Pattern: 3rd Party Registration

Context

You have applied either the Client-side Service Discovery pattern (client-side-discovery.html) or the Server-side Service Discovery pattern (server-side-discovery.html). Service instances must be registered with the service registry (service-registry.html) on startup so that they can be discovered and unregistered on shutdown.

Problem

How are service instances registered with and unregistered from the service registry?

Forces

- Service instances must be registered with the service registry on startup and unregistered on shutdown
- Service instances that crash must be unregistered from the service registry
- Service instances that are running but incapable of handling requests must be unregistered from the service registry

Solution

A 3rd party registrar is responsible for registering and unregistering a service instance with the service registry. When the service instance starts up, the registrar registers the service instance with the service registry. When the service instance shuts downs, the registrar unregisters the service instance from the service registry.

Examples

Examples of the 3rd Party Registration pattern include:

- Netflix Prana (https://github.com/Netflix/Prana) - a “side car” application that runs along side a non-JVM application and registers the application with Eureka.
- AWS Autoscaling Groups (http://aws.amazon.com/autoscaling/) automatically (un)registers EC2 instances with Elastic Load Balancer
- Joyent’s Container buddy (https://github.com/joyent/containerbuddy) runs in a Docker container as the parent process for the service and registers it with the registry
- Registrator (https://github.com/gliderlabs/registrator) - registers and unregisters Docker containers with various service registries
- Clustering frameworks such as Kubernetes (https://github.com/GoogleCloudPlatform/kubernetes/blob/master/docs/services.md) and Marathon (https://mesosphere.github.io/marathon/docs/service-discovery-load-balancing.html) (un)register service instances with the built-in/implicit registry

Resulting context

The benefits of the 3rd Party Registration pattern include:

- The service code is less complex than when using the Self Registration pattern (self-registration.html) since its not responsible for registering itself
- The registrar can perform health checks on a service instance and register/unregister the instance based the health check

There are also some drawbacks:

- The 3rd party registrar might only have superficial knowledge of the state of the service instance, e.g. RUNNING or NOT RUNNING and so might not know whether it can handle requests. However, as mentioned above some registrars such as Netflix Prana perform a health check in order to determine the availability of the service instance.
- Unless the registrar is part of the infrastructure it's another component that must be installed, configured and maintained. Also, since it's a critical system component it needs to be highly available.

Related patterns

- Service Registry (service-registry.html)
- Client Side Discovery (client-side-discovery.html)
- Server Side Discovery (server-side-discovery.html)
- Self Registration (self-registration.html) is an alternative solution