Pattern: Service Instance per VM

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 virtual machine image and deploy each service instance as a separate VM

Examples

- Netflix packages each service as an EC2 AMI and deploys each service instance as a EC2 instance.

Resulting context

The benefits of this approach include:

- Its straightforward to scale the service by increasing the number of instances. Amazon Autoscaling Groups can even do this automatically based on load.
- The VM 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 VM imposes limits on the CPU and memory consumed by a service instance
- IaaS solutions such as AWS provide a mature and feature rich infrastructure for deploying and managing virtual machines. For example,
    - Elastic Load Balancer -
    - Autoscaling groups
    - …

The drawbacks of this approach include:

- Building a VM image is slow and time consuming

Related patterns

- This pattern is a refinement of the Single Service per Host pattern (single-service-per-host.html)
- The Service Instance per Container pattern (service-per-container.html) is an alternative solution
- The Serverless deployment pattern (/patterns/deployment/serverless-deployment.html) is an alternative solution.