KUBERNETES

(Part 2)

Prepared by: Aamir Pinger

fb.com/AamirPingerOfficial

linkedin.com/in/AamirPinger

github.com/AamirPinger

Minikube

- Kubernetes architecture consist of Master node and worker nodes
- Minikube is a tool that makes it easy to run Kubernetes locally
- Minikube runs a single-node Kubernetes cluster on your laptop to use kubernetes for practice or development
- To start Kubernetes cluster using minikube

```
aamir@ap-linux:~$ minikube start
😄  minikube v1.1.1 on linux (amd64)
🔥  Creating virtualbox VM (CPUs=2, Memory=2048MB, Disk=20000MB) ...
🐳  Configuring environment for Kubernetes v1.14.3 on Docker 18.09.6
🚜  Pulling images ...
🚀  Launching Kubernetes ...
⌛  Verifying: apiserver proxy etcd scheduler controller dns
🏄  Done! kubectl is now configured to use "minikube"
```

Source: https://kubernetes.io/docs/setup/learning-environment/minikube/

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Minikube

- To check the status of minikube

```
aamir@ap-linux:~$ minikube status
host: Running
kubelet: Running
apiserver: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.100
```

- To check addresses of the kubernetes master and services

```
aamir@ap-linux:~$ kubectl cluster-info
Kubernetes master is running at https://192.168.99.100:8443
KubeDNS is running at https://192.168.99.100:8443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

NODES

Nodes

- Group of server is called a cluster
- Kubernetes cluster is consist of master node and worker nodes
- Master node act as a manager
- Worker nodes are server or system on which kubernetes deploys and run our applications
- To list down nodes in kubernetes cluster

```
aamir@ap-linux:~$ kubectl get nodes
NAME        STATUS      ROLES       AGE   VERSION
Minikube    Ready       master      1m    v1.10.0
```

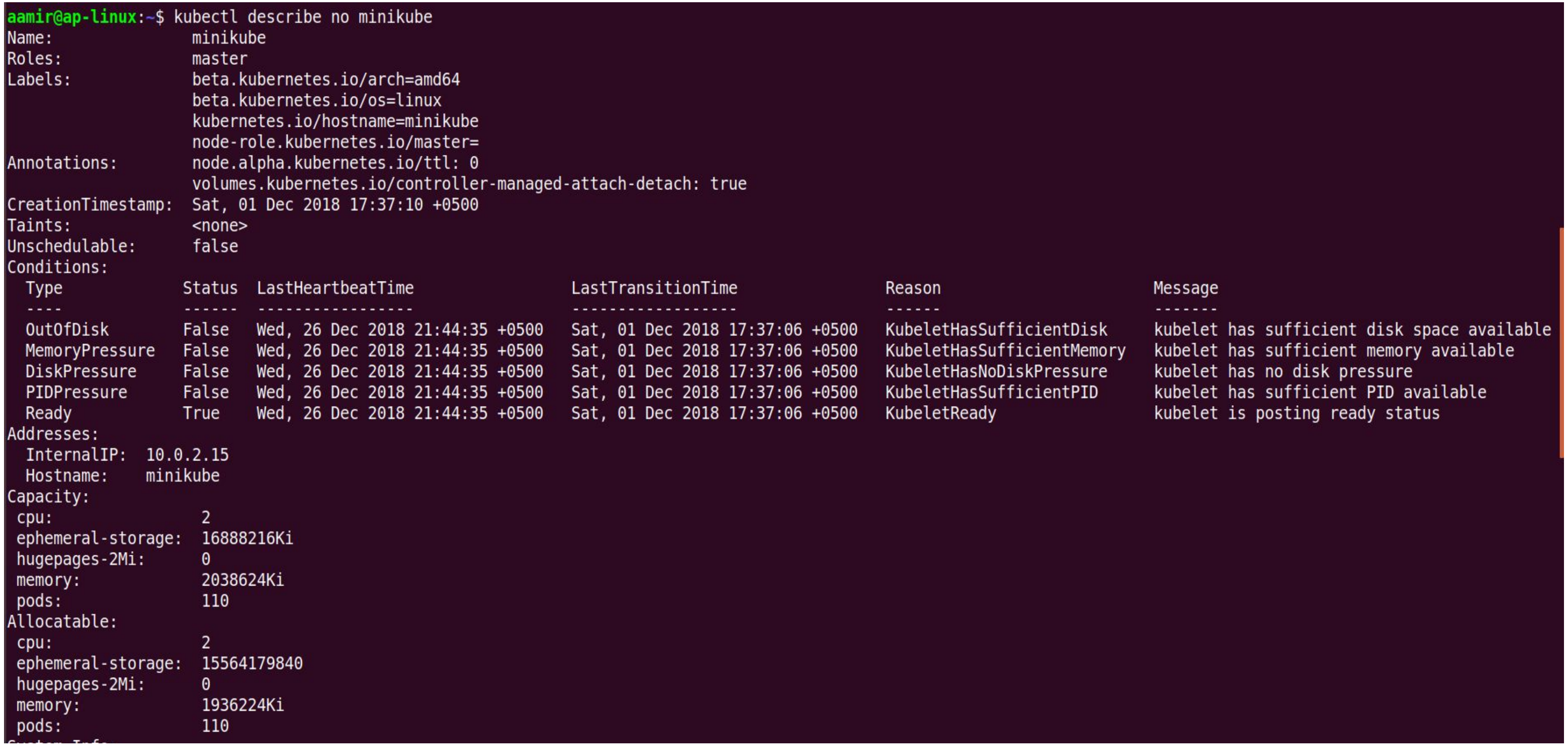
You also can use **kubectl get no** command

Nodes

- To check the additional detail of node

```
aamir@ap-linux:~$ kubectl describe nodes minikube
```

- This command show many important things related to worker node, few are
  - Labels
  - Annotations
  - Unschedulable
  - Capacity
  - System Info
  - Allocated resources
  - Events etc



```
aamir@ap-linux:~$ kubectl describe no minikube
Name:               minikube
Roles:              master
Labels:             beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/os=linux
                    kubernetes.io/hostname=minikube
                    node-role.kubernetes.io/master=
Annotations:        node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Sat, 01 Dec 2018 17:37:10 +0500
Taints:             <none>
Unschedulable:      false
Conditions:
  Type             Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----             ------  -----------------                 ------------------                ------                       -------
  OutOfDisk        False   Wed, 26 Dec 2018 21:44:35 +0500   Sat, 01 Dec 2018 17:37:06 +0500   KubeletHasSufficientDisk     kubelet has sufficient disk space available
  MemoryPressure   False   Wed, 26 Dec 2018 21:44:35 +0500   Sat, 01 Dec 2018 17:37:06 +0500   KubeletHasSufficientMemory   kubelet has sufficient memory available
  DiskPressure     False   Wed, 26 Dec 2018 21:44:35 +0500   Sat, 01 Dec 2018 17:37:06 +0500   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure      False   Wed, 26 Dec 2018 21:44:35 +0500   Sat, 01 Dec 2018 17:37:06 +0500   KubeletHasSufficientPID      kubelet has sufficient PID available
  Ready            True    Wed, 26 Dec 2018 21:44:35 +0500   Sat, 01 Dec 2018 17:37:06 +0500   KubeletReady                 kubelet is posting ready status
Addresses:
  InternalIP:  10.0.2.15
  Hostname:    minikube
Capacity:
 cpu:                2
 ephemeral-storage:  16888216Ki
 hugepages-2Mi:      0
 memory:             2038624Ki
 pods:               110
Allocatable:
 cpu:                2
 ephemeral-storage:  15564179840
 hugepages-2Mi:      0
 memory:             1936224Ki
 pods:               110
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ALIAS

Alias

- As you will go further you will see you will be typing kubernetes long command again and again
- We can use Linux alias command to make the long commands shorter. For example

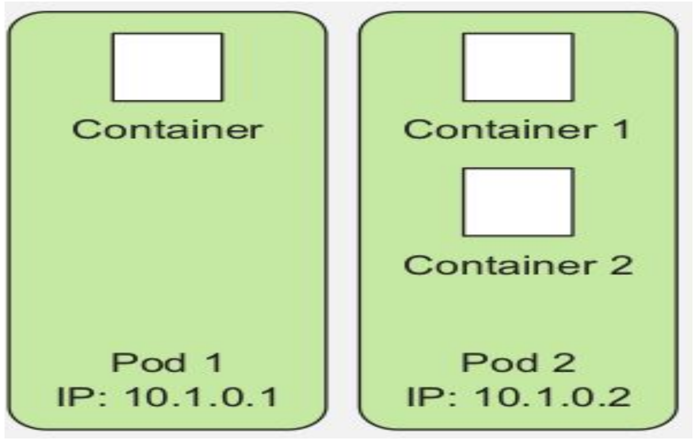
```
aamir@ap-linux:~$ alias kg=”kubectl get”
aamir@ap-linux:~$ kg nodes
NAME        STATUS      ROLES       AGE   VERSION
Minikube    Ready       master      1m     v1.14.3

aamir@ap-linux:~$ kg no
NAME        STATUS      ROLES       AGE   VERSION
Minikube    Ready       master      1m     v1.14.3
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

PODS

Pods

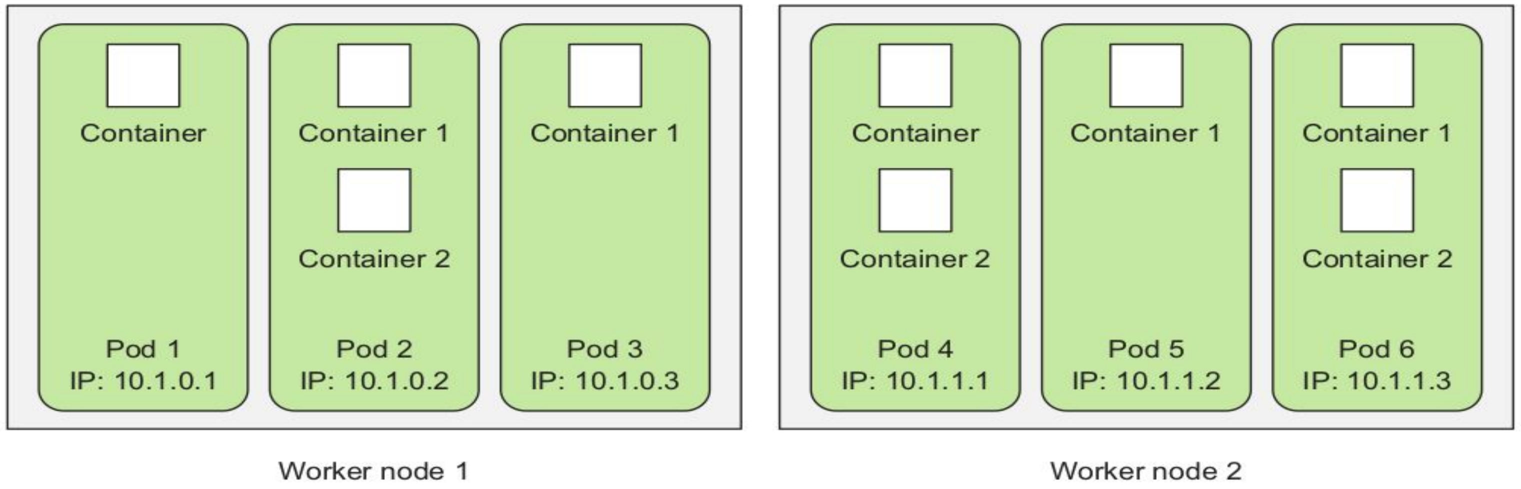
- To run your application on kubernetes most basic condition is your application should be containerized
- But Kubernetes do not deploy your app's container directly like docker does
- Instead, kubernetes wraps our app container or group of containers together
- This container wrapper is called a Pod



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods

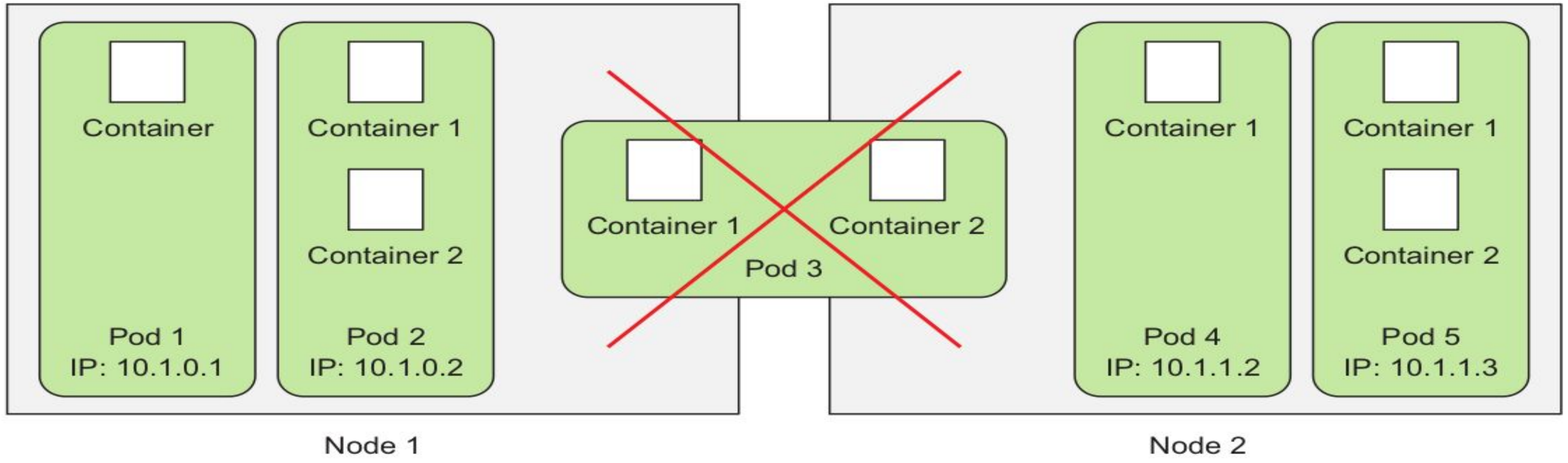
- Each pod is like a separate logical machine with its own IP, hostname, processes, and so on, running a single application



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods

- All the containers in a pod will appear to be running on the same logical machine
- A Pods with multiple container will always run on a same worker node



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

WHY PODS?

?

Why do we even need pods?

Why can’t we use containers directly?

- As we’re not supposed to group multiple processes into a single container
- We will be needing another higher-level construct that will allow you to bind containers together and manage them as a single unit

?

Why would we even need to run multiple containers together?

- You may have a supporting container with your main application container
- For example a log collection, network proxy etc

?

Can’t we put all our processes into a single container?

- Yes we can do that, but we don’t
- We try to divide our application in smaller microservices
- This makes our app’s smaller in size fast to load

Pods

- A pod of containers allows you to run closely related processes together
- Kubernetes provide these containers with (almost) the same environment as if they were all running in a single container, while keeping them somewhat isolated
- “Somewhat isolated”, this is because you want containers inside each group to share certain resources, although not all, so that they’re not fully isolated
- For example if your main application’s container write logs in some file and you want another application to use some part of that log file and make some report

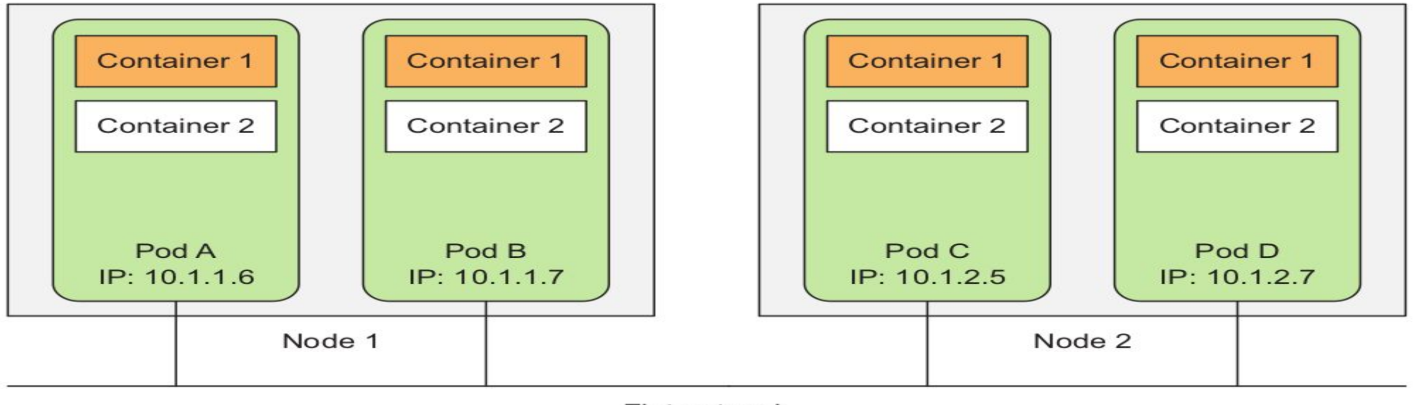
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods

- Thanks to kubernetes which provides volume concept by which we can share files directories between containers with in the pods
- This is also one of the reason why pods having multiple container does not spread over different nodes. They always co-located on same worker node

Pods

- Same like dockers, in kubernetes all containers in a Pod also uses same IP but different port numbers
- Port conflicts if same port number allocated only concerns containers in the same pod
- Containers of different pods can never run into port conflicts, because each pod has a separate port space



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

SHOULD WE RUN MULTI-TIER APPS INTO SINGLE PODS

?

Pods

- Yes you can BUT it’s not the right way
- For example: If you have a two-node Kubernetes cluster and we have learned multiple container in a single pod always runs on a single worker node
- Disadvantages
  - A single worker node will be utilize
  - Not taking advantage of the computational resources (CPU and memory)
  - You have at your disposal on the second node

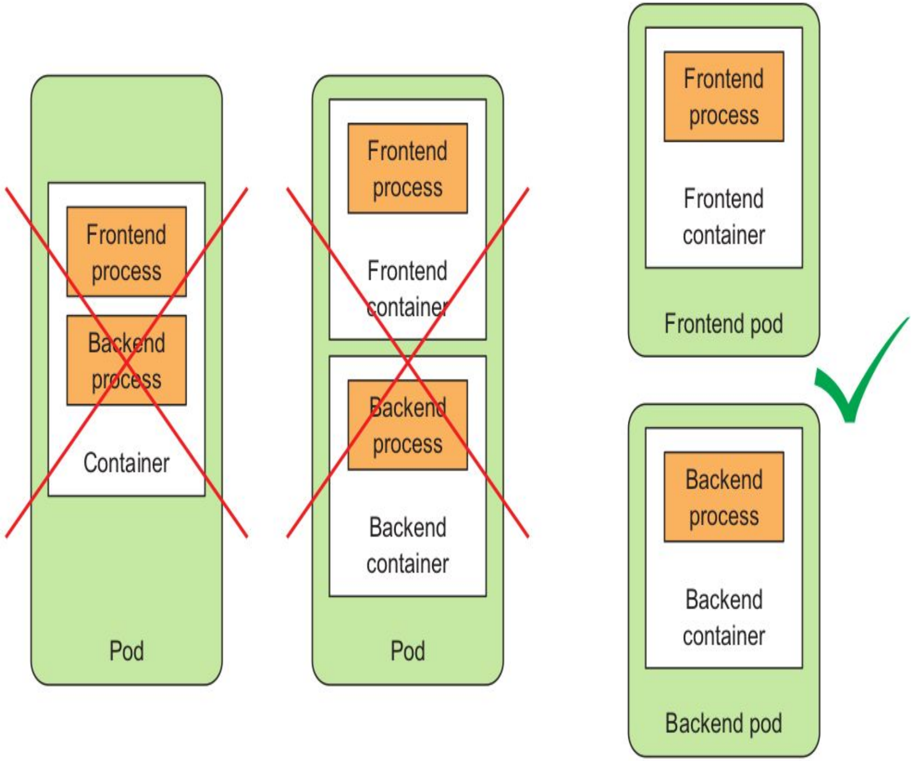
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

When to put two containers on single pod?

?

Pods

- When deciding you always need to ask yourself the following questions:
  - Do they need to be run together or can they run on different hosts?
  - Do they represent a single whole or are they independent components?
  - Must they be scaled together or individually?



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

LET’S CREATE A POD

POD Definition

- To define a POD one way is to define POD configuration in YAML file or JSON format
- **YAML** ("YAML Ain't Markup Language") is commonly used for human-readable configuration files
- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format
- Kuberenet official documentation recommends to write your configuration files using YAML rather than JSON as YAML tends to be more user-friendly

Sample structure in YAML

```
kind: <Resource Type>
apiVersion: <Kubernete API Version>

metadata:
  name: <Resource name>

spec:
  containers:

    - name: <Container name>
      image: <Container Image>

status:
  containerStatuses:

       ...
  podIP: <Pod IP>
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

POD Definition

- **Kubernetes API version** resource belongs to
  - **Type** of resource
    - Pod
    - Deployments
    - Jobs
    - Others
  - **Metadata** includes the
    - Name
    - Namespace
    - Labels
    - Other information about the pod

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

POD Definition

- **Spec** contains the actual description of the pod's contents, such as
    - Pod's containers
    - Volumes
    - Security Context
    - Other data

- **Status** contains the current information about the running pod, such as
    - Pod's Condition
    - Every container's description and status
    - Pod's internal IP and other basic info

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Creating Pods with YAML

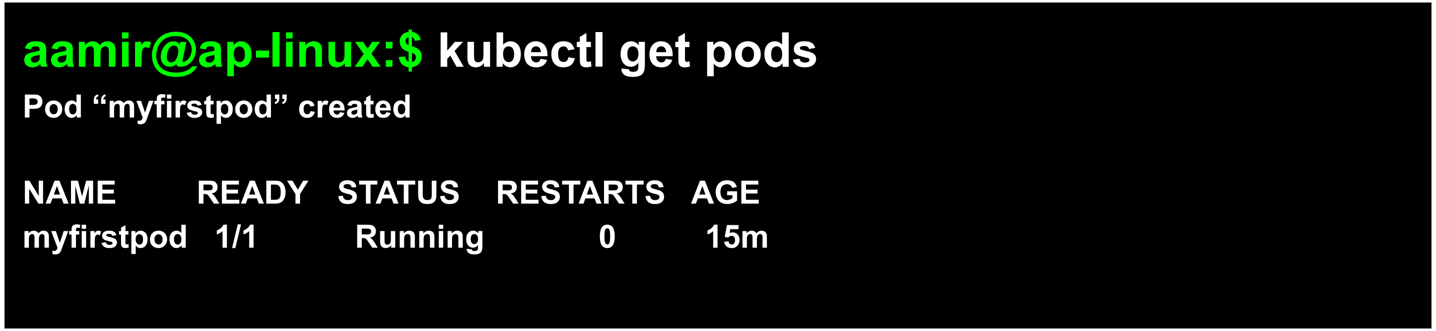
```
kind: Pod
apiVersion: v1
metadata:
  name: myfirstpod
spec:
  containers:
    - name: container1
      image: aamirpinger/helloworld
      ports:
      - containerPort: 80
```

```
aamir@ap-linux:$ kubectl create -f myfirstpod.yaml
Pod “myfirstpod” created
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

POD Listing

Pod Listing



```
aamir@ap-linux:$ kubectl get pods
Pod “myfirstpod” created

NAME         READY   STATUS    RESTARTS   AGE
myfirstpod   1/1       Running          0        15m
```

You can even write **kubectl get po**

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

YAML FROM POD

YAML FROM POD

- We created pod with YAML, What if we have already a pod and we wanted to check pod insights

```
aamir@ap-linux:$ kubectl get pod myfirstpod -o yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-07-04T06:36:20Z"
  name: myfirstpod
  namespace: default
  resourceVersion: "1994"
  selfLink: /api/v1/namespaces/default/pods/myfirstpod
  uid: 08b74341-9e26-11e9-b5e1-080027713168
spec:
  containers:
  - image: aamirpinger/helloworld

...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

POD CREATION WITHOUT YAML

Pod Creation Without Yaml

- To create a pod without writing YAML file we can use

Kubectl run <pod_name> --image=<image_name>

--port=<container_exposed_port> --restart=Never

```
aamir@ap-linux:$ kubectl run mysecondpod
--image=aamirpinger/helloworld --port=80 --restart=Never
pod/mysecondpod created

aamir@ap-linux:$ kubectl get pods
NAME                READY    STATUS               RESTARTS   AGE
myfirstpod          1/1      Running                  0       70m
mysecondpod         0/1      ContainerCreating        0       4s
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

PORT FORWARDING

Port Forwarding

- Our pod is created which got our helloworld application but still not accessible at browser

kubectl port-forward <pod_name> <external_port>:<container_port>

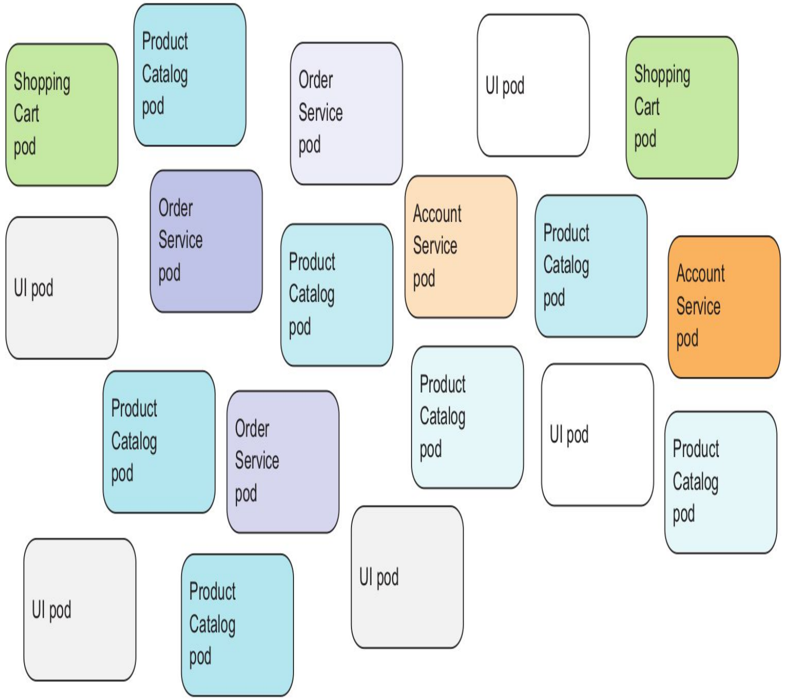
```
aamir@ap-linux:$ kubectl port-forward mysecondpod 6500:80
Forwarding from 127.0.0.1:6500 -> 80
Forwarding from [::1]:6500 -> 80
```

http://localhost:6500

LABELS

Labels

- Uptil now we have successfully create a pod having one container
- Theoretically, we have learnt that kubernetes does help creating nth number of replicas
- Theoretically, we have also learnt that kubernetes can groups pods providing at a single static IP address
- Think of situation where you have hundreds of pods with multiple replicas



Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

?

Question arises

What mechanism kubernetes use for organizing them?

?

Question arises

How group of pods can be managed with a single action instead of having to perform the action for each pod individually

Labels

- Kubernetes organizes pods with the help of labels
- Labels are simple key/value pair that kubernetes uses with almost all the resources it creates to group them
- Kubernetes gives option to filter the resources by providing label criteria
- Multiple labels can be assigned to any resource and they become part of resource group having with same labels
- Labels can be assigned at a time of resource creation or even you can assign or change label value after creation of the resource

Creating Pods With Labels

```
kind: Pod
apiVersion: v1
metadata:
  name: myfirstpodwithlabel
  labels:
    type: backend
    env: production
spec:
  containers:
   - name: container1
     image: aamirpinger/helloworld
     ports:
     - containerPort: 80
```

```
aamir@ap-linux:$ kubectl create -f myfirstpodwithlabel.yaml
Pod "myfirstpodwithlabel" created
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Creating Pods With Labels

- To create a pod without writing YAML file we can use

Kubectl run <pod_name> --image=<image_name> --port=<container_exposed_port> --restart=Never **--labels=key1=val1,key2=val2…,keyN=valN**

```
aamir@ap-linux:$ kubectl run anotherpodwithlabel
--image=aamirpinger/helloworld --port 80 --restart=Never
--labels=type=frontend,env=develop

pod/anotherpodwithlabel created
```

PODS LISTING WITH LABELS

Pods Listing With Labels

- To get the labels in the listing

```
aamir@ap-linux:$ kubectl get po --show-labels
NAME                  READY    STATUS     RESTARTS   AGE   LABELS
myfirstpod            1/1      Running           0   70m   <none>
anotherpodwithlabel   0/1      Running           0   4s    env=develop,type=frontend
...
```

- To make label as a column

```
aamir@ap-linux:$ kubectl get po -L env,type
NAME                  READY    STATUS     RESTARTS   AGE         ENV          TYPE
myfirstpod            1/1      Running           0   70m
myfirstpodwithlabel   1/1      Running           0   7m8s        production
anotherpodwithlabel   0/1      Running           0   4s          develop      frontend
...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

PODS LABELS AT RUNTIME

Pods Listing With Labels

- Assigning labels to existing pods

```
aamir@ap-linux:$ kubectl label pod myfirstpod app=helloworld type=frontend
aamir@ap-linux:$ kubectl get po --show-labels
NAME                  READY     STATUS          RESTARTS   AGE   LABELS
myfirstpod            1/1       Running                0   70m   app=helloworld,type=frontend
anotherpodwithlabel   1/1       Running                0   4s    env=develop,type=frontend
```

- Modifying existing labels of existing pods

```
aamir@ap-linux:$ kubectl label pod anotherpodwithlabel env=prod --overwrite
aamir@ap-linux:$ kubectl get po --show-labels
NAME                  READY     STATUS          RESTARTS   AGE   LABELS
myfirstpod            1/1       Running                0   71m   app=helloworld,type=frontend
anotherpodwithlabel   1/1       Running                0   1m    env=prod,type=frontend

...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

PODS LISTING WITH LABEL SELECTOR

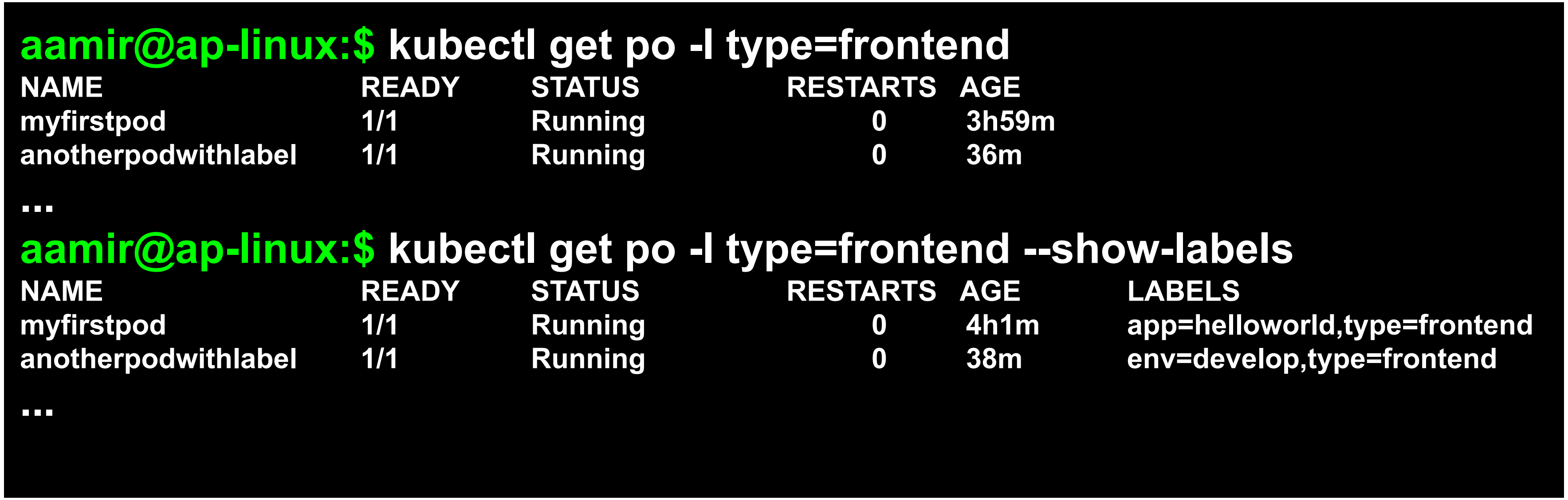
Pods Listing With Label Selector

- Label selector can be used as criteria for filtering any resources
- We can use label as criteria to check if
  - Contains (or doesn’t contain) a label with a certain key
  - Contains a label with a certain key and value
  - Contains a label with a certain key, but with a value not equal to the one you specify
  - Contains (or doesn’t contain) any one of supplied value in any particular label key

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods Listing With Labels

- Contains a label with a certain key/value



```
aamir@ap-linux:$ kubectl get po -l type=frontend
NAME                  READY     STATUS          RESTARTS   AGE
myfirstpod            1/1       Running                0    3h59m
anotherpodwithlabel   1/1       Running                0    36m

...
aamir@ap-linux:$ kubectl get po -l type=frontend --show-labels
NAME                  READY     STATUS          RESTARTS   AGE      LABELS
myfirstpod            1/1       Running                0    4h1m    app=helloworld,type=frontend
anotherpodwithlabel   1/1       Running                0    38m     env=develop,type=frontend

...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods Listing With Labels

- Contains a multiple labels with a certain key/value

```
aamir@ap-linux:$ kubectl get po -l env=prod,type=frontend --show-labels
NAME                    READY     STATUS        RESTARTS   AGE      LABELS
anotherpodwithlabel     1/1       Running              0    53m     env=prod,type=frontend
```

- Does NOT contains a label value against certain key

```
aamir@ap-linux:$ kubectl get po -l type!=frontend --show-labels
NAME                    READY     STATUS        RESTARTS   AGE      LABELS
myfirstpodwithlabel     1/1       Running              0    37m     type=backend,env=production
mysecondpod             1/1       Running              0    138m    run=mysecondpod
...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods Listing With Labels

- Does contains a label irrespective of the value

```
aamir@ap-linux:$ kubectl get po -l type --show-labels
NAME                  READY    STATUS         RESTARTS   AGE       LABELS
myfirstpod            1/1      Running               0    4h10m     app=helloworld,type=frontend
anotherpodwithlabel   1/1      Running               0    47m       env=prod,type=frontend
...
```

- Does NOT contains a label irrespective of the value

```
aamir@ap-linux:$ kubectl get po -l ‘!type’ --show-labels
NAME                  READY    STATUS         RESTARTS   AGE       LABELS
mysecondpod           1/1      Running               0    138m      run=mysecondpod
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pods Listing With Labels

- Contains any value from provided list of values against any key

```
aamir@ap-linux:$ kubectl get po -l 'type in (backend,frontend)' --show-labels
NAME                   READY     STATUS        RESTARTS   AGE       LABELS
myfirstpod             1/1       Running       0          4h29m     app=helloworld,type=frontend
myfirstpodwithlabel    1/1       Running       0          62m       env=production,type=backend
anotherpodwithlabel    1/1       Running       0          53m       env=prod,type=frontend
```

- Does NOT contains any value from provided list of values against any key

```
aamir@ap-linux:$ kubectl get po -l 'type notin(backend,frontend)' --show-labels
NAME                   READY     STATUS        RESTARTS   AGE       LABELS
mysecondpod            1/1       Running       0          165m      run=mysecondpod
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

PODS SCHEDULING WITH NODE SELECTOR

Pods Scheduling With Node Selector

```
kind: Pod
apiVersion: v1
metadata:
  name: podwithnodeselector
spec:
  nodeSelector:
    disk_type: ”ssd”
  containers:
    - name: container1
      image: aamirpinger/helloworld
      ports:
      - containerPort: 80
```

This will deploy this new pod on a worker node where label is disk_type=ssd

ANNOTATION

Annotation

- Kubernetes also provide annotation feature with labels
- Annotation are basically words that explanation or comment on something
- For example, annotation can be used to write creator’s name or contact or about application it running
- Labels are meant to hold limited information whereas you can have larger annotation with any resource
- Annotation are way to add some extra information to the resource but it cannot be used to group or filter like labels

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pod Creation With Annotation

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: podwithannotation
 annotations:
   createdBy: Aamir Pinger
   email: aamirpinger@yahoo.com
spec:
   containers:
    - name: container1
      image: aamirpinger/helloworld
      ports:
      - containerPort: 80
```

```
aamir@ap-linux:$ kubectl create -f podwithannotation.yaml
Pod “podwithannotation” created
```

Annotation Assignment To Any Resource

- To annotate any existing resource

```
aamir@ap-linux:$ kubectl annotate pod myfirstpod createdBy="Aamir
Pinger" email="aamirpinger@yahoo.com"
pod/myfirstpod annotated
```

- Let’s check by re generating YAML from pod

```
aamir@ap-linux:$ kubectl get pod myfirstpod -o yaml
email="aamirpinger@yahoo.com"
apiVersion: v1
kind: Pod
metadata:
  annotations:
    createdBy: Aamir Pinger
    email: aamirpinger@yahoo.com
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

DESCRIBING POD’S INSIGHTS

DESCRIBING POD’S INSIGHTS

- We can have any resource insight with kubectl describe command

```
aamir@ap-linux:$ kubectl describe pod myfirstpod
Name:              myfirstpod
Namespace:         default
Priority:          0
PriorityClassName:  <none>
Node:              minikube/10.0.2.15
Start Time:        Thu, 04 Jul 2019 11:36:20 +0500
Labels:            app=helloworld
                   type=frontend
Annotations:       createdBy: Aamir Pinger
                   email: aamirpinger@yahoo.com
Status:            Running
IP:                172.17.0.7
Containers:
  container1:

...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

NAMESPACE

Namespace

- With labels we have seen how easy is to group resources
- But what if any label overlaps, for example
    - type=frontend is assigned to few pods with app_name=web and env=development
    - type=frontend is assigned to few pods with app_name=web and env=production
- Name of particular resource will always be unique, what about setting the similar environment as production when in development, QA phase
- But what about times when you want to split objects into separate, non-overlapping groups? You may want to only operate inside one group at a time

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Namespace

- For this and other reasons, Kubernetes also groups objects into namespaces
- Namespace is a kind of virtualbox which isolate self contain resources with other namespace
- You can easily separate the scope of resources using namespaces. E.g. Resource in namespace for development phase cannot harm resources in namespace for production
- Similarly you can split them into multiple namespaces, which also allows you to use the same resource names multiple times (across different namespaces)

Namespace

- To create a namespace

```
aamir@ap-linux:$ kubectl create namespace production
namespace/production created
```

Kubectl create ns production can also be used

- To get a list of namespaces

```
aamir@ap-linux:$ kubectl get namespace
NAME               STATUS     AGE
default            Active     37h
kube-node-lease    Active     37h
kube-public        Active     37h
kube-system        Active     37h
production         Active     51s
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Namespace

- To create a pod in any particular namespace

```
aamir@ap-linux:$ kubectl run myfirstpod --image aamirpinger/helloworld
--restart=Never --namespace=production
pod/myfirstpod created
```

- To get any resource from any particular namespace

```
aamir@ap-linux:$ kubectl get pod --namespace production
NAME         READY      STATUS    RESTARTS    AGE
myfirstpod   1/1        Running      0        4m45s
```

- To get any resource from all namespaces

```
aamir@ap-linux:$ kubectl get pod --all-namespace
NAMESPACE     NAME                    READY      STATUS    RESTARTS   AGE
default       myfirstpod               1/1       Running      0         36h ...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

DELETING A RESOURCE

DELETING A RESOURCE

Kubectl delete <resource_type> <resource_name>

--namespace=<optional_namespace_name>

- To delete pod from default namespace

```
aamir@ap-linux:$ kubectl delete pod myfirstpod
Pod “myfirstpod” deleted
```

- To delete pod from from any particular namespace

```
aamir@ap-linux:$ kubectl delete pod myfirstpod --namespace production
Pod “myfirstpod” deleted
```

DELETING A RESOURCE

- To delete pod based on label criteria

```
aamir@ap-linux:$ kubectl delete pod -l kubectl delete pod -l 'type
in(frontend,backend)'
pod "anotherpodwithlabel" deleted
pod "myfirstpodwithlabel" deleted
```

- To delete all pods

```
aamir@ap-linux:$ kubectl delete pod --all
pod "anotherpodwithlabel" deleted
pod "myfirstpodwithlabel" deleted
```

- To delete pod from from any particular namespace

```
aamir@ap-linux:$ kubectl delete ns production
namespace "production" deleted
```

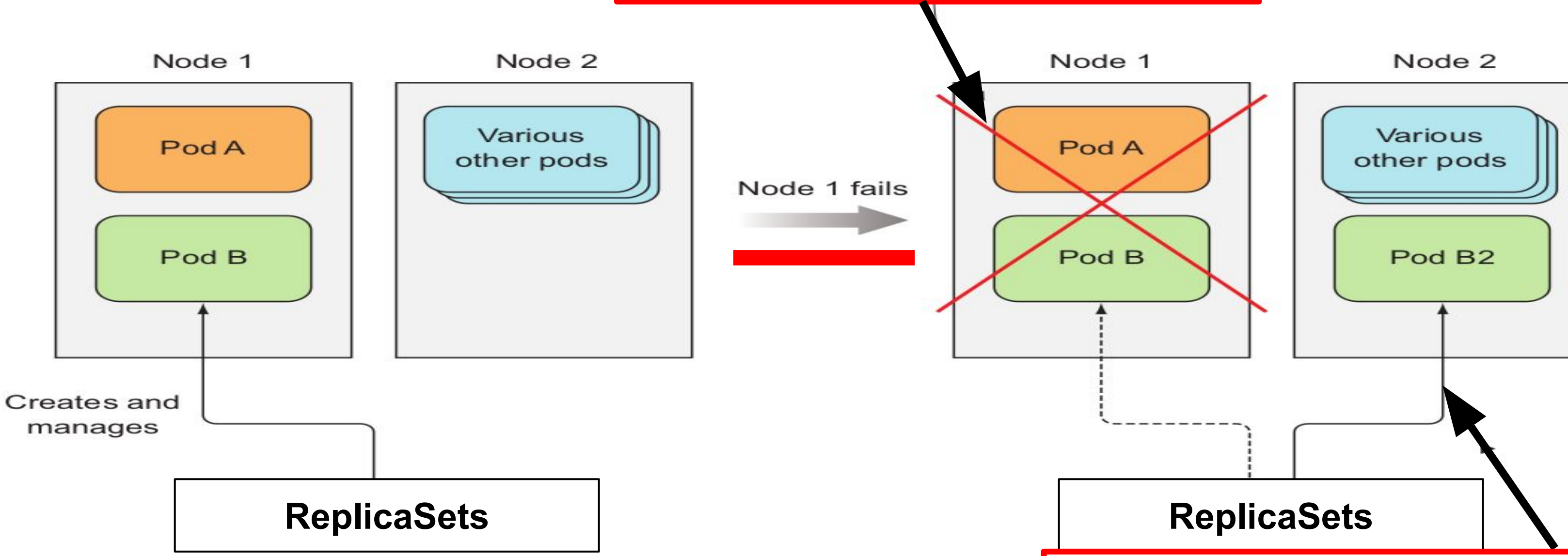
ReplicaSets

ReplicaSets

- ReplicaSet is also one of the resource in kubernetes like pods and others are
- ReplicaSet is the resource that help creating and managing multiple copies of application (replicas) in kubernetes
- When we create any pod through ReplicaSet It ensures its pods are always kept running
- If the pod disappears for any reason, such as
  - If the worker node disappears from the cluster
  - If the pod was evicted from the worker node
- The ReplicaSet notices the missing pod and creates a replacement pod

ReplicaSets

Pod A goes down with Node 1 and not recreated, because it was not created with ReplicaSets



Pod B is recreate on Node 2 because ReplicaSet make sure its Pod are kept runnig

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- A ReplicaSet has three essential parts

1. Label selector
    - Determines what pods are in the ReplicaSet scope
2. Replica count
    - Specifies the desired number of pods that should be running
3. Pod template
    - ReplicaSet uses to create new pod

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

replicaset.yaml

```
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
name: my-replica-set
spec:
replicas: 3
selector:
  matchLabels:
    app: myapp
template:
  metadata:
    labels:
      app: myapp
```

These two must match, otherwise infinite number of pod will be created

```
metadata:
  ...
spec:
  containers:
  - name: my-rs-container
    image: aamirpinger/flag:latest
    ports:
    - containerPort: 80
```

```
aamir@ap-linux:~$ kubectl create -f replicaset.yaml
replicaset.apps/my-replica-set created
```

ReplicaSets

- To get the list of ReplicaSets resource

```
aamir@ap-linux:~$ kubectl get replicasets
NAME            DESIRED   CURRENT   READY   AGE
my-replica-set  3                   3       3       5m6s
```

- Kubectl get rs can also be used
- To get the list of ReplicaSets resource

```
aamir@ap-linux:~$ kubectl get pod
NAME                    READY    STATUS    RESTARTS    AGE
my-replica-set-m644t     1/1     Running      0        8m55s
my-replica-set-4m4kt     1/1     Running      0        8m55s
my-replica-set-njndl     1/1     Running      0        8m55s
....
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- Let’s manually delete one pod and check what effect it gets on pods listing

```
aamir@ap-linux:~$ kubectl delete pod my-replica-set-m644t
pod "my-replica-set-m644t" deleted
```

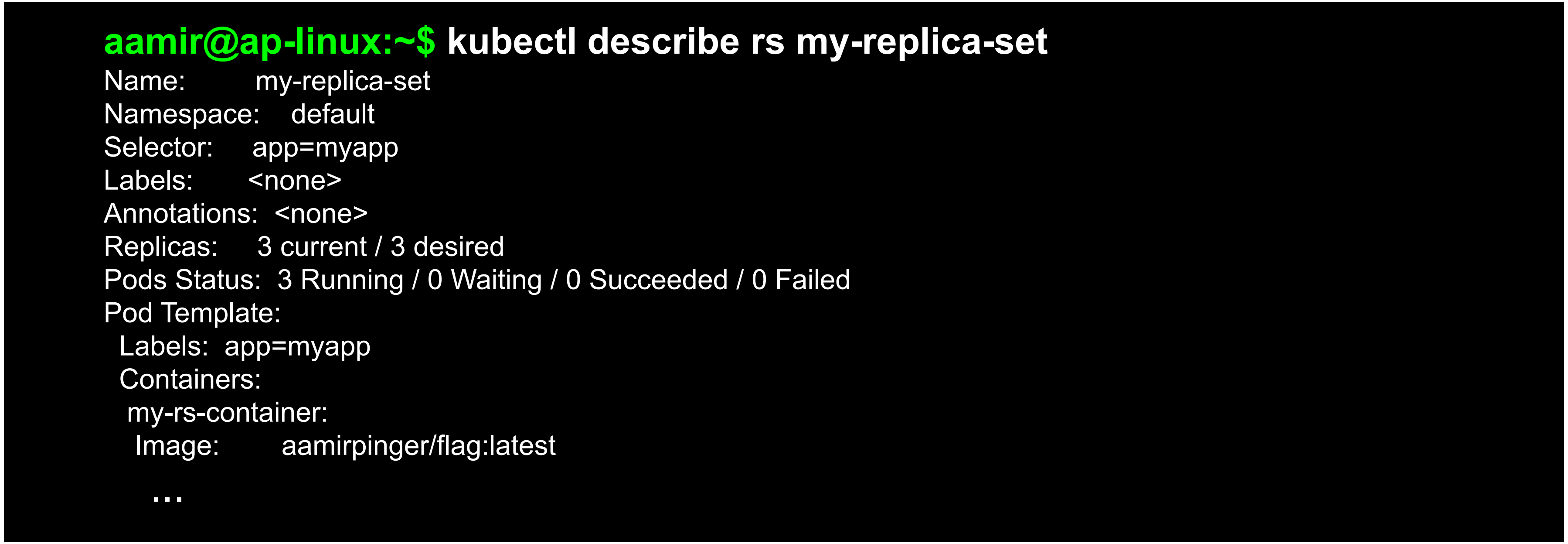
- Let’s get the list of ReplicaSet and pods together

```
aamir@ap-linux:~$ kubectl get rs,pod
NAME                          READY   STATUS              RESTARTS   AGE
pod/my-replica-set-4m4kt      1/1     Running             0          8m55s
pod/my-replica-set-njndl      1/1     Running             0          8m55s
pod/my-replica-set-snhjt      0/1     ContainerCreating   0          3s
....
NAME                                DESIRED   CURRENT   READY   AGE
replicaset.extensions/my-replica-set   3         3         2       13m
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- Let’s manually delete one pod and check what effect it gets on pods listing



```
aamir@ap-linux:~$ kubectl describe rs my-replica-set
Name:         my-replica-set
Namespace:    default
Selector:     app=myapp
Labels:       <none>
Annotations:  <none>
Replicas:     3 current / 3 desired
Pods Status:  3 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:  app=myapp
  Containers:
   my-rs-container:
    Image:       aamirpinger/flag:latest
      ...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- Let’s make a Pod out of scope of ReplicaSets by changing label of any pod

```
aamir@ap-linux:~$ kubectl label pod my-replica-set-njndl app=myappNEW --overwrite
pod/my-replica-set-njndl labeled
```

- Let’s get the list of ReplicaSet and pods together

```
aamir@ap-linux:~$ kubectl get rs,pod --show-labels
NAME                          READY     STATUS              RESTARTS      AGE   LABELS
pod/my-replica-set-4m4kt       1/1      Running                0          18m   app=myapp
pod/my-replica-set-8xtwv       0/1      ContainerCreating      0          4s    app=myapp
pod/my-replica-set-njndl       1/1      Running                0          18m   app=myappNEW
pod/my-replica-set-snhjt       1/1      Running                0          13m   app=myapp
....
NAME                                    DESIRED   CURRENT   READY   AGE   LABELS
replicaset.extensions/my-replica-set        3          3       2     13m  <none>
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- To edit any resource you can also use

  kubectl edit <resource type> <resource name>

- Let’s make the lable again app=myapp of pod my-replica-set-njndl

```
aamir@ap-linux:~$ kubectl edit pod my-replica-set-njndl
pod/my-replica-set-njndl edited
aamir@ap-linux:~$ kubectl get rs,pod --show-labels
NAME                          READY     STATUS        RESTARTS      AGE   LABELS
pod/my-replica-set-4m4kt       1/1      Running          0          26m   app=myapp
pod/my-replica-set-8xtwv       0/1      Terminating      0          10m   app=myapp
pod/my-replica-set-njndl       1/1      Running          0          26m   app=myapp
pod/my-replica-set-snhjt       1/1      Running          0          20m   app=myapp
....
NAME                                DESIRED   CURRENT   READY   AGE   LABELS
replicaset.extensions/my-replica-set     3          3       3     26m   <none>
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- We used matchLabels in selector of ReplicaSet configuration in which we can add multiple labels
- Based on those labels ReplicaSet will look for pods that have all those labels together and group them
- In ReplicaSet you can even add additional expressions to the selector

```
...
metadata:
name: my-replica-set
spec:
replicas: 3
selector:
  matchExpressions:
  - key: app
    operator: In
    values:
    - myapp
template:
  metadata:
    labels:
...
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

ReplicaSets

- There are 4 operators for matchExpressions
  - **In,**
  - **notIn,**
  - **Exists,**
  - **DoesNotExist**
- It’s important to remember that If you specify both matchLabels and matchExpressions then all the labels must match to group the resource

ReplicaSets

- All of the three parts of ReplicaSet can be modified at runtime
- Only changes to the replica count will affect existing pods
- Changes to label selector and pod template in ReplicaSet will only affect new containers
- Due to change in label selector or pod label itself, If existing pods fall out of the scope of the ReplicaSet, so the controller stops caring about them

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

POD SCALING

Pod Scaling

- ReplicaSets make sure a desired number of pod instances is always running
- Scaling number of pods up and down can be done anytime
- If you will scale up, ReplicaSet will add more pod
- If you will scale down, ReplicaSet will terminate pods to match the numbers

```
aamir@ap-linux:~$  kubectl scale rs my-replica-set --replicas=5
replicaset.extensions/my-replica-set scaled
aamir@ap-linux:~$  kubectl get rs
NAME                                  DESIRED   CURRENT   READY   AGE
replicaset.extensions/my-replica-set        5         5        5   4h17m
aamir@ap-linux:~$  kubectl scale rs my-replica-set --replicas=2
replicaset.extensions/my-replica-set scaled
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Pod Scaling

- Even if you delete or add a pod manually with the same labels used by replicaSets to group the pods, it will automatically add or terminate pods to match the exact numbers provided to replicaSets
- When you delete a replicaSet all the pods under that replicaSet will also get terminated
- Kubernetes does provide --cascade=false option for deleting only replicaSet and not the pods under it

```
aamir@ap-linux:~$  kubectl delete rs my-replica-set --cascade=false
replicaset.extensions "my-replica-set" deleted
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

JOB

Job

- Job is another resource in kubernetes
- It’s basically a Pod which we create under the type (kind) of Job
- A job resource is used to create a pod which terminates automatically when the defined job of that pods successfully completed
- A job resource creates one or more pods and ensures that a specified number of them successfully terminate

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job

- The Job object will start a new Pod if the first pod fails or is deleted (for example due to a node hardware failure or a node reboot)
- A Job can also be used to run multiple pods in parallel
- Deleting a Job resource will cleanup the pods it created

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: whalesay
spec:
 template:
   spec:
     containers:
     - name: whalesay
       image: docker/whalesay
       command: ["cowsay",  "This is a Kubernetes Job!" ]
     restartPolicy: Never
 backoffLimit: 4
 activeDeadlineSeconds: 60
```

`restartPolicy: Never` or `OnFailure` is allowed

- **OnFailure** - the Pod stays on the node, but the Container get a restart
- **Never** - the Job controller starts a new Pod and leave the unsuccessful pod as it is

Source 1: https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/ **Source 2**: https://eksworkshop.com/batch/jobs/

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: whalesay
spec:
 template:
   spec:
     containers:
     - name: whalesay
       image: docker/whalesay
       command: ["cowsay",  "This is a Kubernetes Job!" ]
     restartPolicy: Never
 backoffLimit: 4
 activeDeadlineSeconds: 60
```

backoffLimit

- Specify the number of retries before considering a Job as failed
- 6 is default retries set back-off limit by kubernetes
- Failed Pods associated with the Job are recreated by the Job controller with an exponential back-off delay (10s, 20s, 40s …) capped at six minutes

Source 1: https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/ **Source 2**: https://eksworkshop.com/batch/jobs/

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job

```yaml
apiVersion: batch/v1
kind: Job
metadata:
 name: whalesay
spec:
 template:
   spec:
     containers:
     - name: whalesay
       image: docker/whalesay
       command: ["cowsay",  "This is a Kubernetes Job!" ]
     restartPolicy: Never
 backoffLimit: 4
 activeDeadlineSeconds : 60
```

`activeDeadlineSeconds`

- Applies to the duration of the job
- Once a Job reaches activeDeadlineSeconds, all of its Pods are terminated and the Job status will become type: Failed with reason: DeadlineExceeded

Source 1: https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/ **Source 2**: https://eksworkshop.com/batch/jobs/

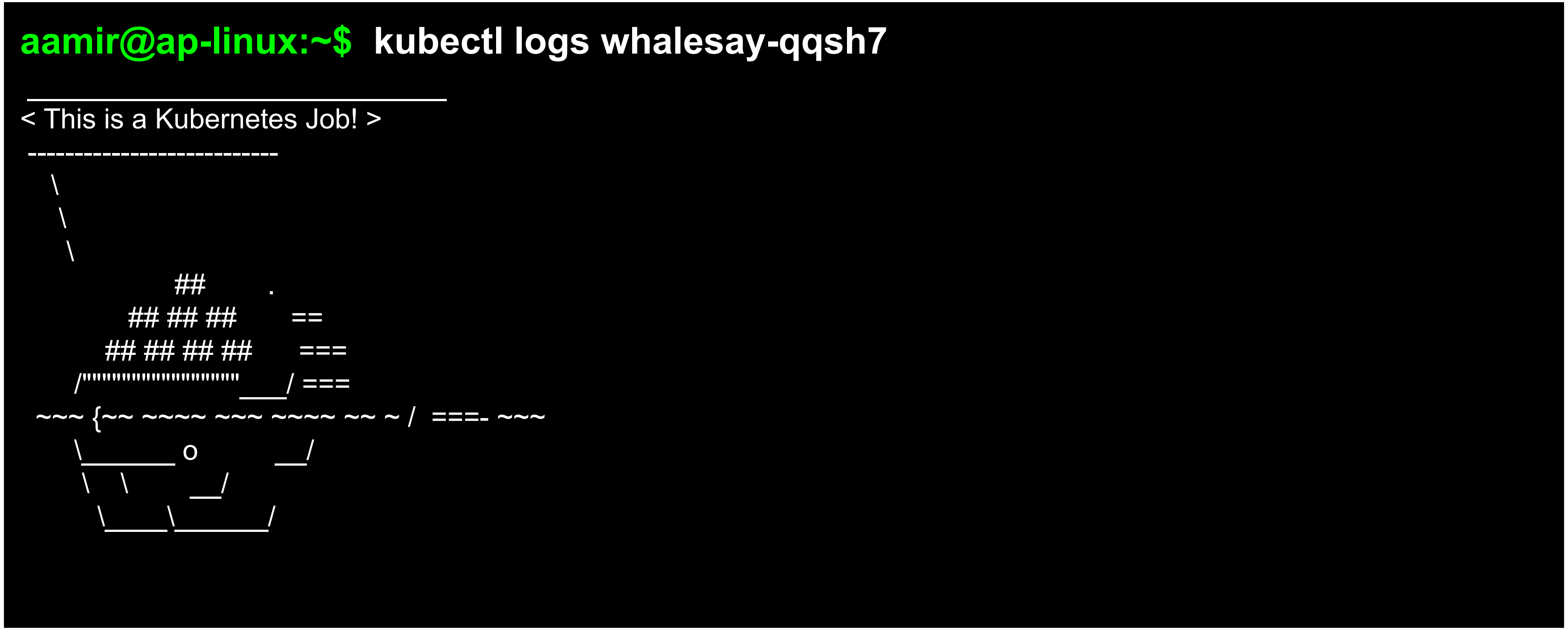
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job

```
aamir@ap-linux:~$  kubectl create -f job.yaml
job.batch/whalesay created
aamir@ap-linux:~$  kubectl get po,job
NAME                      READY      STATUS          RESTARTS        AGE
pod/whalesay-qqsh7         0/1       Completed              0        10s

…
NAME                      COMPLETIONS   DURATION      AGE
job.batch/whalesay               1/1         6s       10s
aamir@ap-linux:~$  kubectl describe job whalesay
Name:                   whalesay

…
Active Deadline Seconds:  60s
Pods Statuses:          0 Running / 1 Succeeded / 0 Failed
Pod Template:

…
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Job



```
aamir@ap-linux:~$  kubectl logs whalesay-qqsh7
 ____________________________
< This is a Kubernetes Job! >
 ----------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
```

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

CRONJOB

CronJob

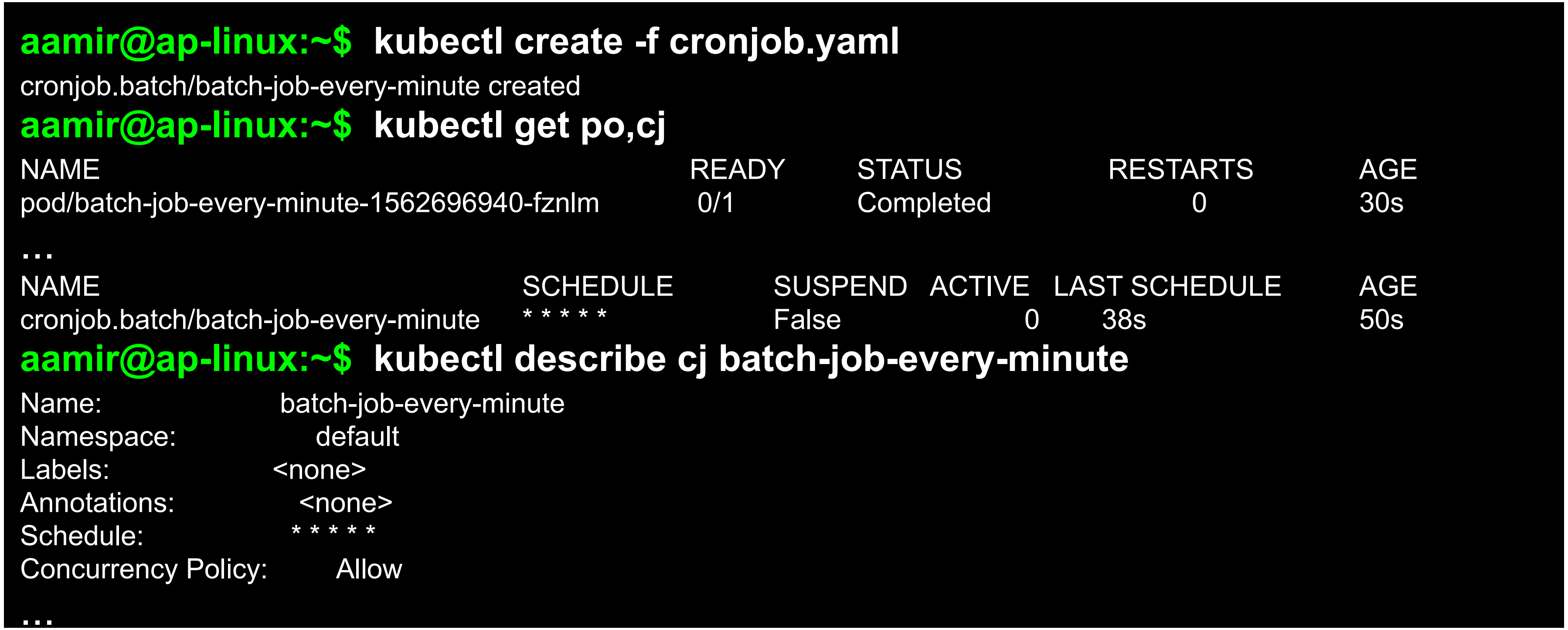
- This works almost similar to Kubernetes Job resource
- Only difference is Job resource create Pod instantly and once job is completed it does not recreate Job resource or pod
- CronJob resource is used to schedule Job at later time and can be set to initiate the Job again on provided time gap
- CronJob always creates only a single Job resource for each execution configured in the schedule

CronJob

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
 name: batch-job-every-minute
spec:
 schedule: "* * * * *"
 jobTemplate:
   spec:
     template:
       metadata:
         labels:
           app: periodic-batch-job
       spec:
         restartPolicy: OnFailure
         containers:
         - name: main
           image: docker/whalesay
           command: ["cowsay",  "This is a CronJob!"]
```

```
aamir@ap-linux:~$  kubectl create -f cronjob.yaml
cronjob.batch/batch-job-every-minute created
```

CronJob



```
aamir@ap-linux:~$  kubectl create -f cronjob.yaml
cronjob.batch/batch-job-every-minute created
aamir@ap-linux:~$  kubectl get po,cj
NAME                                          READY     STATUS          RESTARTS       AGE
pod/batch-job-every-minute-1562696940-fznlm    0/1      Completed           0          30s

…
NAME                               SCHEDULE        SUSPEND   ACTIVE   LAST SCHEDULE     AGE
cronjob.batch/batch-job-every-minute   * * * * *       False         0      38s         50s
aamir@ap-linux:~$  kubectl describe cj batch-job-every-minute
Name:                    batch-job-every-minute
Namespace:                   default
Labels:                  <none>
Annotations:                 <none>
Schedule:                  * * * * *
Concurrency Policy:          Allow
…
```

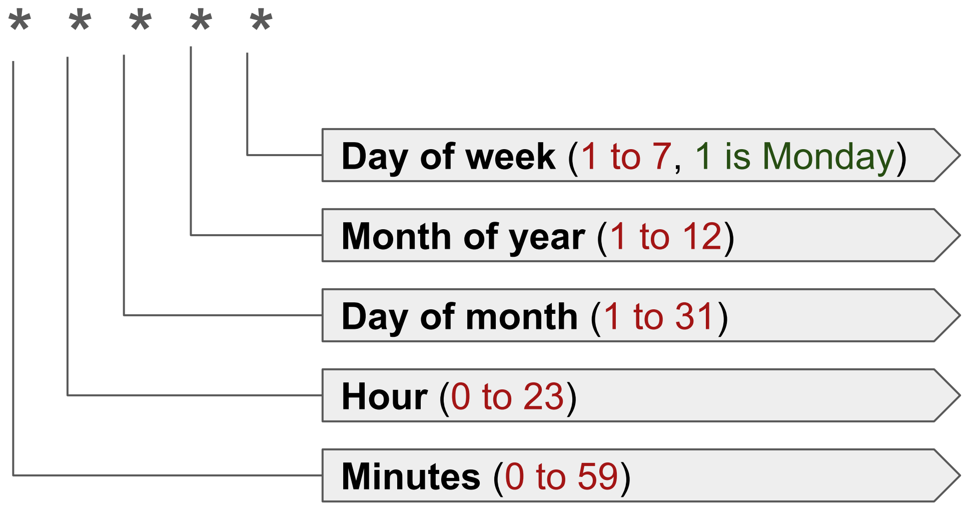
Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

CronJob

```
aamir@ap-linux:~$  kubectl logs batch-job-every-minute-1562697180-fz59v

 _______________________________
< This is a Kubernetes CronJob! >
 -------------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /"""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/

aamir@ap-linux:~$  kubectl delete cj batch-job-every-minute
[illegible]
```

CronJob

Schedule pattern



Examples:

Every Minute

*** * * * ***

Everyday 5am and 5pm

0 5,17 * * *

Every midnight in weekdays

0 0 * * 1-5

Every 15 minutes

***/15 * * * ***

Every hour of alternate days

0 * */2 * *

Source: Kubernetes in Action Book by Marko Luksa (Manning Publications)

Thank you and God bless you all!

fb.com/AamirPingerOfficial

linkedin.com/in/AamirPinger

github.com/AamirPinger