

# Coders Classroom

Node.JS, Docker, Kubernetes, Angular, C#.Net, ASP.Net, SqlServer, HTML, Javascript, JQuery, MVC, and WCF

Sunday, March 14, 2021

## Kubernetes for Developers #9: Kubernetes Pod Lifecycle

A Pod is a group of one or more containers (consider docker containers) with shared storage and network resources.

- **Network:** Pods get unique IP address automatically and all the containers in the Pod communicate each other using **localhost** and **Port**
- **Storage:** Pods can be attached to Volumes that can be shared among the containers

A Pod is designed to run a single instance of an application inside node of Kubernetes cluster.

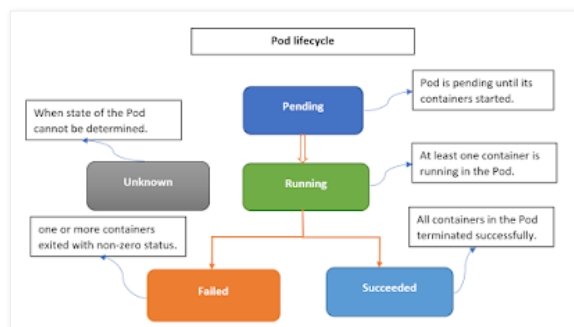
### Types of Pod:

- **Single container Pod:** The “one-container-per-pod” is the most common use case and Kubernetes manage Pod rather than container directly.
- **Multi container Pod:** A Pod can group multiple containers with shared storage volumes and network resources. Generally, we name it as Primary container and Sidecar container.

**Ex:** Primary container used to serve data stored in the filesystem/volume to outer world whereas Sidecar container used to refresh the filesystem/volume.

### Pod lifecycle:

Pods are ephemeral and not designed to run forever, when Pod is terminated it cannot be repair themselves rather it gets deleted and recreated based on Pod policy.



Pod lifecycle starts from “**Pending**” phase, moving through “**Running**” if at least one container starts and then will move to “**Succeeded**” or “**Failed**” phase depending on the container exit status.

Pod phases are continuing to update when,

- Kubelet constantly monitor container states and send info back to KubeAPI Server for updating Pod phase.
- Kubelet stops reporting to the KubeAPI Server.

Pod phases:

<b>Pending</b>	Pod has been created by the cluster, but one or more of its
----------------	---

### Search This Blog

### most popular post

#### Kubernetes for Developers #1: Kubernetes Architecture and Features

Kubernetes(K8) is an open-source container orchestration tool used for automating deployment, scaling and management of containerized applic...

### About Me



**Rama Subba Reddy**

[View my complete profile](#)

### Blog Archive

- ▼ 2021 (7)
  - April (2)
  - ▼ March (3)
    - Kubernetes for Developers #11: Pod Organization us...
    - Kubernetes for Developers #10: Kubernetes Pod YAML...
    - Kubernetes for Developers #9: Kubernetes Pod Lifec...
  - February (1)
  - January (1)
- 2020 (29)
- 2018 (4)
- 2017 (21)
- 2016 (6)
- 2015 (1)
- 2014 (8)
- 2013 (12)

### Labels

.env file (1) Angular (5) Angular 2/4 (1) AngularJS (4) Babel (1) babel-node (1) C#.NET (14) debug (4) Design Patterns (4) Design Patterns Cheat Sheet (1) Design Patterns Real Time Examples (1) Design Tools (3) Developer Productivity Tool (8) Dist folder (1) Docker (23) Docker Assignment

	containers are not yet running. This phase includes time spent being scheduled on a node and downloading images
<b>Running</b>	The Pod has been allotted to a node; all the containers have been created. At least one container is still running, or is in the process of starting or restarting
<b>Succeeded</b>	All containers in the Pod have terminated successfully
<b>Failed</b>	One or more containers terminated with non-zero status
<b>Unknown</b>	The state of the Pod cannot be determined. This occurs due to error while communicating with the node

### Container States:

The way Kubernetes maintain Pod phases, it maintains state of each container in the Pod.

Once the scheduler assigns a Pod to a Node, the kubelet starts creating containers for that Pod using a container runtime. There are 3 possible states for the container.

<b>Waiting</b>	When the container still pulling image, applying Secret data etc.
<b>Running</b>	When the container executing without any issues
<b>Terminated</b>	When the container exited with non-zero status

```
// Create Pod using imperative way
> kubectl run pod-nginx --image=nginx
```

```
# Create Pod using Declarative way (.yaml file )
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
      ports:
        - containerPort: 80
          protocol: TCP
```

```
// Create Pod using Declarative way
> kubectl apply -f ./pod-nginx.yaml
```

```
// View all running pods and their status
> kubectl get po

// View specific pod running status
// syntax: kubectl get po <pod-name>
> kubectl get po pod-nginx

// Get running Pod definition
// syntax: kubectl get po <pod-name> -o <outout-format>
> kubectl get po pod-nginx -o yaml

// Get Pod phase
> kubectl get po pod-nginx -o yaml | grep phase

// Describe Pod in-detail
// syntax: kubectl describe po <pod-name>
> kubectl describe po pod-nginx

// View logs from Pod running container
// syntax: kubectl logs <pod-name>
> kubectl logs pod-nginx

// View logs from Pod when multiple running containers
// syntax: kubectl logs <pod-name> -c <container-name>
```

(22) [Docker Bind Mount](#) (1) [Docker Bridge Network](#) (3) [Docker Compose](#) (8) [Docker Container Linking](#) (1) [Docker Tutorials](#) (12) [Docker User Defined Bridge Network](#) (1) [Docker Volume](#) (1) [Dockerfile](#) (5) [ES6](#) (2) [express.js](#) (1) [Extensions](#) (1) [Gang of Four](#) (1) [Git](#) (1) [GitHub](#) (1) [http-server](#) (1) [HybridApplication](#) (1) [Ionic](#) (1) [Javascript](#) (12) [K8 architecture](#) (2) [K8 Liveness Probe](#) (1) [K8 Pod](#) (5) [K8 Pod Labels](#) (1) [K8 Readiness Probe](#) (1) [kubectl](#) (8) [kubernetes](#) (13) [kubernetes features](#) (2) [MySQL](#) (7) [Mysql Backup](#) (1) [Nginx](#) (2) [Node.js](#) (10) [Nodemon](#) (4) [OAuth](#) (1) [OOPs](#) (2) [OWIN](#) (1) [Performance](#) (2) [singleton](#) (1) [SQL](#) (2) [Stored Procedure](#) (1) [TokenAuthentication](#) (2) [Visual Studio](#) (4) [Visual Studio Code](#) (3) [VS Code](#) (4) [WCF Tutorial](#) (3) [WebAPI](#) (3)

```
> kubectl logs pod-nginx -c nginx

// Stream logs from Pod
> kubectl logs -f pod-nginx

// Expose Pod for debugging or testing purpose using port-forward proxy
// Syntax: kubectl port-forward <pod-name> <host-port>:<container-port>
// Ex: http://localhost:8444
> kubectl port-forward pod-nginx 8444:80

// Shell to a running Pod
> kubectl exec -it pod-nginx --sh

// Shell to specific container when multiple containers running in the Pod
// Syntax: kubectl exec -it <pod-name> --container <container-name> -- sh
> kubectl exec -it pod-nginx --container nginx --sh

// View last 100 messages from Pod
> kubectl logs --tail=100 pod-nginx

// Delete a Pod
> kubectl delete po pod-nginx
```

Kubernetes for Developers Journey.

- [Kubernetes for Developers #12: Effective way of using K8 Liveness Probe](#)
- [Kubernetes for Developers #11: Pod Organization using Labels](#)
- [Kubernetes for Developers #10: Kubernetes Pod YAML manifest in-detail](#)
- [Kubernetes for Developers #9: Kubernetes Pod Lifecycle](#)
- [Kubernetes for Developers #8: Kubernetes Object Name, Labels, Selectors and Namespace](#)
- [Kubernetes for Developers #7: Imperative vs. Declarative Kubernetes Objects](#)
- [Kubernetes for Developers #6: Kubernetes Objects](#)
- [Kubernetes for Developers #5: Kubernetes Web UI Dashboard](#)
- [Kubernetes for Developers #4: Enable kubectl bash autocompletion](#)
- [Kubernetes for Developers #3: kubectl CLI](#)
- [Kubernetes for Developers #2: Kubernetes for Local Development](#)
- [Kubernetes for Developers #1: Kubernetes Architecture and Features](#)

Happy Coding :)


Posted by [Rama Subba Reddy](#) at [4:51 AM](#)



Labels: [K8 Pod](#), [kubernetes](#)

## No comments:

## Post a Comment

 Comment as: abdm03142014 ▾ Sign out

Publish Preview ☐ Notify me

[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

