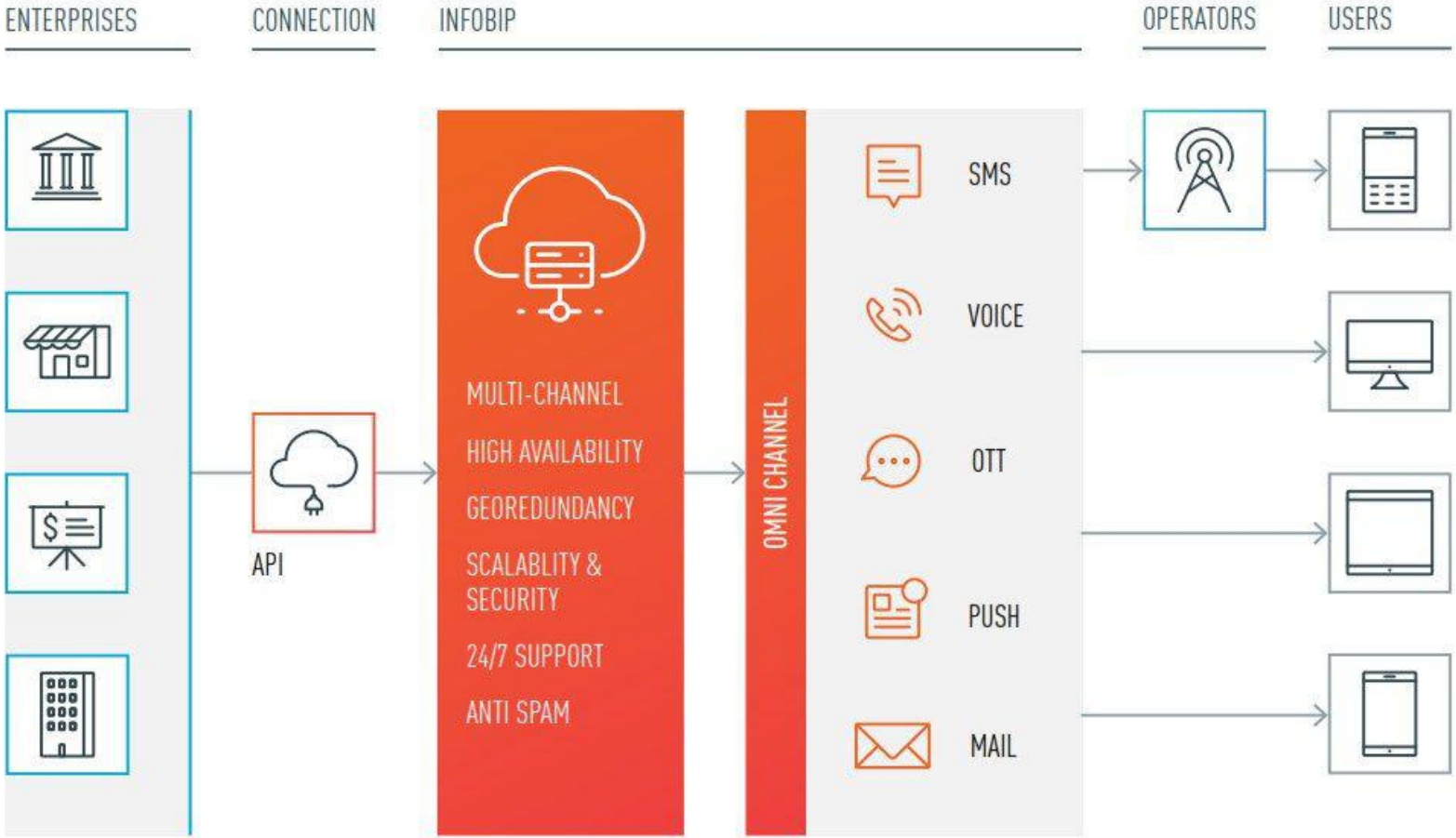
Non-JVM applications in Java microservices

Andrey Markelov - Infobip

DevOps Pro Moscow 2018


ENTERPRISES
CONNECTION
INFOBIP
OPERATORS
USERS
API
MULTI-CHANNEL
HIGH AVAILABILITY
GEOREDUNDANCY
SCALABLITY & SECURITY
24/7 SUPPORT
ANTI SPAM
OMNI CHANNEL
SMS
VOICE
OTT
PUSH
MAIL

IT Infrastructure

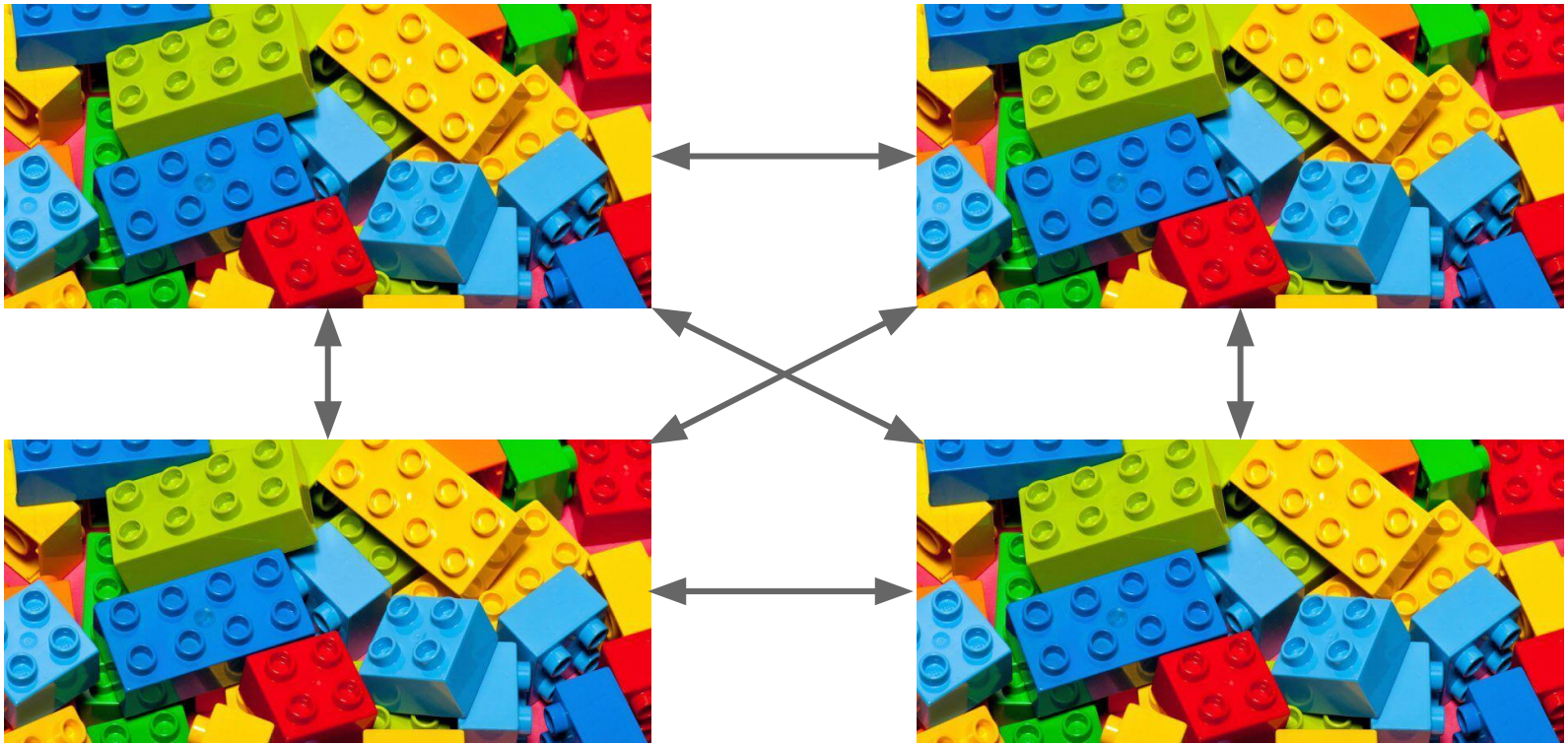
- 9 data centers around the world
- 500+ unique services in production
- 1400+ service instances
- 5TB+ monthly traffic

IT Infrastructure

- 9 data centers around the world
- 500+ unique services in production
- 1400+ service instances
- 5TB+ monthly traffic
- We want to grow

Microservices

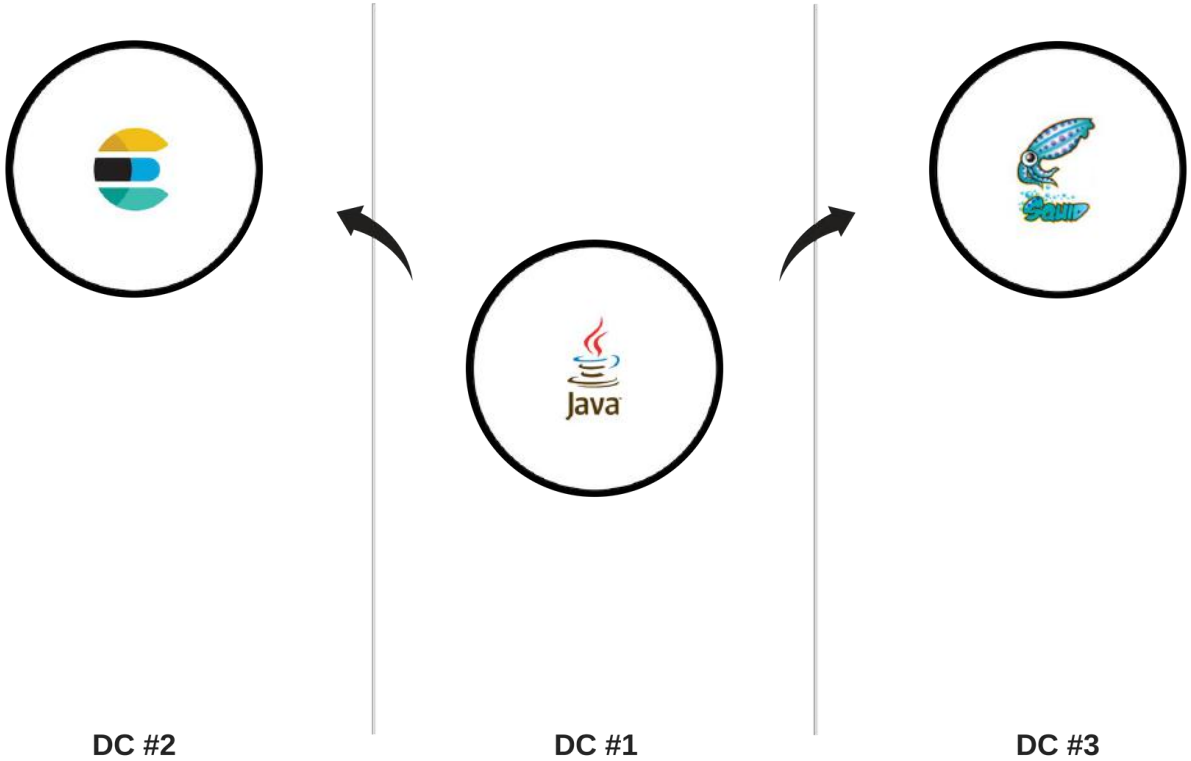
Microservices Multi-DC



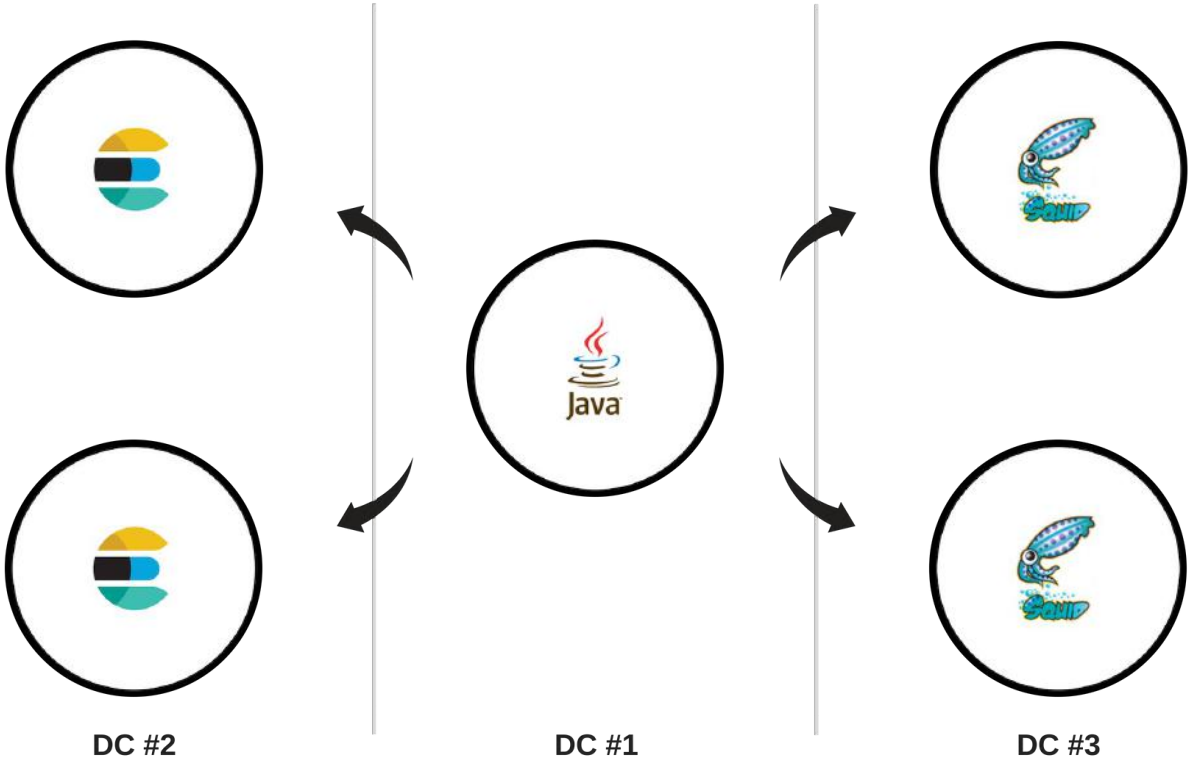
Problems

- Discovering
- Monitoring

Discovering



Discovering



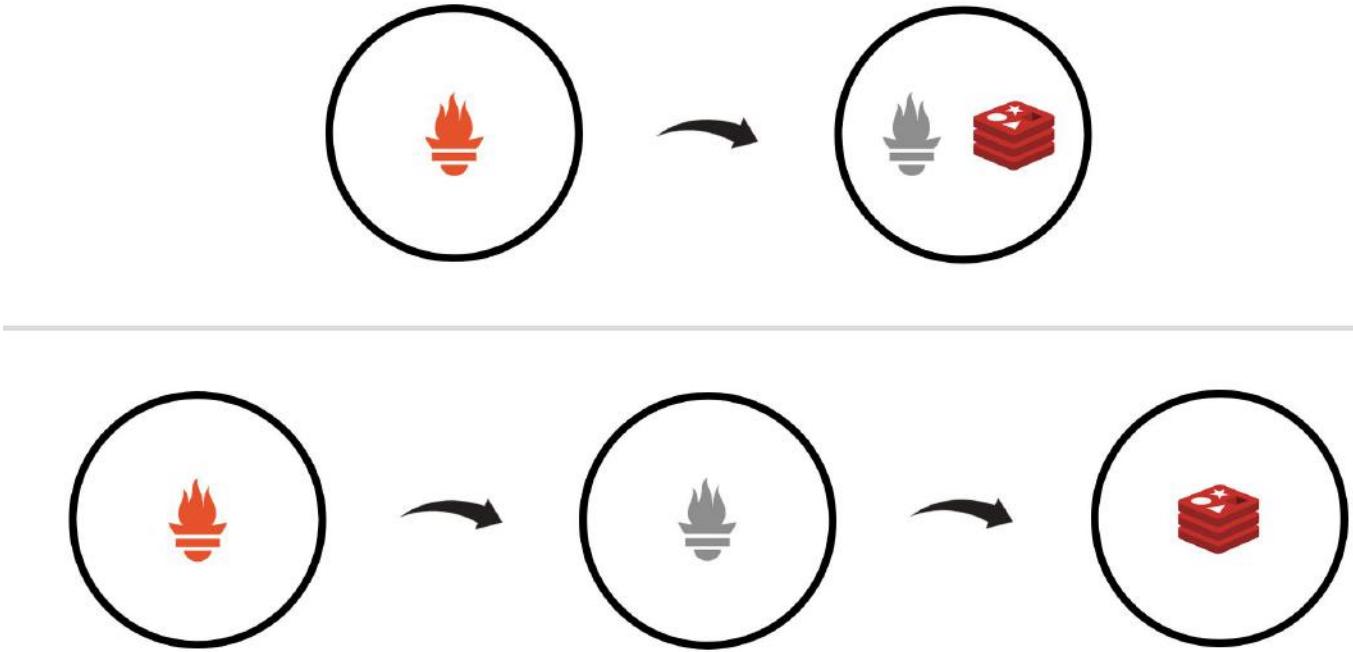
Discovering Problems:

- **New instances should be added manually**
  `redis.uri=redis-sentinel://host1:26379,host2:26379?sentinelMasterId=name`
- **Take care about IP addresses if used**

Discovering Problems:

- **New instances should be added manually**
  `redis.uri=redis-sentinel://host1:26379,host2:26379?sentinelMasterId=name`
- **Take care about IP addresses if used**
- **Too environment variables an application**

Monitoring



Monitoring Problems:

- Add to Prometheus exporter manually
- Exporter fails or application fails?
- Where to place exporter (separate VM)?

Goals

- Autodiscovery non-JVM applications
- Client-side balancing for non-JVM applications
- Monitoring like any other Java application
- Provides application status and health checks
- Perform action on in/out balancer events

Solution

Deploy non-JVM applications with sidecar

Sidecar pattern

Deploy components of an application into a separate process or container to provide isolation and encapsulation. This pattern can also enable applications to be composed of heterogeneous components and technologies.

Context and Problem

Applications and services often require related functionality, such as monitoring, logging, configuration, and networking services. These peripheral tasks can be implemented as separate components or services.

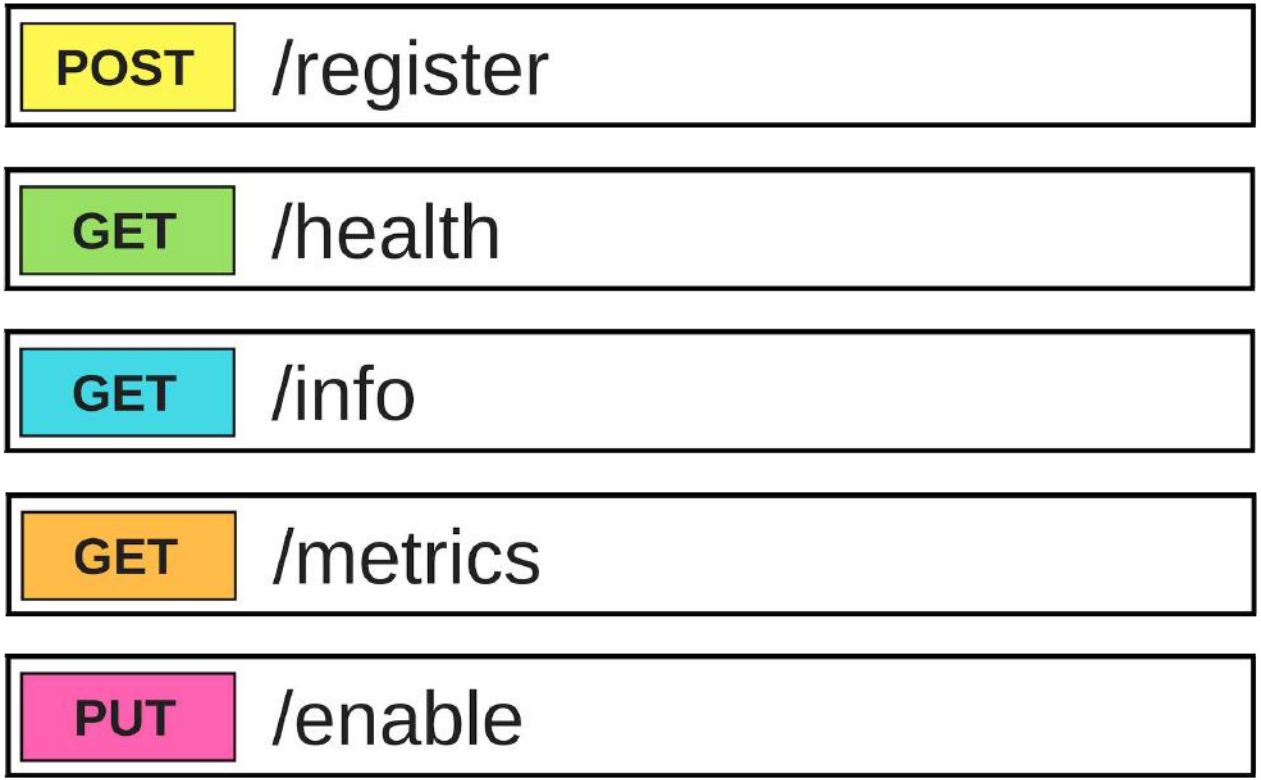
Solution

Co-locate a cohesive set of tasks with the primary application, but place them inside their own process or container, providing a homogeneous interface for platform services across languages.

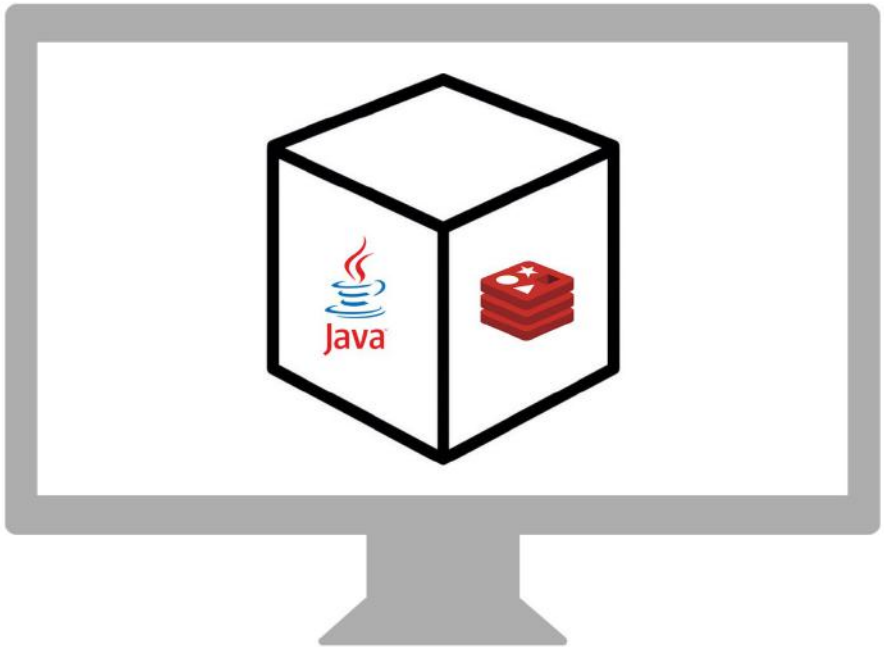
https://docs.microsoft.com/en-us/azure/architecture/patterns/sidecar

Sidecar features:

- Register in service-discovery on start
- Provide endpoints health?check and status
- Provide endpoints for enable/disable
- Provide endpoint for metrics (for Prometheus)


POST
/register
GET
/health
GET
/info
GET
/metrics
PUT
/enable

Deploy inside one container



Deploy inside one container

Advantages:

- Specialized sidecar for application
- Command line tools
- Use application files

Drawbacks:

- Hard to maintain

Deploy inside one container - health check

```
@Override
public Health health() {
    try {
        Process process = Runtime.getRuntime().exec("redis-cli ping");
        String output = IOUtils.toString(process.getInputStream());
        if (output.indexOf("PONG") != -1) {
            return Health.up().build();
        } else {
            return Health.down().withDetail("reason", output).build();
        }
    } catch (IOException e) {
        return Health.down().withException(e).build();
    }
}
```

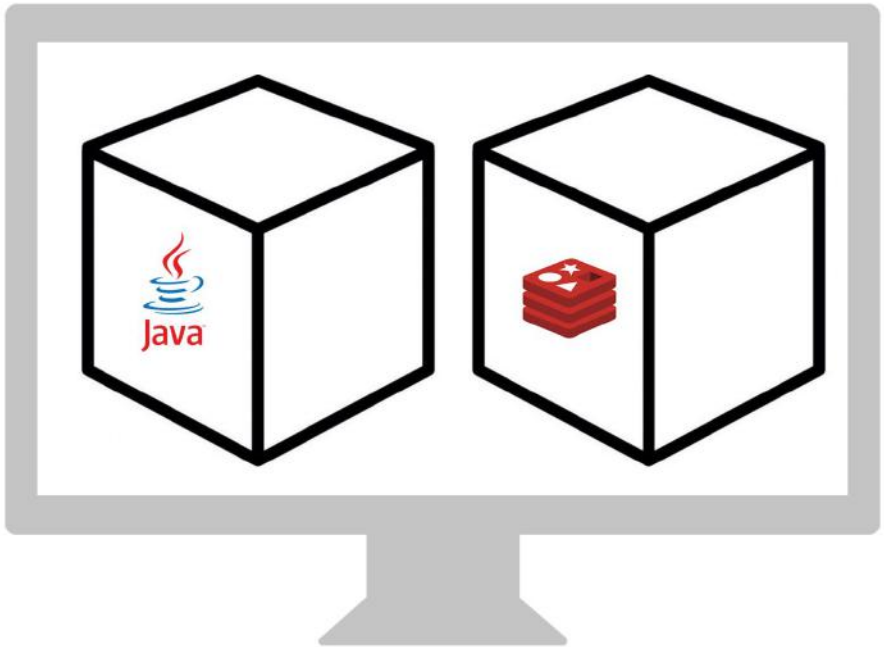
Deploy inside one container - metrics

```
@Override
public List<MetricFamilySamples> collect() {
   try {
       Process process = Runtime.getRuntime().exec("redis-cli info");
       String output = IOUtils.toString(process.getInputStream());

       String[] lines = output.split("\\r?\\n");
       for (String line : lines) {
           if (line.startsWith("used_memory:")) {
               usedMemory.set(Double.parseDouble(line.substring("used_memory:".length())));
           }
           if (line.startsWith("db0:keys=")) {
               keysCount.set(Double.parseDouble(line.substring("db0:keys=".length(), line.indexOf(","))));
           }
       }
   } catch (Exception ex) {
       // log or something
   }

   List<MetricFamilySamples> result = new ArrayList<>();
   result.addAll(usedMemory.collect());
   result.addAll(keysCount.collect());
   return result;
}
```

Deploy inside one container – metrics example

```
# HELP jvm_buffer_memory_used_bytes An estimate of the memory that the Java virtual machine is using for this buffer pool
# TYPE jvm_buffer_memory_used_bytes gauge
jvm_buffer_memory_used_bytes{id="direct",} 81920.0
jvm_buffer_memory_used_bytes{id="mapped",} 0.0
# HELP redis_used_memory How much memory used
# TYPE redis_used_memory gauge
redis_used_memory 814888.0
# HELP redis_keys_count How many keys
# TYPE redis_keys_count gauge
redis_keys_count 0.0
# HELP jvm_memory_committed_bytes The amount of memory in bytes that is committed for the Java virtual machine to use
# TYPE jvm_memory_committed_bytes gauge
```

Deploy in separate containers



Deploy in separate containers

Map to the same directory:

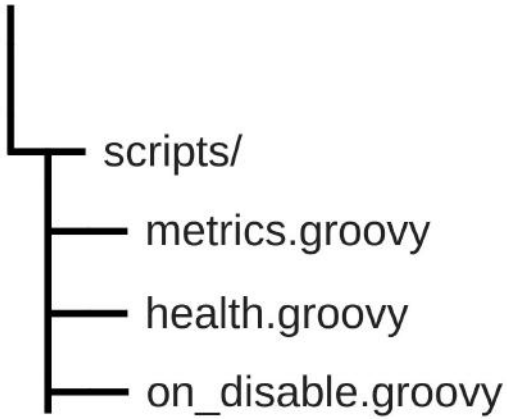
1. Dockerfile

   ```
   VOLUME /scripts
   ```

2. Run docker with --volumes-from

   ```
   docker run --volumes-from <container>
   ```

Deploy in separate containers - structure



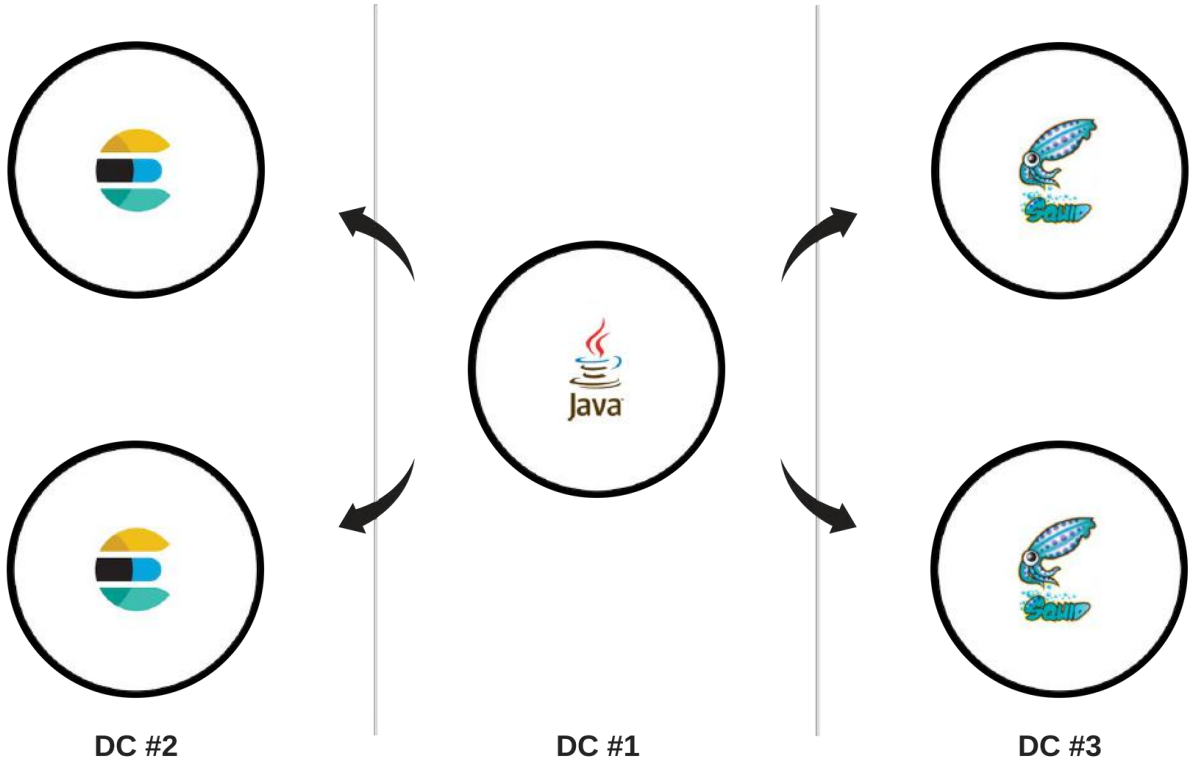
Deploy in separate containers - Dockerfile

```
FROM redis:4.0.11
COPY ./scripts/health.groovy /scripts/
COPY ./start.sh /start.sh
RUN chmod +x /start.sh
EXPOSE 6379
VOLUME /scripts
ENTRYPOINT ["/start.sh"]
CMD []
```

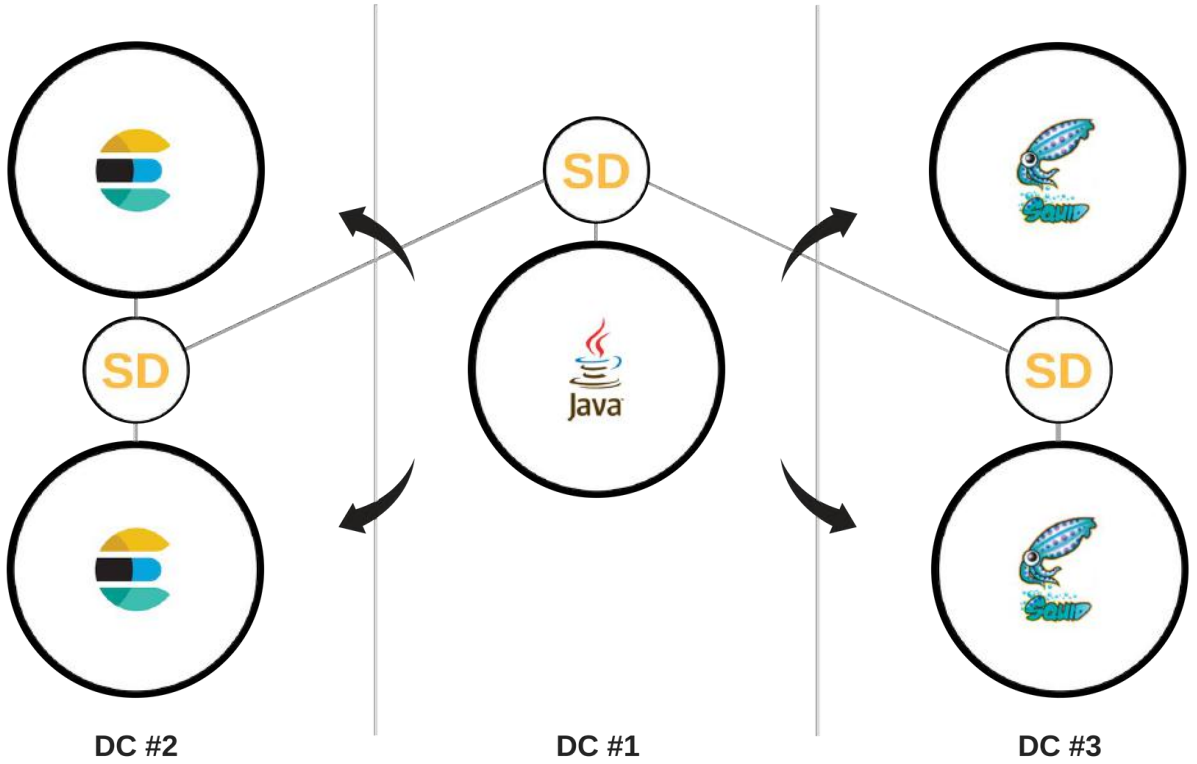
Deploy in separate containers - health.groovy

```
def socket = new Socket()
try {
   socket.connect(new InetSocketAddress('localhost', 6379), 5000)
   socket.withStreams { inputStream, outputStream ->
       outputStream.write("*1\r\n$4\r\nPING\r\n".getBytes())
       outputStream.flush()
       def line = inputStream.newReader().readLine()
       if (line.contains("PONG")) {
           return true
       }
...
```

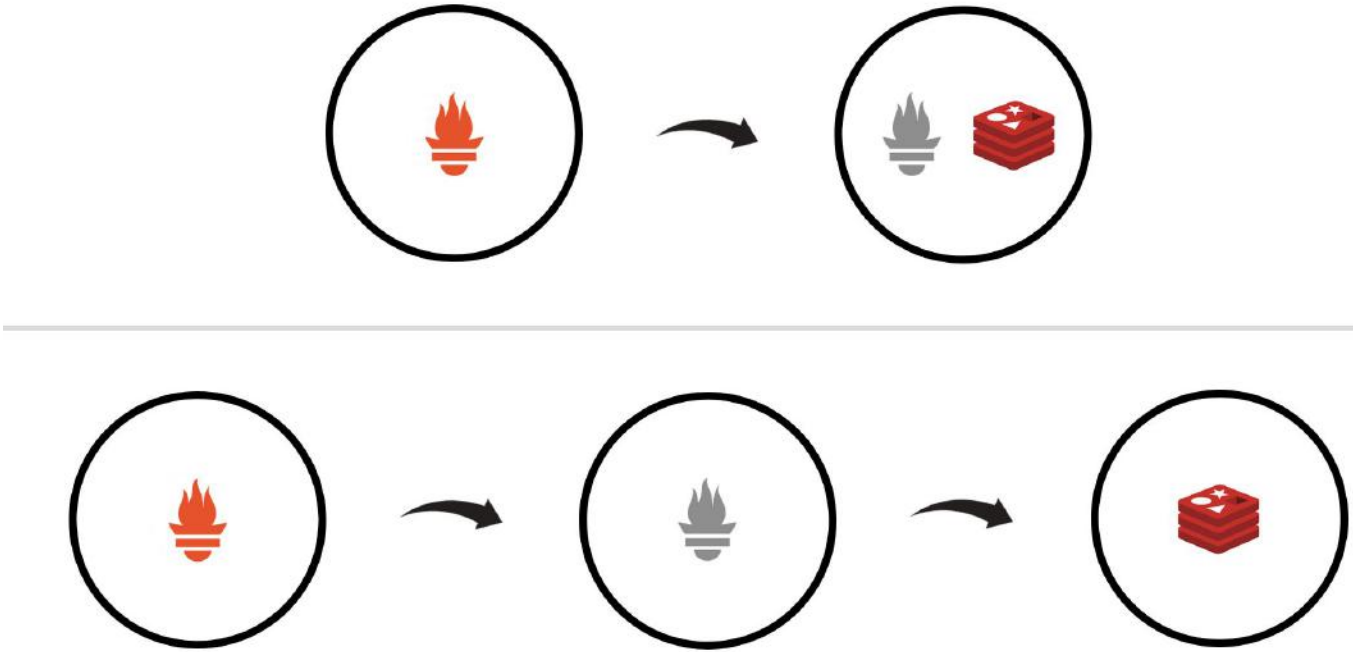
Discovering



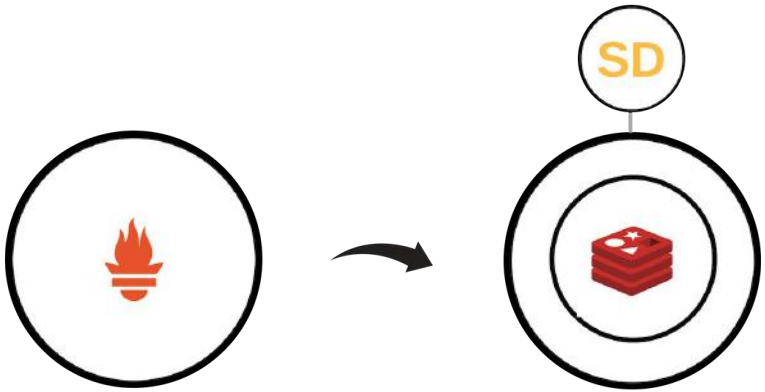
Discovering



Monitoring



Monitoring



Implementations

- AirBNB - Nerve and Synapse
- Netflix - Prana

Resume

- Service discovery for non-JVM
- Health check from the box
- Monitoring from the box

Q/A

Email: andrey.v.markelov@gmail.com

Github: https://github.com/AndreyVMarkelov/devopspro2018-sidecar-demo