8 STEPS TO CLOUD MIGRATION WITH ISTIO SERVICE MESH

By Ravi Verma, CTO, IMESH

Table of Contents

01

Cloud migration is the need of the hour, but

It is a basic norm that end customers today want on-demand and uninterrupted services, top-notch experience and convenience using technology. This has also become the main differentiating factors for large and small companies across industries.

Enterprises are rapidly migrating their traditional workloads to the cloud (AWS/GCP/Azure) and container technologies (Kubernetes) to win customers. However, migration projects are usually full of complications and turbulence; many projects do not end well. As per cloud migration reports from Mckinsey and CIOdive magazines, 90% of the CIOs face challenges completing cloud migration on the stipulated time and budget.

There can be many reasons for the migration project crash-landing, such as:

- Lack of planning and design
- Non-alignment of various stakeholders
- Limited time and budget
- Unseen network and security complexities

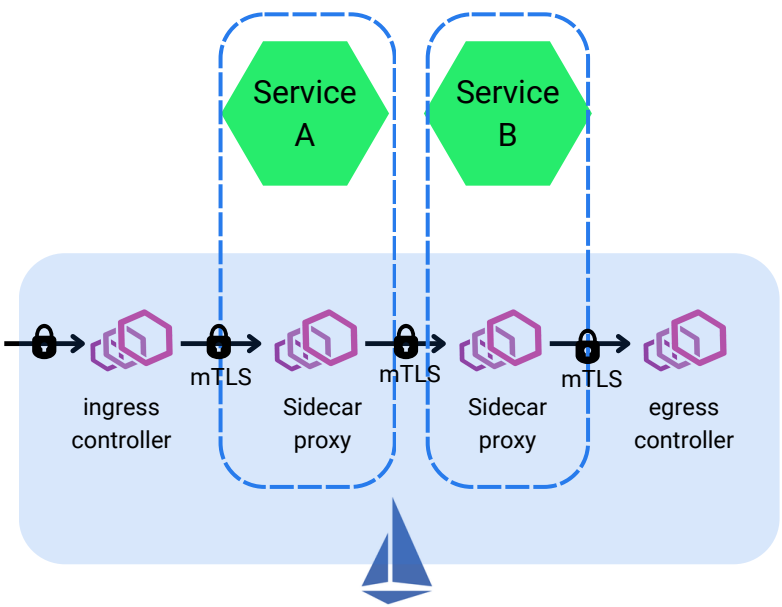
In this eBook, we will talk about how to eliminate one of the key challenges- network and security complexities and move workloads to the cloud happily.

Cloud migration with Istio

Intro to Istio service mesh

Istio is an open-source service mesh used to simplify network complexities in microservices. It is a network layer to manage communication between services and secure data in transit. Architects and engineers use Istio to build cloud-native applications or follow microservice architecture approaches.

Istio service mesh provides a control plane to define and implement the way microservices communicate with each other. Istio provides a data layer based on a foundation of network proxy instances derived from the Envoy proxy. Envoy, an L4 and L7 proxy, is responsible for all service interactions in Kubernetes or virtual machines (VMs).



Using Istio service mesh, platform teams can address needs for traffic management, service security, and application monitoring. Istio service mesh enables developers to develop business logic for loosely coupled microservices without worrying about communication logic and security. Istio is designed to run in various environments like on-premises, multi-cloud, Kubernetes containers, and virtual machines (VMs). Istio helps platform teams manage and monitor all the service traffic across clusters and data centers.

Various components such as API gateway and ingress controller make Istio suitable for cloud migration. While API gateway is well ideal for traffic shifting & shaping between services spread across data centers, ingress is convenient for traffic turning & shaping between services in a cluster.

Reasons to use Istio for cloud migration

Istio service mesh provides the infrastructure layer to abstract networks from core business logic. And this means the migration team can handle network complexities separately, while developers can still work on new feature development instead of swarming into migration projects.

Traditional monolithic applications are not modularised and hence may only involve a little network communication. In a microservices architecture, all the communication between services takes place over a network; hence, the network complexities increase in a magnitudinal manner. Therefore Istio service mesh is used to reduce the operational complexities of the network by providing the ability to monitor and manage traffic and secure data in transit.

Istio service mesh provides the following features which it is highly relevant for any cloud migration projects:

- Traffic splitting rules among services implementing progressive deployment strategies such as canary.
- Advanced traffic management features include timeouts to a service, frequent retries, and planned automatic failover of high availability (HA) systems.
- Tracking of network requests and traces each call across multiple services.
- Observe telemetries such as latency, saturation, traffic health, and errors to aid SREs with troubleshooting.
- Authentication, authorization, and access control policies for implementing zero-trust networks.

Istio service mesh helps enterprises by routing traffic between legacy environments and the newly deployed cloud application without compromising security. In this case, no application business logic or configuration has to be changed, as the network is entirely decoupled from the app layer.

Note stateless application migrations are straightforward with Istio service mesh; however, in complicated cases like statefulset applications or tightly coupled applications, additional planning and research have to be done before migration. Because in statefulset, data has to be migrated while complying with consistency and integrity, and in the case of tightly coupled features, all the individual components must be migrated to the cloud simultaneously.

8 Steps to migrate on-prem to cloud (Improve & Move strategy)

1. Prioritize of application for migration
2. Provision a target environment- cluster & VM
3. Install & configure Istio
4. Expose legacy services through the Istio API gateway
5. Deploy services in the target environment
6. Split traffic with Istio and test the performance and quality
7. Change the traffic route to the target environment
8. Retire the legacy environment

Step1: Prioritization of application for migration

Understanding which application to migrate to the cloud first is an essential and primary element of any cloud migration project. Lifting and shifting the entire application at a time is only sometimes advisable. Divide-and-conquer approach will work well to reduce the inherent complexity of cloud migrations. While planning to migrate, consider using a framework- "***Business relevance, Data dependence***"- to prioritize components to migrate.

1. **Business relevance**

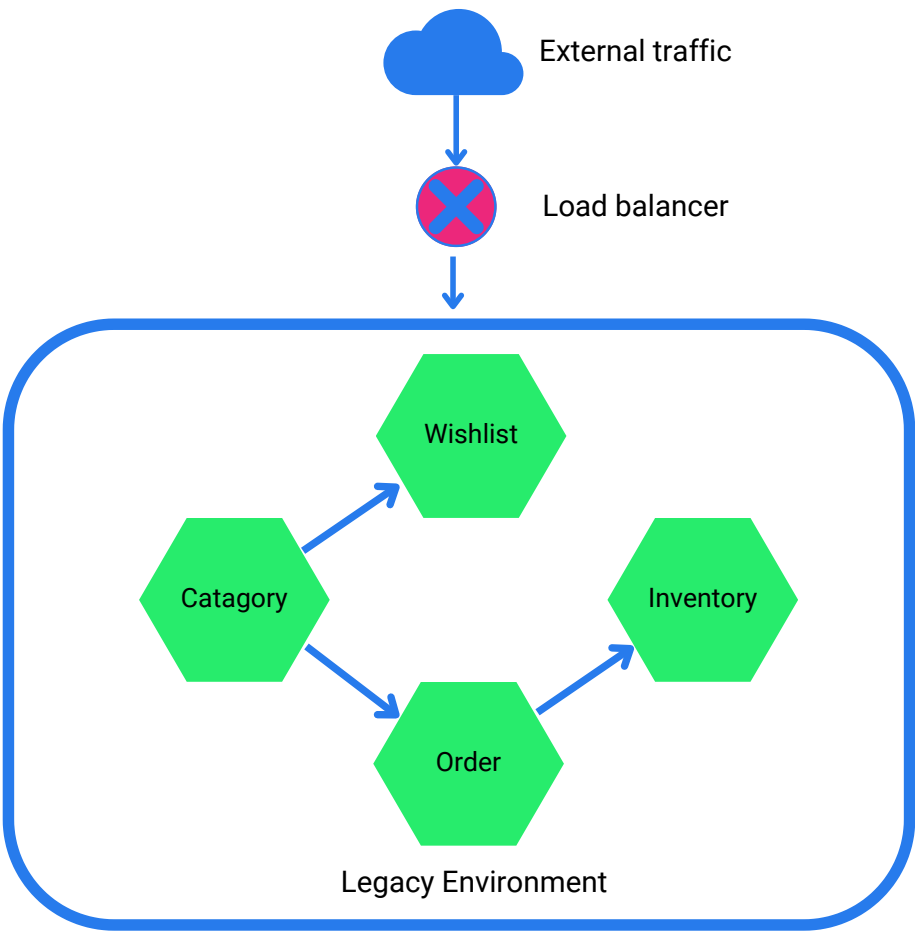
Cloud migration is usually a one-off event for an organization, project, or team. Hence there will be people getting exposed to the migration for the first time, and there will be mistakes. Therefore any component with higher business relevance or mission critical where downtime can severely affect business should not be considered for migration in the first place. Similarly, if any lightweight component with less business context is chosen, team members will have less learning experience. Choose a feature with a medium business impact and several use cases for migration.

2. **Data dependence**

Understand and note the data dependencies of each component or feature you plan to migrate. You can choose to evaluate the degree of data dependencies with multiple sub-factors, such as the amount of transaction a component/feature carries out, the number of other features dependent on the components, the criticality of the data the feature deals with, and any network prerequisites for data transactions.

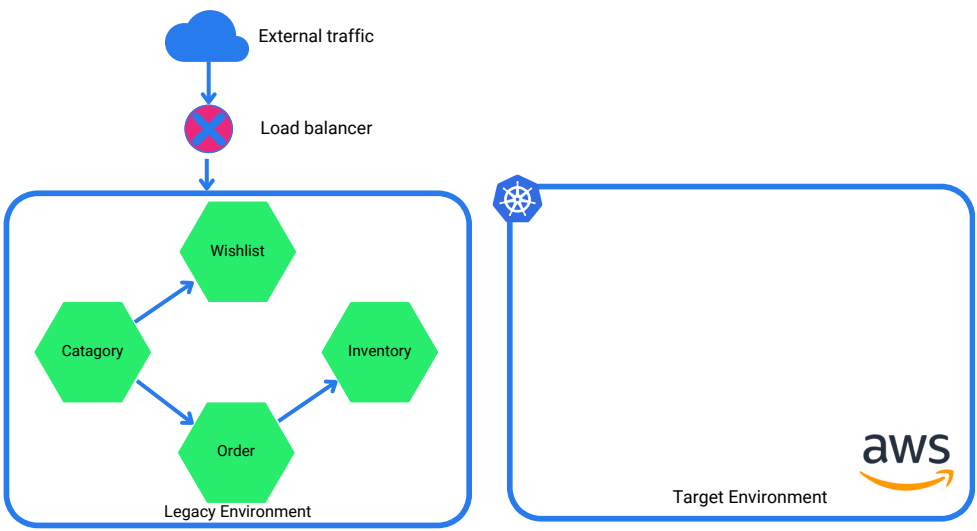
Step1: Prioritization of application for migration

Most migrations fail because of migrating a highly data-dependent component first, because of high network latency of the new component in the cloud, or data integrity and synchronization challenges during migration. Consider a component for migration that has fewer data dependencies.



Step2: Provision the target environment

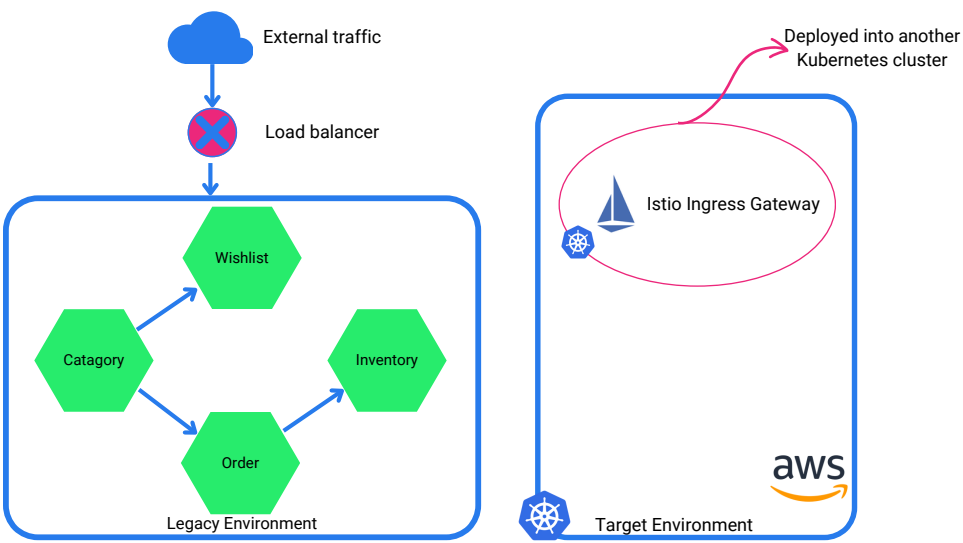
Once you prioritize a component or feature of your application to migrate, you can provision a target environment per your requirements- software, hardware, and network configuration. Use infrastructure-as-code methodology to version control, maintain, and reuse the infrastructure configuration. The diagram below represents an empty target Kubernetes cluster where apps can be deployed.



Step3: Install & configure Istio

After provisioning the target cluster, the next step is to install the Istio service mesh, which covers the legacy and target environments. Istio will not automatically discover all the microservices running in a legacy environment. Hence first register the services in the legacy environment into Istio. If your legacy environment runs on Kubernetes, it will be easy for Istio to discover all the benefits automatically; Istio treats Kubernetes as first-class citizens.

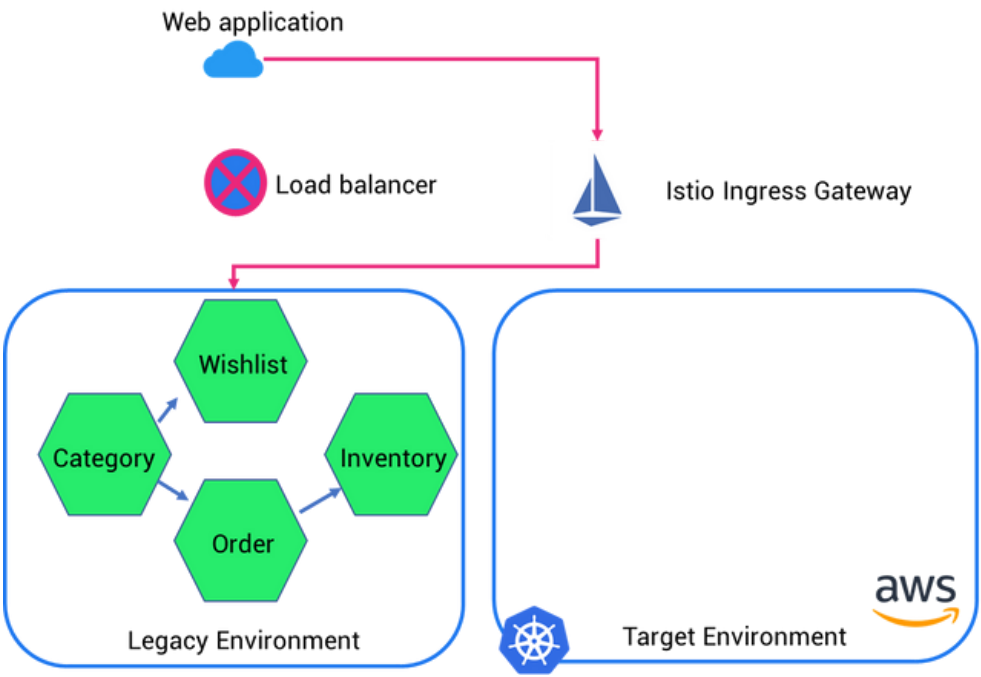
The service mesh will not receive any production traffic. The client will be using the load balancer to access the legacy environment. So now, you should configure the Istio ingress gateway to get the production traffic and send it to the services in the legacy environment.



Step4: Expose legacy services through the Istio API gateway

When your legacy environment and the microservices are registered into Istio, you can use the Istio ingress gateway to route the production traffic to the legacy environment instead of the old load balancers. There will be no service downtime. Hence customer experience will remain as it is. Clients will not know the routing has changed because they will still use legacy environment features.

The following diagram shows that services running in the legacy environment are exposed through Istio.

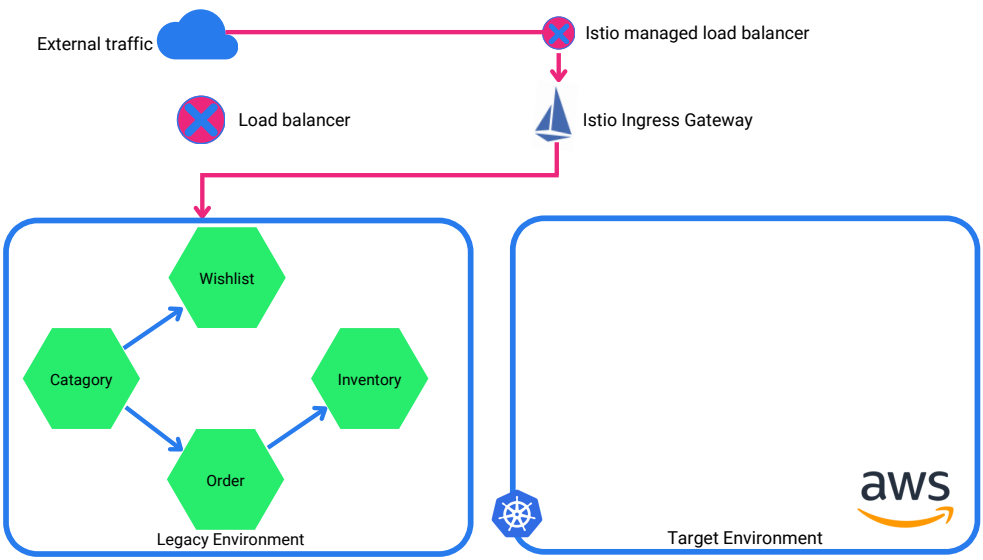


Step5: Deploy services in the target environment

In this step, deploy all the microservices for the components you want to migrate. Create container images for each element and manifest files for deployment. Use a delivery pipeline or GitOps tool like Argo CD to deploy apps into the target cluster so that you can quickly roll back your application in case of issues.

Note: In case you are migrating a stateful application, you have to deploy the database to the target environment first to avoid downtime and deal with synchronization.

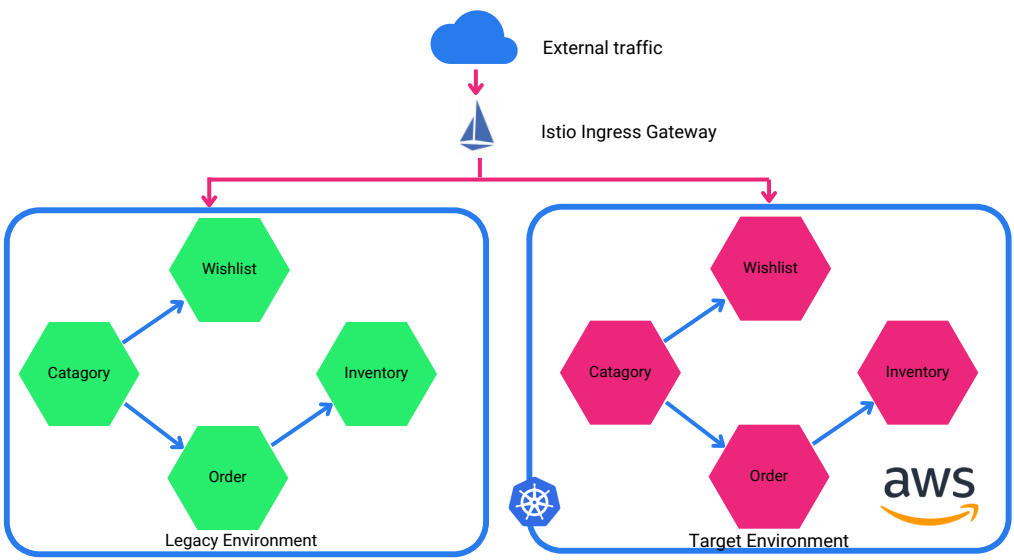
After the deployment, the microservices running in the target clusters will not receive any production traffic. So we have to configure the Istio ingress gateway to allow a certain amount of traffic to the new cloud environment.



Step6: Split traffic with Istio and test the performance and quality

In this step, you can set up routing rules to split the production traffic between the legacy and cloud environments. The rule of thumb is to allow a small portion of traffic, as low as 3-5%, to the cloud environment using Istio.

Initially, you need to test if the new application is working fine regarding its performance, quality, and behavior. Apart from conducting sanity or smoke testing, you can collect metrics and logs for a certain period to estimate the system's performance. In the testing phase, you can write additional network logic to allow or deny traffic from specific sources.

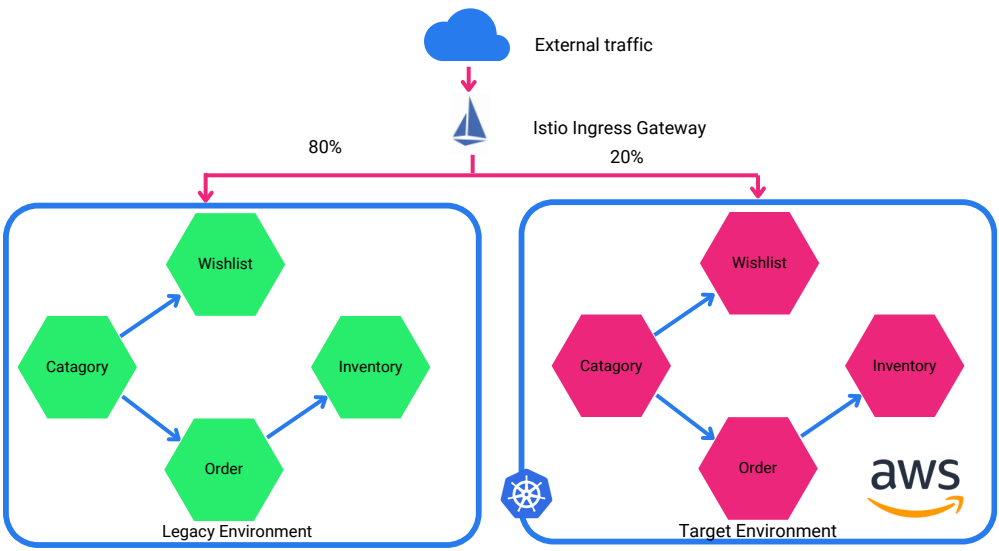


Step6: Split traffic with Istio and test the performance and quality

Once you build confidence in the new target environment, you can increase the traffic gradually (say by another 5% or 10%) and perform the testing and monitoring exercise again.

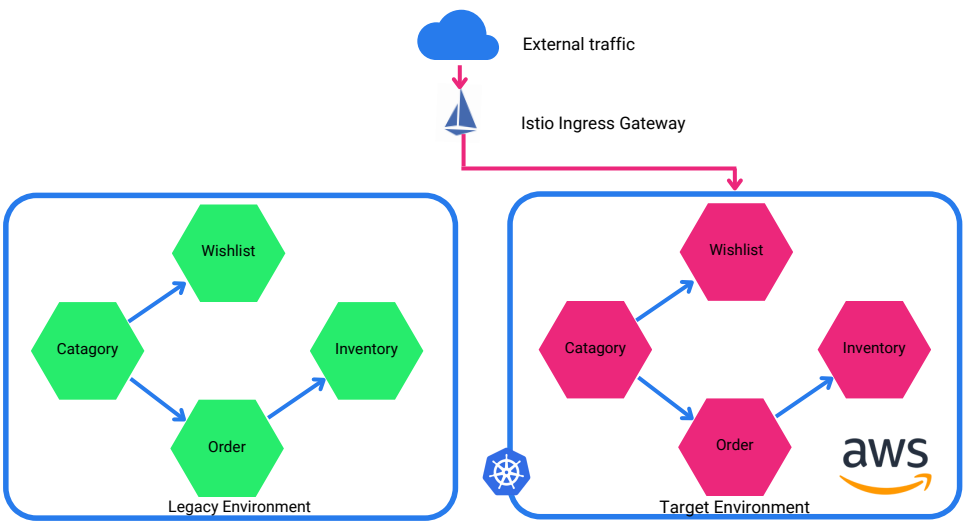
In the case of a stateful application, more planning and testing on the database stack must be done because there can be multiple data sources from legacy and target systems.

The following diagram shows how traffic is split between the microservices running in the cloud and the legacy environment.



Step7: Change the traffic route to the target environment

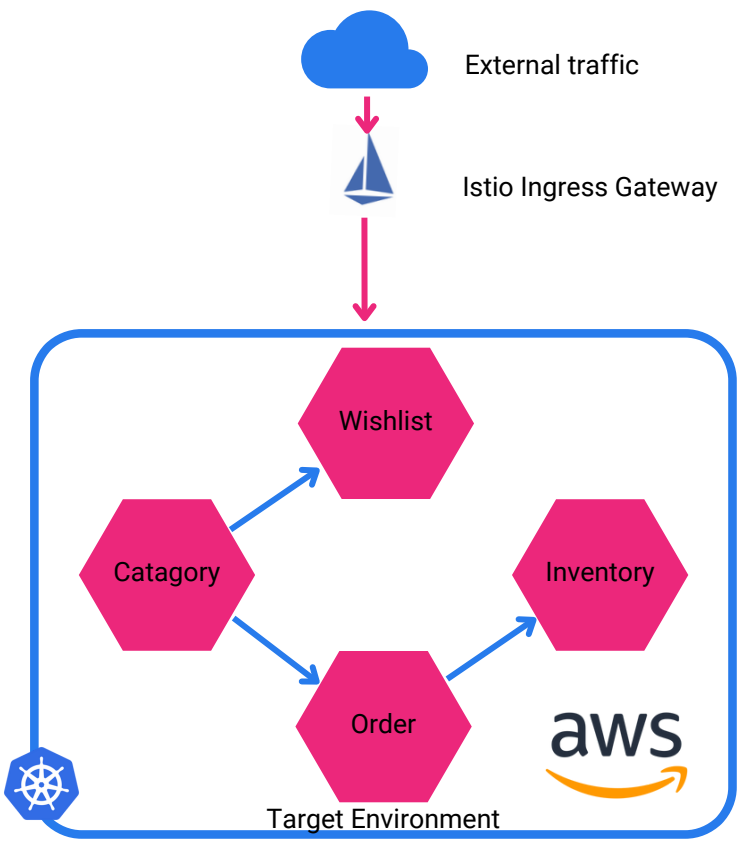
With each iteration of increasing traffic and testing, there will be a point in time to route 100% of the traffic to the new cloud environment. The image below represents traffic routed to the new environment, whereas the legacy system is still running for emergencies or rollback plans.



Step8: Retire the legacy environment

Wait for a few days, and ensure no change failures or significant incidents due to the migration; you can decide to phase out the legacy environment. You double-check Istio is not routing any portion of traffic to legacy environments and the target system is battle-proof, serving customers without any hurdles.

The image below represents the running target environment only.



Conclusion

A cloud migration strategy requires careful planning, execution, and testing. The most crucial goal of cloud migration is to get a high-available and scalable system without zero downtime or affecting customer experience. And thus, architects, cloud engineers, and developers must consider the network as the most crucial element of the migration strategy. They can abstract the network complications from the core business logic to migrate their workloads to the cloud seamlessly.

Authors

Ravi Verma, CTO, IMESH

Ravi is the CTO of IMESH. Ravi, a technology visionary, brings 12+ years of experience in software development and cloud architecture in enterprise software. He has led R&D divisions at Samsung and GE Healthcare and architected high-performance, secure and scalable systems for Baxter and Aricent. His passion and interest lie in network and security. Ravi frequently discusses open-source technologies such as Kubernetes, Istio, and Envoy Proxy from the CNCF landscape.

Ashish Ashutosh Das, VP Engineering, IMESH

Ashish has 12+ years of experience as software development and solution architect in a global IT solution provider. He has primarily designed end-to-end enterprise solutions, leveraging open source technologies, for telecom and retail companies across Africa and the UK region. He has provided cloud migration strategies and transformed the IT landscape for large companies like British Telecom, and Vodafone. His interest is in cloud and microservices, and often speaks about Kubernetes and service mesh.

About IMESH

IMESH offers Kubernetes-native application network and security platform to manage multi-cloud and hybrid cloud environments. The IMESH platform is built on top of Istio service mesh and Envoy API gateway and helps cloud, platform and security teams to make Kubernetes application more secure, manageable, and reliable.

Visit: https://imesh.ai/
email: contact@imesh.ai