**App Mesh**

links :

1. <https://aws.amazon.com/blogs/compute/learning-aws-app-mesh/>
2. <https://aws.amazon.com/app-mesh/>

Resources :

Saurav-poc-key ->

Private key file link : <https://drive.google.com/open?id=1OiPVEHpO3xC_Cli7PRdg-GFQJM2V3Q5Q>

Saurav-poc-key1 ->

Private key file link : <https://drive.google.com/open?id=1DNqA239R3-D6wgRbdz_DlLuV9fFx3aPQ>

Ec2 Instances :

1. ec2-100-26-198-128.compute-1.amazonaws.com ->having spring boot application

ssh -v -i "saurav-poc-key.pem" [ec2-user@ec2-100-26-198-128.compute-1.amazonaws.com](mailto:ec2-user@ec2-100-26-198-128.compute-1.amazonaws.com)

1. ec2-3-92-188-175.compute-1.amazonaws.com -> having spring boot application

ssh -v -i "saurav-poc-key.pem" [ec2-user@ec2-3-92-188-175.compute-1.amazonaws.com](mailto:ec2-user@ec2-100-26-198-128.compute-1.amazonaws.com)

1. ec2-54-234-152-69.compute-1.amazonaws.com -> client application from where we hit rest api’s

ssh -v -i "saurav-poc-key.pem" [ec2-user@ec2-54-234-152-69.compute-1.amazonaws.com](mailto:ec2-user@ec2-100-26-198-128.compute-1.amazonaws.com)

App mesh :

Name : apps

-> curl [ec2-100-26-198-128.compute-1.amazonaws.com:80/](http://ec2-100-26-198-128.compute-1.amazonaws.com/) → hit spring boot api

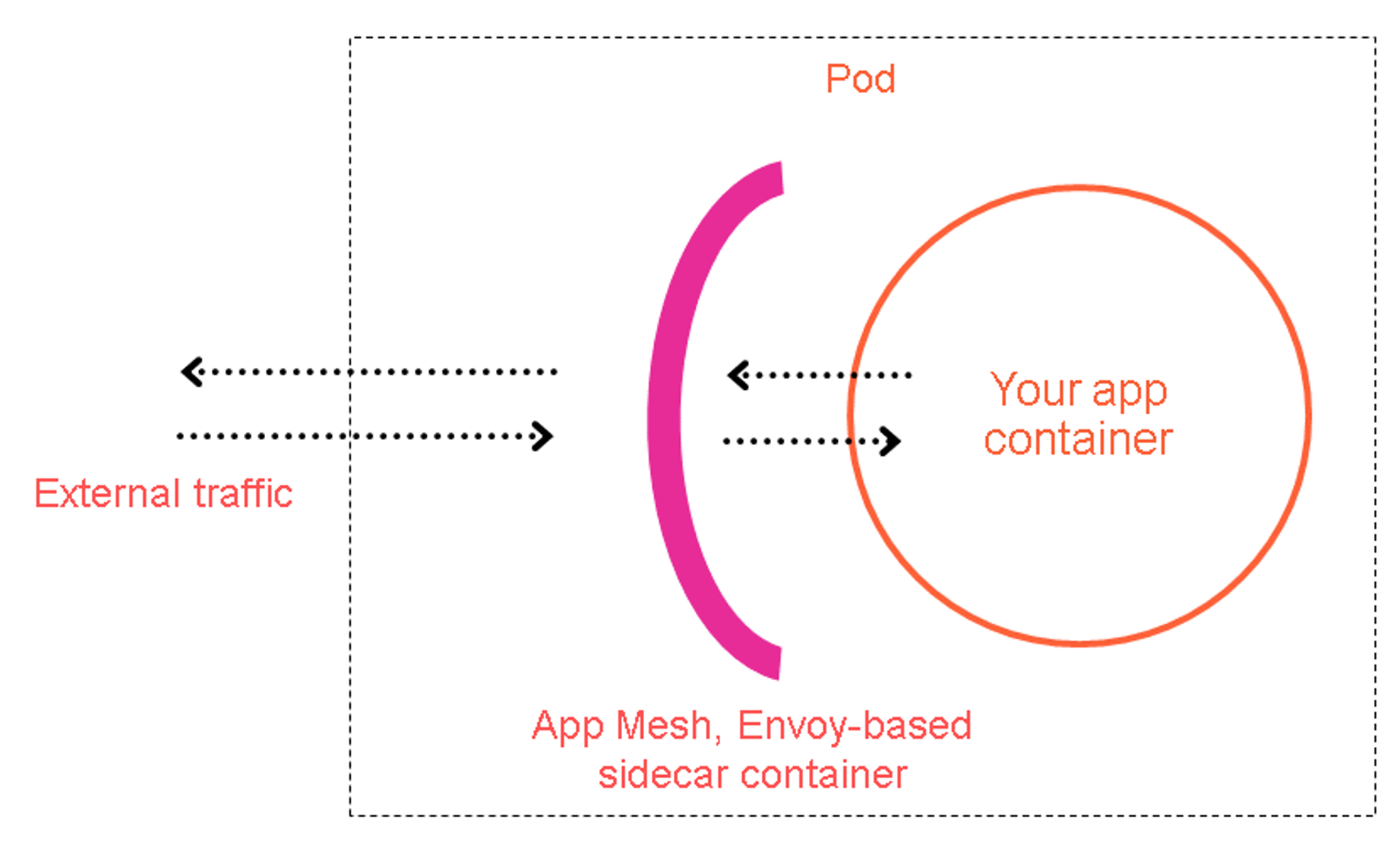
-> sudo java -jar target/spring-boot-web-application-example-0.0.1-SNAPSHOT.war --server.port=80 → run spring boot application

-> AWS announced [AWS App Mesh](https://aws.amazon.com/app-mesh), a service mesh that provides application-level networking. App Mesh makes it easy for your services to communicate with each other across multiple types of compute infrastructure, including:

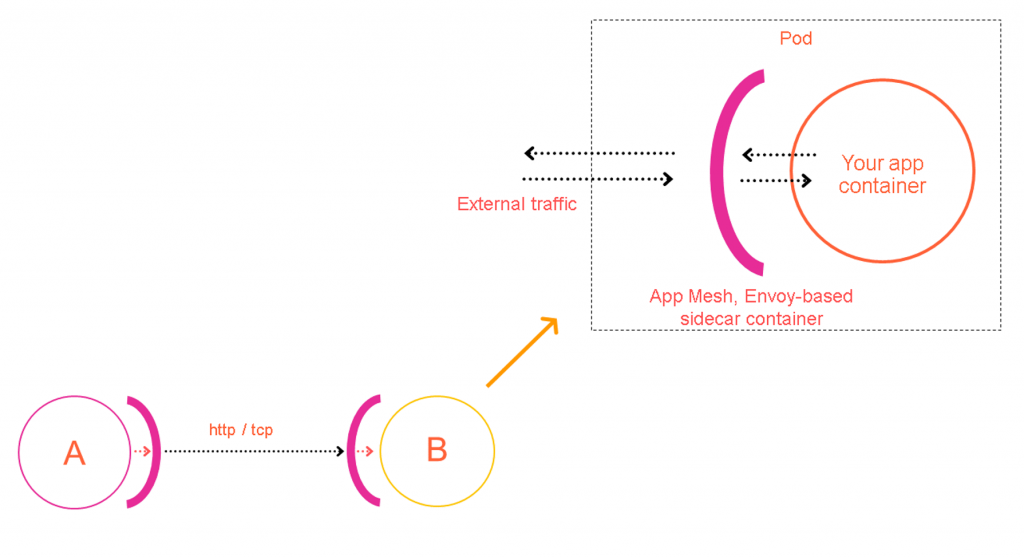
1. [Amazon EKS](https://aws.amazon.com/eks/)
2. [Amazon ECS](https://aws.amazon.com/ecs/)
3. [Kubernetes](https://kubernetes.io/) on [Amazon EC2](https://aws.amazon.com/ec2/)

-> App Mesh standardizes how your services communicate, giving you end-to-end visibility and ensuring high availability for your applications. Service meshes like App Mesh help you run and monitor HTTP and TCP services at scale.

-> Using the open source [Envoy](https://www.envoyproxy.io/) proxy, App Mesh gives you access to a wide range of tools from AWS partners and the open source community. Because all traffic in and out of each service goes through the Envoy proxy, all traffic can be routed, shaped, measured, and logged. This extra level of indirection lets you build your services in any language desired without having to use a common set of communication libraries.

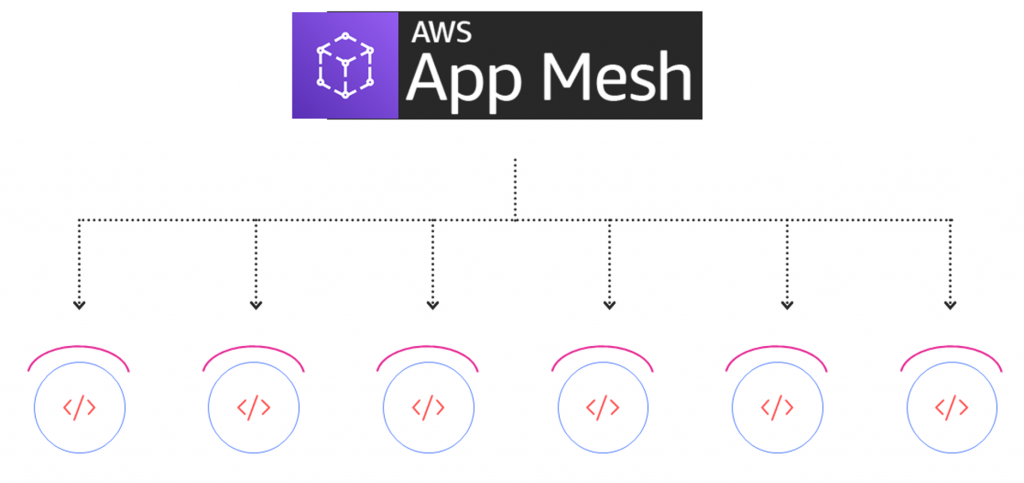


-> With a service mesh, you can decouple your microservices’ observability, analytics, and routing logic from the underlying infrastructure and application layers.



-> App Mesh implements this sidecar proxy via the production-proven Envoy proxy. Envoy is arguably the most popular open-source service proxy.

-> In the following diagram, a sidecar runs alongside each container in your application to provide its proxying logic, syncing each of their unique configurations from the App Mesh control plane.



-> Each one of these proxies must have its own unique configuration ruleset pushed to it to operate correctly. To achieve this, DevOps teams can push their intended ruleset configuration to the App Mesh API. From there, the App Mesh control plane reliably keeps all proxy instances up-to-date with their desired configurations. App Mesh dynamically scales to hundreds of thousands of pods, tasks, EC2 instances, and Lambda functions, adjusting configuration changes accordingly as instances scale up, down, and restart.

## **App Mesh components**

-> App Mesh is made up of the following components:

* [Service mesh](https://docs.aws.amazon.com/app-mesh/latest/userguide/meshes.html): A logical boundary for network traffic between the services that reside within it.
* [Virtual nodes](https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual_nodes.html): A logical pointer to a Kubernetes service, or an App Mesh virtual service.
* [Virtual routers](https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual_routers.html): Handles traffic for one or more virtual services within your mesh.
* [Routes](https://docs.aws.amazon.com/app-mesh/latest/userguide/routes.html): Associated with a virtual router, it directs traffic that matches a service name prefix to one or more virtual nodes.
* [Virtual services](https://docs.aws.amazon.com/app-mesh/latest/userguide/virtual_services.html): An abstraction of a real service that is either provided by a virtual node directly, or indirectly by means of a virtual router.
* **App Mesh sidecar:** The App Mesh sidecar container configures your pods to use the [App Mesh service mesh](https://docs.aws.amazon.com/app-mesh/latest/userguide/envoy.html) traffic rules set up for your virtual routers and virtual nodes.
* **App Mesh injector:** Makes it easy to auto-inject the App Mesh sidecars into your pods.
* **App Mesh custom resource definitions:** (CRD) Provided to implement App Mesh CRUD and configuration operations directly from the kubectl CLI. Alternatively, you may use the latest version of the [AWS CLI](https://aws.amazon.com/cli/).