

Agenda

Cross-Cutting Concern & Observability Patterns

- Cross-Cutting Concern Patterns
 - Externalized Configuration
 - Service Discovery
 - Circuit Breaker
 - Retry Pattern
- Observability Patterns
 - Microservices Logging
 - Microservices Tracing
 - Microservices Monitoring

Cross Cutting Concern Patterns

Problem: How to handle cross cutting concerns?

- **Externalized configuration** - externalize all configuration such as database location and credentials
- **Service discovery** - How does the client of an RPI-based service discover the network location of a service instance?
- **Circuit Breaker** - invoke a remote service via a proxy that fails immediately when the failure rate of the remote call exceeds a threshold

Externalized Configuration: Azure Key Vault



- Azure Key Vault is a safe place to store passwords, connection strings, access codes, and certificate keys
- Fully managed by Azure, so you don't have to worry about the underlying infrastructure.
- You can use Azure Key Vault from your .NET application using the *Microsoft.Azure.KeyVault* NuGet package

Key Vault Permission

Key vault

Search (Ctrl+ /)

Keys

Secrets

Certificates

Access policies

Networking

Security

Properties

Locks

Monitoring

Insights

Alerts

Metrics

Diagnostic settings

Logs

Workbooks

Access policies

Enable Access to:

☒ Azure Virtual Machines for deployment ⓘ

☒ Azure Resource Manager for template deployment ⓘ

☒ Azure Disk Encryption for volume encryption ⓘ

Permission model

☒ Vault access policy

☐ Azure role-based access control

+ Add Access Policy

Current Access Policies

Name	Email	Key Permissions
APPLICATION		
WebAppDNTAD		9 selected
USER		
Shailendra Chauhan	pro.shailendra_gmail....	9 selected

> dntkeyvault101 > dntkeyvault101 >

Add access policy ...

Add access policy

Configure from template (optional)

Key permissions

Secret permissions

Certificate permissions

Select principal *

Authorized application ⓘ

Add

Key, Secret, & Certificate Management

9 selected

7 selected

15 selected

DNTWebApp101

Object ID: dfa53f7f-2730-4602-a428-cba6b3832d47

None selected

Requesting Token from Azure AD

POST <https://login.microsoftonline.com/d95c2423-fd55-4495-8a6f-0a9283991dca/oauth2/v2.0/token>

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings

☐ none ☐ form-data ☒ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE
<input checked="" type="checkbox"/>	grant_type	client_credentials
<input checked="" type="checkbox"/>	client_id	45514a65-339c-4554-bab1-e798aa0e47d2
<input checked="" type="checkbox"/>	client_secret	sxo4--60-WTQe52L3-540K4P7akmRKS61g
<input checked="" type="checkbox"/>	scope	https://vault.azure.net/.default

Body Cookies (3) Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
5 {  "access_token":  
    "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6Imt  
nMkxZczJUMENUaklmaWRydDZKSXluZW4zOCJ9.  
eyJhdWQiOiJodHRwczovL3ZhdWx0LmF6dXJlLn51dCIsIm1z
```

```
grant_type:client_credentials  
client_id:45514a65-339c-4554-bab1-e798aa0e47d2  
client_secret:sxo4--60-WTQe52L3-540K4P7akmRKS61g  
scope:https://vault.azure.net/.default
```

Accessing Key Vault Secrets and Keys

GET <https://dntkeyvauly.vault.azure.net/secrets/dbconnection?api-version=2016-10-01>

Params ☒ Authorization **Headers (8)** Body Pre-request Script Tests Settings

Headers ☒ 7 hidden

	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImtnMkxZczJUMENUaklr...
	Key	Value

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "value": "dnt@#12345678",
3   "id": "https://dntkeyvauly.vault.azure.net/s
4   "attributes": {
5     "enabled": true,
6     "created": 1604336842,
```

GET <https://dntkeyvauly.vault.azure.net/keys/mykey?api-version=2016-10-01>

Params ☒ Authorization **Headers (8)** Body Pre-request Script Tests Settings

Headers ☒ 7 hidden

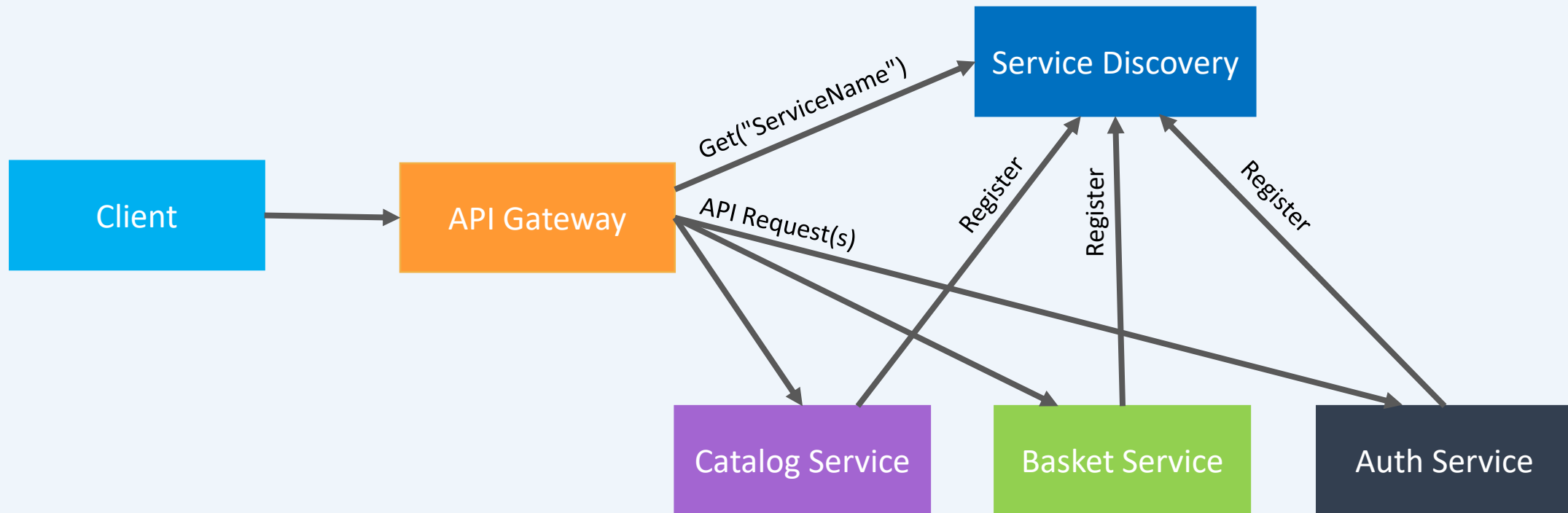
	KEY	VALUE
<input checked="" type="checkbox"/>	Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6ImtnMkxZczJUMENUaklr...
	Key	Value

Body Cookies Headers (13) Test Results

Pretty Raw Preview Visualize JSON

```
2   "key": {
3     "kid": "https://dntkeyvauly.vault.azure.
4     "kty": "RSA",
5     "key_ops": [
6       "sign",
7       "verify",
8       "wrapKey",
9       "unwrapKey",
```

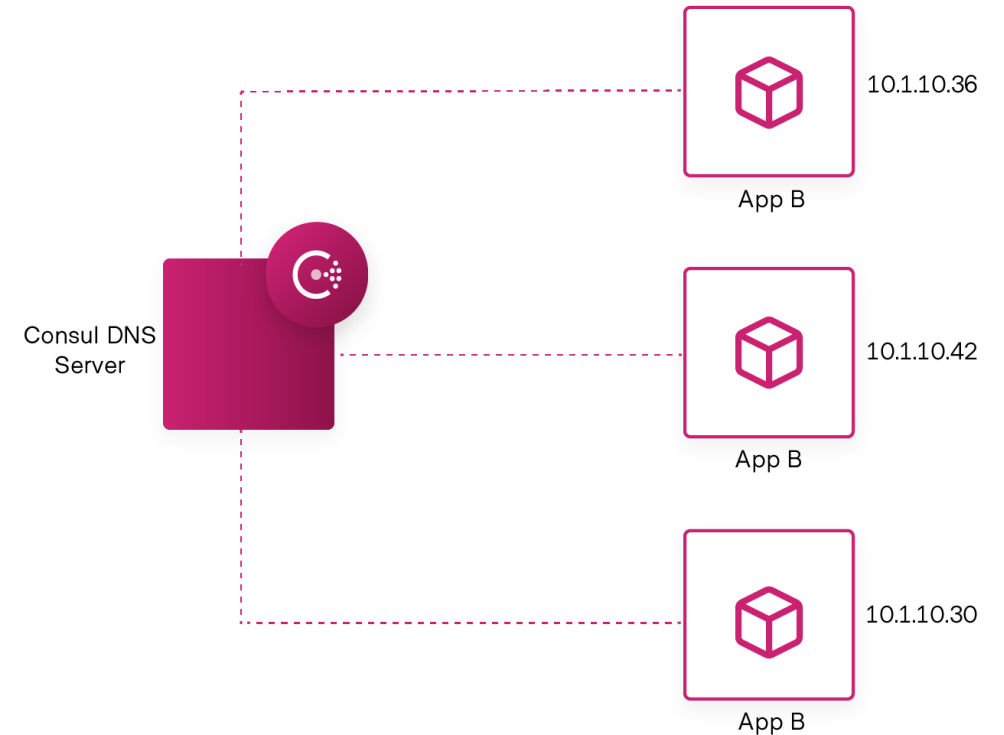
Service Discovery



Service Discovery Using Consul

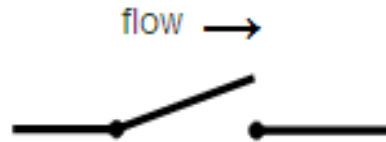
- Discover, Register, and Resolve services for application workloads across any cloud.
- Automatically add and remove services based on health checking.

```
> docker pull consul  
> docker run -p 8500:8500 consul  
  
//inside docker instance  
> consul services deregister -id=web
```



Circuit Breaker Pattern

- A circuit breaker is an electronic component like fuse that opens a circuit so that no current can flow through it. As a result, it prevents the damage caused by excess current from an overload or short circuit.



- To prevent an application from trying to invoke a remote service or access a shared resource if this operation is highly likely to fail.
- The circuit-breaker tracks the number of times an API call has failed. Once it crosses a threshold number of failures in a row, it doesn't even try to call the API for subsequent requests.



- Polly is a .NET library that allows developers to express policies such as Retry, Circuit Breaker, Timeout, Bulkhead Isolation, and Fallback in a fluent and thread-safe manner.
- If the failure scenario is occurring due to load issues (e.g. a database is trying to handle too many operations), a circuit-breaker policy will prevent the situation from getting worse whereas a retry policy would almost certainly have cause a bigger problem

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddHttpClient("basketapi")
        .SetHandlerLifetime(TimeSpan.FromMinutes(5))
        .AddPolicyHandler(GetCircuitBreakerPolicy());
}
```

```
static IAsyncPolicy<HttpResponseMessage> GetCircuitBreakerPolicy()
{
    return HttpPolicyExtensions
        .HandleTransientHttpError()
        .CircuitBreakerAsync(2, TimeSpan.FromSeconds(10));
}
```

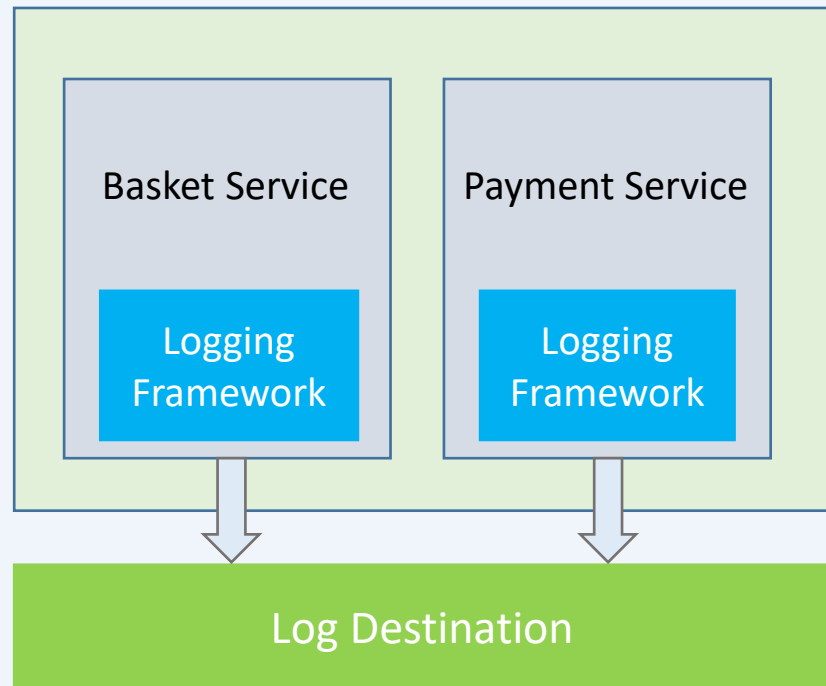
Observability Patterns

Problem: How to understand the behavior of an application and troubleshoot problems?

- **Log aggregation** - aggregate the application logs
- **Application metrics** - instrument a service's code to gather statistics about operations
- **Exception tracking** - report all exceptions to a centralized exception tracking service that aggregates and tracks exceptions and notifies developers.
- **Health check API** - service API (e.g. HTTP endpoint) that returns the health of the service and can be pinged, for example, by a monitoring service
- **Audit logging** - record user activity in a database

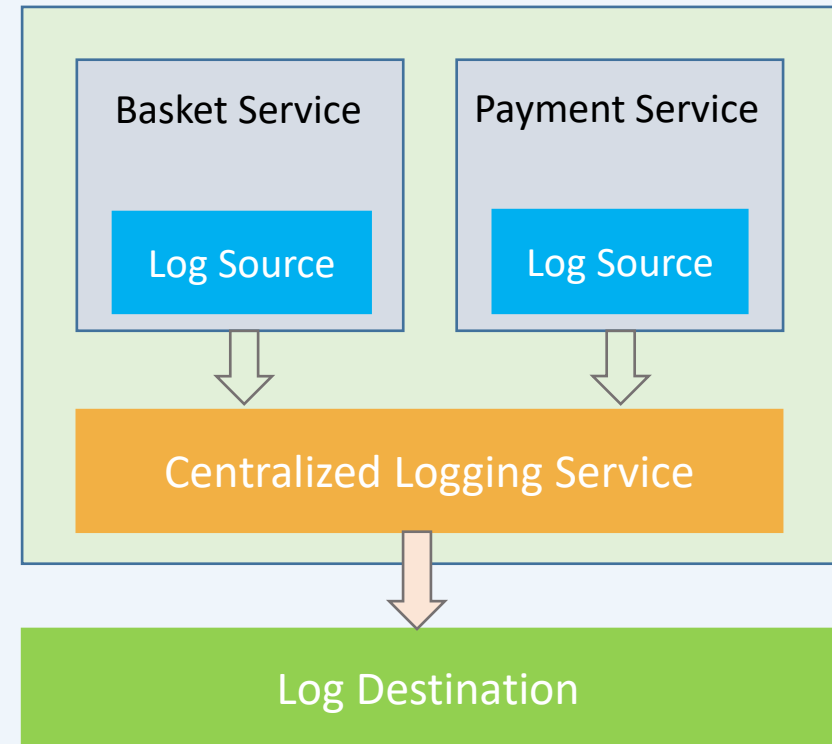
Microservices Logging

Microservice Architecture



Logging in Each Service

Microservice Architecture



Centralized Logging Service

Logging with ELK Stack



- **Elastic Search** is a NoSQL database that is capable of taking all logs, index them and making them searchable.
- **LogStash** is a log collector and a known format for storing logs in a structured JSON format.
- **Kibana** is a UI that can connect to Elastic Search and provide a usable interface for searching logs, making queries, and making dashboards and charts out of those queries.

ELK Stack Docker Setup

> docker-compose up -f docker-compose.yml

---- Or ----

> docker-compose up

- Accessing Dashboard
 - Kibana: *localhost:9200*
 - Elastic Search UI: *localhost:5601*

Open Telemetry



- An observability framework for cloud-native software.
- Used to the instrument, generate, collect, and export telemetry data (metrics, logs, and traces) for analysis in order to understand your software's performance and behavior.
- Supports all officially supported versions of .NET Core and .NET Framework except for .NET Framework 3.5 SP1.
- Zipkin is a distributed tracing system. It helps gather timing data needed to troubleshoot latency problems in service architectures.

```
docker run -d -p 9411:9411 openzipkin/zipkin
```