# Purpose

In this lab we will enable our applications to register themselves with a [Eureka](https://github.com/Netflix/eureka/wiki) service registry on PCF.

At the end of the lab, each application will be able to discover microservices by name, reducing the configuration needed to deploy our system.

# Discussion points

* Local development with SCS (platform-services)
* Service discovery basics ([diagram](https://docs.pivotal.io/spring-cloud-services/common/service-registry/))
* What is client side load balancing?
* Steeltoe currently supports [random load balancing](https://steeltoe.io/docs/steeltoe-discovery/#1-2-6-discovering-services)
* Steeltoe does not [currently support automatic retries](https://github.com/SteeltoeOSS/Samples/issues/29).

# Start Eureka server

For the following labs we will run a local version of our Spring Cloud Services. These Java applications have been provided for you in the PlatformServices directory.

1. Run the Eureka server using the Gradle build tool:
2. **cd** ~/workspace/pal-tracker-distributed/PlatformServices
3. ./gradlew EurekaServer:bootRun

You may see some errors in the console output while the Eureka server boots. This is Eureka trying to replicate service registry information and does not affect the operation of the server.

1. Navigate to [localhost:8761](http://localhost:8761/) to see your Eureka dashboard.

# Add Eureka to applications

Now that we have Eureka running locally we are ready to configure our applications to use it.

1. Add the Cloud Foundry Configuration and Service Discovery packages to all projects in the **Applications** directory. For example for the allocations microservice:
2. dotnet add ~/workspace/pal-tracker-distributed/Applications/AllocationsServer \
3. package Pivotal.Discovery.ClientCore --version 2.1.1
4. dotnet add ~/workspace/pal-tracker-distributed/Applications/AllocationsServer \
5. package Steeltoe.Extensions.Configuration.CloudFoundryCore --version 2.1.1
6. In our architecture, registration service is the only one that needs to be discoverable. Update appsettings.json in **RegistrationServer** as follows:
7. {
8. // ...
9. + "eureka": {
10. + "client": {
11. + "serviceUrl": "http://localhost:8761/eureka/",
12. + "shouldRegisterWithEureka": true,
13. + "shouldFetchRegistry": false
14. + },
15. + "instance": {
16. + "appName": "registration-service",
17. + "hostname": "localhost",
18. + "port": 8883
19. + }
20. + }
21. }

This configuration:

* + Instructs the application to register itself with Eureka under the name "registration-service"
  + Provides location of the Eureka server with which to register
  + Instructs the application to **not** fetch the registry (registration service does not need to discover other services)
  + Provides hostname and port of the application so that Eureka can properly route requests

1. Update appsettings.json in **AllocationServer**, **BacklogServer**, and **TimesheetsServer** as follows:
2. {
3. // ...
4. - "REGISTRATION\_SERVER\_ENDPOINT" : "http://localhost:8883"
5. + "REGISTRATION\_SERVER\_ENDPOINT": "http://registration-service/",
6. + "eureka": {
7. + "client": {
8. + "serviceUrl": "http://localhost:8761/eureka/",
9. + "shouldRegisterWithEureka": false,
10. + "shouldFetchRegistry": true
11. + }
12. + }
13. }

This configuration:

* + Provides the location of the Eureka server
  + Instructs the application **not** to register itself with Eureka (only registration service needs to be discoverable)
  + Instructs the application to fetch the registry so it can discover the registration service.
  + Configures REGISTRATION\_SERVER\_ENDPOINT to refer to the registration service by it's name rather than the full url

1. Update Program.cs of **AllocationServer**, **BacklogServer**, **TimesheetsServer**, and **RegistrationServer** to configure dynamic port binding via UseCloudFoundryHosting()
2. namespace ...
3. public class Program
4. {
5. // ...
6. public static IWebHostBuilder WebHostBuilder(string[] args) =>
7. WebHost.CreateDefaultBuilder(args)
8. + .UseCloudFoundryHosting()
9. .AddCloudFoundry()
10. .UseStartup<Startup>();
11. }
12. Update Startup.cs of **AllocationServer**, **BacklogServer**, **TimesheetsServer**, and **RegistrationServer** to add the discovery client to the dependency injection container and to use the discovery client service.
13. + using Pivotal.Discovery.Client;
14. namespace ...
15. {
16. public class Startup
17. {
18. // ...
19. public void ConfigureServices(IServiceCollection services)
20. {
21. // ...
22. + services.AddDiscoveryClient(Configuration);
23. }
24. public void Configure(IApplicationBuilder app, IHostingEnvironment env, ILoggerFactory loggerFactory)
25. {
26. // ...
27. app.UseMvc();
28. + app.UseDiscoveryClient();
29. }
30. }
31. }

Note: Be sure to use the extension functions in the Pivotal.Discovery.Client namespace.

1. Again in Startup.cs of **AllocationServer**, **BacklogServer**, and **TimesheetsServer**, update how the HttpClient is constructed:
2. + using Steeltoe.Common.Discovery;
3. namespace ...
4. {
5. public void ConfigureServices(IServiceCollection services)
6. {
7. services.AddDiscoveryClient(Configuration);
8. // ...
9. services.AddSingleton<IProjectClient>(sp =>
10. {
11. - var httpClient = new HttpClient
12. + var handler = new DiscoveryHttpClientHandler(sp.GetService<IDiscoveryClient>());
13. + var httpClient = new HttpClient(handler, false)
14. {
15. BaseAddress = new Uri(Configuration.GetValue<string>("REGISTRATION\_SERVER\_ENDPOINT"))
16. };
17. return new ProjectClient(httpClient);
18. });
19. }
20. }

Note that while we are changing how HttpClient is constructed, the **ProjectClient** implementation remains unchanged. This means that Components do not depend on the discovery client, only Applications do.

1. With Eureka still running, start your microservices. You should see the registration service appear on your [Eureka dashboard](http://localhost:8761/).

Make sure the applications are working by exercising endpoints.

If you encounter the following error on a Linux or Mac machine, make sure that your machine's hostname is mapped to 127.0.0.0 in **/etc/hosts**.

Unhandled Exception: System.AggregateException: One or more errors

occurred. (Device not configured) --->

System.Net.Internals.SocketExceptionFactory+ExtendedSocketException:

Device not configured

1. Finally, update **FlowTest.cs** to disable reliance on Eureka in the integration test.
2. public FlowTest()
3. {
4. \_registrationServer = TestAppServerBuilder()
5. .AppName("RegistrationServer")
6. .Port(8883)
7. .Database("tracker\_registration\_dotnet\_test")
8. + .SetEnvironmentVariable("EUREKA\_\_CLIENT\_\_SHOULDREGISTERWITHEUREKA", "false")
9. .Build();
10. \_allocationsServer = TestAppServerBuilder()
11. .AppName("AllocationsServer")
12. .Port(8881)
13. .Database("tracker\_allocations\_dotnet\_test")
14. .SetEnvironmentVariable("REGISTRATION\_SERVER\_ENDPOINT", \_registrationServer.Url())
15. + .SetEnvironmentVariable("EUREKA\_\_CLIENT\_\_SHOULDFETCHREGISTRY", "false")
16. .Build();
17. \_backlogServer = TestAppServerBuilder()
18. .AppName("BacklogServer")
19. .Port(8882)
20. .Database("tracker\_backlog\_dotnet\_test")
21. .SetEnvironmentVariable("REGISTRATION\_SERVER\_ENDPOINT", \_registrationServer.Url())
22. + .SetEnvironmentVariable("EUREKA\_\_CLIENT\_\_SHOULDFETCHREGISTRY", "false")
23. .Build();
24. \_timesheetsServer = TestAppServerBuilder()
25. .AppName("TimesheetsServer")
26. .Port(8884)
27. .Database("tracker\_timesheets\_dotnet\_test")
28. .SetEnvironmentVariable("REGISTRATION\_SERVER\_ENDPOINT", \_registrationServer.Url())
29. + .SetEnvironmentVariable("EUREKA\_\_CLIENT\_\_SHOULDFETCHREGISTRY", "false")
30. .Build();
31. }

# Deploy

Now deploy your applications to Cloud Foundry

1. Create the service instance
2. cf create-service p-service-registry standard tracker-service-registry

Creation of the service will take a few minutes. Monitor the status with cf services.

1. Update all four manifests to **add** the service binding.
2. services:
3. *# ...*
4. - tracker-service-registry
5. Because the registration server url is now being resolved via service discovery, we no longer need to explicitly set it using an environment variable. Remove the REGISTRATION\_SERVER\_ENDPOINT environment variable from the manifests of **AllocationServer**, **BacklogServer**, **TimesheetsServer**.
6. # ...
7. path:
8. - env:
9. - REGISTRATION\_SERVER\_ENDPOINT: http://FILL\_ME\_IN/
10. services:
11. Remove the REGISTRATION\_SERVER\_ENDPOINT environment variable from each of the applications that have it set. For example:
12. cf **unset**-env tracker-timesheets REGISTRATION\_SERVER\_ENDPOINT
13. Commit your changes and push your code to GitHub. CircleCI will trigger the build and deploy all four applications.

Make sure the applications are working on Cloud Foundry by exercising endpoints.

# Assignment

Submit the following assignment:

**cd** ~/workspace/assignment-submission

./gradlew dotnetCloudNativeDeveloperDistributedSystemWithServiceDiscovery \

-PregistrationServerUrl=https://<registration-app-url> \

-PbacklogServerUrl=https://<backlog-app-url> \

-PallocationsServerUrl=https://<allocations-app-url> \

-PtimesheetsServerUrl=https://<timesheets-app-url>

# Extra

If you are finished with this assignment before the rest of the class is done, try deploying your local eureka server to Cloud Foundry and configure your apps to use it. What are the pain points here? If you still have time, try scaling the instances and making each instance [peer aware](http://projects.spring.io/spring-cloud/spring-cloud.html#_peer_awareness).