Exam Objective 5

# Use Consul Service Mesh

# Use Consul Service Mesh

**Objective 6a:** Understand Consul Connect service mesh high level architecture

**Objective 6b:** Describe configuration for registering a proxy

**Objective 6c:** Describe intentions for Consul Connect service mesh

**Objective 6d:** Check intentions in both the Consul CLI and UI

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Difficulty Level

# What is Consul Service Mesh?

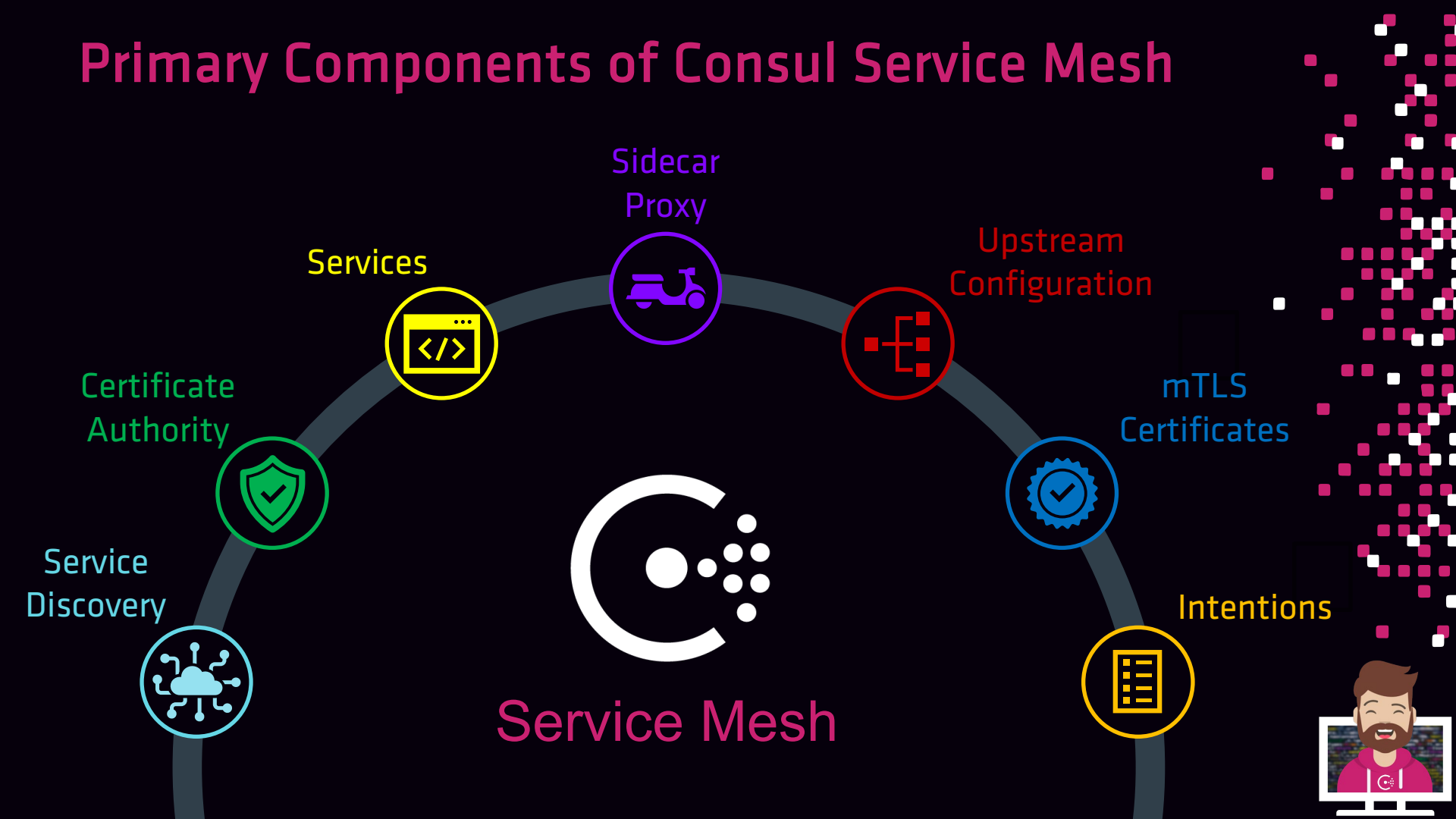
## Provides service-to-service connection authorization and encryption

- Uses mTLS for authorization and encryption
- Applications can be written for native support using SDK or...
- ...use a sidecar proxy architecture (most common)

## Applications may or may not be aware that Consul service mesh is present

- Traffic between apps flow through the sidecar proxy
- The proxy enables authenticated and encrypted communication (mTLS) between services
- Could provide encryption between services that wouldn't otherwise be encrypted

# Primary Components of Consul Service Mesh



# Intro to Consul Service Mesh

## Certificate Authority issues mTLS certificates

- mTLS is the core of Consul Service Mesh
- Consul has a built-in certificate authority that can be used
- Has support to "outsource" CA functionality to HashiCorp Vault or other solutions

## mTLS Certificates

- Provides authentication by validating the certificate against the CA
- Enables encryption between the services

# Intro to Consul Service Mesh

## Intentions define access control for Services

- Determines what services can establish connections to other services
- Top-down ruleset using Allow or Deny intentions
- Can be configured via API, CLI, or UI

## Sidecar Proxy

- Service proxy running alongside the core application
- Primary sidecar proxy used today is Envoy (envoyproxy.io)
- Consul also has a built-in sidecar proxy (but not as feature-rich)
- Other proxies can be used as well

# Intro to Consul Service Mesh

## Service Mesh is platform agnostic

- Manage services running on physical networks, public cloud, software-defined networks, or even cross-cloud

## Service Mesh enables Layer 7 observability

- Proxies see all traffic between services and can collect metrics
- Metrics can be sent to an external monitoring tool, like Prometheus

## Connect must be enabled in the agent configuration (connect stanza)

- Connect is enabled by default if using `-dev` mode

# Intro to Consul Service Mesh

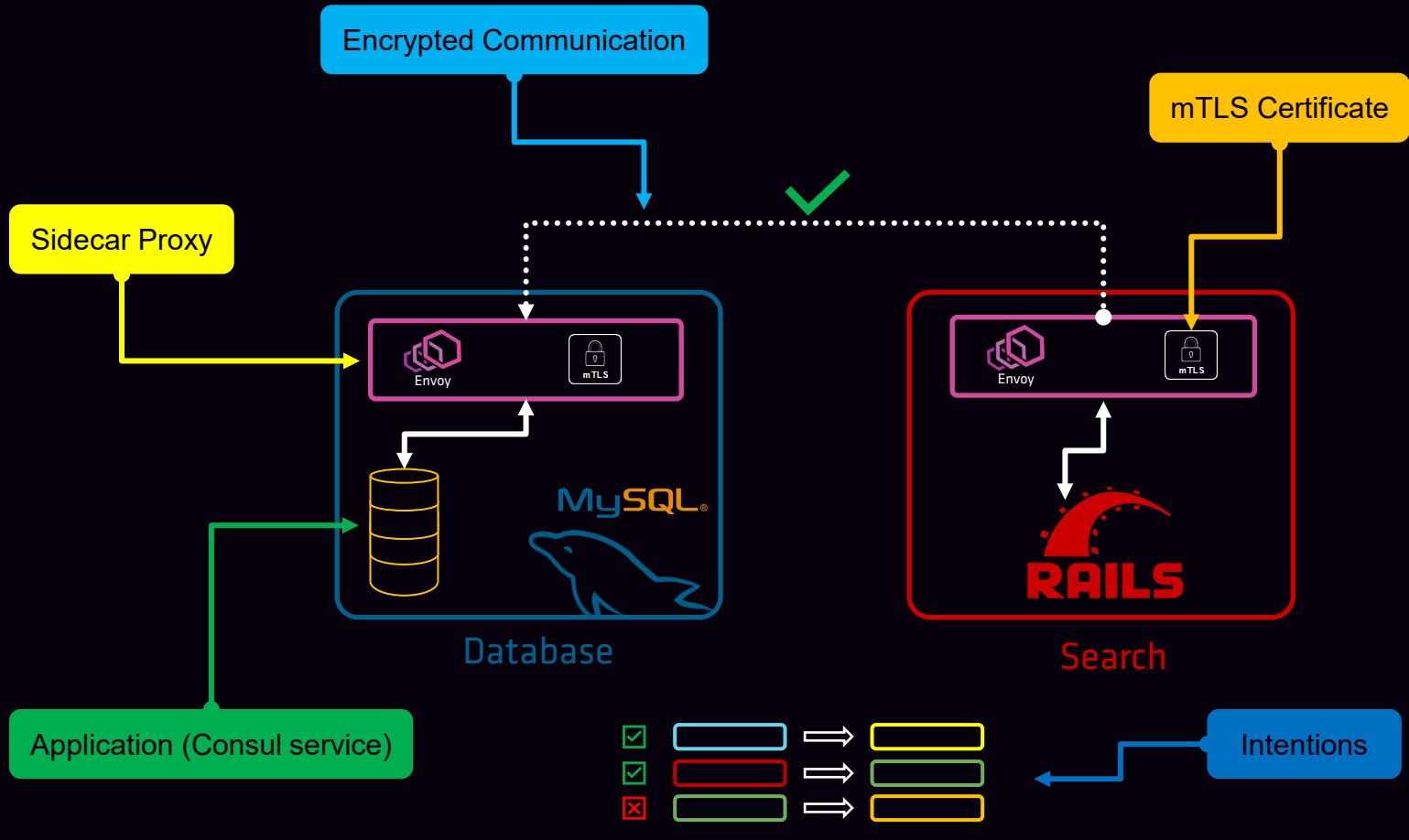
## Upstream vs. Downstream

- Upstream – the target service that another service depends on
- Downstream – the service that is dependent on the target service

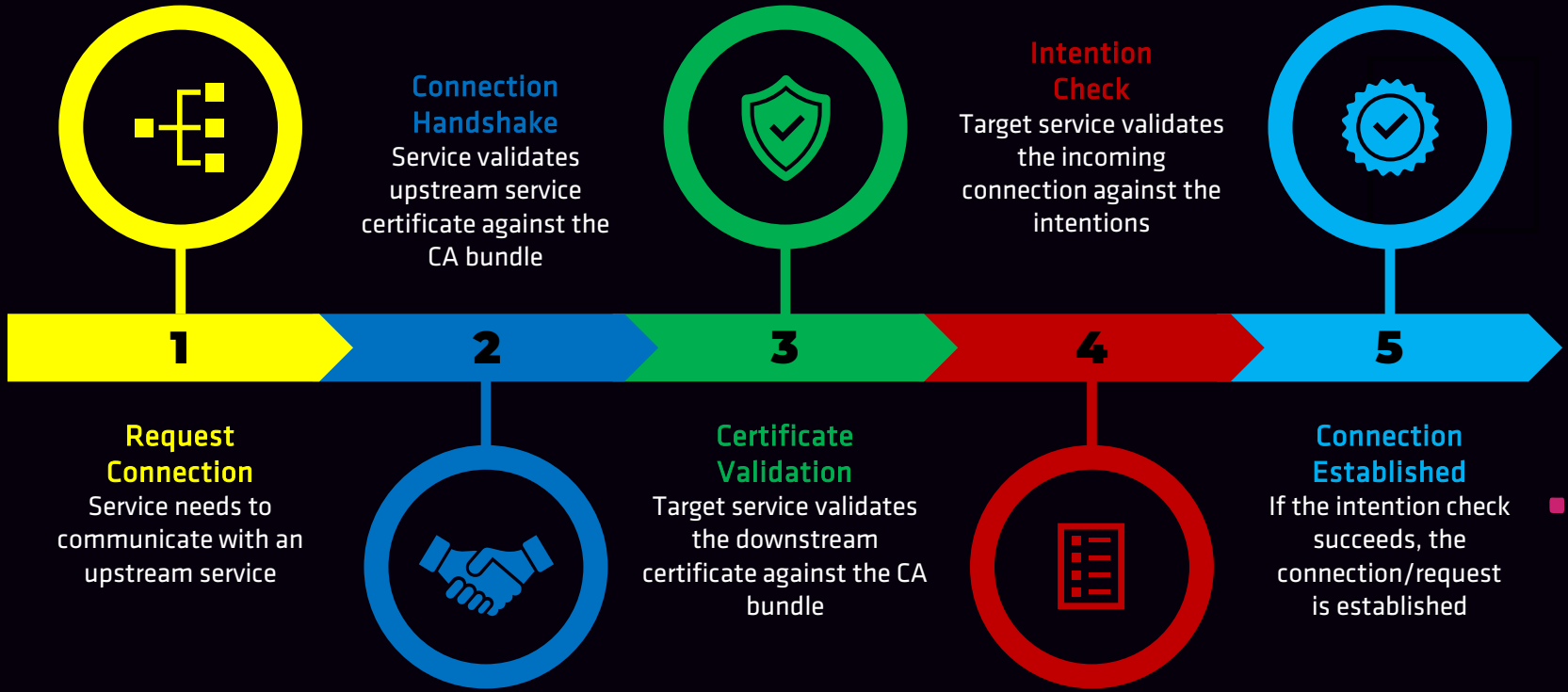

The DATABASE SERVICE is *upstream* from the WEB SERVICE since the WEB SERVICE depends on it

WEB SERVICE depends on the DATABASE SERVICE and therefore is *downstream* from the DATABASE SERVICE

# Consul Service Mesh - High-Level Architecture



# Consul Service Mesh - Workflow




# Consul Service Mesh – Other Components

## L7 Traffic Management

- "Carve up" traffic across the pool of services vs. just using round robin
- Sometimes called traffic splitting

## Service Mesh Gateways

- Enables routing between federated service mesh datacenters where private connectivity may not be established or feasible
- Ingress gateways and terminating gateways(k8s)

- Observability
  - Consul 1.9.0 includes new topology visualizations to show a service's connectivity

# Registering a Service Proxy

- Just like a service, a sidecar proxy must be registered with Consul
  - Registration does not start the sidecar proxy

- Registration is most commonly done using a configuration file
  - Usually, the same config file as the service using the connect parameter and related options

```
"service": {
  "name": "front-end-eCommerce",
  "port": 8080,
  "connect": {
    "sidecar_service": {}
  }
}
```

# Registering a Service Proxy

```
{
  "service": {
    "id": "web-01",
    "name": "front-end-eCommerce",
    "tags": ["v7.05", "production"],
    "address": "",
    "port": 8080,
    "connect": {
      "sidecar_service": {
        "proxy": {
          "upstreams": [
            {
              "destination_name": "db01",
            }
          ]
        }
      }
    }
  }
}
```

The Service to Register

Connect Stanza

Upstream Configuration

Upstream Service(s)

Unrelated - This is why I hate JSON ☺

# Consul Service Mesh - Intentions

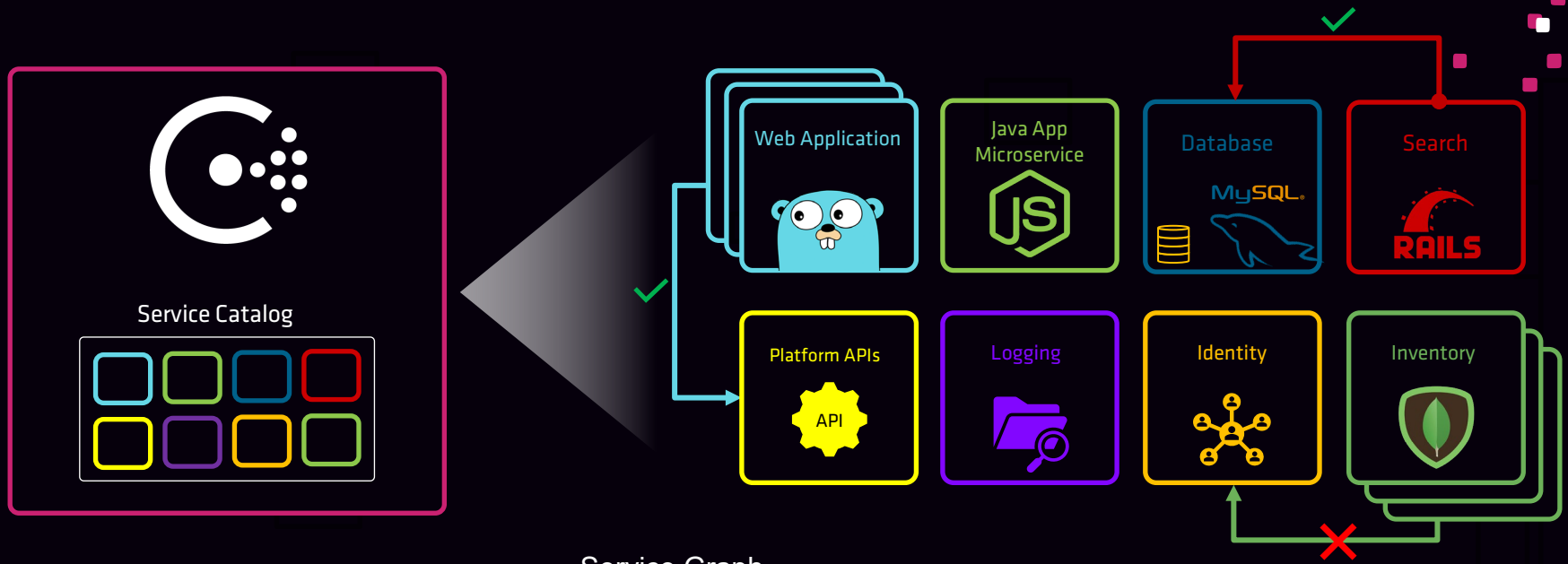
## Intentions define access control for Services

- Uses a service graph to determine what services are allowed to establish connections to other services
- Enforced at the destination/target service on inbound connections, proxy requests, or within a natively integrated app (SDK)

## Enforcing Intentions

- Default behavior is controlled by the default ACL policy
  - 'Allow all' default means all connections are allowed by default
  - 'Deny all' default means all connections are denied by default
- Only one intention controls authorization at any given time

# Consul Service Mesh - Intentions




Service Graph

| | | |
|---|---|---|
| ✓ | Web Application | Platform APIs |
| ✓ | Search | Database |
| ✗ | Inventory | Identity |

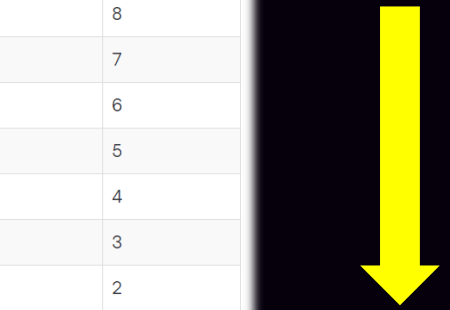
# Consul Service Mesh - Intentions

## Precedence and Match Order

- Top-down ruleset using Allow or Deny intentions
- Precedence cannot be overridden

| Source Namespace | Source Name | Destination Namespace | Destination Name | Precedence |
|---|---|---|---|---|
| Exact | Exact | Exact | Exact | 9 |
| Exact | * | Exact | Exact | 8 |
| * | * | Exact | Exact | 7 |
| Exact | Exact | Exact | * | 6 |
| Exact | * | Exact | * | 5 |
| * | * | Exact | * | 4 |
| Exact | Exact | * | * | 3 |
| Exact | * | * | * | 2 |
| * | * | * | * | 1 |

Evaluated First



Evaluated Last

# Consul Service Mesh - Intentions

## Controlling Authorization

- Authorization is control using either L4 or L7 depending on the protocol being used
  - L4 – identity based (TLS) – all or nothing access control based on new connections
  - L7 – application-aware – can be based on L7 request attributes based on new requests

# Managing Consul Service Mesh Intentions

## Configuring and Managing Intentions

- Can be configured via API, CLI, or UI
- Intentions can be managed on any interface (i.e. intentions created using the API can be seen/managed in the UI
- Changing an intention does not affect existing connections, only new connections

- Intentions are managed primarily using the service-intentions config entries or the UI
  - Simpler tasks can be done using the older API or CLI

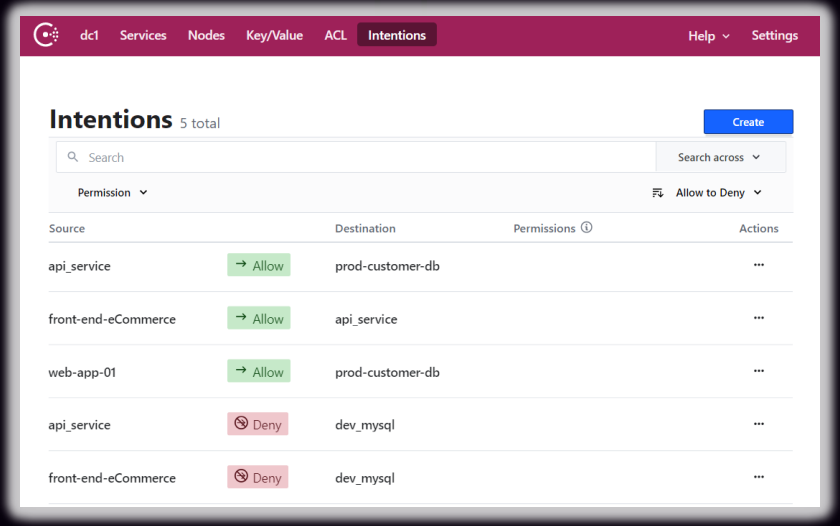
```
Kind = "service-intentions"
Name = "db-01"
Sources = [
  {
    Name   = "web-01"
    Action = "deny"
  }
]
```

Connections from web-01 to db-01 will be rejected

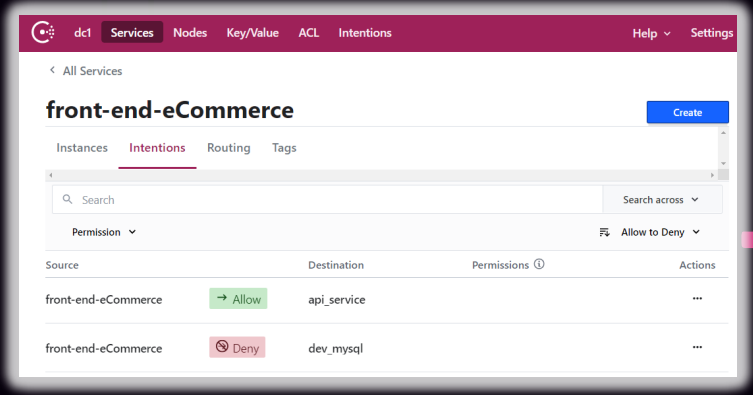
# Managing Consul Service Mesh Intentions

Configuring Intentions with the UI

Create, view, and manage intentions in the Intentions 'tab" within the UI




Intentions can be viewed within the affected service as well




# Managing Consul Service Mesh Intentions

Configuring Intentions with the API

**FYI:** /connect/intentions was deprecated in Consul 1.9.0

## Create Intensions with the Consul API

- Method: PUT
- Endpoint: /v1/connect/intentions/exact
- Response: Returns 'true' if intention was created successfully

Terminal

```
$ curl \
    --request PUT \
    --data @payload.json \
    https://consul.example.com:8500/v1/connect/intentions/exact?source=web-01&destination=db-01
```

Terminal

```
$ cat payload.json
{
  "SourceType": "consul",
  "Action": "allow"
}
```

Action: Allow or Deny

https://www.consul.io/api-docs/connect/intentions#upsert-intention-by-name

# Managing Consul Service Mesh Intentions

Configuring Intentions with the CLI

Use `consul intention` command:

- `get` – show information about the intention
- `check` – validate whether a connection is allowed
- `create` – create a new intention (default is allow)
- `delete` – delete an existing intention
- `match` – show intentions matching a source or destination
- `list` – lists ALL intentions

Terminal

```
$ export CONSUL_HTTP_TOKEN=aba7cbe5-879b-999a-07cc-2efd9ac0ffe

$ consul intention create web-01 db-01
Created: web-01 => db-01 (allow)

$ consul intention create -deny web-01 db-01
Created: web-01 => db-01 (deny)
```

Needed if ACLs are enabled

Allow

Deny (must include -deny)

# Use Consul Service Mesh

**Objective 6a:** Understand Consul Connect service mesh high level architecture

**Objective 6b:** Describe configuration for registering a proxy

**Objective 6c:** Describe intentions for Consul Connect service mesh

**Objective 6d:** Check intentions in both the Consul CLI and UI

| 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|

Difficulty Level

# END OF SECTION