KUBERNETES - PODS

Github repo..

https://github.com/vsaini44/KubernetesRepo.git

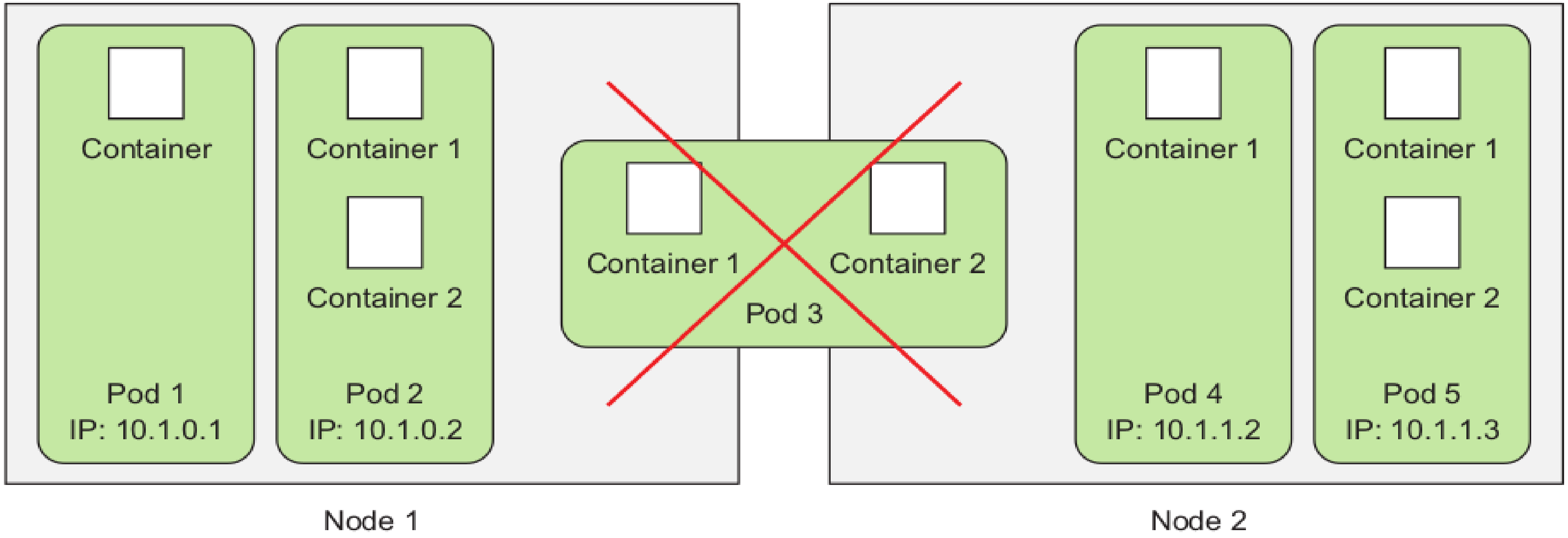
What is POD ?

A pod is a collection of containers sharing a network and mount namespace and is the basic unit of deployment in Kubernetes.

Instead of deploying containers individually,you always deploy and operate on a pod of containers.

Pod ?

The key thing about pods is that when a pod does contain multiple containers, all of them are always run on a single worker node—it never spans multipleworker nodes



Why Pods ?

Because you’re not supposed to group multiple processes into a single container, it’s obvious you need another higher-level construct that will allow you to bind containers together and manage them as a single unit. This is the reasoning behind pods.

A pod of containers allows you to run closely related processes together and provide them with (almost) the same environment as if they were all running in a single container, while keeping them somewhat isolated. This way, you get the best of both worlds.

Why Container grouped in Pod ?

To recap how containers should be grouped into pods—when deciding whether to put two containers into a single pod or into two separate pods, 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?**

Creating pods from YAML

Pods and other Kubernetes resources are usually created by posting a JSON or YAML manifest to the Kubernetes REST API endpoint.

To configure all aspects of each type of resource, you’ll need to know and understand the Kubernetes API object definitions.

Api docs: http://kubernetes.io/docs/reference/

Sample YAML File

```yaml
apiVersion: v1
kind: Pod

metadata:
  name: pod1

spec:
  containers:
    - name: cont1
      image: nginx
```

Anatomy of YAML file

- **ApiVersion: Kubernetes API Version**
- **Kind : Type of the resource described in YAML.**
- **Metadata: Data about data.**
- **Spec: Actual description of the Pod’s content.**
- **Status: Current information of the running instance.**

Basic Commands

- **Kubectl explain**
- **Kubectl create**
- **Kubectl describe**
- **Kubectl get**
- **Kubectl exec**
- **Kubectl delete**
- **Kubectl logs**
- **Kubectl port-forward**

Understanding labels

Labels are a simple, yet incredibly powerful, Kubernetes feature for organizing not only pods, but all other Kubernetes resources.

A label is an arbitrary key-value pair you attach to a resource, which is then utilized when selecting resources using label selectors

Basic Pod Scheduling

You could schedule a pod to an exact node by matching the node label inside nodeselector property in pod defination.

Annotating Pods

In addition to labels, pods and other objects can also contain annotations.

Annotations are also key-value pairs, so in essence, they’re similar to labels, but they aren’t meant to hold identifying information.

NameSpace

Namespaces are intended for use in environments with many users spread across multiple teams, or projects.

Namespaces are a way to divide cluster resources between multiple users.

Summary

- What is Pod ?
- YAML resource defination file anotomy
- Labels and Label Selector
- Basic scheduling with NodeSelector
- Annotations for larger information
- Namespace for isolation