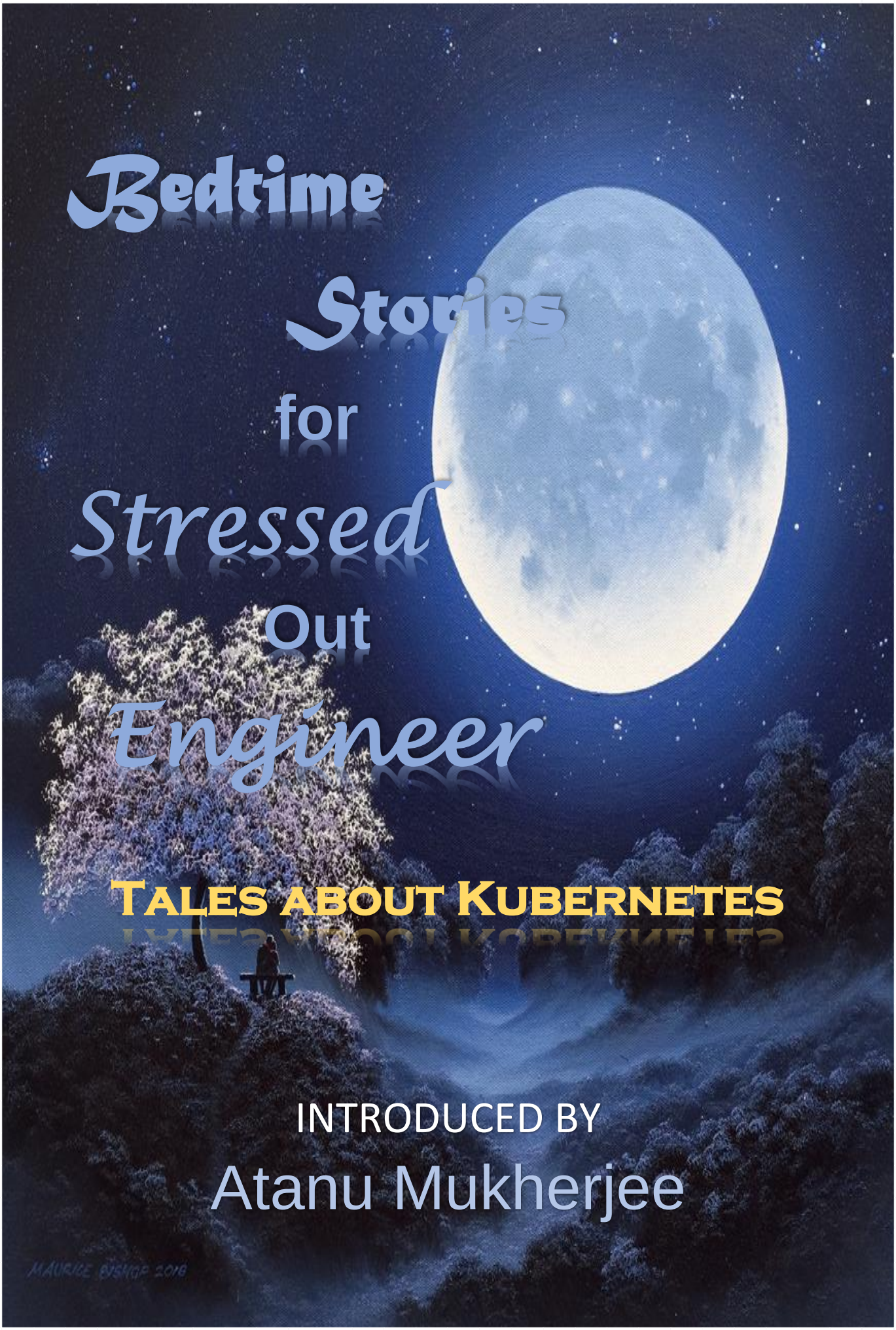

Bedtime
Stories
for
Stressed
Out
Engineer
Tales about Kubernetes
INTRODUCED BY
Atanu Mukherjee

PREFACE

It is a great opportunity for me to share the concept of Kubernetes with the Engineers who want to know about Kubernetes. They can just go through the book and know about the basic objects and resources of Kubernetes. I try to present and explain the concept in the form of a story. This book gives you a basic understanding of Kubernetes and its object. I have shared some basic lists of commands which are used in our day-to-day life.

I got inspired to share the concept of Kubernetes through stories from CNCF, how they have explained some concepts of Kubernetes in their documents. I want to especially thank CNCF for the concepts. It has helped me to share with all the Engineers who want to learn Kubernetes.

I want to thank my daughter who keeps on asking about the tool which made me write this book in the form of a story and share it with others.

I want to thank my wife who helped me in reviewing the book, to checking for grammatical mistakes.

And not least, I want to thank, everyone who encouraged me to learn Kubernetes and today reach a point where I can share my knowledge.

Myself is an Engineer working as a DevOps Consultant with 13 plus years of experience. I have worked with multiple clients and have exposure to many DevOps tools with practical knowledge.

- Atanu Mukherjee

HAPPY LEARNING

Due to Covid-19 whole world suffered. Because of this many businesses were affected, and lots of changes happened which we have never thought of. As I am an Engineer and working in IT things change for me as well. We have been told to work from home to support the business. It's new for all of us basically when you have kids at home.

I too have a kid and she is 4 years old. So, you can imagine that there are lots of questions at this age. I started working from home attending calls and meetings, interacting with colleagues.

For the past 3 years, I am working with Kubernetes and my last project was very hectic, but the main thing, I was enjoying my project and I was happy we have Kubernetes in the project and have full authority to manage that as SME. Always in the call, I am discussing Kubernetes, its internal back-end, how to design and bring the best practice of it into the project. All these were noticed by my daughter and especially always hearing the word "**Kubernetes**"

One day she asks me strangely about the Kubernetes with clear pronunciation,

DADA what is **KUBERNETES**, you always talk about?

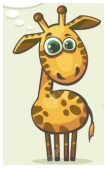
I replied:

Kubernetes is an opensource orchestration system for container. It handles scheduling on to nodes in a compute cluster in actively manage workload to ensure that the state matches the user declared intension.
It uses the concept of pods, services etc. to makeup K8s resources.
Kubernetes is a portable, extensible, opensource platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

My Daughter replied: What was that!!!!!!

Then I thought of explaining her in story form.

Once upon a time, there was an App named "**Giffy**". She was written in simple PHP.



She lived on hosting provider and was sharing the home(environment) with Other scary apps, which she was not aware of.
She wishes she could have her own environment and webserver, she called as her own home.

Apps and Environment:

- Every Apps has an environment. The environment in which user runs its application. The Software Development Environment is the collection of hardware and software tools.

- For the PHP app the environment might be webserver.

One Day, **Mr. Wale** came and suggested to **Giffy** that she can move to a container.
The Container was good, but it was a fancy little room floating in an ocean.

Containers:

A container is a standard unit of software that packages up code and all its dependencies so that application run quickly and reliably from one computing environment to another.

Corns:

- Need to manged
- Networking is hard
- It must be scheduled, loadbalanced, distributed etc.

Giffy was very sad. **Mr. Wale** was very sorry about her sadness. Before **Giffy** could tell him anything he disappear into the ocean.

Then there arrived Captain **Mr.Owlly** with a gigantic Ship.

The Ship was made up of dozens of rafts with all latched together, but. From the outside, it looked like one huge boat.

Mr. Owlly: Hello PHP App my name is Captain Mr.Owlly said the wise old captain.

Kubernetes:

The name Kubernetes originates from Greek, meaning helmsman or pilot. K8s as an abbreviation result from counting the eight letters between the "K" and the "s". Google open-sourced the Kubernetes project in 2014.

It provides the robust platform for running 1000 of container in a Kubernetes environment with easy to manged, networking etc.

PHP App: I am Giffy.

Mr.Owlly: Nice to meet you. And stick a name tag on her.

Labels:

- Kubernetes uses labels.
- Labels are key/value pairs that are attached to objects.
- Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users.
- We can use Labels to link one resource to other.

Captain Mr. Owlly secure the container and moved it into the pod in the ship. Giffy moves to the boat and feels like he was at home.

Pods:

- Pods are the smallest, most basic deployable objects in Kubernetes. A Pod represents a single instance of a running process in your cluster. Pods contain one or more containers.
- Pods encapsulate an application's container storage resources, a unique network IP and the configuration options on how the container should run.

One day, Giffy was very curious and thinking something. She asked the captain,

Giffy: Captain what if, I can clone myself.

Mr.Owlly: That was easy and he introduce to replicaset.

Replicaset:

A ReplicaSet is a process that runs multiple instances of a Pod and keeps the specified number of Pods constant. Its purpose is to maintain the specified number of Pod instances running in a cluster at any given time to prevent users from losing access to their application when a Pod fails or is inaccessible.

Giffy was very happy with so many replicas and was thinking that he got many friends to talk to. But sometimes they were very noisy and not in control and argument happened between them as to who had to do the things first.

Mr.Owlly decided to bring someone who can **control them and Deploy them to some process**. So, he introduces to **Giffy "Mr. King"**.

Deployments:

A Kubernetes deployment is a resource object in Kubernetes that provides declarative updates to applications. A deployment allows you to describe an application's life cycle, such as which images to use for the app, the number of pods there should be, and the way in which they should be updated.

Many days and nights passed, the Little App was happy with the POD and her Replicas and happy with Deployment process.
But Giffy is getting only his self-accompany and it's not all of it.

Just another day, Mr.Owlly get to know that Giffy was too sad. He came to Giffy and smiled and told

Mr.Owlly: I have just a thing for you.

And sooner he opens the tunnel between Giffy's Deployment and the rest of the ship and said again,

Mr.Owlly: Even your clone come and go you can discover other pods in the ship and they can discover you.

Services:

- An abstract way to expose an application running on a set of Pods as a network service.
- With Kubernetes you don't need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.
- Pods can come and go but your Service Ip will remain the same. That the way it helps other app pod to communicate with your pod no matter if there is new pod.

One day **Giffy** was having an inner urge, that she should talk to **Mr.Owlly**. So, she went to **Mr. Owlly**

Giffy: I want to speak to you regarding something urgent.

Mr. Owlly: Smiled and asked her, what's now, wellllll is that something bothering you!!!!!

Giffy: Yes

Mr.Owlly: Shoot your Question.

Giffy: I have some important documents which contain passwords usernames. Is there any place where can I keep it safely?

Mr.Owlly: Again Smiled and tell told yes of course. There is a place called secret over here you can hide the document there.

Mr.Owlly further added you can see them with secret Gogol and nobody else can find that or see those secrets.

Giffy wearing secret glasses to see her documents

Secrets:

Secrets can be defined as Kubernetes objects used to store sensitive data such as user name and passwords with encryption.

Suddenly One day some water started to leak into **Giffy's** Pod. She started feeling insecure. She contacted **Mr.Owlly** and asked for help.

Mr.Owlly: Don't worry, they are bags in your pod that have some important things such as extra wood, nails, hammer, glue, etc. to fix this leak. You can configure the leak which those things.

ConfigMap:

- A ConfigMap is a dictionary of configuration settings. This dictionary consists of key-value pairs of strings. Kubernetes provides these values to your containers. Like with other dictionaries (maps, hashes, ...) the key lets you get and set the configuration value.

- Use a ConfigMap to keep your application code separate from your configuration.

One day **Giffy** saw a strange thing, she saw that **Mr.Zebbi** was cleaning some deck of the ship. She was curious, as she saw there another copy of **Mr.Zebbi** in another deck on the ship, there are multiple copies of **Zebbi** in each deck and they are alone doing the task and not leaving the place if they got tired it has been replaced by another copy of **Zebbi**.

Giffy ask Zebbi

Giffy: Hello! I am Giffy, what's your name

Zebbi: replied my name is Zebbi

Giffy: Why are you cleaning the deck all alone and you are not leaving the place.

Zebbi: It's our job. I have to make sure so that I can present in each deck make it clean and have to take care of the parcels as well. I have my copy in all the decks. We cannot leave the deck for a single moment.

DaemonSet:

The DaemonSet feature is used to ensure that some or all of your pods are scheduled and running on every single available node. This essentially runs a copy of the desired pod across all nodes.

- When a new node is added to a Kubernetes cluster, a new pod will be added to that newly attached node.
- When a node is removed, the DaemonSet controller ensures that the pod associated with that node is garbage collected. Deleting a DaemonSet will clean up all the pods that DaemonSet has created.

DaemonSets are an integral part of the Kubernetes cluster facilitating administrators to easily configure services (pods) across all or a subset of nodes.

One day Giffy meet Goldy and they became best friends.
Goldy gives Giffy one beautiful Gift. But Giffy doesn't know how to take it and where to keep it. She was very sad and started crying.

Goldy: Why are you so sad and crying?

Giffy replied

Giffy: I am staying in a small pod and I have no place to keep your beautiful present, which you have given me with love.

Goldy suggested

Goldy: Why not you put that in volume.

VOLUMES:

A Kubernetes volume is **a directory that contains data accessible to containers in a given Pod in the orchestration and scheduling platform.** Volumes provide a plug-in mechanism to connect ephemeral containers with persistent data stores elsewhere.

Giffy loved life in the Ship and enjoyed the company of all her friends.

But she started thinking of those past scary Apps, she thought to have a little bit of privacy.

Namespace:

A **Kubernetes namespace provides the scope for Pods, Services, and Deployments in the cluster.** Users interacting with one namespace do not see the content in another namespace.

Giffy with her friends sails in the ship in the big once. There are many more adventures to come.

Likewise, the Kubernetes world is very big and there are many other objects and features and integration to come.

Happy Learning…….

Cheat Sheet

➔ This will display the current context
```
kubectl config current-context
```

➔ This will display the list of context from the kubeconfig file
```
kubectl config get-contexts
```

➔ This will set the default context to <my-cluster>
```
kubectl config use-context <my-cluster>
```

➔ Permanently save the namespace for all subsequent kubectl command in that context
```
kubectl config set-context --current --namespace <my-namespace>
```

➔ To display the nodes in the cluster
```
kubectl get nodes
```

➔ To display the wider information with node output
```
kubectl get nodes -o wide
```

➔ To display pods in specific namespace
```
kubectl get pods -n <namespace_name>
```

➔ To Display all the resources in the cluster
```
kubectl get all
```

➔ To display the deployments
```
kubectl get deployments
```

➔ To display the services
```
kubectl get service
```

➔ To display the namespaces
```
kubectl get namespaces
```

➔ To display the daemonset
```
kubectl get ds
```

➔ To display the endpoint
```
kubectl get ep
```

➔ To display the events in the cluster
```
kubectl get events
```

➔ To check the logs of pod for specific container
```
kubectl logs <pod_name> -c <container_name>
```

➔ To check the running logs
```
kubectl logs -f <pod_name>
```

➔ Details for the pod configuration. We can see the configuration of any object with same command
```
kubectl describe pods <pod_name>
```

➔ Logged into the running pod
```
kubectl exec -it <pod_name> -- /bin/bash
```

➔ Edit the deployment at the runtime
```
kubectl edit deployments <pod_name>
```

➔ Show all logs from pod nginx written in the last hour
```
kubectl logs --since=1h <pod_name>
```

➔ Display latest 20 line of the pod logs

```
kubectl logs --tail-20 <pod_name>
```

➔ Create the pod with imperative command

```
kubectl run <pod_name> --image=<image_name>
```

➔ Create pod with manifest file

```
kubectl apply -f <manifest.yaml>
```

➔ To display the manifest file of the resource

```
kubectl get pod <pod_name> -o yaml
```

➔ Imperative command to link deployment with service Before creating service redirects the command output to manifest file, recheck and then apply.

```
kubectl expose deployment <deployment_name> --port=xxxx -target-port --type NodePort --name
<service_name> --dry-run=client -oyaml > manifest.yaml
```

➔ Scale the deployment

```
kubectl scale deployment <deployment_name> --replicas=2
```

➔ Delete the object.

```
kubectl delete po <pod_name>
```

I have tried to mention all the basic commands of Kubernetes in this Cheat-Sheet. I will be preparing a more detailed command sheet and will try to include all the objects in that Cheat Sheet.