Pattern: Service instance per container

Context

You have applied the Microservice architecture pattern (/patterns/microservices.html) and architected your system as a set of services. Each service is deployed as a set of service instances for throughput and availability.

Problem

How are services packaged and deployed?

Forces

- Services are written using a variety of languages, frameworks, and framework versions
- Each service consists of multiple service instances for throughput and availability
- Service must be independently deployable and scalable
- Service instances need to be isolated from one another
- You need to be able to quickly build and deploy a service
- You need to be able to constrain the resources (CPU and memory) consumed by a service
- You need to monitor the behavior of each service instance
- You want deployment to reliable
- You must deploy the application as cost-effectively as possible

Solution

Package the service as a (Docker) container image and deploy each service instance as a container

Examples

Docker is becoming an extremely popular way of packaging and deploying services. Each service is packaged as a Docker image and each service instance is a Docker container. There are several Docker clustering frameworks including:

- Kubernetes (http://kubernetes.io/)
- Marathon/Mesos (https://mesosphere.github.io/marathon/)
- Amazon EC2 Container Service (http://aws.amazon.com/ecs/)

Resulting context

The benefits of this approach include:

- It is straightforward to scale up and down a service by changing the number of container instances.
- The container encapsulates the details of the technology used to build the service. All services are, for example, started and stopped in exactly the same way.
- Each service instance is isolated
- A container imposes limits on the CPU and memory consumed by a service instance
- Containers are extremely fast to build and start. For example, it’s 100x faster to package an application as a Docker container than it is to package it as an AMI. Docker containers also start much faster than a VM since only the application process starts rather than an entire OS.

The drawbacks of this approach include:

- The infrastructure for deploying containers is not as rich as the infrastructure for deploying virtual machines.

Related patterns

- This pattern is a refinement of the Service Instance per Host pattern (single-service-per-host.html)
- The Service Instance per VM pattern (service-per-vm.html) is an alternative solution
- The Serverless deployment pattern (serverless-deployment.html) is an alternative solution.