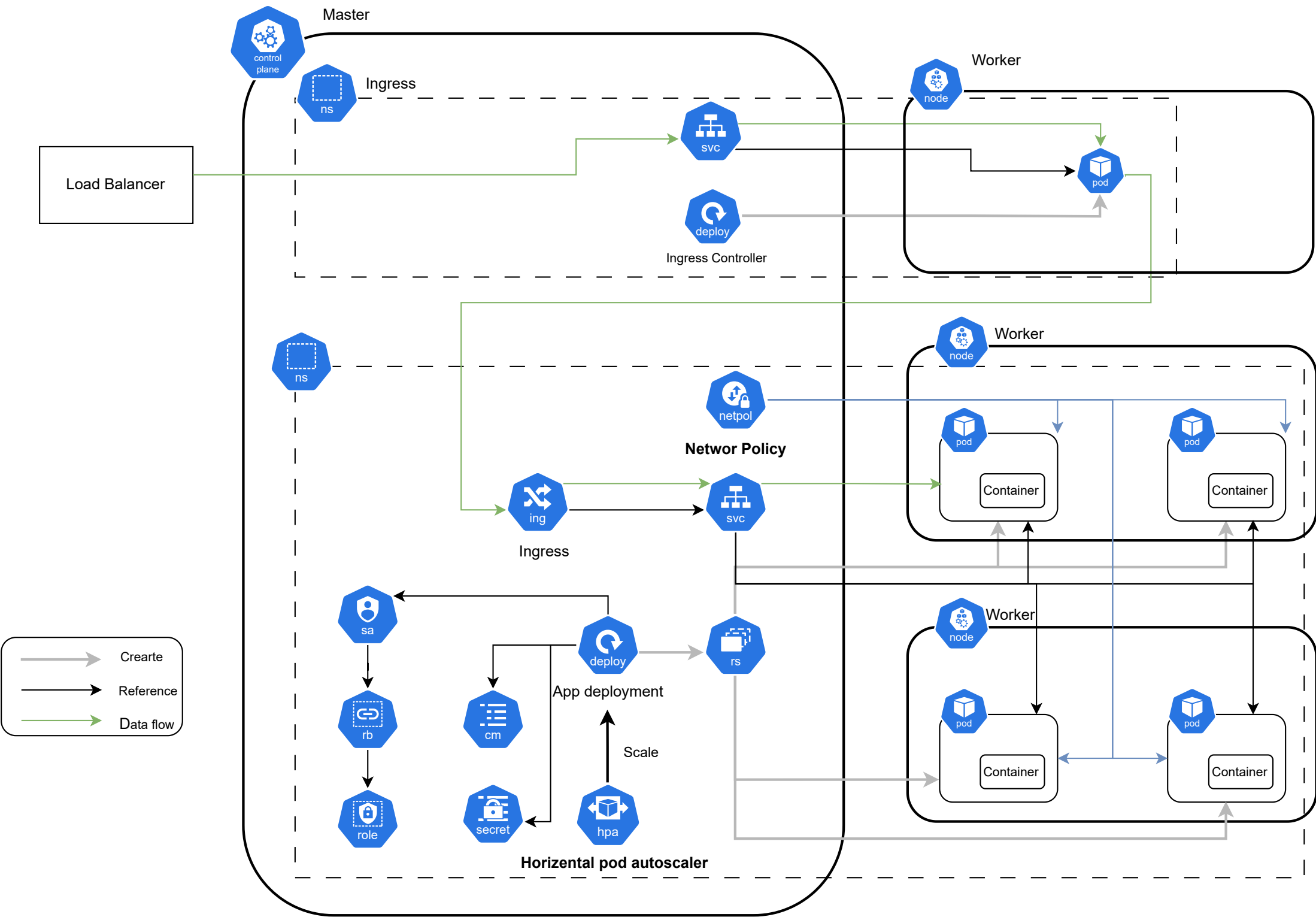**High Availability**:
We ensure high availability by running multiple replicas of our application using the Deployment object. We use a Rolling Update strategy to ensure that the application remains available during updates and upgrades. Additionally, we use an Ingress Controller to manage external access to services in the cluster,providing a single entry point and enabling load balancing for increased availability.

Scalability:
We achieve scalability through horizontal pod autoscaling, which monitors the CPU and memory usage of pods. When the usage exceeds a specified target, the autoscaler automatically scales the deployment by creating new pods. This ensures that our application can handle increased traffic by dynamically adjusting its resources.

Master

control plane

Worker

node

Ingress

ns

svc

pod

deploy

Ingress Controller

Load Balancer

ns

Worker

node

netpol

Networ Policy

pod

pod

Container

Container

ing

svc

Ingress

sa

deploy

rs

App deployment

cm

Worker

node

pod

pod

rb

Container

Container

role

secret

hpa

Scale

Horizental pod autoscaler

Crearte

Reference

Data flow

Security:

To ensure security in our cluster, we use network policies to secure communication between pods. Additionally, we employ roles and role bindings to grant only the necessary permissions to our deployments within the cluster, and we use secrets to secure our credentials.

Data Flow:

External requests to the cluster flow through the load balancer, which redirects them to the Ingress Controller. The Ingress Controller, based on defined rules in the Ingress object, forwards the requests to the appropriate service, which then directs the traffic to one of the specific pods. Note that while the diagram shows the service passing the request to only one pod for clarity, in reality, it can be redirected to any one of the pods for load balancing.