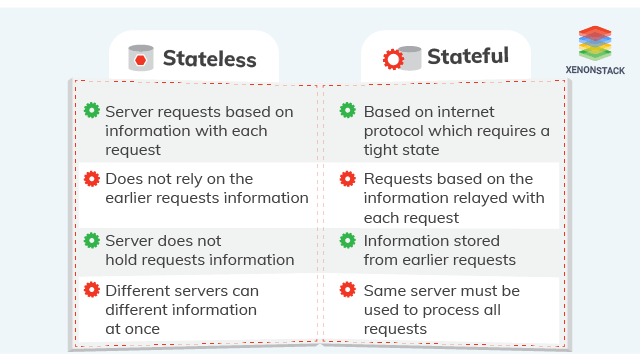
K8s

Kubernextras

# State management

## Stateful vs Stateless

A stateless application is one which depends on no persistent storage. The only thing your cluster is responsible for is the code, and other static content, being hosted on it. A stateful application, on the other hand, has several other parameters it is supposed to look after in the cluster.

Deploying a stateless application such as a rest api on K8s is simple enough since its not storing anything, but stateful applications are another thing. The problem is with scaled applications you basically have clones of services, which means that whenever you make requests your requests would probably not be handled by the same instance. Which means you need to have some sort of mechanism that allows the instances to access user session data. The instances need to have some central repository to access previous transactions, or you can just make sure that the request gets routed to the same instance.

In k8s, deploying these applications uses different objects. Stateless applications uses deployments, which creates pods “randomly” (there’s no specific order or data that needs to be retained) because it doesn’t matter which instance handles a request. The instances are interchangeable.

For stateful applications, you’d use StatefulSets. You still create multiple instances of the same image but the replica pods are unique. There is a pod identifier and it is persistent for any rescheduling! Which means that you will be accessing the same instance.

# Security

## Secret management

There’s a K8s object called a secret that you can refer to by name. You set the secret value via the kubectl and refer to it in your object config file. Note that storing secrets using secret management is more secure than storing sensitive info in your images (there are ways to inspect the file system in a running container using docker commands which means with some digging around you could find sensitive info). Kinda like how you can set config via Azure App Services, K8s secret management stores your secrets and encrypts it for you.

# Scalability and Failover

Failover is a concept of switching to a backup if something breaks. Just like using loadbalancers to route requests to active pods, spin up a new pod when one instance goes down, etc.

## Ingress

*Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster. Traffic routing is controlled by rules defined on the Ingress resource.*

Doesn’t work by itself, you need to set up an ingress controller. An Ingress controller is responsible for fulfilling the Ingress, usually with a load balancer, though it may also configure your edge router or additional frontends to help handle the traffic.

Basically a gateway to your cluster via HTTP/HTTPS.

## Autoscaling

Autoscaling in K8s just like autoscaling anywhere (\*ahem AWS), you automatically horizontally scale.

The Horizontal Pod Autoscaler is implemented as a Kubernetes API resource and a controller. The resource determines the behavior of the controller. The controller periodically adjusts the number of replicas in a replication controller or deployment to match the observed metrics such as average CPU utilisation, average memory utilisation or any other custom metric to the target specified by the user.

Resources:

<https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/>

<https://www.youtube.com/watch?v=pPQKAR1pA9U>

<https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/>

<https://platform9.com/blog/kubernetes-secrets-management/>

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>

<https://kubernetes.io/docs/concepts/services-networking/ingress/>