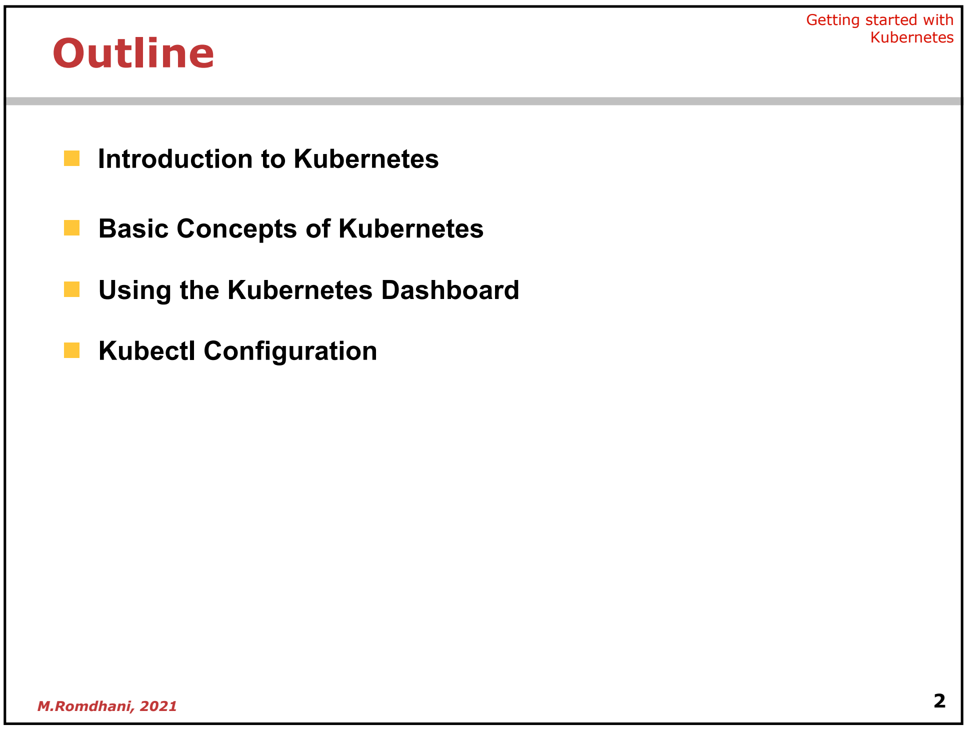
Unit 1

Getting Started with Kubernetes

Business Training

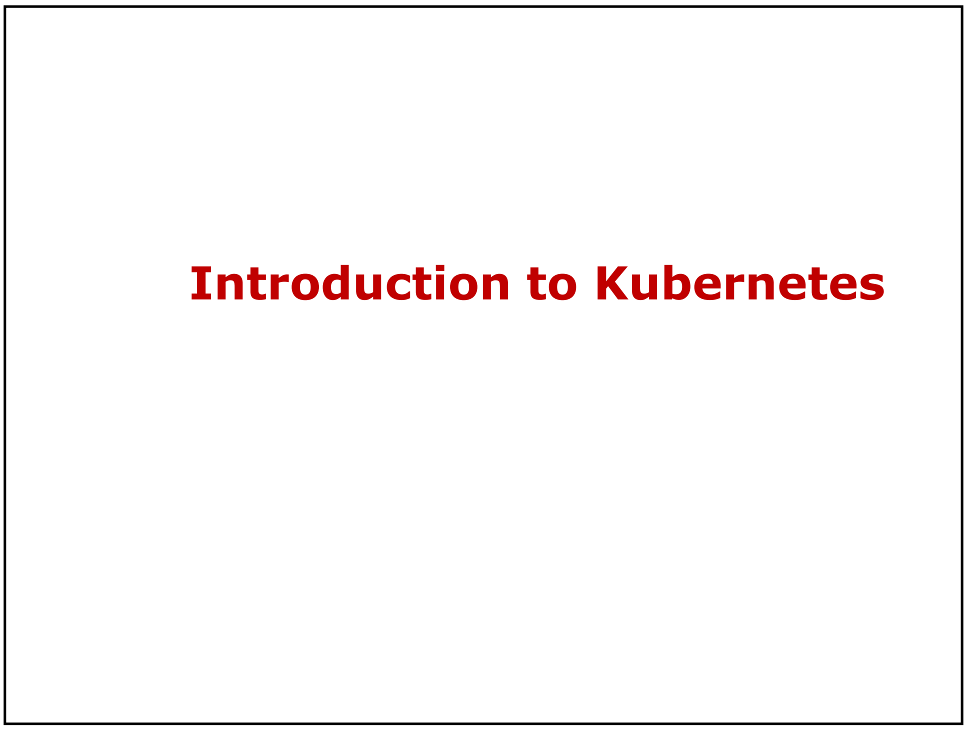
1



Outline

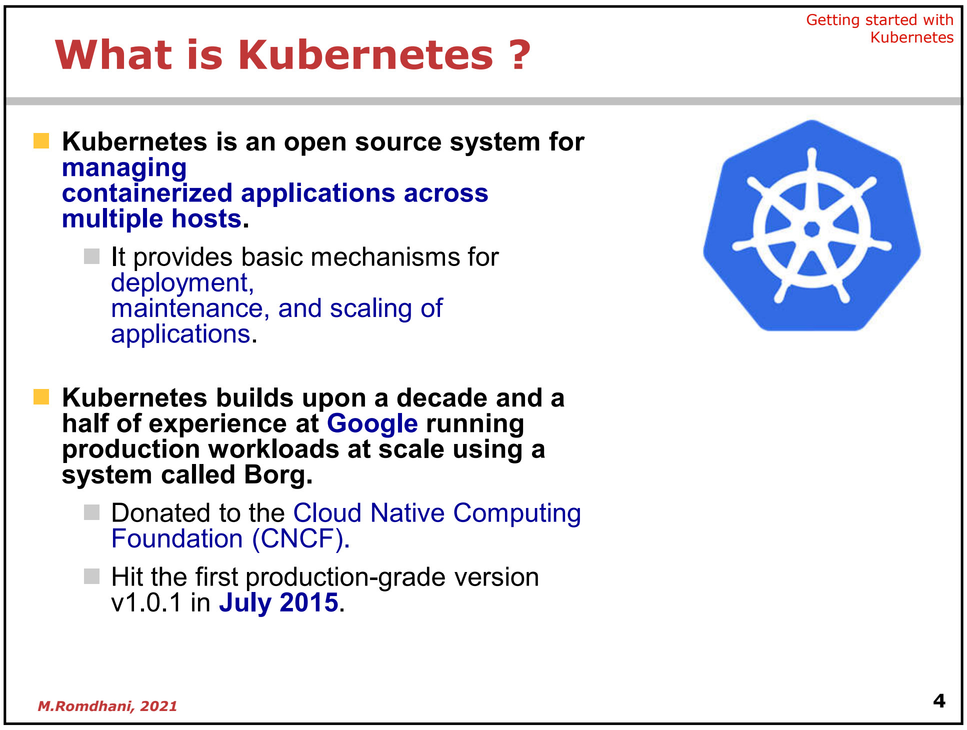
- Introduction to Kubernetes
- Basic Concepts of Kubernetes
- Using the Kubernetes Dashboard
- Kubectl Configuration

2



Introduction to Kubernetes

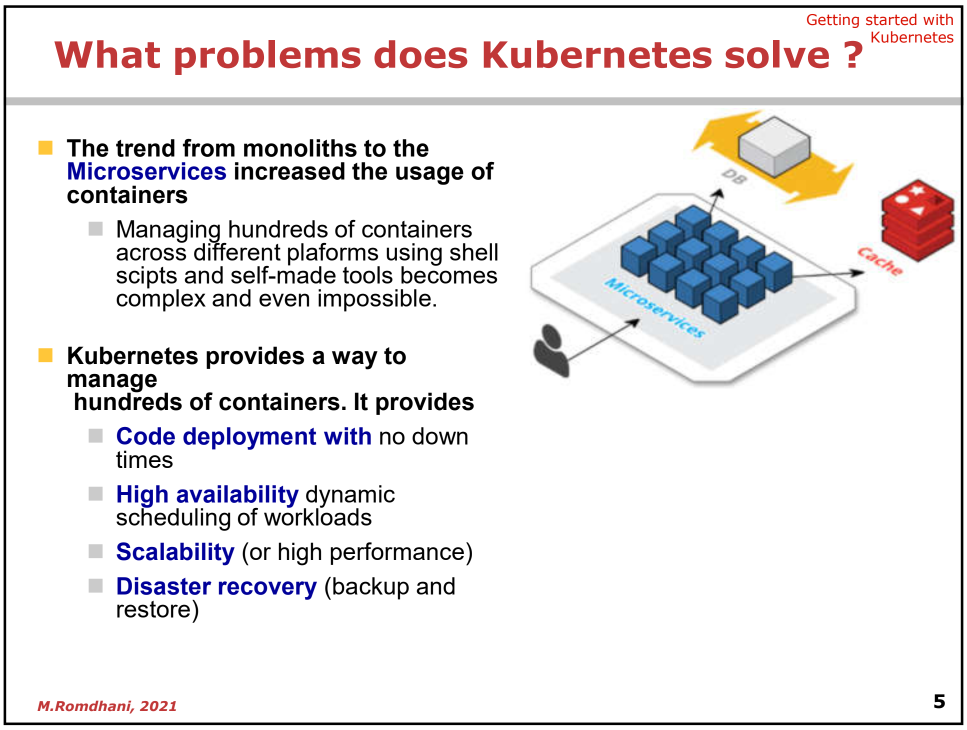
3



What is Kubernetes ?

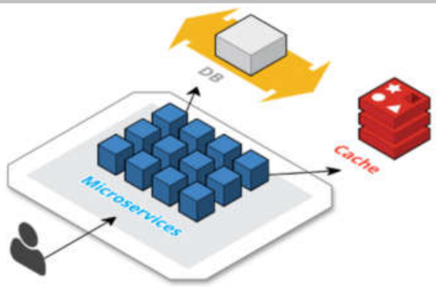
- **Kubernetes is an open source system for managing containerized applications across multiple hosts.**
  - It provides basic mechanisms for deployment, maintenance, and scaling of applications.
- **Kubernetes builds upon a decade and a half of experience at Google running production workloads at scale using a system called Borg.**
  - Donated to the Cloud Native Computing Foundation (CNCF).
  - Hit the first production-grade version v1.0.1 in **July 2015**.

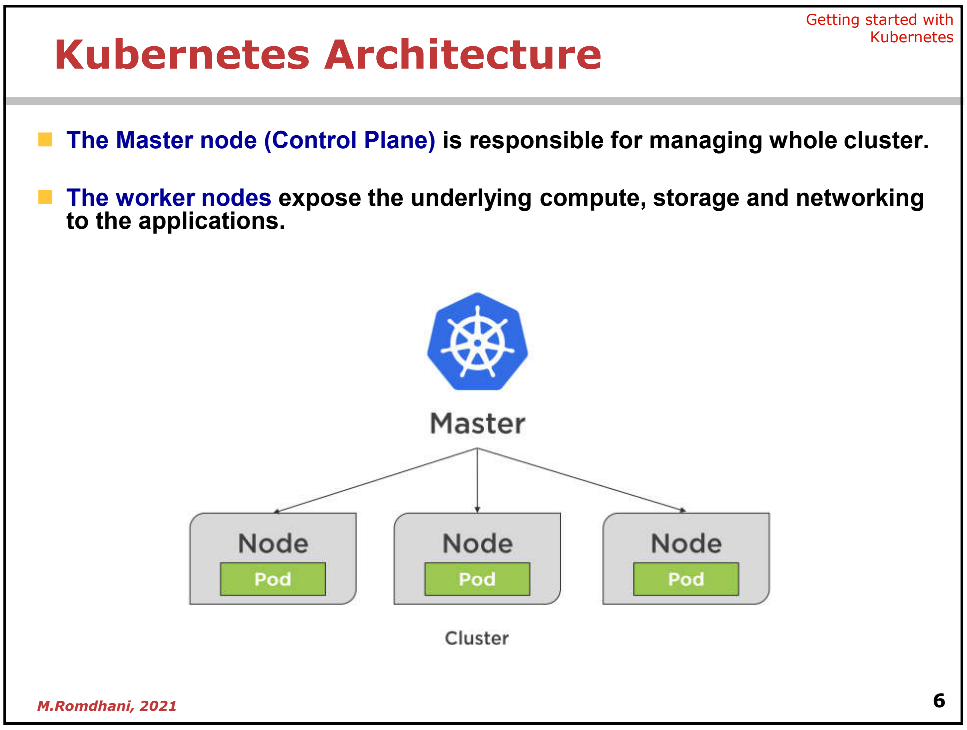
4



What problems does Kubernetes solve ?

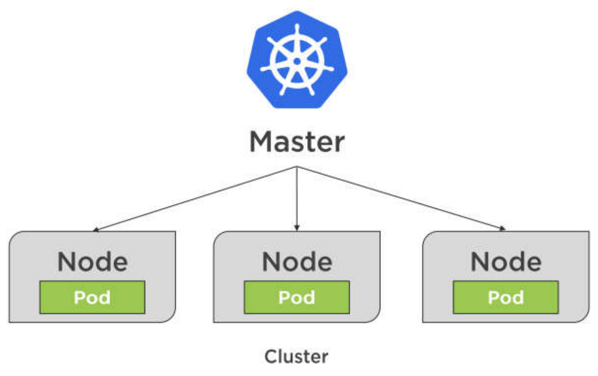
- **The trend from monoliths to the Microservices increased the usage of containers**
  - Managing hundreds of containers across different plaforms using shell scipts and self-made tools becomes complex and even impossible.
- **Kubernetes provides a way to manage hundreds of containers. It provides**
  - **Code deployment with** no down times
  - **High availability** dynamic scheduling of workloads
  - **Scalability** (or high performance)
  - **Disaster recovery** (backup and restore)



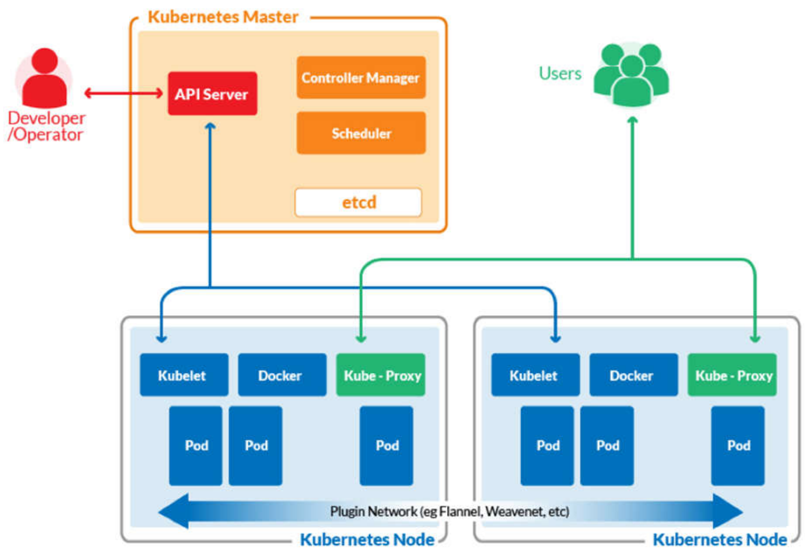


Kubernetes Architecture

- **The Master node (Control Plane) is responsible for managing whole cluster.**
- **The worker nodes expose the underlying compute, storage and networking to the applications.**

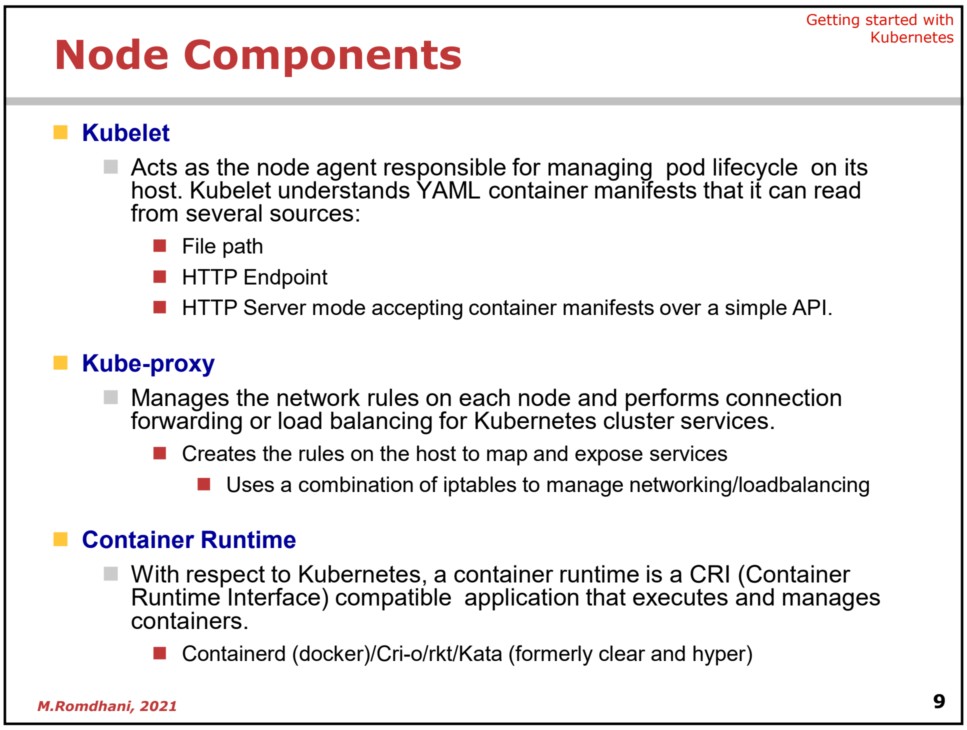


Kubernetes Architecture



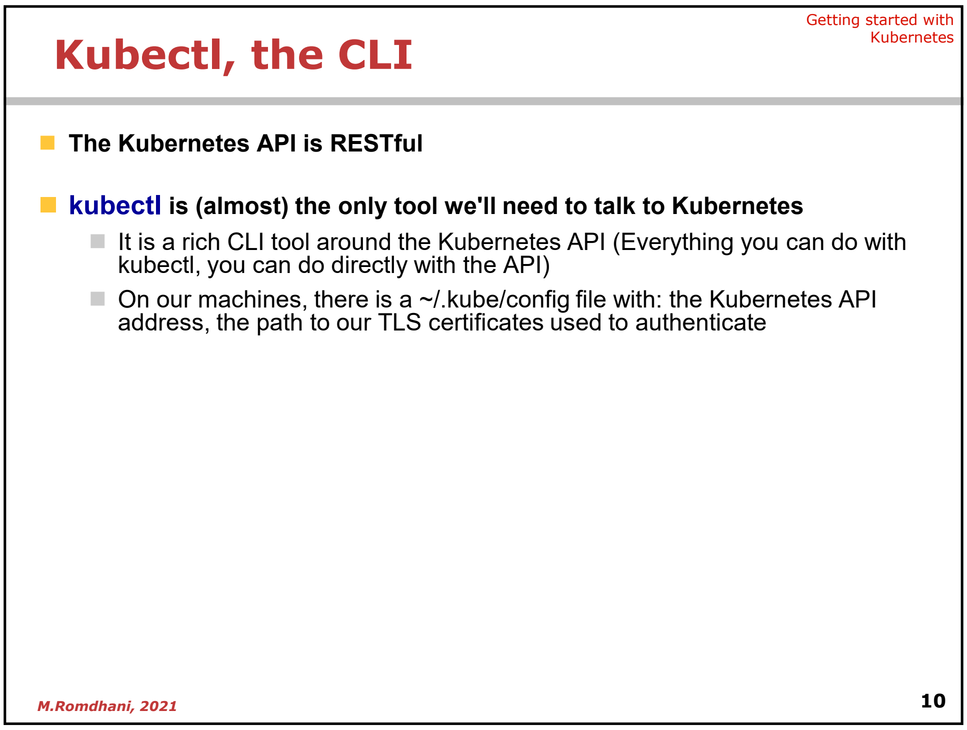
The kubernetes Master Compoents

- **The API Server**
  - The apiserver provides a facing REST interface into the kubernetes control plane and datastore. All clients, including nodes, users and other applications interact with kubernetes strictly through the API Server.
- **Etcd**
  - Etcd acts as the cluster datastore; providing a strong, consistent and highly available key-value store used for persisting cluster state.
- **The Controller Manager**
  - Controller managers are in charge of regulating the state of the system. They implement a set of the control loops that allows for Kubernetes to enforce the target desired state.
- **The Scheduler**
  - The Kubernetes scheduler is in charge of scheduling pods onto nodes. It needs to take into account individual and collective resource requirements, quality of service requirements, …



Node Components

- **Kubelet**
  - Acts as the node agent responsible for managing pod lifecycle on its host. Kubelet understands YAML container manifests that it can read from several sources:
    - File path
    - HTTP Endpoint
    - HTTP Server mode accepting container manifests over a simple API.
- **Kube-proxy**
  - Manages the network rules on each node and performs connection forwarding or load balancing for Kubernetes cluster services.
    - Creates the rules on the host to map and expose services
      - Uses a combination of iptables to manage networking/loadbalancing
- **Container Runtime**
  - With respect to Kubernetes, a container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
    - Containerd (docker)/Cri-o/rkt/Kata (formerly clear and hyper)



Kubectl, the CLI

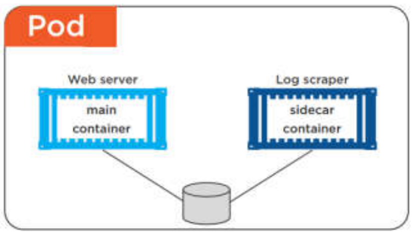
- **The Kubernetes API is RESTful**
- **kubectl is (almost) the only tool we'll need to talk to Kubernetes**
  - It is a rich CLI tool around the Kubernetes API (Everything you can do with kubectl, you can do directly with the API)
  - On our machines, there is a ~/.kube/config file with: the Kubernetes API address, the path to our TLS certificates used to authenticate

Basic Concepts of Kubernetes

11

What is a Pod ?

- **A Pod is the basic building block of Kubernetes–the smallest and simplest unit in the Kubernetes object model that you create or deploy.**
  - A Kubernetes pod is a group of containers that are deployed together on the same host and share storage and networking resources. it's very common to have a group of containers work together to produce an artifact or process a set of work.

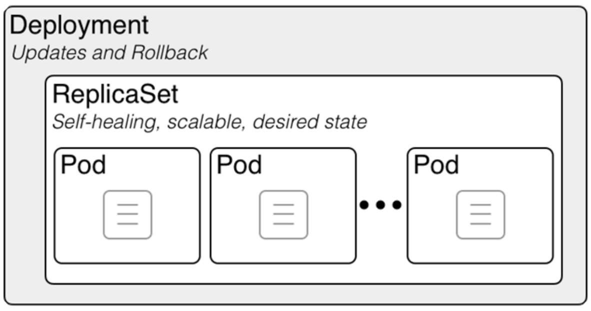


- **Containers within a Pod share an IP address and port space, and can find each other via localhost.**
- **Pods aren't intended to be treated as durable entities. They are ephemeral.**
- **Pods serve as unit of deployment, horizontal scaling, and replication.**

12

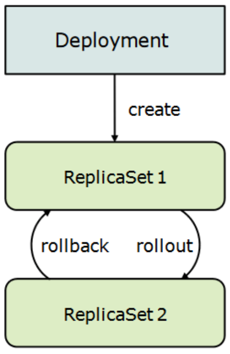
Pods/Replicasets/Deployments

- **Pods are the basic, atomically deployable unit**
- **ReplicaSet are responsible for achieving and reconciling the desired state of an application service.**
- **The Deployment augments a ReplicaSet by providing rolling update and rollback functionality on top of it.**



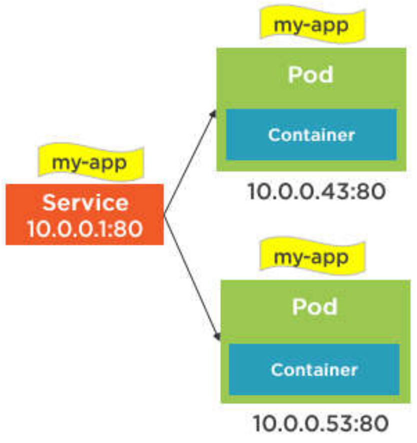
What is a deployment ?

- **ReplicaSets**
  - A ReplicaSet is a set of Pod templates that describes a set of Pod replicas. It uses a template that describes what each Pod must contain. The ReplicaSet ensures that a specified number of Pod replicas are running at any time. As such, it is often used to guarantee the availability of a specified number of identical Pods.
- **Deployment**
  - A Deployment provides declarative updates for Pods and ReplicaSets.
  - You describe a desired state in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.



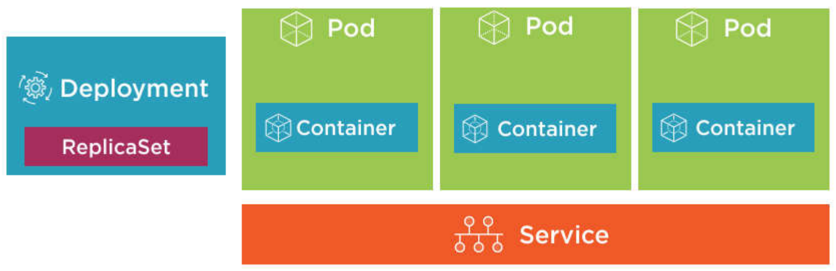
What is a Service ?

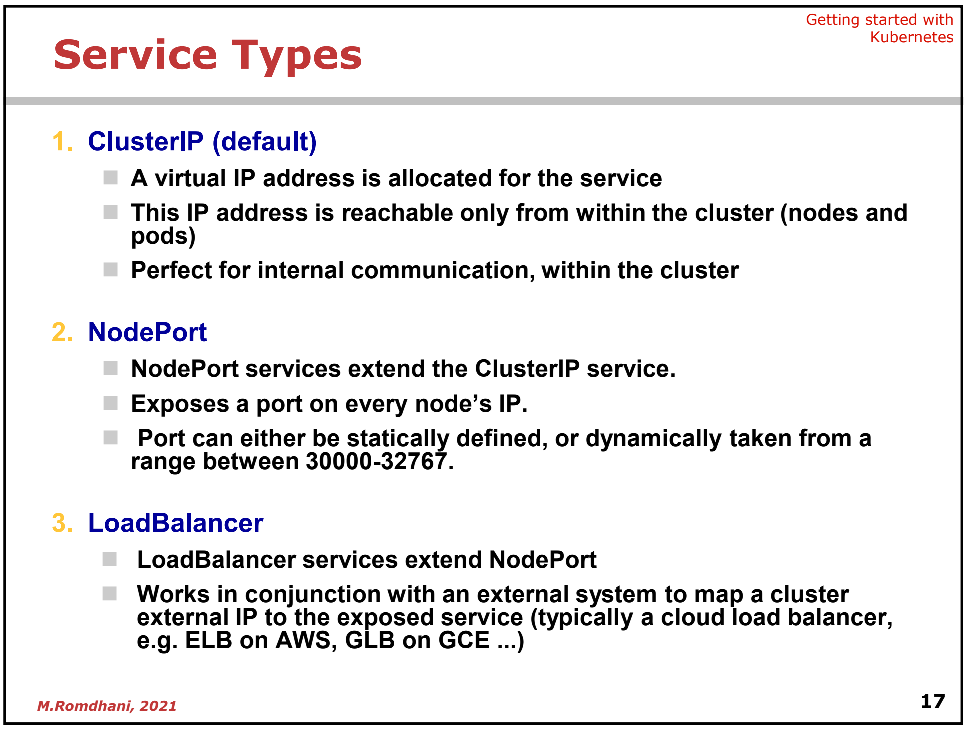
- A service is an abstract way to expose an application running on a set of Pods as a network service.
  - It acts as the unified method of accessing replicated pods.
- Services provide important features that are standardized across the cluster:
  - Services abstract Pod IP Addresses from consumers
  - Load-balancing between Pods
  - Node's kube-proxy creates virtual IP for services
  - Services are not ephemeral
- Rely on labels to associate service with a Pod



Workload Concepts, Recap

- ReplicaSet ensures that a specified number of Pod replicas are running at any given time.
- Deployment is used to change the current state to the desired state.





Getting started with Kubernetes

Service Types

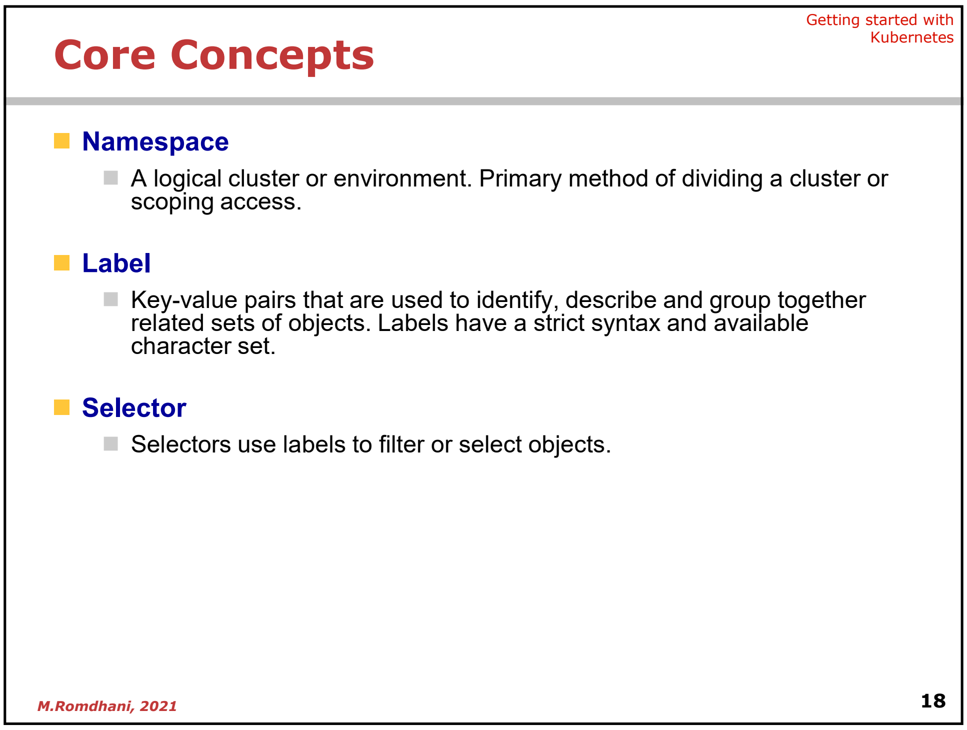
1. **ClusterIP (default)**
   - **A virtual IP address is allocated for the service**
   - **This IP address is reachable only from within the cluster (nodes and pods)**
   - **Perfect for internal communication, within the cluster**

2. **NodePort**
   - **NodePort services extend the ClusterIP service.**
   - **Exposes a port on every node's IP.**
   - **Port can either be statically defined, or dynamically taken from a range between 30000-32767.**

3. **LoadBalancer**
   - **LoadBalancer services extend NodePort**
   - **Works in conjunction with an external system to map a cluster external IP to the exposed service (typically a cloud load balancer, e.g. ELB on AWS, GLB on GCE ...)**

M.Romdhani, 2021 **17**

17



Getting started with Kubernetes

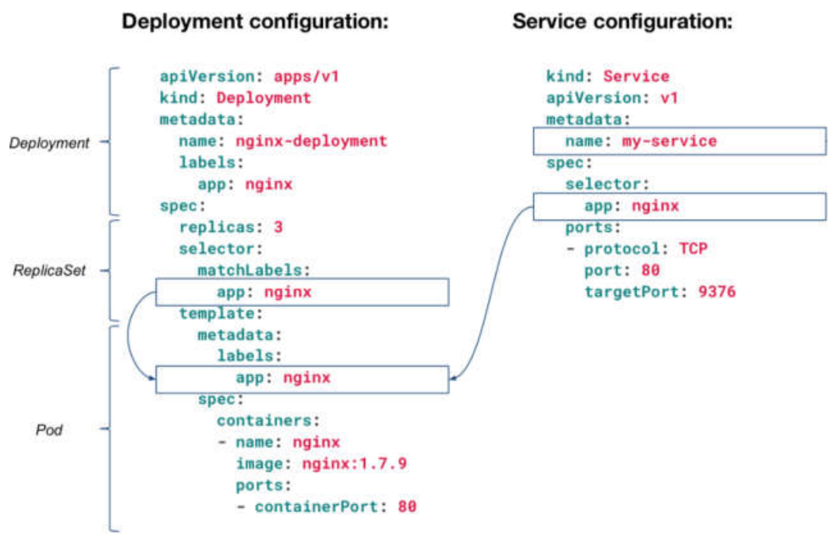
Core Concepts

- **Namespace**
  - A logical cluster or environment. Primary method of dividing a cluster or scoping access.
- **Label**
  - Key-value pairs that are used to identify, describe and group together related sets of objects. Labels have a strict syntax and available character set.
- **Selector**
  - Selectors use labels to filter or select objects.

M.Romdhani, 2021 **18**

18

Labels and Selectors



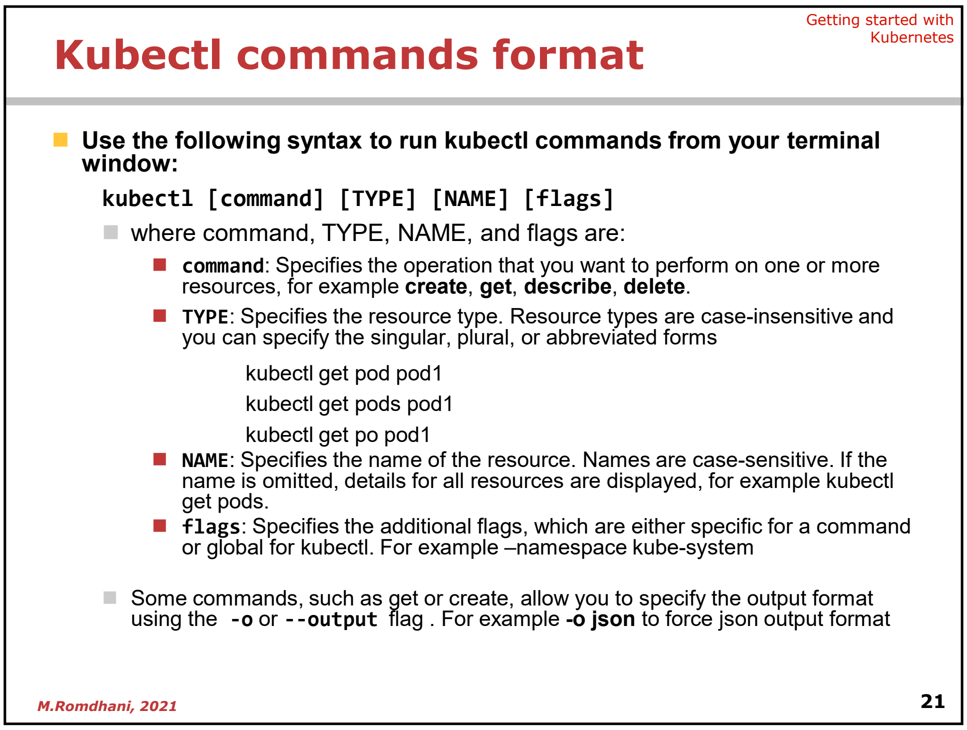
Deployment configuration:

```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:1.7.9
        ports:
        - containerPort: 80
```

Service configuration:

```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

Basic Commands of Kubectl



Kubectl commands format

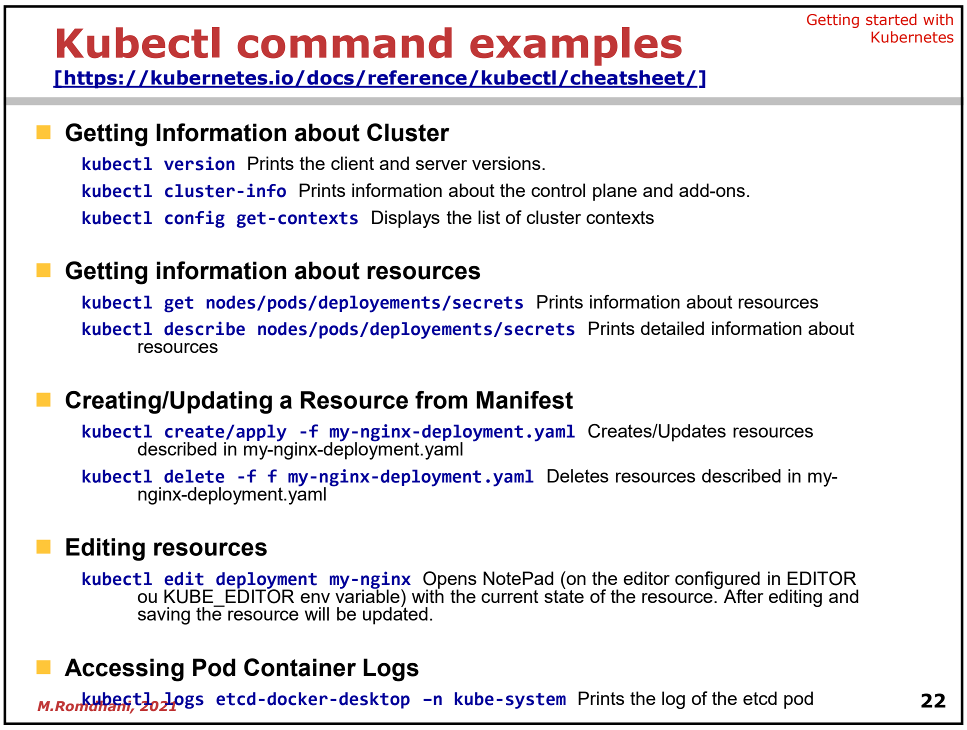
- **Use the following syntax to run kubectl commands from your terminal window:**

  `kubectl [command] [TYPE] [NAME] [flags]`

  - where command, TYPE, NAME, and flags are:
    - **`command`**: Specifies the operation that you want to perform on one or more resources, for example **create**, **get**, **describe**, **delete**.
    - **`TYPE`**: Specifies the resource type. Resource types are case-insensitive and you can specify the singular, plural, or abbreviated forms

      ```
      kubectl get pod pod1
      kubectl get pods pod1
      kubectl get po pod1
      ```
    - **`NAME`**: Specifies the name of the resource. Names are case-sensitive. If the name is omitted, details for all resources are displayed, for example kubectl get pods.
    - **`flags`**: Specifies the additional flags, which are either specific for a command or global for kubectl. For example –namespace kube-system

  - Some commands, such as get or create, allow you to specify the output format using the **`-o`** or **`--output`** flag . For example **-o json** to force json output format



Kubectl command examples

[https://kubernetes.io/docs/reference/kubectl/cheatsheet/]

- **Getting Information about Cluster**

  `kubectl version` Prints the client and server versions.

  `kubectl cluster-info` Prints information about the control plane and add-ons.

  `kubectl config get-contexts` Displays the list of cluster contexts

- **Getting information about resources**

  `kubectl get nodes/pods/deployements/secrets` Prints information about resources

  `kubectl describe nodes/pods/deployements/secrets` Prints detailed information about resources

- **Creating/Updating a Resource from Manifest**

  `kubectl create/apply -f my-nginx-deployment.yaml` Creates/Updates resources described in my-nginx-deployment.yaml

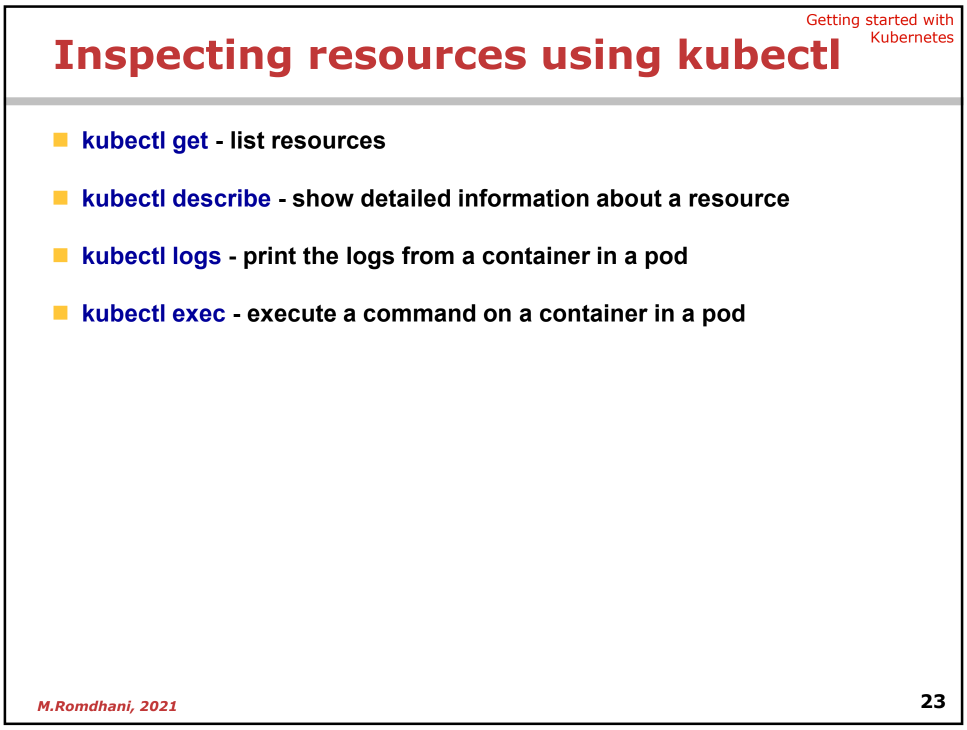
  `kubectl delete -f f my-nginx-deployment.yaml` Deletes resources described in my-nginx-deployment.yaml

- **Editing resources**

  `kubectl edit deployment my-nginx` Opens NotePad (on the editor configured in EDITOR ou KUBE_EDITOR env variable) with the current state of the resource. After editing and saving the resource will be updated.

- **Accessing Pod Container Logs**

  `kubectl logs etcd-docker-desktop -n kube-system` Prints the log of the etcd pod



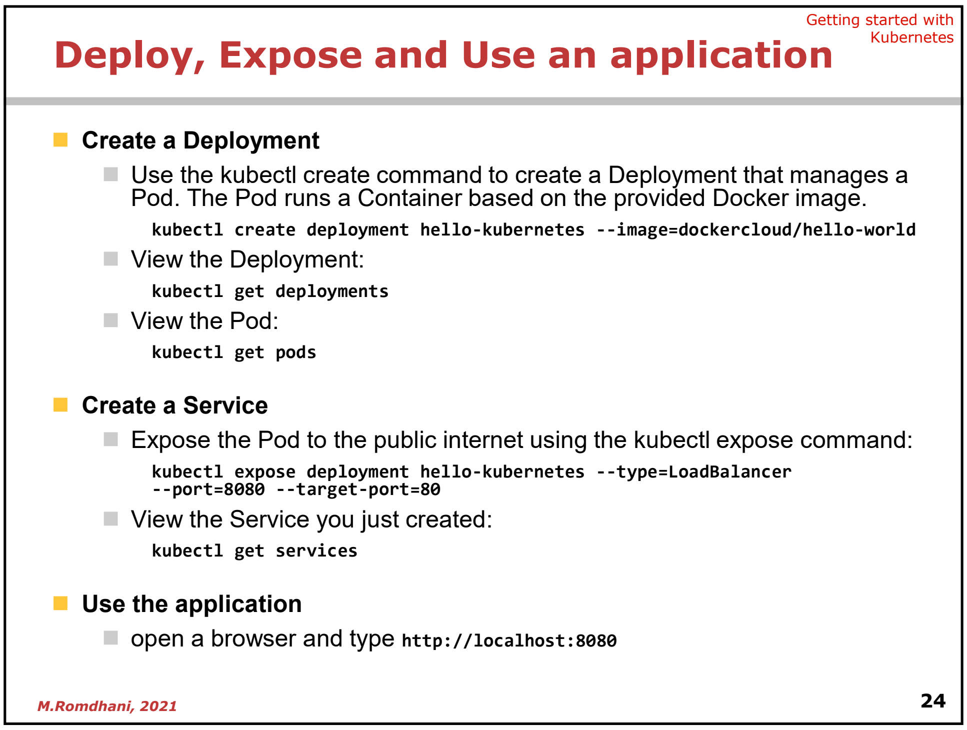
Getting started with Kubernetes

Inspecting resources using kubectl

- **kubectl get - list resources**
- **kubectl describe - show detailed information about a resource**
- **kubectl logs - print the logs from a container in a pod**
- **kubectl exec - execute a command on a container in a pod**

M.Romdhani, 2021 **23**

23



Getting started with Kubernetes

Deploy, Expose and Use an application

- **Create a Deployment**
  - Use the kubectl create command to create a Deployment that manages a Pod. The Pod runs a Container based on the provided Docker image.

    ```
    kubectl create deployment hello-kubernetes --image=dockercloud/hello-world
    ```
  - View the Deployment:

    ```
    kubectl get deployments
    ```
  - View the Pod:

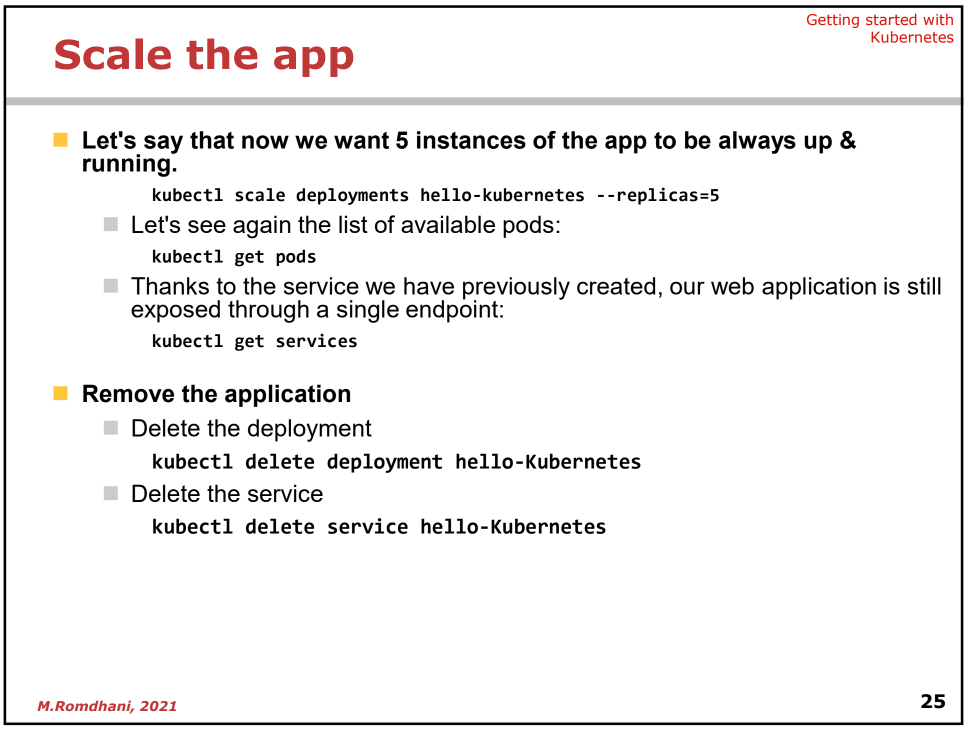
    ```
    kubectl get pods
    ```
- **Create a Service**
  - Expose the Pod to the public internet using the kubectl expose command:

    ```
    kubectl expose deployment hello-kubernetes --type=LoadBalancer
    --port=8080 --target-port=80
    ```
  - View the Service you just created:

    ```
    kubectl get services
    ```
- **Use the application**
  - open a browser and type `http://localhost:8080`

M.Romdhani, 2021 **24**

24



Scale the app

- **Let's say that now we want 5 instances of the app to be always up & running.**

  ```
  kubectl scale deployments hello-kubernetes --replicas=5
  ```

  - Let's see again the list of available pods:

    ```
    kubectl get pods
    ```

  - Thanks to the service we have previously created, our web application is still exposed through a single endpoint:

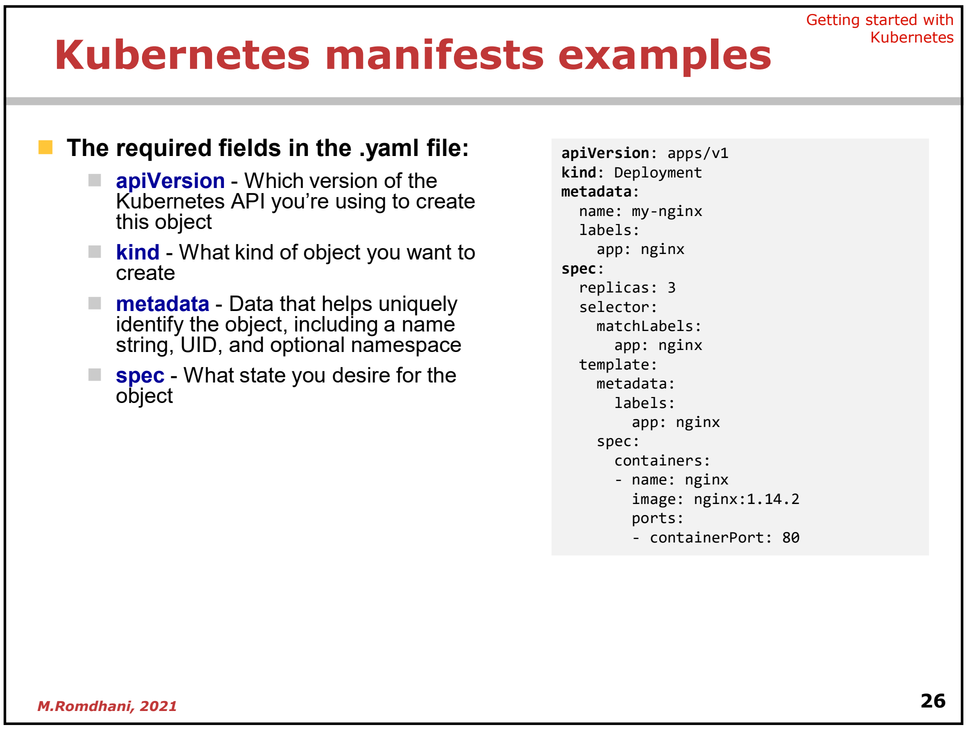
    ```
    kubectl get services
    ```

- **Remove the application**
  - Delete the deployment

    ```
    kubectl delete deployment hello-Kubernetes
    ```

  - Delete the service

    ```
    kubectl delete service hello-Kubernetes
    ```



Kubernetes manifests examples

- **The required fields in the .yaml file:**
  - apiVersion - Which version of the Kubernetes API you're using to create this object
  - kind - What kind of object you want to create
  - metadata - Data that helps uniquely identify the object, including a name string, UID, and optional namespace
  - spec - What state you desire for the object

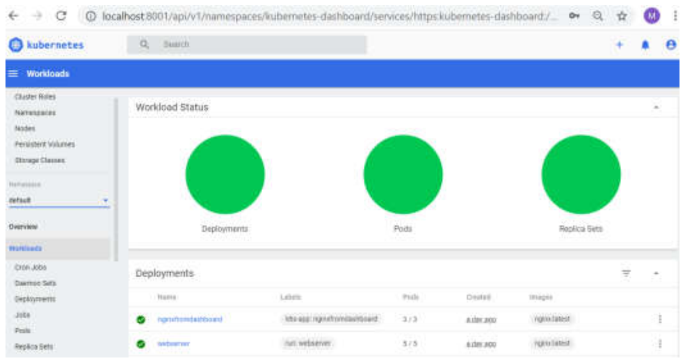
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Using the Kubernetes dashboard

27

Kubernetes Dashboard

- **Dashboard is a web-based Kubernetes user interface.**
  - You can use Dashboard to deploy containerized applications to a Kubernetes cluster, troubleshoot your containerized application, and manage the cluster resources.
  - You can use Dashboard to get an overview of applications running on your cluster, as well as for creating or modifying individual Kubernetes resources (such as Deployments, Services, etc).



	Name	Labels	Pods	Created	Images
✓	nginxfromdashboard	k8s-app: nginxfromdashboard	3 / 3	a day ago	nginx:latest
✓	webserver	run: webserver	5 / 5	a day ago	nginx:latest

28

Installing Kubernetes Dashboard

- **To deploy it, run the following command:**

```
$ kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.0.0/aio/deploy/recommended.yaml
```

  - On minikube : to launch the dashboard, just run `$ minikube dashboard`

- **To access Dashboard from your local workstation, run the following command:**

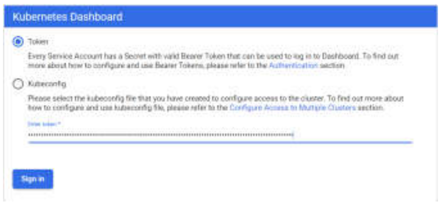
```
$ kubectl proxy
```

  - It will proxy server between your machine and Kubernetes API server.

- **To view the dashboard in the browser, navigate to the following address in the browser of your Master VM:**

```
http://localhost:8001/api/v1/namespaces/kubernetes-
dashboard/services/https:kubernetes-dashboard:/proxy/
```

  - You will then be prompted with this page, to enter



Kubernetes Dashboard

[✓ Token]

Every Service Account has a Secret with valid Bearer Token that can be used to log in to Dashboard. To find out more about how to configure and use Bearer Tokens, please refer to the Authentication section.

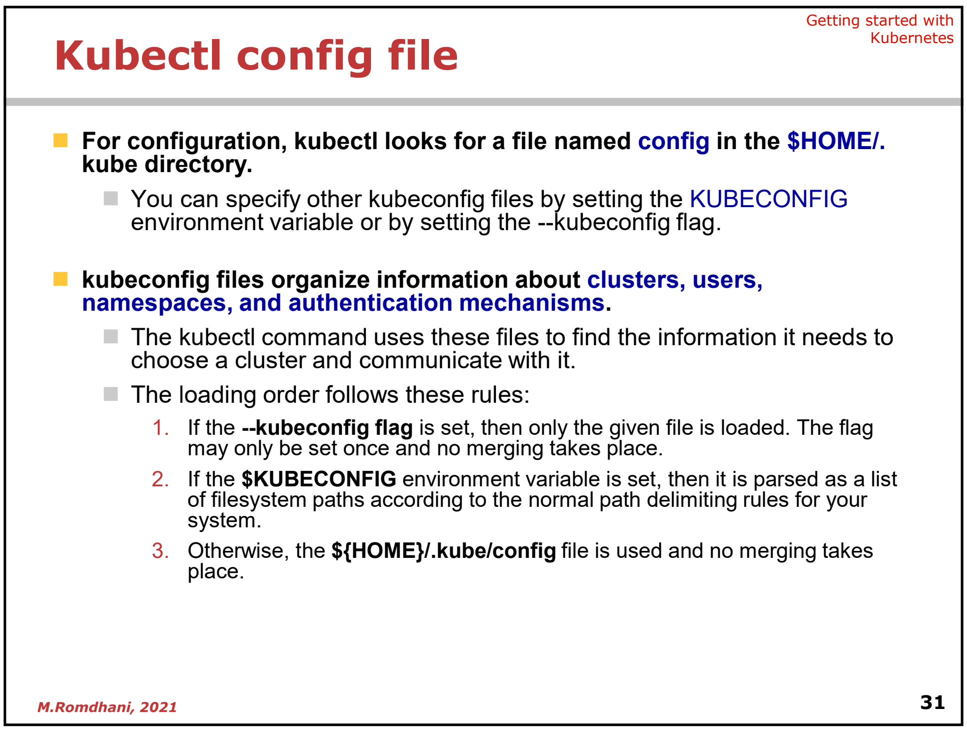
Kubeconfig

Please select the kubeconfig file that you have created to configure access to the cluster. To find out more about how to configure and use kubeconfig file, please refer to the Configure Access to Multiple Clusters section.

Enter token: *

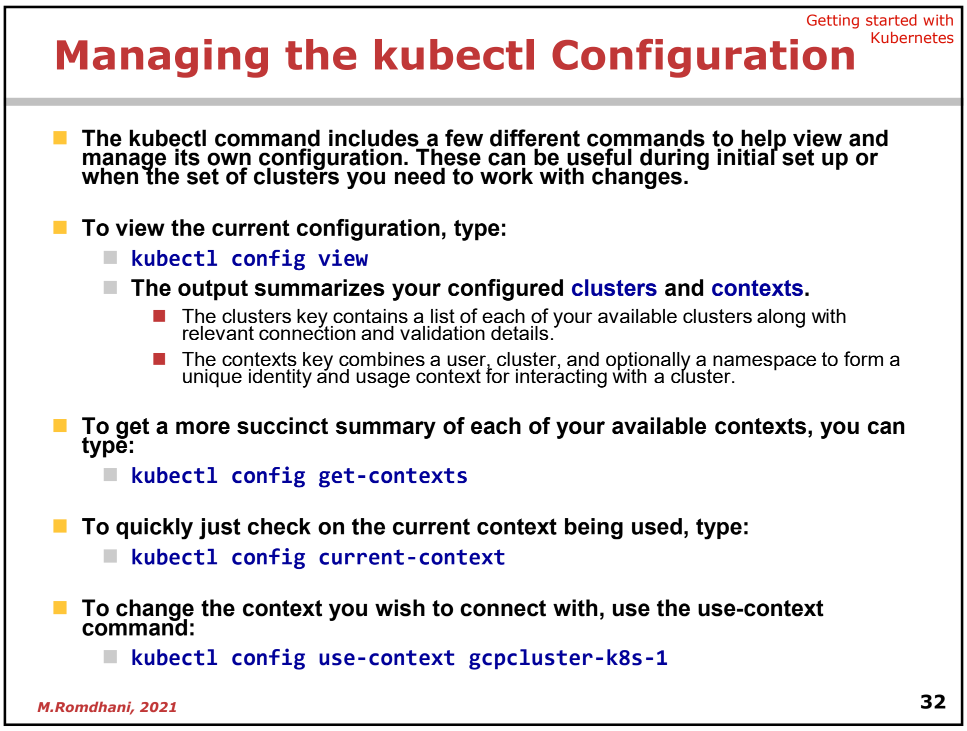
- **Get an Authentication Token**
  - **Get account**
    - `kubectl -n kubernetes-dashboard describe secret default`

Kubectl Configuration



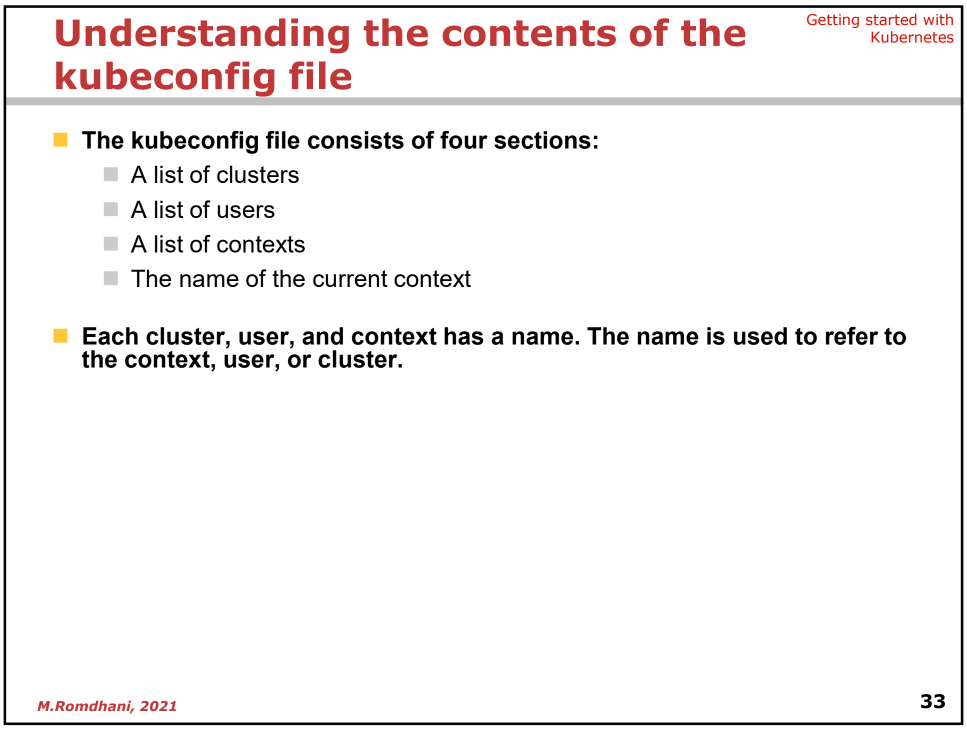
Kubectl config file

- **For configuration, kubectl looks for a file named config in the $HOME/. kube directory.**
  - You can specify other kubeconfig files by setting the KUBECONFIG environment variable or by setting the --kubeconfig flag.
- **kubeconfig files organize information about clusters, users, namespaces, and authentication mechanisms.**
  - The kubectl command uses these files to find the information it needs to choose a cluster and communicate with it.
  - The loading order follows these rules:
    1. If the **--kubeconfig flag** is set, then only the given file is loaded. The flag may only be set once and no merging takes place.
    2. If the **$KUBECONFIG** environment variable is set, then it is parsed as a list of filesystem paths according to the normal path delimiting rules for your system.
    3. Otherwise, the **${HOME}/.kube/config** file is used and no merging takes place.



Managing the kubectl Configuration

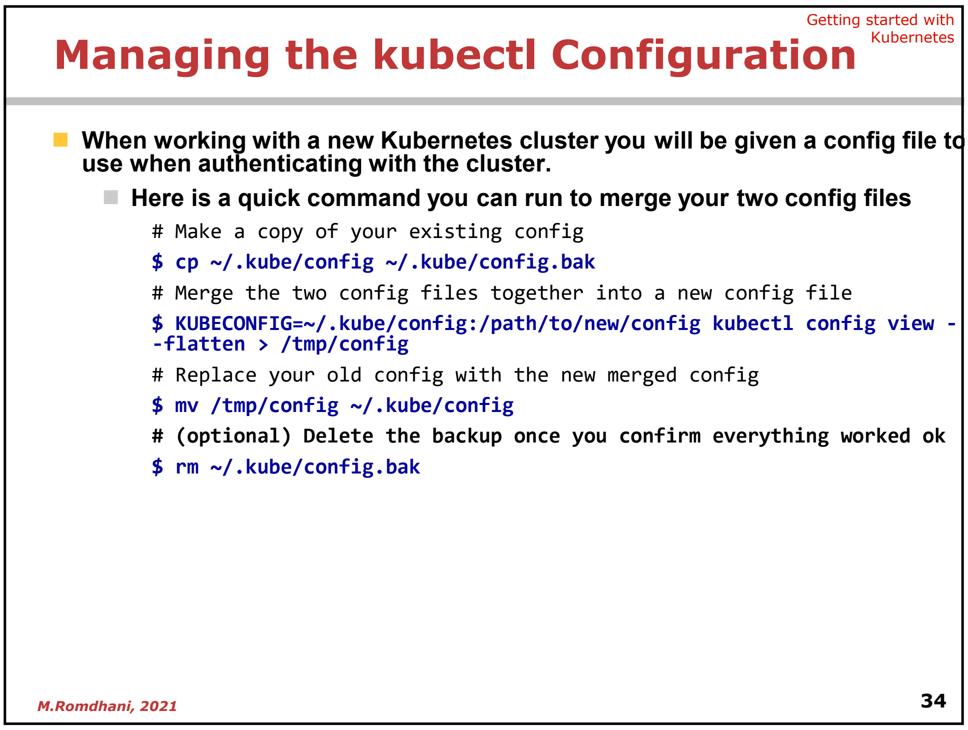
- **The kubectl command includes a few different commands to help view and manage its own configuration. These can be useful during initial set up or when the set of clusters you need to work with changes.**
- **To view the current configuration, type:**
  - `kubectl config view`
  - **The output summarizes your configured clusters and contexts.**
    - The clusters key contains a list of each of your available clusters along with relevant connection and validation details.
    - The contexts key combines a user, cluster, and optionally a namespace to form a unique identity and usage context for interacting with a cluster.
- **To get a more succinct summary of each of your available contexts, you can type:**
  - `kubectl config get-contexts`
- **To quickly just check on the current context being used, type:**
  - `kubectl config current-context`
- **To change the context you wish to connect with, use the use-context command:**
  - `kubectl config use-context gcpcluster-k8s-1`



Understanding the contents of the kubeconfig file

- **The kubeconfig file consists of four sections:**
  - A list of clusters
  - A list of users
  - A list of contexts
  - The name of the current context
- **Each cluster, user, and context has a name. The name is used to refer to the context, user, or cluster.**

33

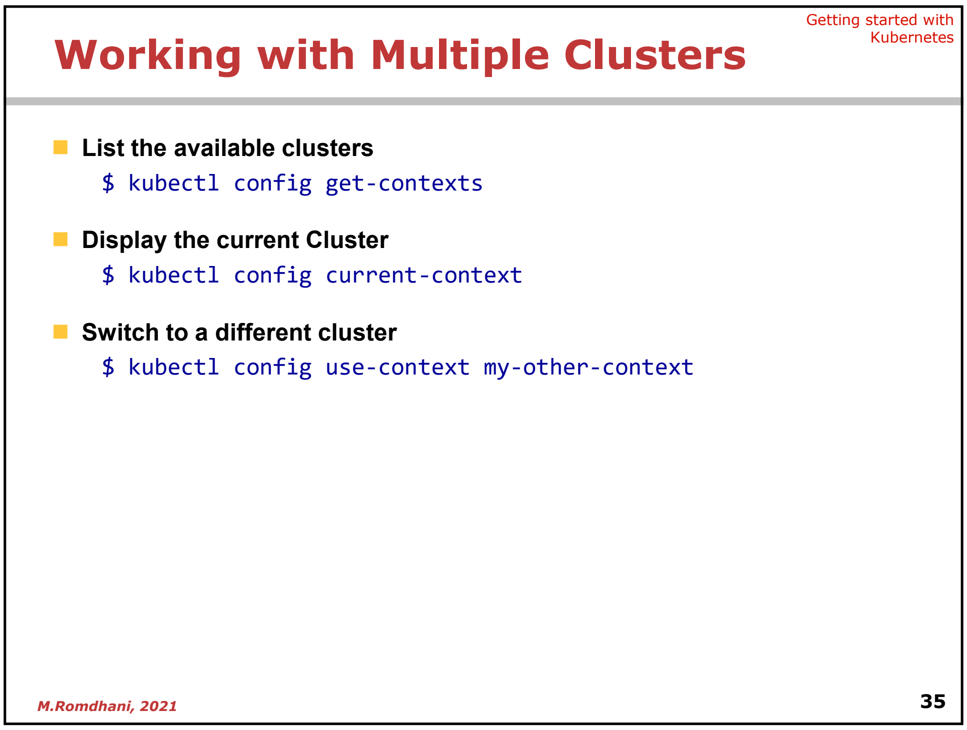


Managing the kubectl Configuration

- **When working with a new Kubernetes cluster you will be given a config file to use when authenticating with the cluster.**
  - **Here is a quick command you can run to merge your two config files**

```
# Make a copy of your existing config
$ cp ~/.kube/config ~/.kube/config.bak
# Merge the two config files together into a new config file
$ KUBECONFIG=~/.kube/config:/path/to/new/config kubectl config view -
-flatten > /tmp/config
# Replace your old config with the new merged config
$ mv /tmp/config ~/.kube/config
# (optional) Delete the backup once you confirm everything worked ok
$ rm ~/.kube/config.bak
```

34



Working with Multiple Clusters

- **List the available clusters**

```
$ kubectl config get-contexts
```

- **Display the current Cluster**

```
$ kubectl config current-context
```

- **Switch to a different cluster**

```
$ kubectl config use-context my-other-context
```

35