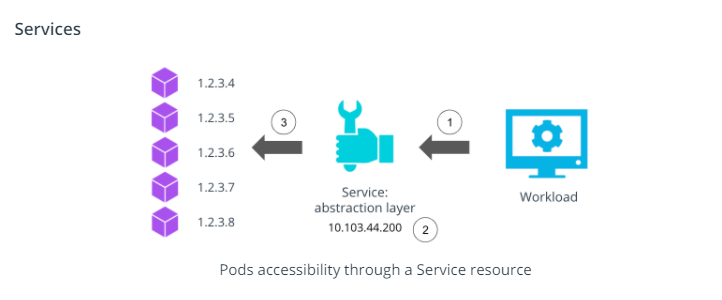
**Kubernetes Resources Part 2**

**Application Reachability**

Summary

Within a cluster, every pod is allocated 1 unique IP which ensures connectivity and reachability to the application inside the pod. This IP is only routable inside the cluster, meaning that external users and services will not be able to connect to the application.

For example, we can connect a workload within the cluster to access a pod directly via its IP. However, if the pod dies, all future requests will fail, as these are routes to an application that is not running. The remediation step is to configure the workload to communicate with a different pod IP. This is a highly manual process, which brings complexity to the accessibility of an application. To automate the reachability to pods, a Service resource is necessary.



A **Service** resource provides an abstraction layer over a collection of pods running an application. A Service is allocated a cluster IP, that can be used to transfer the traffic to any available pods for an application.

As such, as shown in the above image, instead of accessing each pod independently, the workload (1) should access the service IP (2), which routes the requests to available pods (3).

There are 3 widely used Service types:

* **ClusterIP** - exposes the service using an internal cluster IP. If no service type is specified, a ClusterIP service is created by default.
* **NodePort** - expose the service using a port exposed on all nodes in the cluster.
* **LoadBalancer** - exposes the service through a load balancer from a public cloud provider such as AWS, Azure, or GCP. This will allow the external traffic to reach the services within the cluster securely.

To create a service for an existing deployment, use the kubectl expose deployment command, with the following syntax:

*# expose a Deployment through a Service resource*

*# NAME - required; set the name of the deployment to be exposed*

*# --port - required; specify the port that the service should serve on*

*# --target-port - optional; specify the port on the container that the service should direct traffic to*

*# FLAGS - optional; provide extra configuration parameters for the service*

kubectl expose deploy NAME --port=port [--target-port=port] [FLAGS]

*# Some of the widely used FLAGS are:*

--protocol - set the network protocol. Options [TCP|UDP|SCTP]

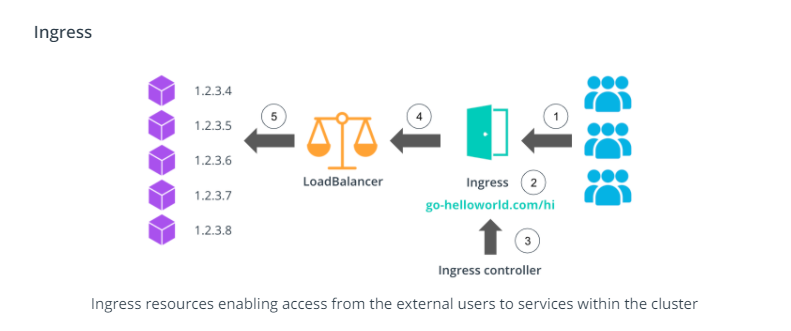
--type - set the type of service. Options [ClusterIP, NodePort, LoadBalancer]

For example, to expose the Go hello-world application through a service, the following command can be used

**# expose the `go-helloworld` deployment on port 8111**

**# note: the application is serving requests on port 6112**

kubectl expose deploy go-helloworld --port=8111 --target-port=6112



To enable the external user to access services within the cluster an **Ingress** resource is necessary. An Ingress exposes HTTP and HTTPS routes to services within the cluster, using a load balancer provisioned by a cloud provider. Additionally, an Ingress resource has a set of rules that are used to map HTTP(S) endpoints to services running in the cluster. To keep the Ingress rules and load balancer up-to-date an Ingress Controller is introduced.

For example, as shown in the image above, the customers will access the *go-helloworld.com/hi* HTTP route (1), which is managed by an Ingress (2). The Ingress Controller (3) examines the configured routes and directs the traffic to a LoadBalancer (4). And finally, the LoadBalancer directs the requests to the pods using a dedicated port (5).

**Application Reachability Demo**

**Summary**

This demo provides a step-by-step guide on how to expose the Go hello-world application through a ClusterIP service. Additionally, an *alpine* pod is used to showcase how a workload can connect to the Go hello-world application through the service IP and port from within the cluster.

**New terms**

* **Service** - an abstraction layer over a collection of pods running an application
* **Ingress** - a mechanism to manage the access from external users and workloads to the services within the cluster

**Further reading**

Explore Kubernetes resources used to connect to an application:

* [Kubernetes Services](https://kubernetes.io/docs/concepts/services-networking/service/)
* [Kubernetes Ingress](https://kubernetes.io/docs/concepts/services-networking/ingress/)