Red Hat Summit

Connect

Red Hat Service Interconnect

Red Hat

Andrzej Kowalczyk

Associate Principal Solution Architect
Red Hat

Applications reside in a diverse mix of environments

Either On-Premises, in the Public Cloud, or at the Edge

Multiple versions of OpenShift

OpenShift 3.x, OpenShift 4.x, ARO, ROSA

Other Kubernetes Offerings

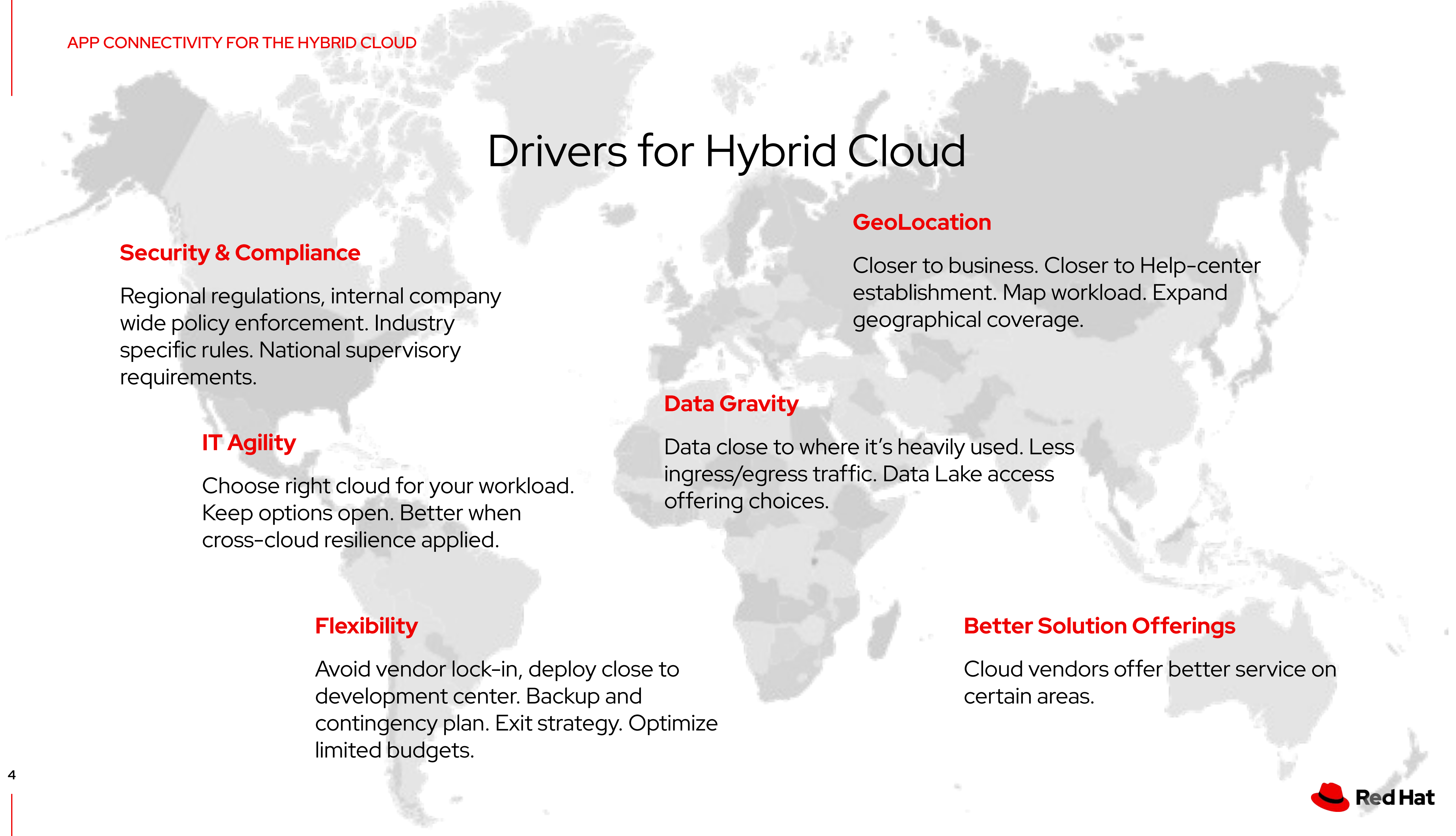
Kubernetes from hyperscalers (Amazon EKS, Azure AKS, Google GKE) Vanilla Kubernetes

Bare metal and VMs

Variety of bare metal and VM environments running existing existing services

Legacy Systems

Old unixes, Mainframes



Drivers for Hybrid Cloud

Security & Compliance

Regional regulations, internal company wide policy enforcement. Industry specific rules. National supervisory requirements.

IT Agility

Choose right cloud for your workload. Keep options open. Better when cross-cloud resilience applied.

Flexibility

Avoid vendor lock-in, deploy close to development center. Backup and contingency plan. Exit strategy. Optimize limited budgets.

GeoLocation

Closer to business. Closer to Help-center establishment. Map workload. Expand geographical coverage.

Data Gravity

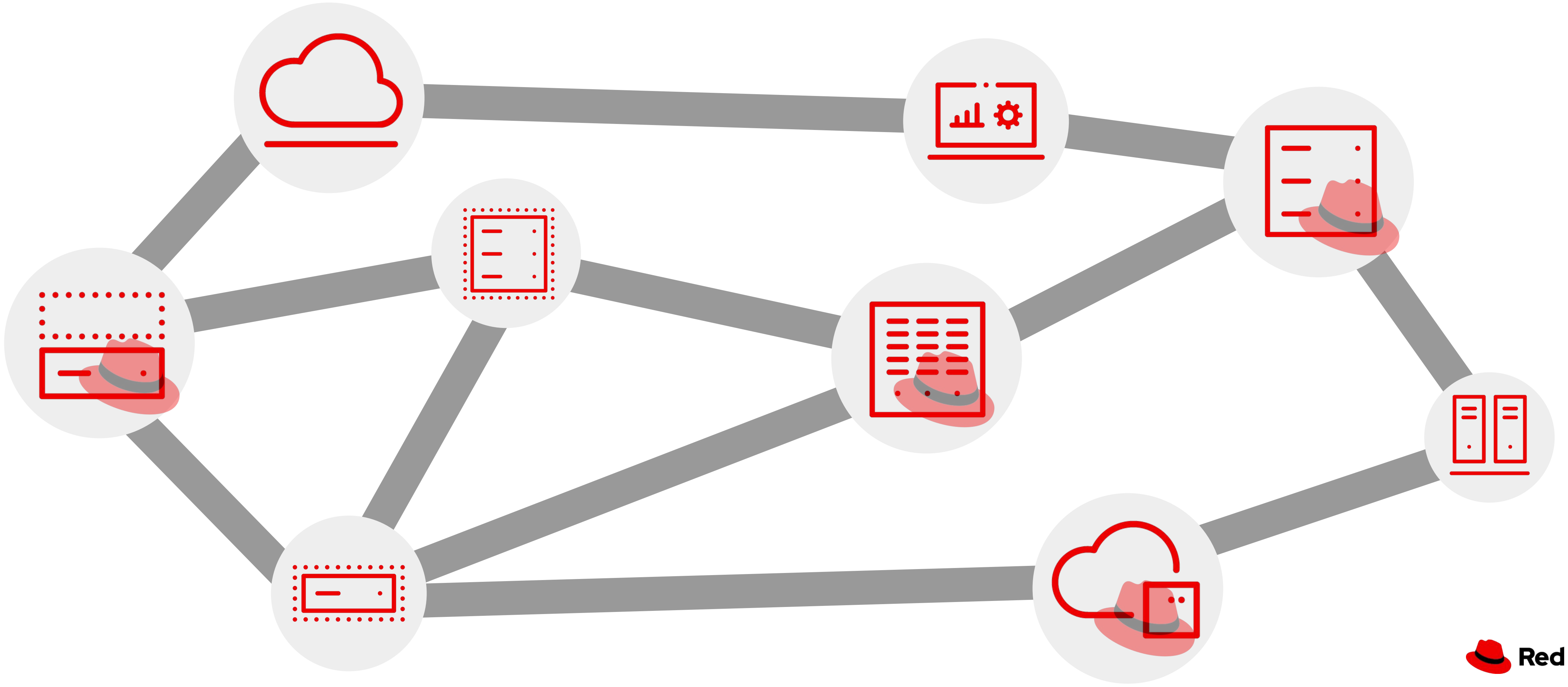
Data close to where it's heavily used. Less ingress/egress traffic. Data Lake access offering choices.

Better Solution Offerings

Cloud vendors offer better service on certain areas.

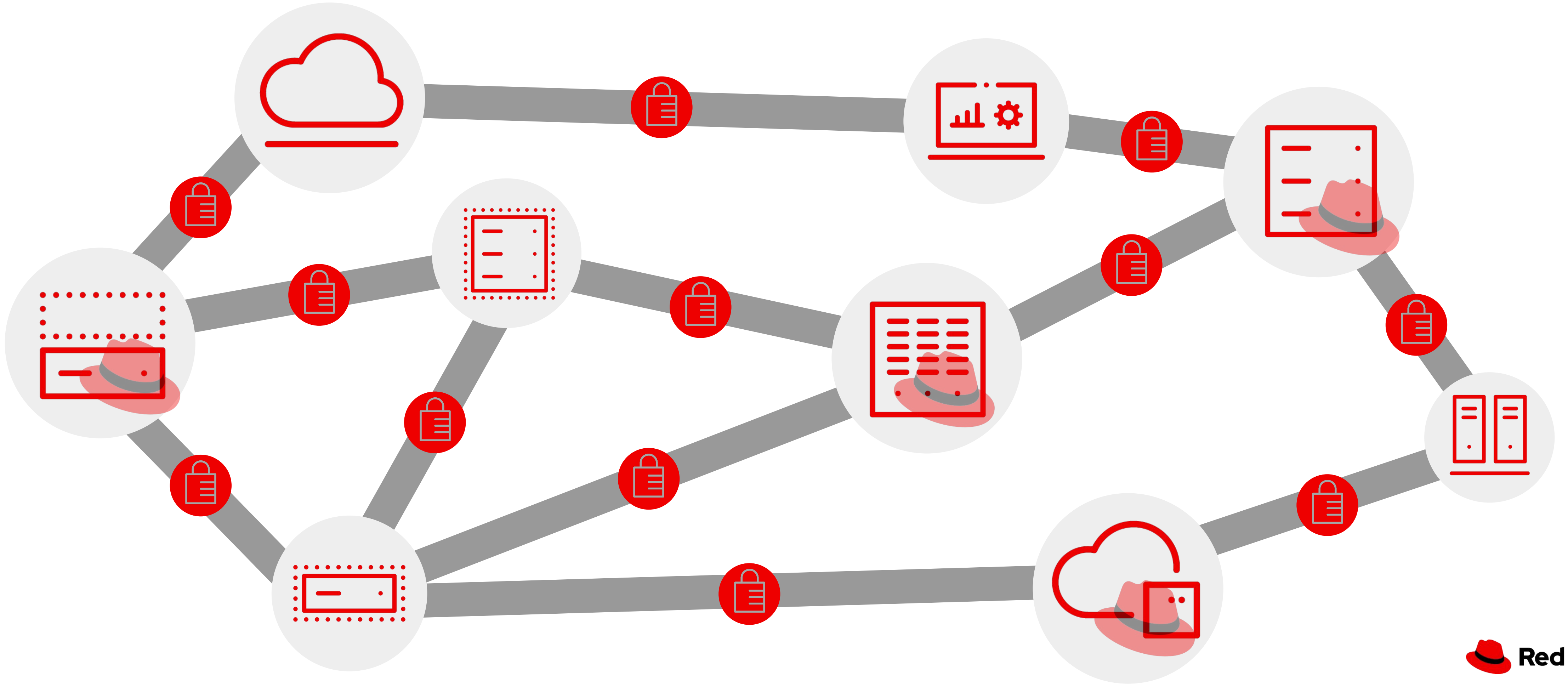
Interconnectivity delivers value

Combining different capabilities helps organizations deliver products and services.



Interconnections must be protected

Interconnections should not compromise the infrastructure or data



Connectivity Options/Choices

Public IP Networks

No network isolation

No connectivity to sites behind NAT or Firewalls

Each IP is a co$t

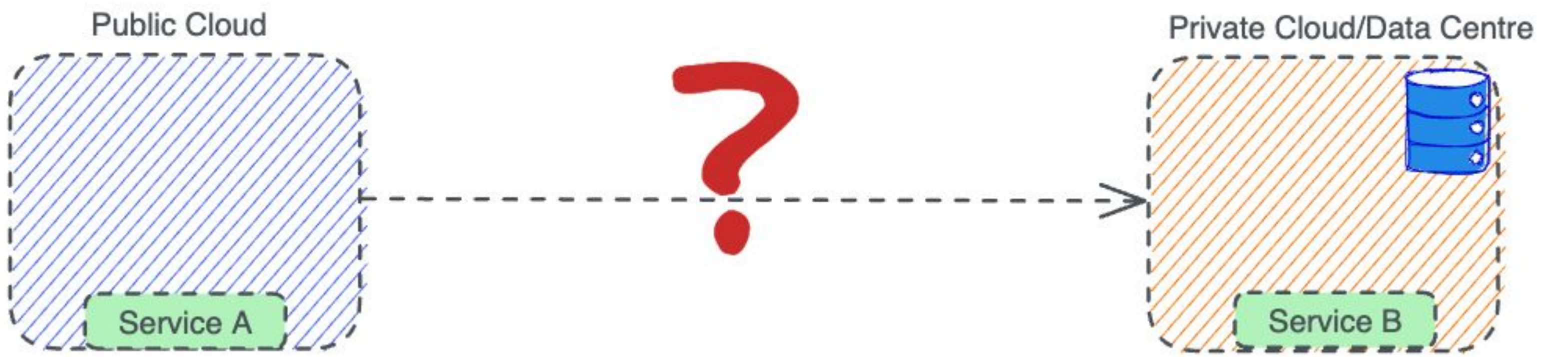
Set up your own VPN network

Network isolation

Complexity (iptables and firewall rules)

Hub-n-spoke topology

Requires administrator privileges



Red Hat Service Interconnect

Larger Provider Networks(AWS VPC)

Network isolation

Vendor lock in

Requires cluster privileges

Each connection is a co$t

Overlay Network (VAN)

Fine-grained network isolation

Low complexity

Developer controlled

Very low cost for additional resource

Characteristics of a Virtual Application Network (VAN)

Overlay Network

VAN is an application-layer(layer 7) network that is overlaid on top of the existing endorsed networks.

Addressing

VAN address references a running process or API endpoints, not a host

Network Portability

Routes application traffic based on the VAN address, not the underlying IP addresses

Multicast/Anycast

VAN addresses are assumed to be multi-access, where multiple destinations can use the same address

Security

All of the inter-site connections in a VAN are locked down using mutual TLS (Transport Layer Security) with a private, dedicated certificate authority

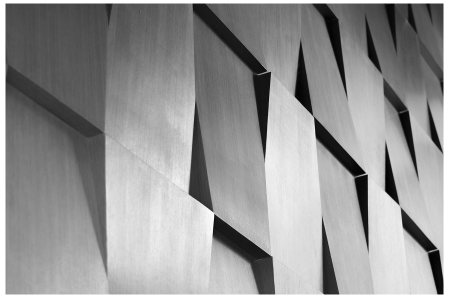
Lightweight & Ephemeral

Easy to set up and easy to clean network. Application networks and service bindings can be transitory

Source:
Insert source data here
Insert source data here

Red Hat Service Interconnect

Application connectivity with Layer 7 VAN across platforms, clusters, and clouds



Application Focused Integration

Individual Apps running on virtually any platform can make native TCP calls locally to any other app running on any other platform securely without special VPNs.

Mutual TLS Encryption

Interconnections use Mutual TLS in order to prevent unauthorized interconnections.

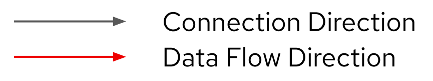
Application Layer Abstraction

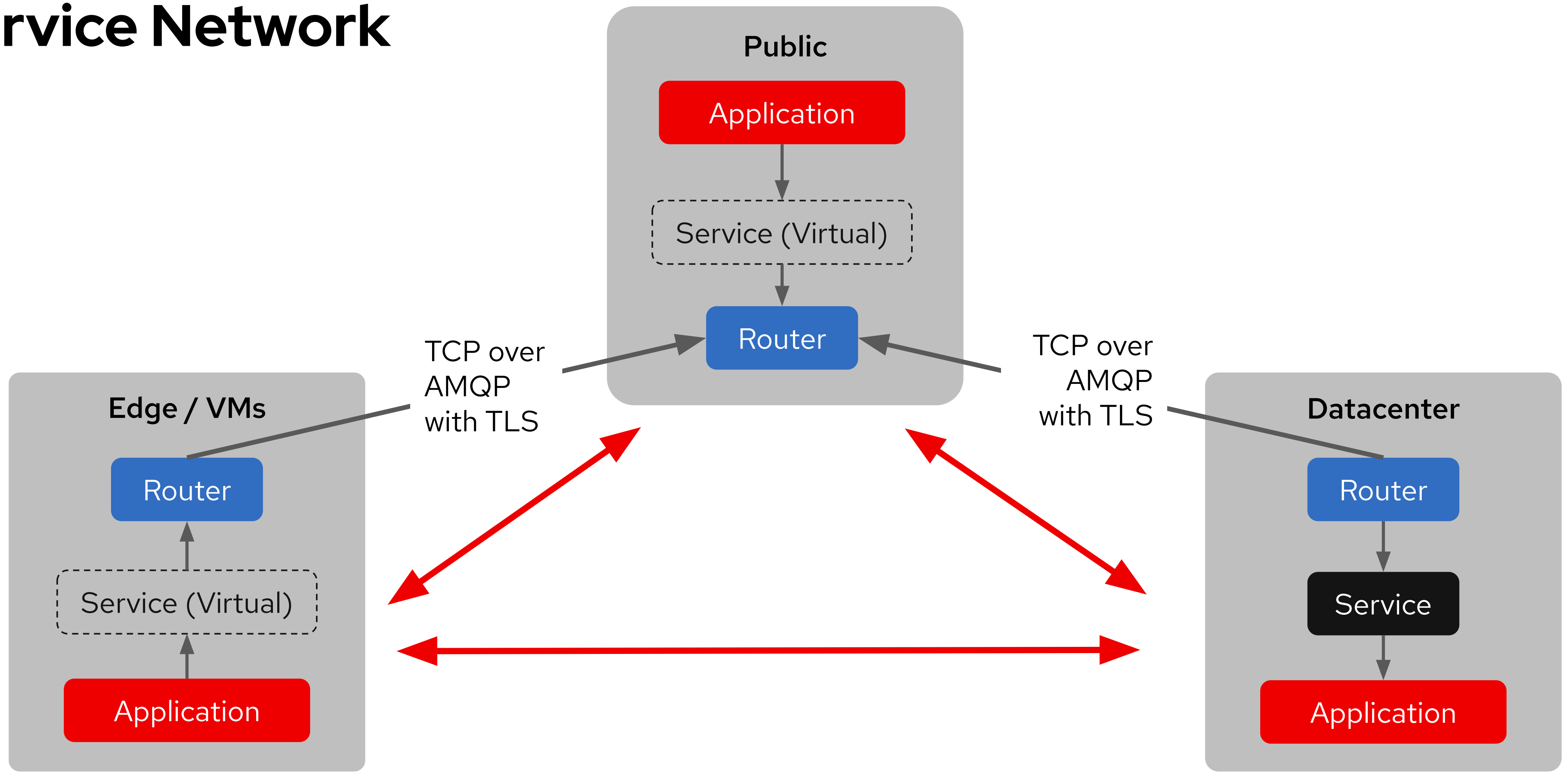
Agnostic of the environment and IP versions (such as IPv4 and IPv6) Enables portability for both applications and its associated networking. Migrations can be easily done without recreating the networking.

Layer 7 Addressing

Instead of routing IP packets between network endpoints, Layer 7 application routers route messages between application addresses

Service Network



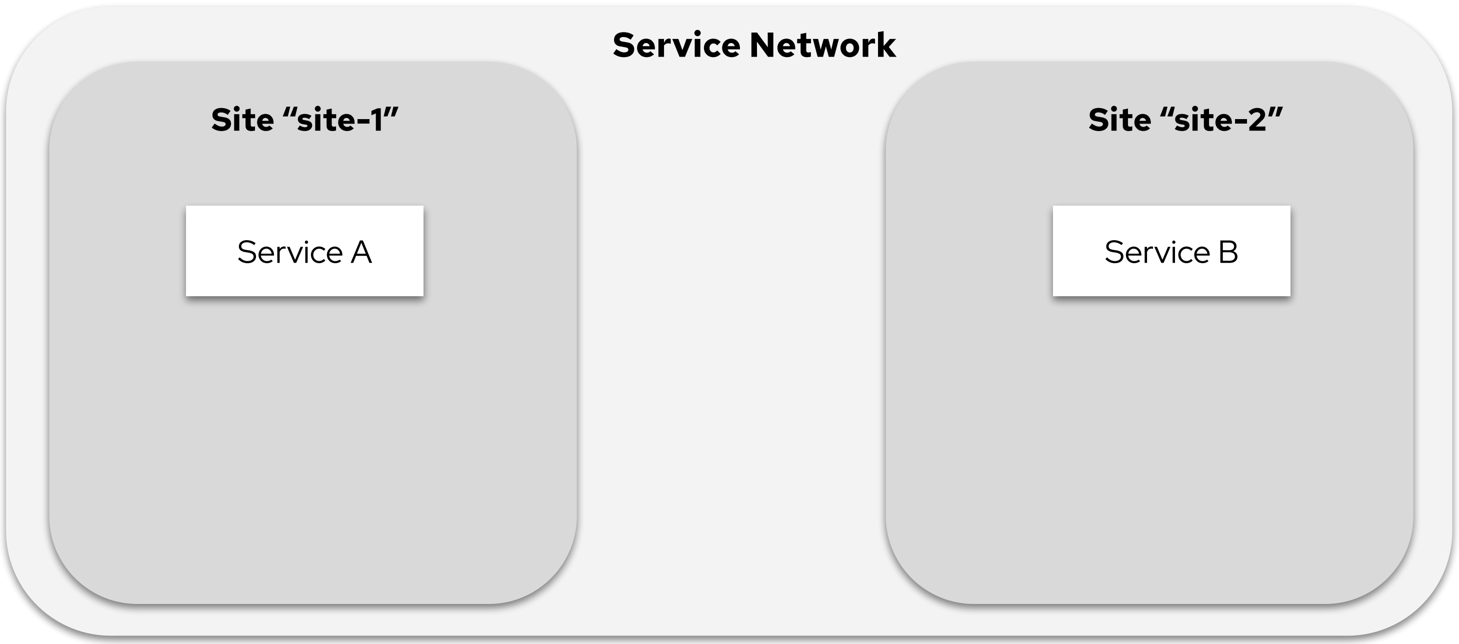


Portable Simple Protected Hybrid

Service Interconnect Concepts and Terminology

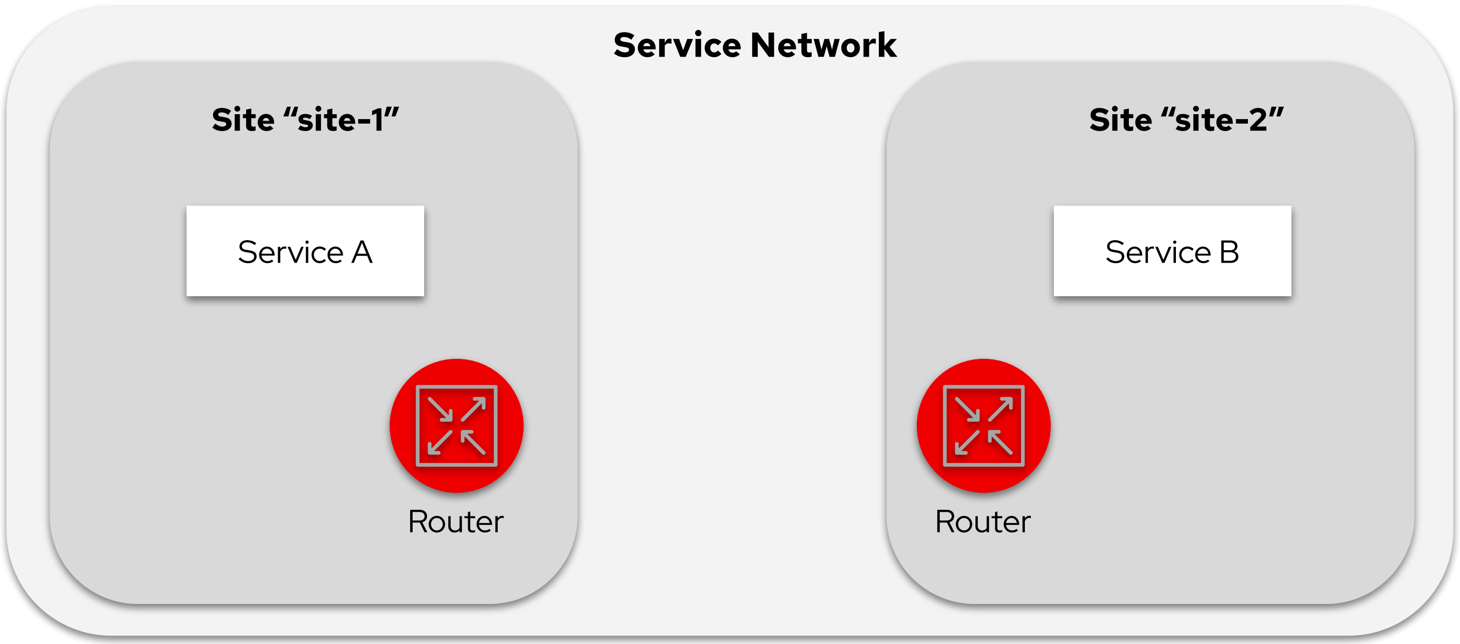
Understanding some key concepts, components and terminology of Red Hat Service Interconnect

Red Hat



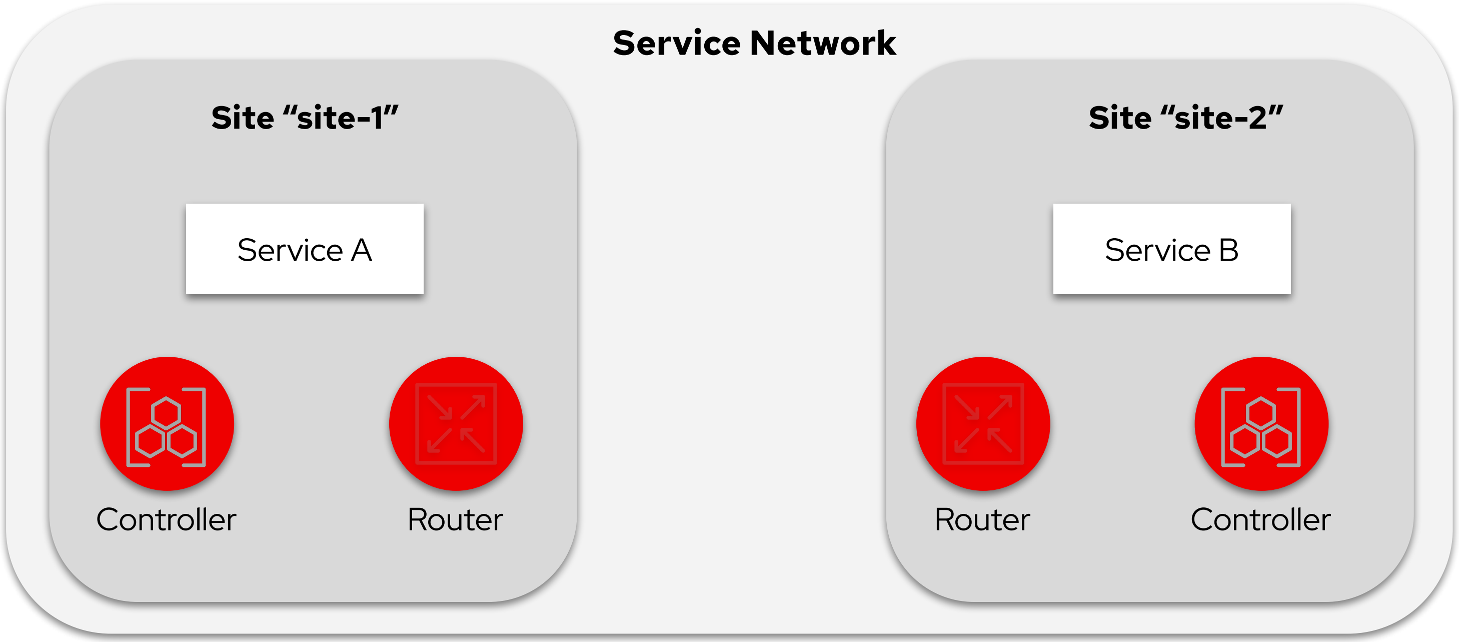
Site

- RHSI network is composed of sites. A site is a place where components of your distributed application are running.
- Site can be a K8s namespace, virtual machine, bare metal
- In this example, "site-1" and "site-2" must be linked to form the network for Service A and B to communicate.



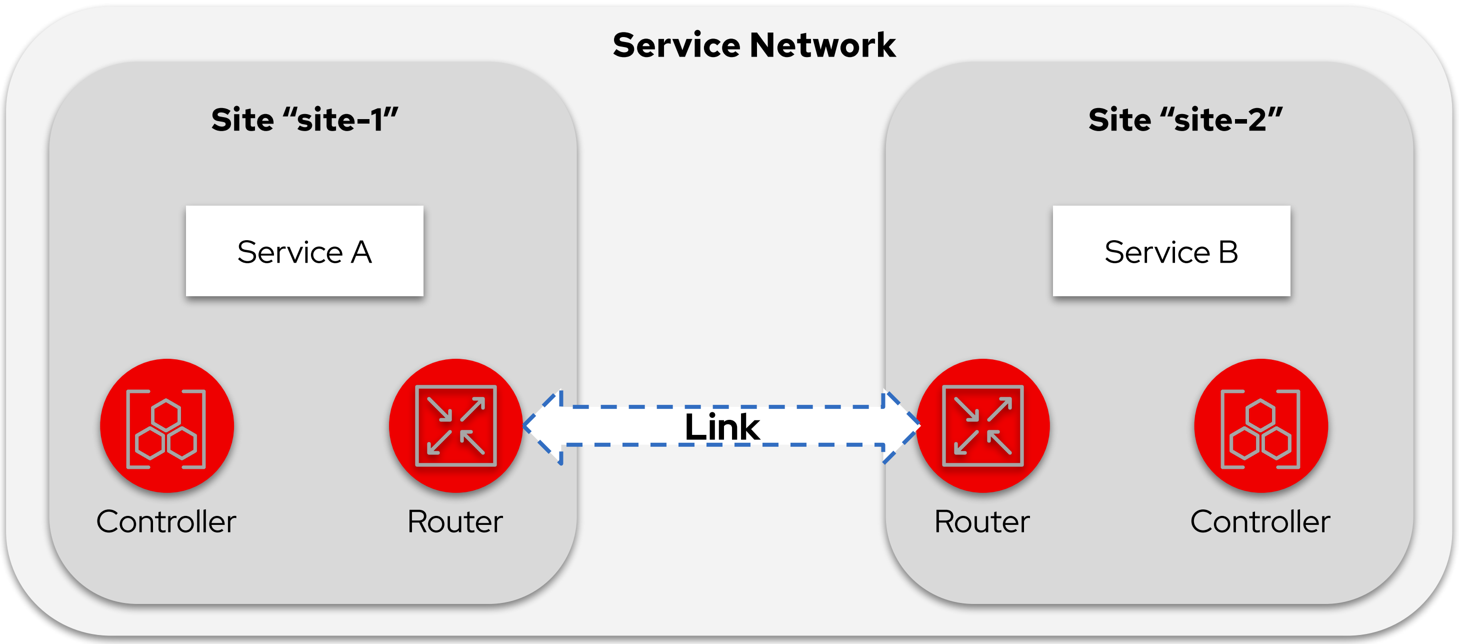
Router (Data Plane)

- Key component for establishing connectivity between sites. Installed in all the sites in the network
- Communication across the network happens between the routers
- Routers establish links with assigned peers
- Determine shortest path based on message exchange
- Exchange target address updates
- Delivery pattern (anycast, multicast)
- Automatic recovery to failure by re-routing
- Dynamic and stateless



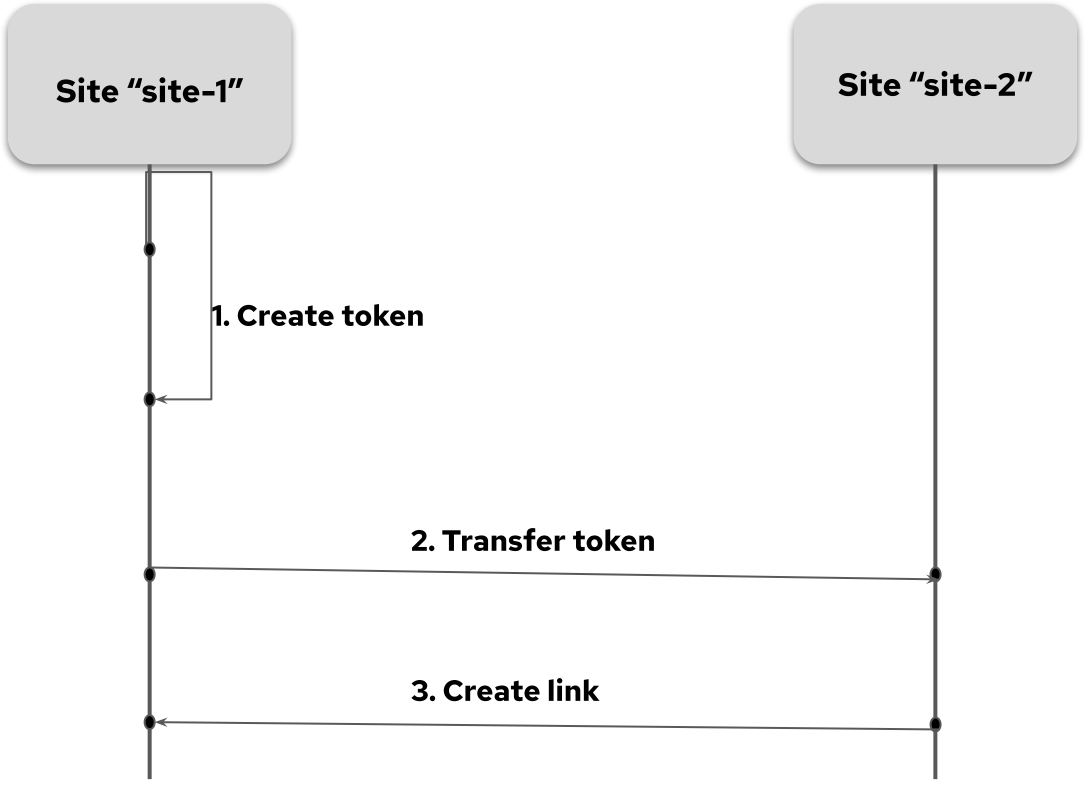
Controller (Control Plane)

- Collection of control loops to monitor K8s resources and services, translates them into router configuration
- Understands the network topology and maintains router configuration
- Expose and communicate service availability across the router network
- Responsible for Service Sync → A Protocol to provide periodic updates on what services are exposed across the network. Can be turned off
- CA for generating tokens
- Certificate for router used on inter-route and edge connections



Link

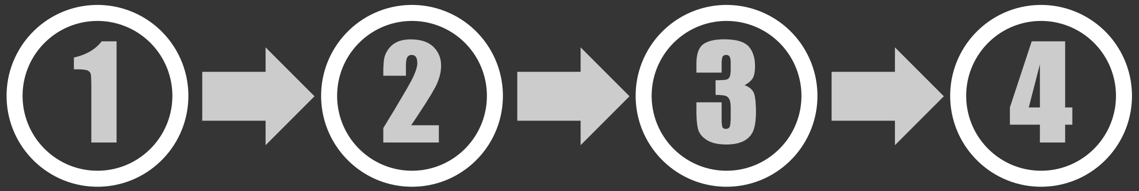
- Sites use links to form a dedicated network for your application. These links are the basis for site-to-site and service-to-service communication.
- A link is a site-to-site communication channel. Links serve as a transport for application traffic such as connections and requests
- Links are always secured using mutual TLS authentication and encryption.
- Uni directional connectivity is enough to establish a bidirectional link



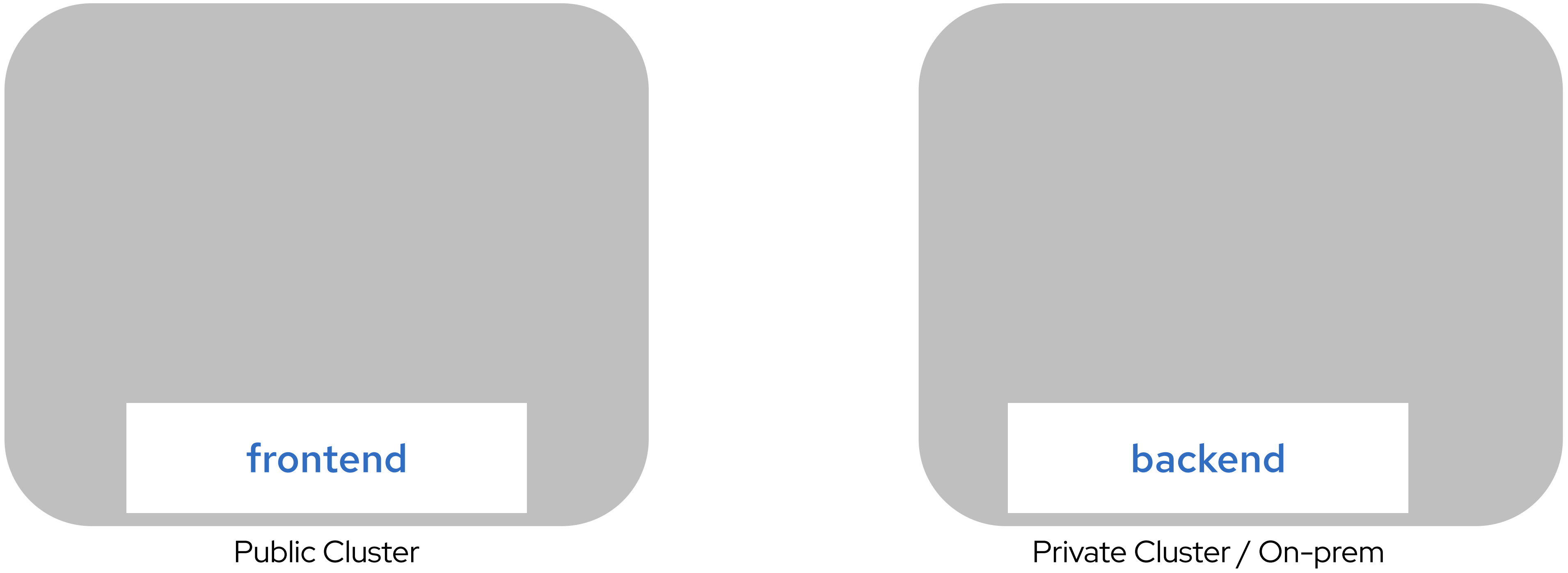
Token

- Creating a link requires explicit permission from the target site. This permission is granted using tokens. A token contains a URL for the target site and a secret key.
- Tokens can be restricted to a chosen number of uses inside a limited time window. By default, tokens allow only one use and expire after 15 minutes.
- In this example, site "site-1" wishes to allow "site-2" to create a link. Site "site-1" creates a token. The owner of "site-1" gives the token to the owner of "site-2". The owner of "site-2" then uses the token to create the link.

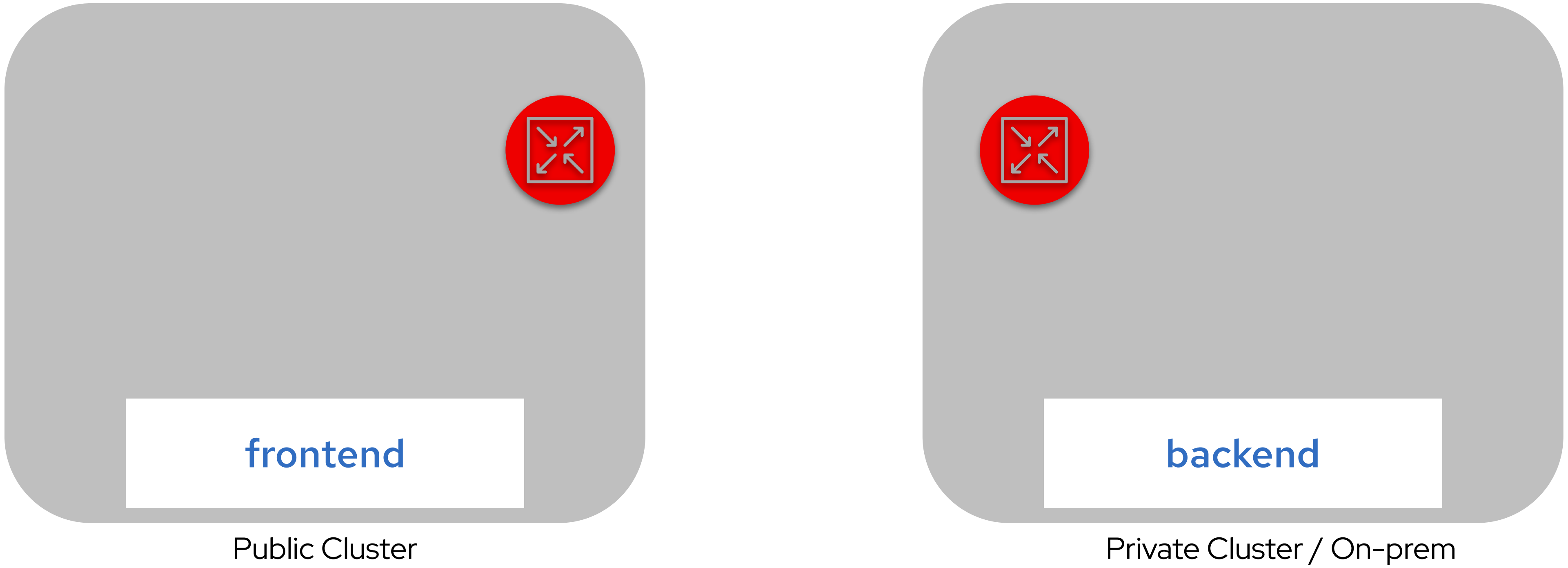
How to in 4 easy steps



Frontend and the backend spread Across Different Environments



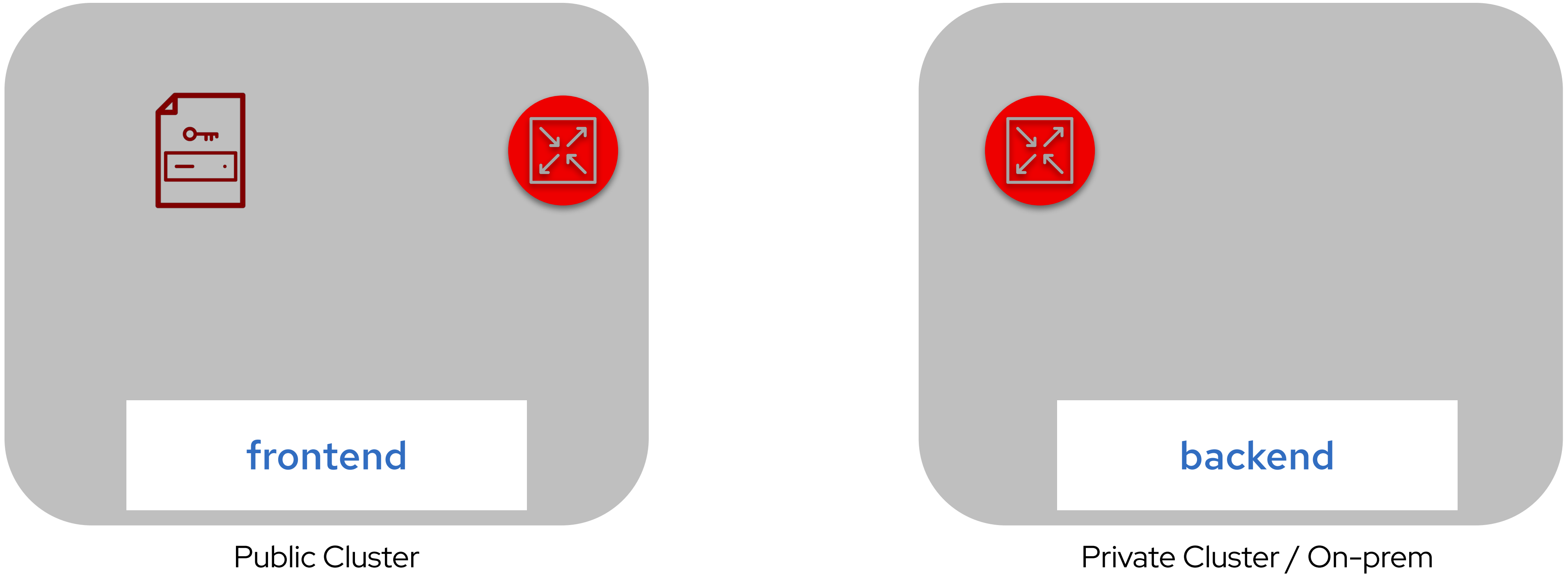
Initialize the Routers



```
$ skupper init
```

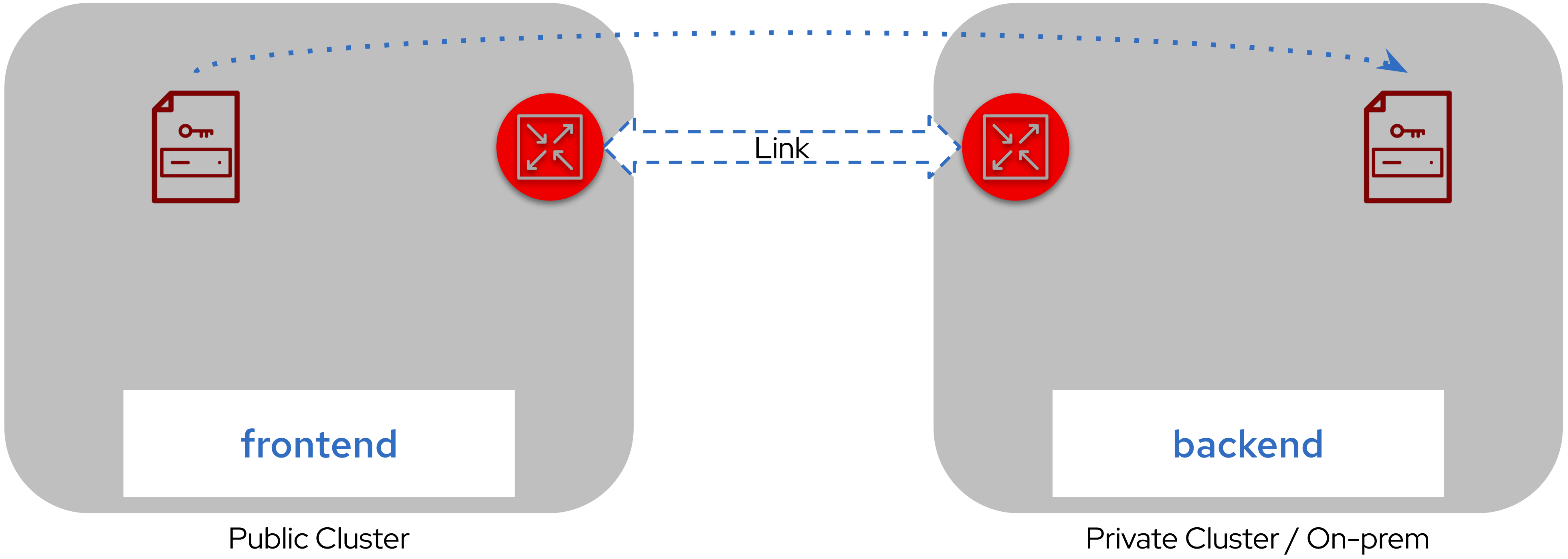
```
$ skupper init
```

Create a secure token



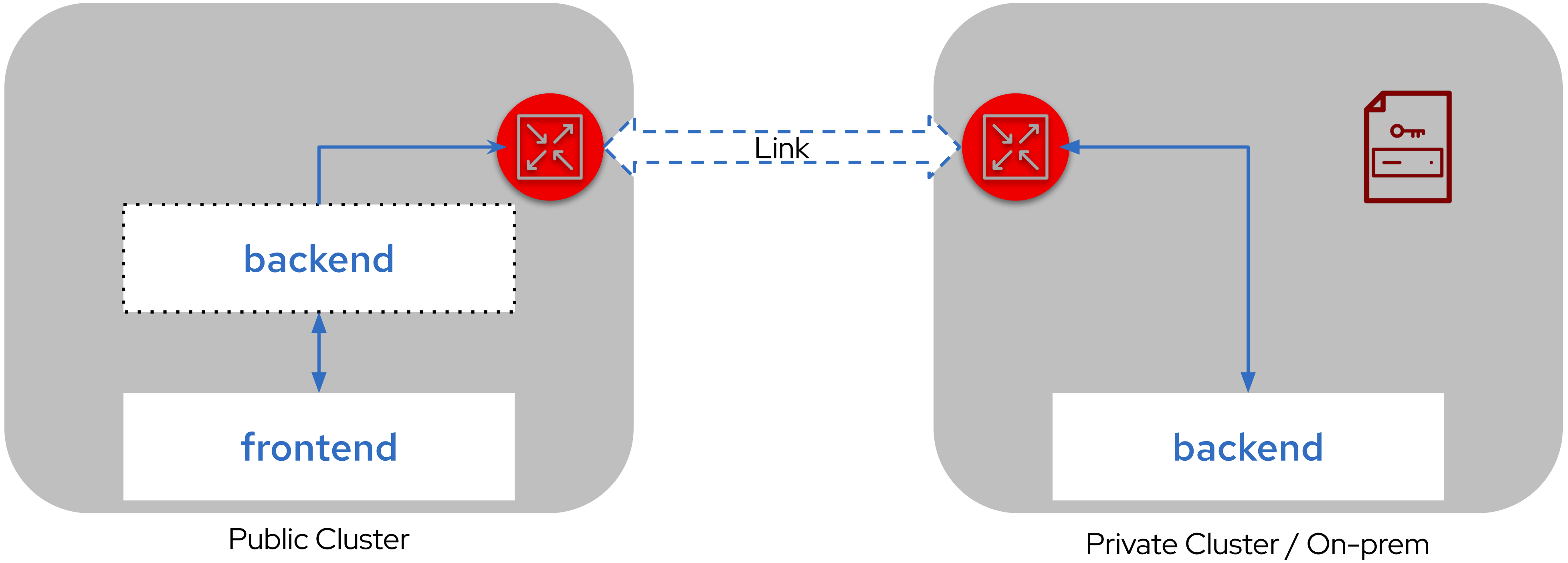
```
$ skupper token create secret.token
```

Transfer the token and link the sites using the token



```
$ skupper link create secret.token
```

Expose only the Required Services



```
$ skupper expose service backend
```

Simplicity

What makes Red Hat Service Interconnect unique is the ability to simplify application connectivity across Red Hat or non-Red Hat environments and platforms.

Eliminates Time Taking Complex Configurations

An application-layer solution can significantly reduce complexity and coordination delay

No code changes

You don't have to change your application code. Services communicate transparently as though they were deployed together in one location.

No network changes

You don't need new firewall rules, and you don't need your infra team to install a gateway. If you can connect (either way), you can create a service network.

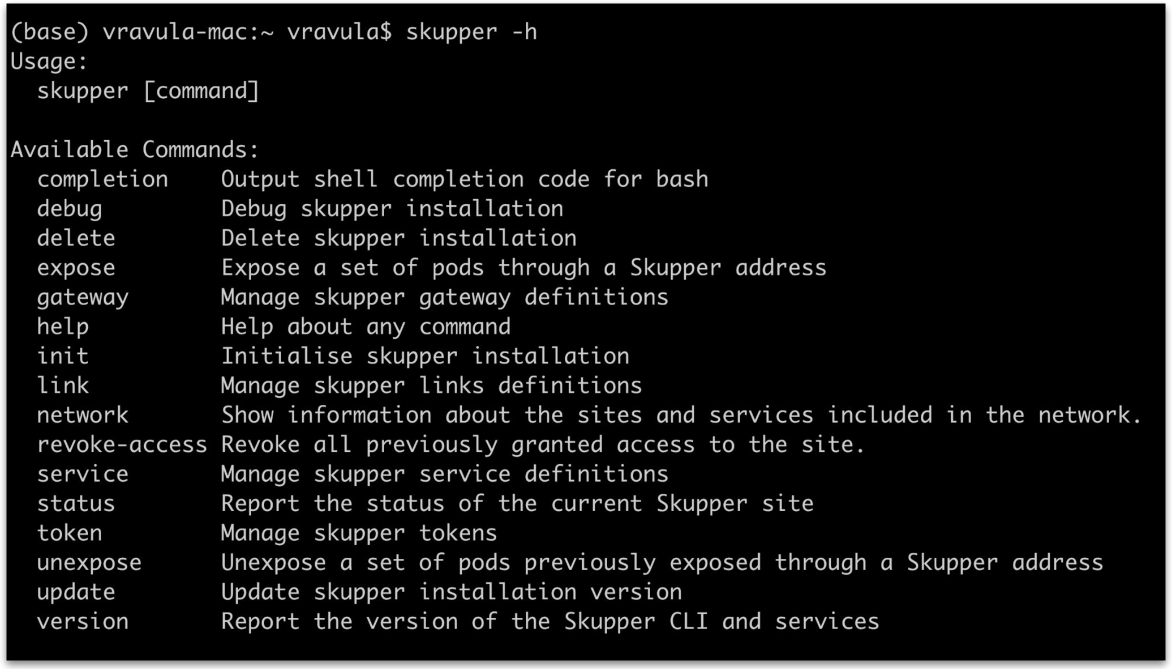
No admin privileges

It requires no elevated privileges to set up. Operates with the same privileges as your application.

Source:
Insert source data here
Insert source data here
Insert source data here

Simple CLI Based Configuration

CLI Command Structure



```
(base) vravula-mac:~ vravula$ skupper -h
Usage:
  skupper [command]

Available Commands:
  completion    Output shell completion code for bash
  debug         Debug skupper installation
  delete        Delete skupper installation
  expose        Expose a set of pods through a Skupper address
  gateway       Manage skupper gateway definitions
  help          Help about any command
  init          Initialise skupper installation
  link          Manage skupper links definitions
  network       Show information about the sites and services included in the network.
  revoke-access Revoke all previously granted access to the site.
  service       Manage skupper service definitions
  status        Report the status of the current Skupper site
  token         Manage skupper tokens
  unexpose      Unexpose a set of pods previously exposed through a Skupper address
  update        Update skupper installation version
  version       Report the version of the Skupper CLI and services
```

Service Management

Control the visibility of individual services in the network

Token Management

Create Secure Tokens for Establishing mTLS connections

Site Lifecycle

Manage the lifecycle of Skupper installations and components

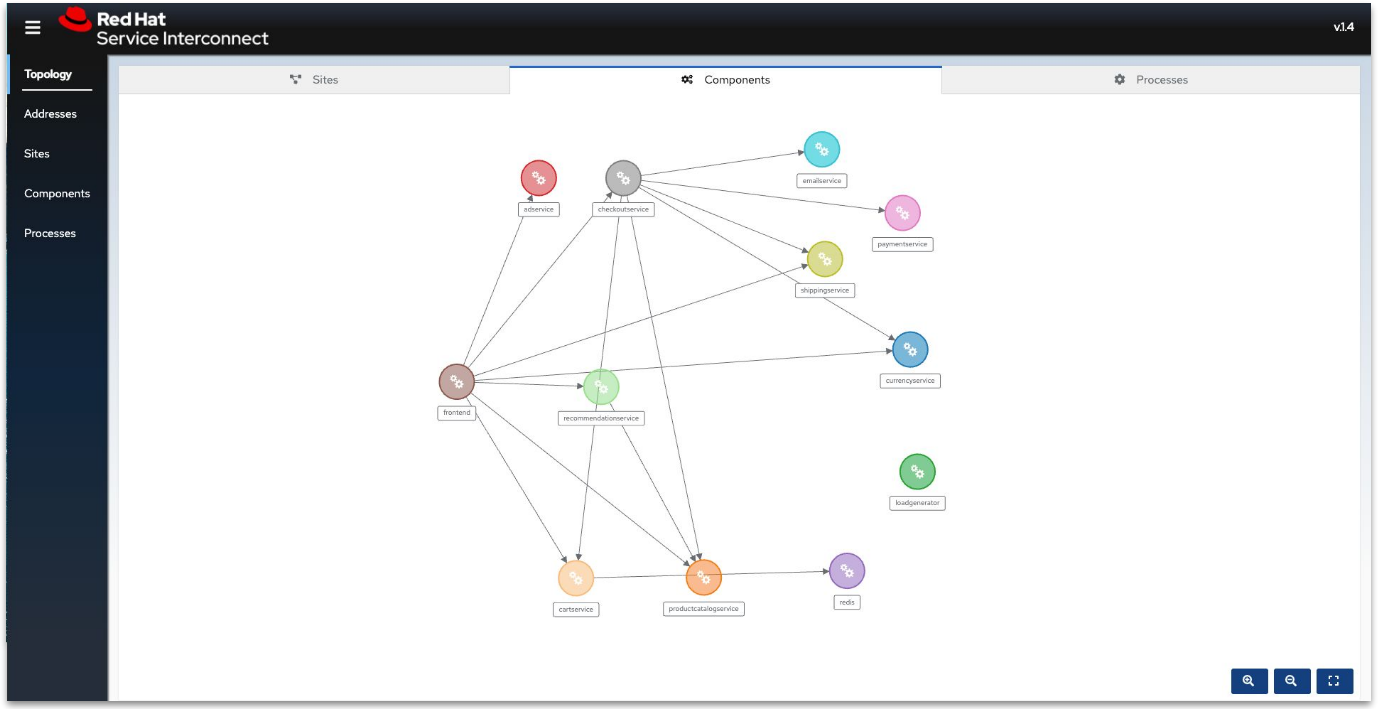
Link Management

Manage the connections and link definitions

Source:
Insert source data here
Insert source data here
Insert source data here

Console

Visualize your connections



- **Topology:** Graphical representation of all the connections
- **Components:** Services that are exposed on the service network, both local and remote.
- **Sites:** Application Interconnect installations on the current service network.
- **Throughput Bytes:** Charts providing traffic related information

Source:
Insert source data here
Insert source data here
Insert source data here

Portable Simple Protected Hybrid

Portability

Applications using Service Interconnect are highly portable from a networking perspective, offering great freedom of operational efficiency and migration.

Red Hat

Some elements in software are still not portable

Portability allows to decouple elements in software

Containers turned computing
PORTABLE

Containers enable to move applications from different environments effortlessly

Object Storage turned storage
PORTABLE

Object Storage enable to move data stored from one location to another easily

Networking is still
NOT PORTABLE

Networking is still the only element in software that is still immutable. It requires a new configuration for a new environment

Service Interconnect changes that

Interconnections follows your application to different environments and platforms

Red Hat
Service Interconnect

Containers turned computing
PORTABLE

Containers enable to move applications from different environments effortlessly

Object Storage turned storage
PORTABLE

Object Storage enable to move data stored from one location to another easily

Networking is now
PORTABLE

Because it operates on Layer 7, it abstracts the underlying networking and helps to re-establish interconnections in different environments

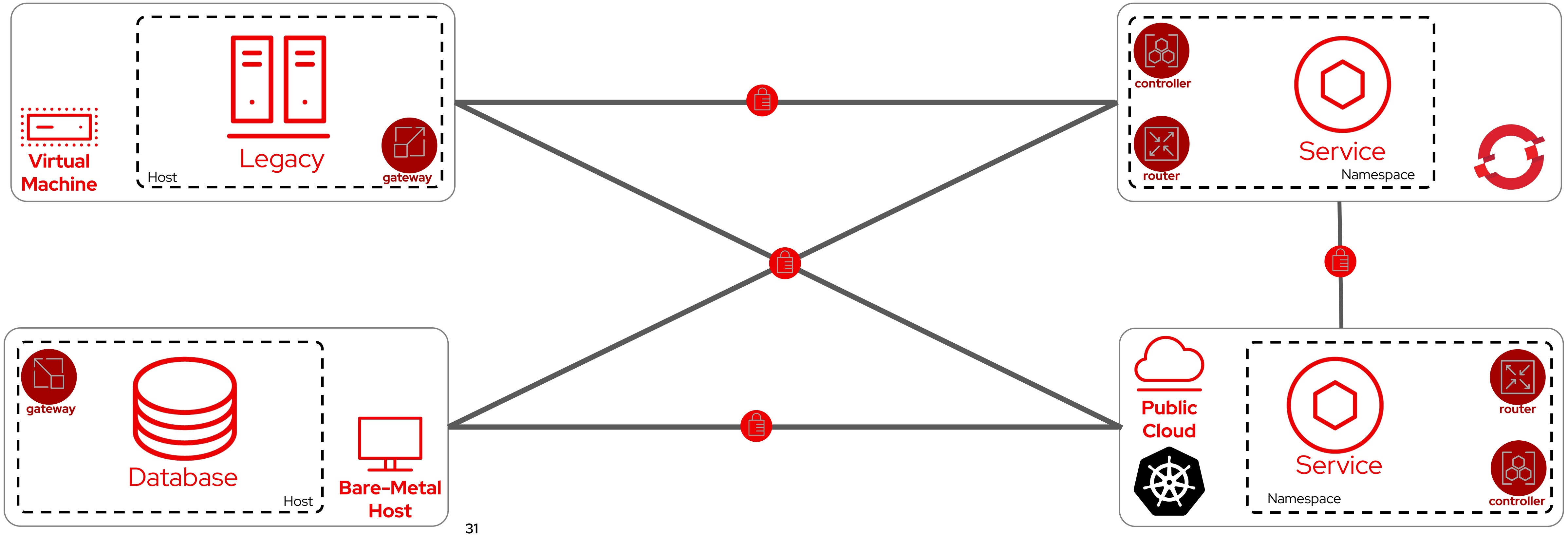
Portable Simple Protected Hybrid

Hybrid

Service Interconnect makes hybrid cloud strategies easier to implement by allowing customers' development teams to easily, rapidly and safely interconnect any Kubernetes cluster, any public cloud, any virtual machine or any bare-metal host.

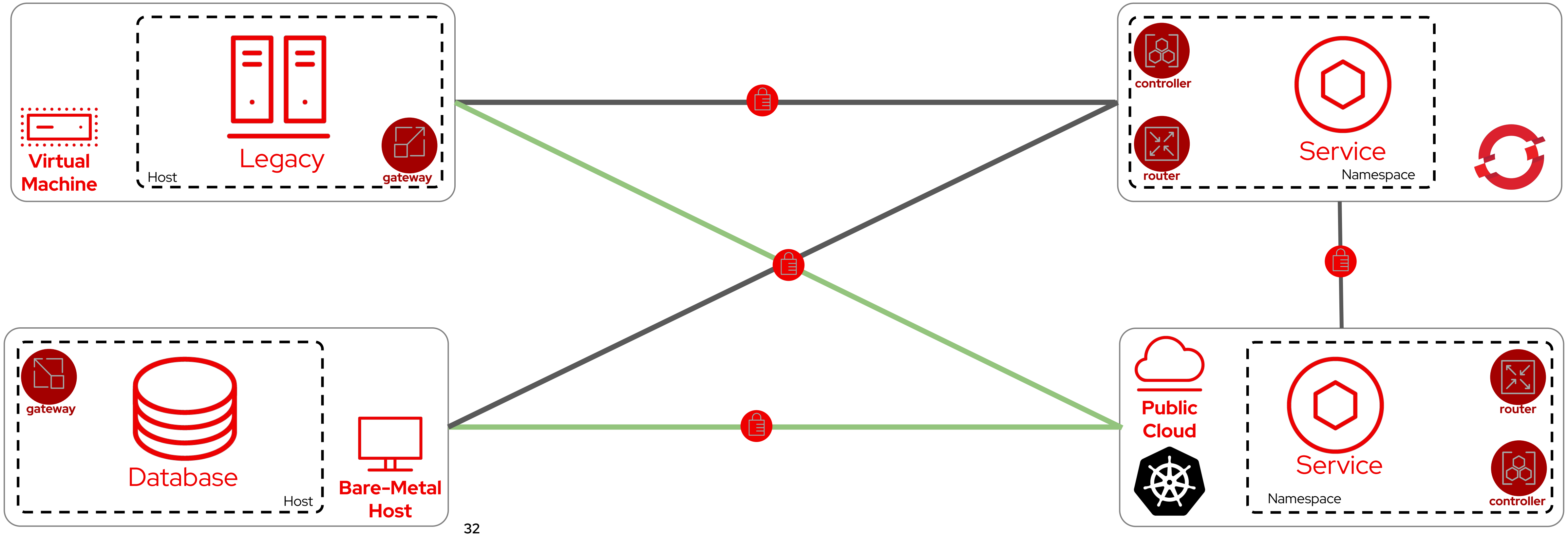
Hybrid interconnections

Linking different applications and services across different environments



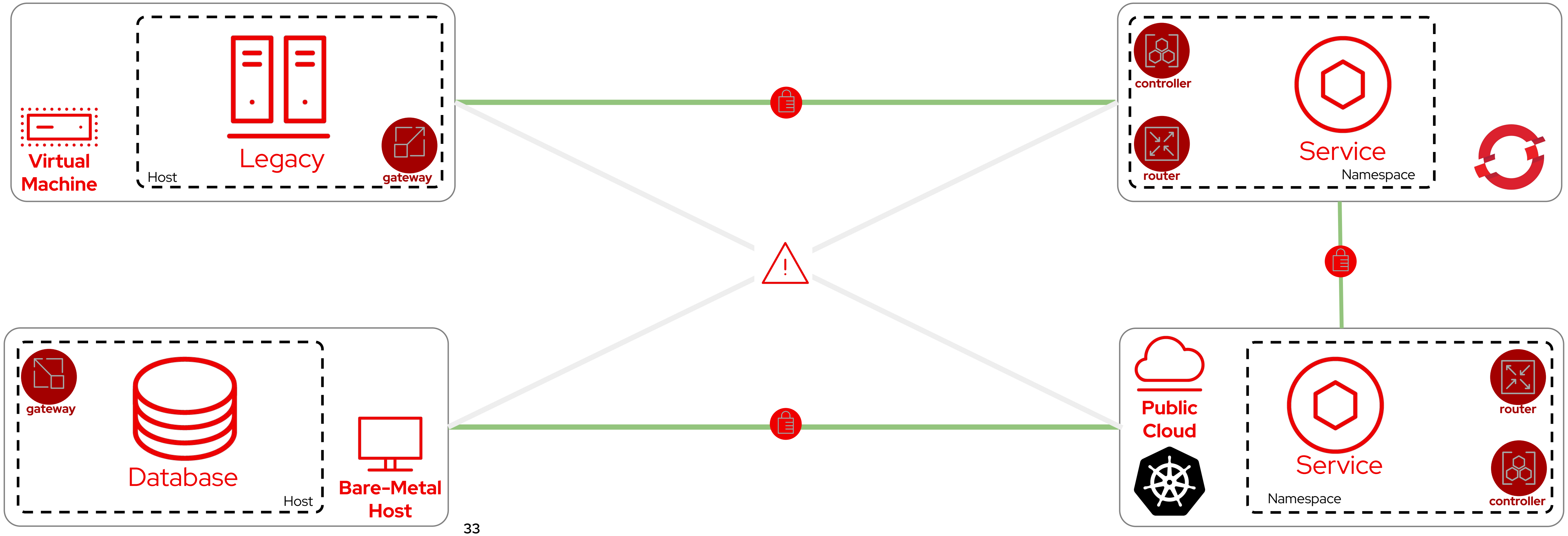
Indirect connections amongst services

Services that are a part of the network and not directly connected can access each other if needed



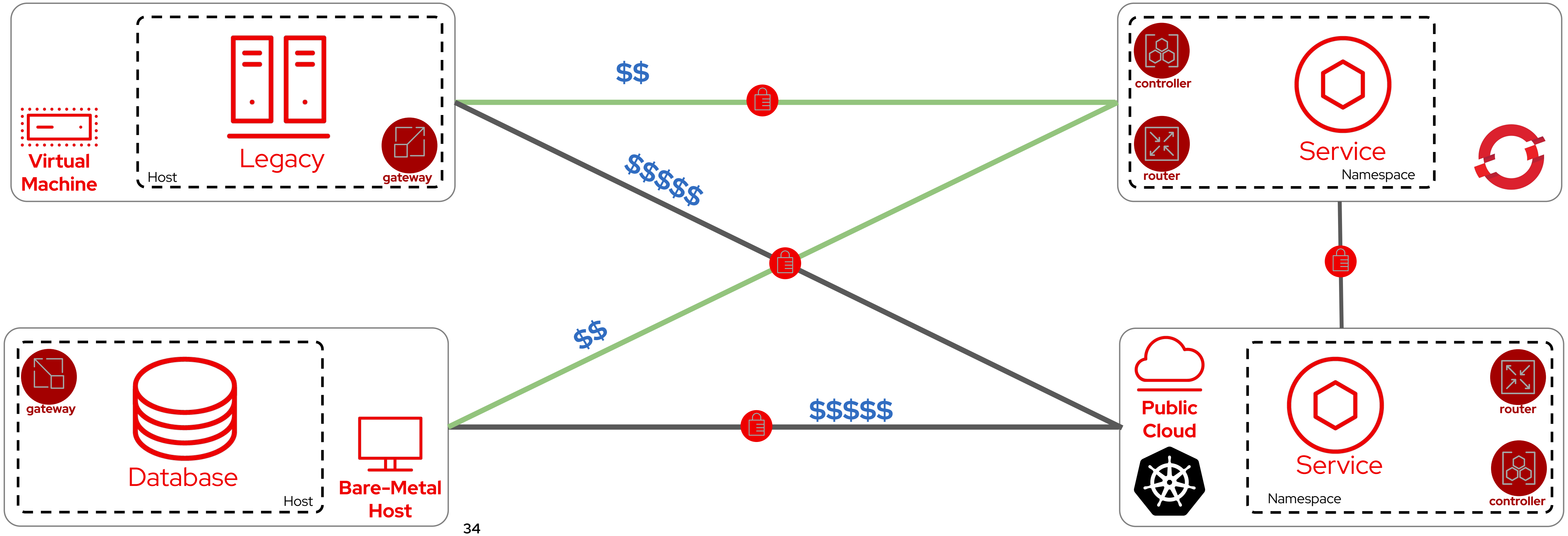
High Availability

In case of a Router outage, alternate path is found



Cost- and locality-aware traffic forwarding

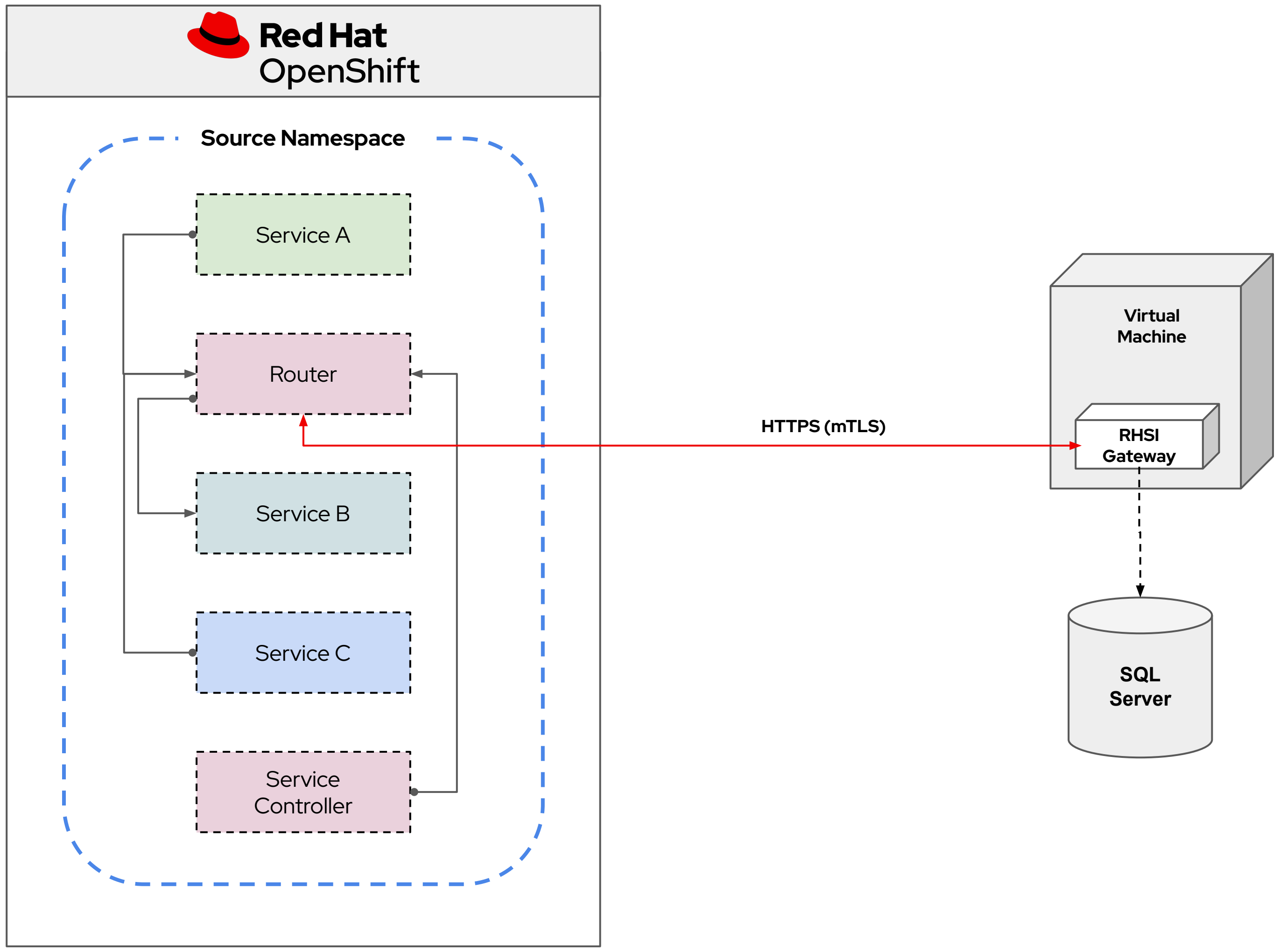
Interconnections find the optimal path to reach a destination



Key Use Cases

Service Interconnect makes hybrid cloud strategies easier to implement by allowing customers' development teams to easily, rapidly and safely interconnect services across any Kubernetes cluster, any public cloud, any virtual machine or any bare-metal host.

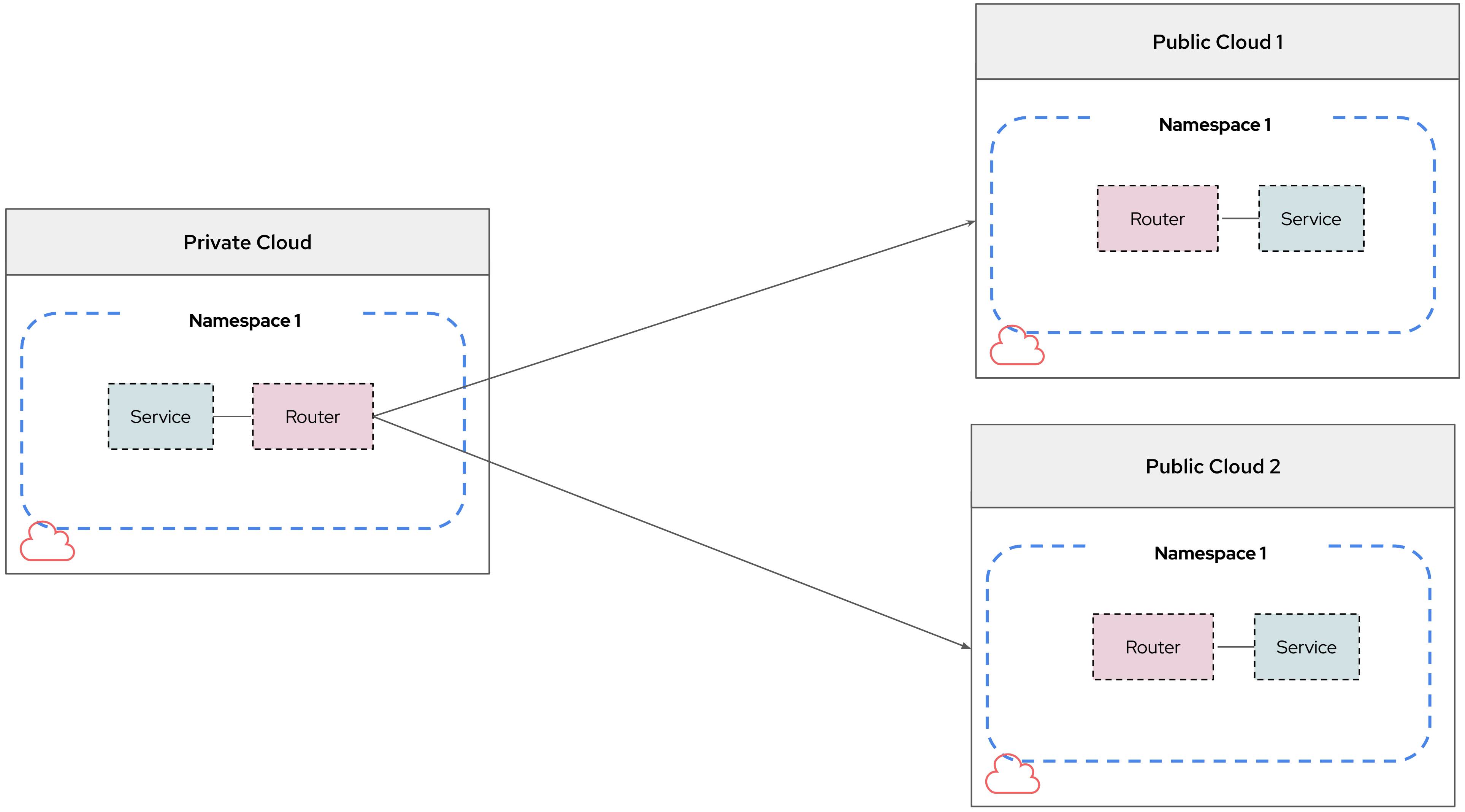
Use Case: Integrate OpenShift with Traditional Applications & Infrastructure



Service Interconnect Gateway enables “locationless” integration with traditional applications

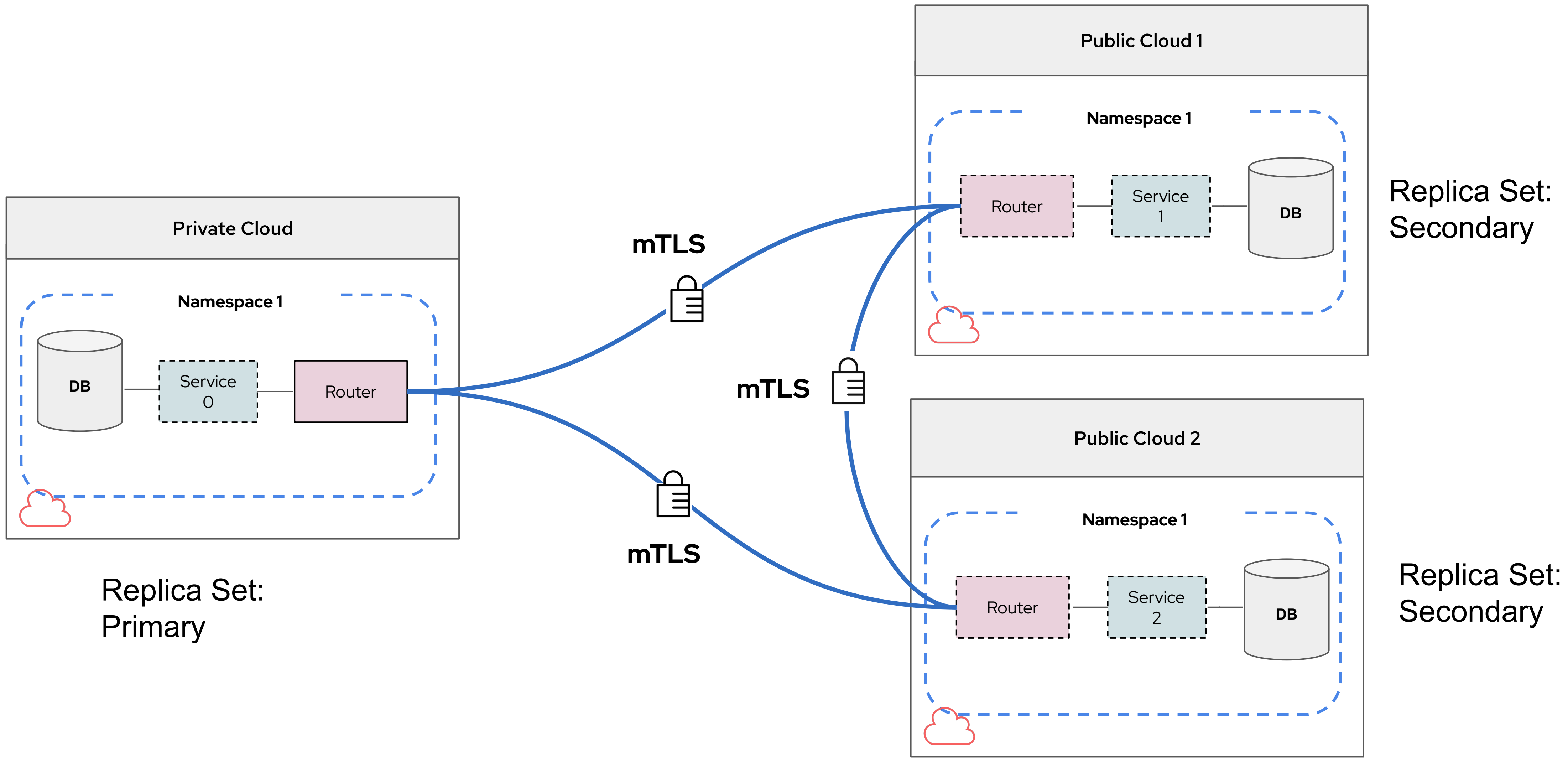
Note: This is a logical network flow. All RHSI network flows ride on top of already endorsed network flows and ingress/egress the cluster via routes on the RHSI Router.

Use Case: High Availability of Services Across Multiple Clusters



Red Hat

Use Case: Distributed Data Replication



Red Hat

Red Hat Summit

Connect

Thank you

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat